Skip Lists and Probabilistic Analysis of Algorithms

by

Thomas Papadakis

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 1993

I hereby declare that I am the sole author of this thesis.

I authorize the University of Waterloo to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the University of Waterloo to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

The University of Waterloo requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

Abstract

This thesis is concerned with various forms of skip lists, and with probabilistic analyses of algorithms. We investigate three topics; one topic from each of these two areas, and another topic common to both of them.

First, we consider Pugh's skip list. We derive exact and asymptotic expressions for the average search cost of a fixed key and of an average key. Our results improve previously known upper bounds of these two average search costs. We also derive exact and asymptotic expressions for the variance of the search cost for the largest key.

Next, we propose several versions of deterministic skip lists. They all have guaranteed logarithmic search and update costs per operation, they lead to an interesting "bridge" structure between the original skip list and standard search trees, they are simpler to implement than standard balanced search trees, and our experimental results suggest that they are also competitive in terms of space and time.

Finally, we consider the elastic-bucket trie, a variant of the standard trie, in which each external node (bucket) has precisely as many key slots as the number of keys stored in it. We examine the number of buckets of each size, and we derive exact and asymptotic expressions for their average values, as well as asymptotic expressions for their variances and covariances under the closely related "Poisson model" of randomness. Our experimental results suggest that maintaining only two bucket sizes may be a very reasonable practical choice.

Acknowledgements

I am very grateful to my supervisor, Prof. Ian Munro, for his continuous help and encouragement. Ian has always been available whenever and for as long as I needed him. With his brilliant ideas, his unmatched intuition on research directions, his unparalleled knowledge on academic and non-academic issues, his generous financial support, and his great sense of humour, Ian helped enormously to make this thesis possible and my stay at Waterloo more enjoyable.

This thesis would also have been impossible without the help from Prof. Hosam Mahmoud. Hosam introduced me to the realm of Mellin transforms during his visit to Waterloo by giving me a preprint of his wonderful book. Since then, he has spent extraordinary amounts of time answering all my questions on related topics, and he has been a constant source of praise and encouragement for me.

Very special thanks go also to Prof. Patricio Poblete for his help, especially with Chapter 5. Patricio has always been available to discuss my ideas, answer my questions, and correct my mistakes.

Many thanks go also to Prof. Robert Sedgewick for his contribution to the material of Chapter 4, especially for the top-down algorithms.

The present form of this thesis has been greatly improved after critical comments made by Ian, Hosam and Patricio, who read earlier drafts of it. I am also grateful to the other members of my committee, Profs. Luc Devroye, Jeff Shallit, and Bruce Richmond, for the careful reading of my thesis and for their useful suggestions.

Finally, I owe many thanks to Maria for her support and understanding during all these years, and to my family in Greece for their continuing and unconditional love. And last, but not least, many thanks to Costas, Tina, and little George for making us feel at home the last couple of years.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The orderly maintenance of a collection (*set*) of items is a fundamental prerequisite of almost any computational task. While airlines have to keep track of available space and fares, governments have to store identification numbers and names of their citizens, and compilers have to maintain a list of the types and values for all encountered variables.

For some computational tasks (such as the above), the ability to *insert* a new item in a set, to *delete* an existing item from a set, and to *search* for an item in a set is probably all that is required. A set of items, together with these three operations, is called a *dictionary*.

A common assumption about dictionaries is that, although each item may consist of several fields, there exists an identifying *key* field. When we search for an item, we actually search for its key, and if we find it, we retrieve its associated fields in one additional operation. Similarly, when we insert (delete) an item, we search for its key, and if the insertion (deletion) is to be successful, we add (remove) the key, along with its associated fields. For our purposes, in this thesis, we are going

to *consider only the key fields* of the items stored in the dictionary. The number of key fields will be denoted by n, throughout our thesis.

To evaluate a dictionary implementation, one may look at its

- time complexity (in the average and in the worst case)
- space complexity (in the average and in the worst case)
- simplicity

The time complexity — at least among theoretical computer scientists — is measured as the number of times a basic operation (e.g. a key comparison) is executed, and it is expressed in terms of a parameter (e.g. number of keys in the dictionary). Similarly, the space complexity is measured as the number of additional "cells" (e.g. memory addresses) required. Since no model for the simplicity of a solution exist, only ad-hoc arguments are possible here.

In this thesis, we derive new results on the time and space complexity of two known dictionary implementations, and we propose a new and simpler implementation.

1.1 History

The oldest and most straightforward dictionary implementation is the sequential search algorithm on an unsorted file. It is very simple and it uses only $\Theta(1)$ extra space, but its disadvantage is that a search requires $\frac{n}{2}$ key comparisons on the average, and n key comparisons in the worst case.

The binary search on sorted files was the next method to be proposed. Knuth gives [26, pp. 417–419] a very interesting account on how tough the notion of a

"sorted" file was a few centuries ago. The first large sorted file of numbers can be traced back to 200 B.C., but the first alphabetized dictionary didn't appear before 1286 (or before the beginning of the 12th century, according to [20, p. 380]), and the first purely English dictionary, in 1604, came with lengthy explanations on what "alphabetical order" meant! In any case, Knuth reports that the first binary search algorithm was discussed in a 1946 paper, but the first correct general version of it was only published in 1962. The binary search method takes $\Theta(1)$ extra space, $\Theta(\lg n)$ key comparisons for a search in the worst and in the average case, but $\Theta(n)$ time for an update (i.e. an insertion or a deletion) in both the worst and the average case.

Another dictionary implementation is the binary search tree. According to Knuth [26, pp. 446–447], this structure was first discussed in 1952, but it wasn't fully described before 1960. Unlike its predecessors, the binary search tree requires 2 extra memory addresses (pointers) per key. In return, it offers $\Theta(\lg n)$ search and update costs in the average case, although its worst case costs are $\Theta(n)$.

In 1962, the two Russians Adel'son-Vel'skii and Landis proposed [1] a method of balancing the binary search tree, achieving thus, for the first time, $\Theta(\lg n)$ search and update costs in the worst (and in the average) case. Their scheme was subsequently called the AVL tree, after their names. A straightforward implementation of this scheme requires 2 extra pointers *plus 2 extra bits* per key.

Additional balanced search trees were proposed in the 1970's. The most well-known of the dozens of such schemes are the 2-3 tree proposed in 1970 by Hopcroft [26, p. 468], the B-tree proposed in 1972 by Bayer and McCreight [6] and the red-black tree proposed in 1978 by Guibas and Sedgewick [24]. All these schemes use $\Theta(n)$ extra space and achieve a $\Theta(\lg n)$ search and update cost in the worst case.

Sleator and Tarjan [44] in 1985 proposed the splay tree, a self-balancing binary search tree. Although a single operation in such a tree may take $\Theta(n)$ time in the worst case, any sequence of m operations is guaranteed to take $\Theta(m \lg n)$ time in the worst case.

A common underlying characteristic of all solutions mentioned so far, is that the only permissible key operations are comparisons between entire keys. The trie, first proposed in late 1950's by de la Briandais [8] and Fredkin [19], is a tree-like data structure, in which, in order to search for a key, one compares one bit (or character) at a time. The keys themselves are stored in the external nodes of the tree. Although the worst case behaviour of tries is equal to the length of the longest key in the structure, on the average they require $\Theta(n)$ extra space and they have $\Theta(\lg n)$ search and update costs. In practice, they do very well when the keys are character strings of variable length or very long; in either case, key-comparison based algorithms perform poorly.

Finally, a totally different solution to the dictionary problem is hashing. Knuth reports [26, p. 541] that this idea was first discussed in 1953 and it was first published in 1956. Hashing is based on arithmetic transformations of the keys (into table addresses). Its efficiency depends on the amount of space one is willing to waste, but in practice it performs extremely well. Unlike all previous methods, hashing cannot perform operations based on the relative order of keys, if usual hashing functions are used. For example, it cannot (efficiently) find the minimum or maximum key in the dictionary, it cannot handle range queries, print the keys in order, etc. Hashing is not considered at all in this thesis.

1.2 Thesis Preview

The main topic of this thesis is the skip list, in various forms; we propose a (new) deterministic version of the skip list as a simple alternative to balanced search trees, and we analyze the time complexity of the original randomized version of the skip list. The techniques used in the latter analysis are also used to analyze the space complexity of bucket tries.

In Chapter 2, we describe the probabilistic models that we will consider in this thesis, and we summarize all mathematical machinery that we are going to use. The latter includes probability and moment generating functions of random variables, basic binomial coefficient identities, well-known Gamma and Psi function properties, exponential generating functions, elementary complex variable theory, together with binomial, Poisson and Mellin transforms, and Rice's method for deriving asymptotics for alternating sums. All of the above are introduced from first principles, and nothing beyond what is discussed in Chapter 2 is needed for the complete understanding of our dissertation. Care has been taken to present separately (Section 2.1) all facts necessary for the understanding of our bare results, independently of the techniques used to derive them.

In Chapter 3, we consider the average search time of the probabilistic skip list (PSL), a data structure recently proposed by Pugh [40]. The main advantage of the PSL is its simplicity; it is not coincidental that only three years after its original publication, it has gained its place in introductory Data Structure and Algorithms textbooks [28, 46, 51]. On the average, the PSL is very competitive with other data structures. In terms of space, it requires $\frac{1}{1-p}$ extra pointers per key, where p is a user-specified parameter with $0 < p < 1$. In terms of time, Pugh showed that a search for the mth key in the structure requires *at most*

$\log_{\frac{1}{p}} n + \frac{1-p}{p} \log_{\frac{1}{p}}(m-1)$ + "constant" key comparisons, and he gave an expression for the constant term. In Chapter 3, we show that the required number of key comparisons is *exactly* $\log_{\frac{1}{p}} n + \frac{1-p}{p} \log_{\frac{1}{p}}(m-1)$ + "constant" + $O\left(\frac{1}{m}\right)$, and we provide a precise expression for the constant term. This expression involves some well-known constants, and some oscillations expressed as a series of terms of rotations of the complex Gamma function. Although this result may sound complicated, its derivation is not, because the search cost in a PSL is related to the depth of a leaf in a trie, a problem well-studied before [26].

Despite the fact that *on the average* the PSL performs reasonably well, in the *worst case* its $\Theta(n \lg n)$ space and $\Theta(n)$ time complexity are unacceptably high. In Chapter 4, we present several simple versions of a deterministic skip list (DSL), all of which require $\Theta(\lg n)$ time for a search or an update in the *worst case* and an additional $\Theta(n)$ number of pointers. The main advantage of our DSL, when compared with other balanced tree schemes also having $\Theta(\lg n)$ worst case costs, is its simplicity. Although one cannot rigorously "measure" the simplicity of a solution, we believe that this issue has unjustifiably been neglected (see also [2]). Even seemingly simple algorithms require sometimes considerable coding efforts, becoming thus costly solutions. For example, for the *very simple* binary search algorithm (on a sorted array), Bentley reported [7] in 1983:

> I 've given this problem as an in-class assignment in courses at Bell Labs and IBM. The professional programmers had one hour (sometimes more) to convert the above description [of the binary search] into a program in the language of their choice; a high-level pseudocode was fine. At the end of the specified time, almost all the programmers reported that they had correct code for the task ... In many different classes and with over a hundred programmers, the results varied little:

> 90 percent of the programmers found [later] bugs in their code (and I wasn't always convinced of the correctness of the code in which no bugs were found).

Next, we shift our attention back into the PSL. Although the expected value of the search cost has been derived (Chapter 3), and additional results on its asymptotic convergence are known [12, 13], its variance seemed to be a hard nut to crack up to now. In Chapter 5, we derive the variance of the search cost for the largest key in the PSL. The entire mathematical machinery stated in Chapter 2 will be used for the derivation of our results, but the results themselves are fairly simple to understand.

In Chapter 6, we follow the same approach as in Chapter 5, and we analyze an aspect of the space complexity in tries. As discussed towards the end of Section 1.1, the keys of a trie are stored in external nodes (buckets). If we now allow more than one key per bucket, we have a bucket trie. If we furthermore allow the buckets to be of variable size, so that no empty key slots exist, then we have an elastic-bucket trie. Despite the plethora of results (46 papers listed in [21, Section 3.4.4]) on the path length, the height, the depth of a leaf and the number of nodes in a trie, nobody had studied the number of buckets of each size up to now. In Chapter 6, we derive the expected value and the variance of an arbitrary linear combination of all bucket sizes. As a consequence of our expression for the variance, we obtain the complete variance-covariance matrix of all bucket sizes.

We conclude this thesis in Chapter 7 with a summary of our results and a discussion of directions for possible future research.

Chapter 2

Mathematical Prerequisites

In this chapter, we give the mathematical machinery that we will use in this thesis; we have assumed the reader's knowledge of the basics, such as big-Oh notation, etc. In the first two sections, we state everything that is required for the *understanding* of our results. In the last section, we present the mathematical tools necessary for the *derivation* of our results. The non-mathematically inclined reader may skip Section 2.3, as well as everything under "proof" in the rest of the thesis.

2.1 Mathematical Notation

The probability that event A occurs will be denoted by $\mathbf{Pr}[A]$. If X is a random variable, then $\mathbf{E}(X)$ and $\mathbf{Var}(X)$ will denote its expected value and variance respectively. For two random variables X and Y, $\mathbf{Cov}(X, Y)$ will denote their covariance.

As we said earlier, n has been reserved for the number of keys in the dictionary. The logarithms of x to the base e and to the base 2 will be denoted by $\ln x$ and

$\lg x$ respectively. Euler's constant, 0.5772..., will be denoted by γ, and i will be reserved for $\sqrt{-1}$. If $\langle a_k \rangle$ and $\langle b_k \rangle$ are two sequences of real numbers, we will write $a_k \sim b_k$ when $\lim_{k\to\infty} \frac{a_k}{b_k} = 1$. Finally, as suggested in [23, p. 24], if a true/false statement P is enclosed in brackets such as $[P]$, the result will be 1 if P is true, and 0 if P is false; for example, Kronecker's delta $\delta_{j,k}$ may be written as $[j=k]$.

If $s=a+ib=r(\cos\theta+i\sin\theta)$ is a complex number, then $\overline{s}$ will be its conjugate $a-ib$, $|s|$ will be its absolute value $r=\sqrt{a^2+b^2}$, $\arg(s)$ will be its argument θ, and $\Re(s)$ will be its real part a. The standard complex Gamma function

$$\Gamma(s) \stackrel{\text{def}}{=} \int_0^\infty e^{-z} z^{s-1} dz, \qquad \text{for } \Re(s) > 0 \tag{2.1}$$

and its log-derivative Psi (or Digamma) function

$$\Psi(s) = \frac{d}{ds} \ln \Gamma(s) = \frac{\Gamma'(s)}{\Gamma(s)} \tag{2.2}$$

show up in the oscillating functions

$$f_{r,x}(l) \stackrel{\text{def}}{=} \frac{2}{\ln x} \sum_{k\geq 1} \Re\left(\Gamma\left(r - i\frac{2k\pi}{\ln x}\right) e^{i\,2k\pi \log_x l}\right), \quad \text{integer } r, \quad \text{real } x, \quad \text{real } l>0 \tag{2.3}$$

(first introduced in Knuth's analysis of radix exchange sort [26, ex. 5.2.2.50]), and

$$g_{r,x}(l) \stackrel{\text{def}}{=} \frac{2}{\ln x} \sum_{k\geq 1} \Re\left(\Psi\left(r - i\frac{2k\pi}{\ln x}\right)\Gamma\left(r - i\frac{2k\pi}{\ln x}\right) e^{i\,2k\pi \log_x l}\right), \text{ integer } r, \text{ real } x, \text{ real } l>0, \tag{2.4}$$

both of which will appear repeatedly in our results.

2.2 Probabilistic Models

We are going to consider two different probabilistic models. In the *fixed population model*, the number of keys in the dictionary is a fixed number n. In the *Poisson*

model (first considered — in computer science — by Fagin et al. [14] in 1979) the number of keys is not fixed; rather, it is a random variable N, following the well-known Poisson distribution

$$\mathbf{Pr}[N=k] = e^{-z}\frac{z^k}{k!}, \qquad \text{for } k = 0, 1, \ldots \tag{2.5}$$

with mean z. In this thesis, the time or space complexity of a particular dictionary implementation will be a random variable, denoted by, say, Y_n under the fixed population model, and by $Y(z)$ under the Poisson model. Sometimes, $Y(z)$ will be referred to as the *Poissonized version* of Y_n. Typically, we will be interested in the derivation of moments of Y_n and $Y(z)$.

Although one may argue that the fixed population model is more realistic than the Poisson model, we are going to see shortly (Section 2.3.6) that the Poisson model may be viewed merely as a mathematical transformation which may facilitate derivation of results under the fixed population model.

2.3 Mathematical Foundations

2.3.1 Moments of Random Variables

Let X be a random variable taking the values $0, 1, 2, \ldots$. The *probability generating function* (pgf) of X is

$$G(z) \stackrel{\text{def}}{=} \mathbf{E}(z^X) = \sum_{k\geq 0} \mathbf{Pr}[X=k]\, z^k. \tag{2.6}$$

For our purposes, the pgf is useful, because it gives us factorial moments of X:

$$\mathbf{E}(X) = G'(1) \tag{2.7}$$

and

$$\mathbf{E}\Big(X(X-1)\Big) = G''(1). \tag{2.8}$$

The *moment generating function* (mgf) of a random variable Y is

$$M(z) \stackrel{\text{def}}{=} \mathbf{E}(e^{zY}) = \sum_{k\geq 0} \mathbf{E}(Y^k)\frac{z^k}{k!}. \tag{2.9}$$

The mgf gives us moments of Y:

$$\mathbf{E}(Y) = M'(0) \tag{2.10}$$

and

$$\mathbf{E}(Y^2) = M''(0). \tag{2.11}$$

2.3.2 Binomial Coefficients

The formal definition of the *binomial coefficient* is

$$\binom{r}{k} \stackrel{\text{def}}{=} \begin{cases} \dfrac{r^{\underline{k}}}{k!}, & \text{integer } k\geq 0 \\ 0, & \text{integer } k<0 \end{cases}, \quad \text{real } r, \tag{2.12}$$

where

$$r^{\underline{k}} \stackrel{\text{def}}{=} r(r-1)\cdots(r-k+1), \qquad \text{real } r, \quad \text{integer } k\geq 0$$

is the kth *falling factorial power* of r. From this definition, we immediately get $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ for integer $n \geq k \geq 0$, $\binom{-1}{k} = (-1)^k$ for $k \geq 0$, and $\binom{k}{k} = [k\geq 0]$. The reader is also assumed to be familiar with elementary properties of binomial coefficients, such as symmetry $\binom{m}{k} = \binom{m}{m-k}$, absorption $\binom{r}{k} = \frac{r}{k}\binom{r-1}{k-1}$, upper negation $\binom{r}{k} = (-1)^k\binom{k-r-1}{k}$, trinomial revision $\binom{r}{m}\binom{m}{k} = \binom{r}{k}\binom{r-k}{m-k}$, the addition formula $\binom{r}{k} = \binom{r-1}{k} + \binom{r-1}{k-1}$, and the binomial theorem

$$\sum_{k=0}^{r}\binom{r}{k}x^k y^{r-k} = (x+y)^r, \qquad \text{integer } r. \tag{2.13}$$

The parallel summation

$$\sum_{k=0}^{m} \binom{r+k}{k} = \binom{r+1+m}{r+1}, \qquad \text{integer } m, \quad \text{real } r \tag{2.14}$$

and the identity [23, eq. (5.24)]

$$\sum_{\text{all } l} \binom{r}{m+l}\binom{s+l}{n}(-1)^l = (-1)^{r+m}\binom{s-m}{n-r}, \quad \text{integers } m, n, \quad \text{integer } r \geq 0, \quad \text{real } s \tag{2.15}$$

will also be used in our proofs.

2.3.3 Properties of Gamma and Psi Functions

The most well-known property of the Gamma function is

$$\Gamma(s+1) = s\Gamma(s), \qquad \text{complex } s \neq 0, -1, -2, \ldots. \tag{2.16}$$

Since

$$\Gamma(1) = 1, \tag{2.17}$$

the Gamma function becomes the factorial function for integer arguments:

$$\Gamma(k) = (k-1)!, \qquad \text{integer } k \geq 1. \tag{2.18}$$

The absolute value of the Gamma function on the imaginary axis is given by

$$|\Gamma(\pm iy)|^2 = \frac{\pi}{y \sinh(\pi y)}, \qquad \text{real } y, \tag{2.19}$$

where $\sinh t = \frac{e^t - e^{-t}}{2}$ is the hyperbolic sine of t; the "$\pm$" in the argument of the Gamma function above follows from

$$\Gamma(\overline{s}) = \overline{\Gamma(s)}, \qquad \text{complex } s. \tag{2.20}$$

From (2.16) and (2.19), we can immediately get

$$|\Gamma(1 \pm iy)|^2 = \frac{y\pi}{\sinh(\pi y)}, \qquad \text{real } y \tag{2.21}$$

and

$$|\Gamma(-1 \pm iy)|^2 = \frac{\pi}{y(1+y^2)\sinh(\pi y)}, \qquad \text{real } y, \tag{2.22}$$

and, working similarly, we can derive $|\Gamma(\pm r \pm iy)|$ for any integer r. The asymptotic approximations

$$\frac{\Gamma(x+a)}{\Gamma(x+b)} = x^{a-b} + O\left(x^{a-b-1}\right) \quad \text{as } x \to \infty, \qquad \text{complex } a, b \tag{2.23}$$

and

$$|\Gamma(x \pm iy)| = O\left(y^{x-\frac{1}{2}} e^{-\frac{\pi y}{2}}\right) \quad \text{as } y \to \infty, \qquad \text{real } x \tag{2.24}$$

of the Gamma function will also be used repeatedly in our derivations.

For the Psi function, we have the recurrence

$$\Psi(s+1) = \Psi(s) + \frac{1}{s}, \qquad \text{complex } s. \tag{2.25}$$

Since

$$\Psi(1) = -\gamma, \tag{2.26}$$

the last recurrence gives the integer values

$$\Psi(k) = -\gamma + H_{k-1}, \qquad \text{integer } k \geq 1, \tag{2.27}$$

where

$$H_l \stackrel{\text{def}}{=} \sum_{j=1}^{l} \frac{1}{j} = \ln l + \gamma + O\left(\frac{1}{l}\right) \qquad \text{as } l \to \infty \tag{2.28}$$

is the first order lth harmonic number. Finally, the asymptotic approximations

$$\Psi(x) = \ln(x-1) + O\left(\frac{1}{x}\right) \qquad \text{as } x \to \infty \tag{2.29}$$

and

$$\frac{d^k}{dx^k}\Psi(x) = O\left(\frac{1}{x^k}\right) \qquad \text{as } x \to \infty \tag{2.30}$$

will also be used in our derivations.

2.3.4 Oscillating Functions

We are now in a position to verify that the functions $f_{r,x}(l)$ and $g_{r,x}(l)$, defined in (2.3) and (2.4), are indeed oscillatory. We have

$$\begin{aligned} f_{r,x}(xl) &= \frac{2}{\ln x}\sum_{k\geq 1}\Re\left(\Gamma\left(r-i\frac{2k\pi}{\ln x}\right)e^{i\,2k\pi\log_x(xl)}\right) \\ &= \frac{2}{\ln x}\sum_{k\geq 1}\Re\left(\Gamma\left(r-i\frac{2k\pi}{\ln x}\right)e^{i\,2k\pi(1+\log_x l)}\right), \end{aligned}$$

and since $e^{i2k\pi}=1$ for all integers k, we get

$$f_{r,x}(xl) = f_{r,x}(l), \qquad \text{integer } r, \quad \text{reals } x,l>1. \tag{2.31}$$

Similarly, we can show that $g_{r,x}(xl)=g_{r,x}(l)$.

We can also see that these functions are bounded by a quantity *independent of* l. Consider, for example, the $f_{0,x}$ function. Since $|\Re(s)|\leq|s|$ for all complex numbers s, and $|e^{ix}|=1$ for all real x, we get

$$\left|\Re\left(\Gamma\left(-i\frac{2k\pi}{\ln x}\right)e^{i\,2k\pi\log_x l}\right)\right| \leq \sqrt{\frac{\ln x}{2}}\left(\frac{1}{k\sinh\left(\frac{2k\pi^2}{\ln x}\right)}\right)^{1/2}, \qquad \text{reals } x,l>1$$

by using property (2.19). It is now easy to see that the series

$$\sum_{k\geq 1}\left(\frac{1}{k\sinh\left(\frac{2k\pi^2}{\ln x}\right)}\right)^{1/2} \tag{2.32}$$

converges, since

$$\left(\frac{1}{k\sinh\left(\frac{2k\pi^2}{\ln x}\right)}\right)^{1/2} = \sqrt{\frac{2}{k}}\left(\frac{e^{\frac{2k\pi^2}{\ln x}}}{e^{\frac{4k\pi^2}{\ln x}}-1}\right)^{1/2} \sim \sqrt{\frac{2}{k}}\,\frac{1}{e^{\frac{k\pi^2}{\ln x}}} < \sqrt{2}\,k\left(\frac{1}{e^{\frac{\pi^2}{\ln x}}}\right)^k,$$

and the series of the last term converges because $e^{\frac{\pi^2}{\ln x}}>1$. Therefore, $f_{0,x}(l)$ is absolutely convergent, and thus

$$|f_{0,x}(l)| \leq \frac{2}{\ln x}\sum_{k\geq 1}\left|\Re\left(\Gamma\left(-i\frac{2k\pi}{\ln x}\right)e^{i\,2k\pi\log_x l}\right)\right|, \qquad \text{reals } x,l>1,$$

$1/x$	$F_{0,x}$	$\limsup_{l\to\infty}\{f_{0,x}(l)\}$	$\liminf_{l\to\infty}\{f_{0,x}(l)\}$
$1-10^{-4}$=.9999	$.1981\times10^{-42,858}$	$.1981\times10^{-42,858}$	$-.1981\times10^{-42,858}$
$1-10^{-3}$=.9990	$.4263\times10^{-4,282}$	$.4263\times10^{-4,282}$	$-.4263\times10^{-4,282}$
$1-10^{-2}$=.9900	$.6535\times10^{-425}$	$.6535\times10^{-425}$	$-.6535\times10^{-425}$
9/10=.9000	$.1280\times10^{-39}$	$.1280\times10^{-39}$	$-.1280\times10^{-39}$
4/5=.8000	$.2618\times10^{-18}$	$.2618\times10^{-18}$	$-.2618\times10^{-18}$
3/4=.7500	$.4700\times10^{-14}$	$.4700\times10^{-14}$	$-.4700\times10^{-14}$
7/10=.7000	$.3217\times10^{-11}$	$.3217\times10^{-11}$	$-.3217\times10^{-11}$
2/3=.6667	$.8428\times10^{-10}$	$.8428\times10^{-10}$	$-.8428\times10^{-10}$
$1-1/e$=.6321	$.1334\times10^{-8}$	$.1334\times10^{-8}$	$-.1334\times10^{-8}$
3/5=.6000	$.1137\times10^{-7}$	$.1137\times10^{-7}$	$-.1137\times10^{-7}$
1/2=.5000	$.1573\times10^{-5}$	$.1573\times10^{-5}$	$-.1573\times10^{-5}$
2/5=.4000	.00004387	.00004387	−.00004387
$1/e$=.3679	.0001035	.0001035	−.0001035
1/3=.3333	.0002394	.0002394	−.0002394
3/10=.3000	.0005020	.0005018	−.0005019
1/4=.2500	.001375	.001375	−.001374
1/5=.2000	.003429	.003429	−.003418
1/10=.1000	.01831	.01812	−.01815
10^{-2}=.0100	.1193	.1019	−.1188
10^{-3}=.0010	.2212	.1705	−.2107
10^{-4}=.0001	.3038	.2194	−.2715

Table 2.1: Range of $f_{0,x}(l)$ and its bound

which implies

$$|f_{0,x}(l)| \;\le\; F_{0,x} \;\stackrel{\text{def}}{=}\; \sqrt{\frac{2}{\ln x}}\sum_{k\ge1}\left(\frac{1}{k\sinh\left(\frac{2k\pi^2}{\ln x}\right)}\right)^{1/2}, \qquad \text{reals } x,l>1 \tag{2.33}$$

from the above. From Table 2.1, we may now observe that the derived bound of the range of $f_{0,x}(l)$ is tight for small values of x. The graph of the function, along with our bound of its range, is given in Figure 2.1.

Working as in the proof of (2.33), we can also show

$$|f_{1,x}(l)| \;\le\; F_{1,x} \;\stackrel{\text{def}}{=}\; \pi\left(\frac{2}{\ln x}\right)^{3/2}\sum_{k\ge1}\left(\frac{k}{\sinh\left(\frac{2k\pi^2}{\ln x}\right)}\right)^{1/2}, \qquad \text{reals } x,l>1$$

by using (2.21), and

$$|f_{-1,x}(l)| \;\le\; F_{-1,x} \;\stackrel{\text{def}}{=}\; \sqrt{2\ln x}\sum_{k\ge1}\left(\frac{1}{k(\ln^2 x+4k^2\pi^2)\sinh\left(\frac{2k\pi^2}{\ln x}\right)}\right)^{1/2}, \quad \text{reals } x,l>1 \tag{2.34}$$

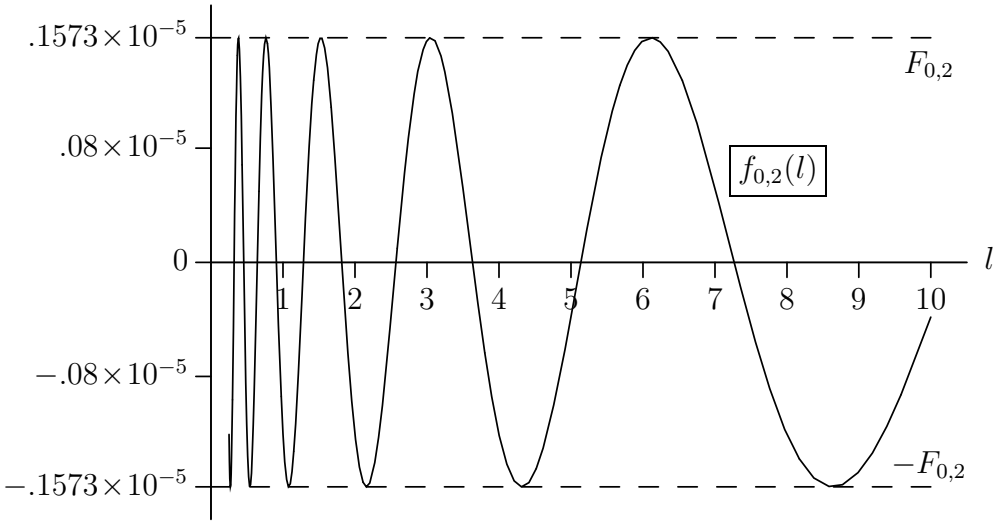


Figure 2.1: Oscillating function $f_{0,2}$ and its bounds

by using (2.22). In general, we can derive similar bounds for $f_{r,x}(l)$ and $g_{r,x}(l)$ for arbitrary r, since from (2.16) we have $\Gamma(r+it)=(r-1+it)^{\underline{r}}\Gamma(it)$ for all integers $r>0$, and $\Gamma(r+it)=\frac{\Gamma(it)}{(-1+it)^{\underline{-r}}}$ for all integers $r<0$.

The last remark, together with the periodicity of $f_{r,x}(l)$ and $g_{r,x}(l)$ from (2.31), justify the use of the term "periodic functions of $\log_x l$, with small magnitude" for expressions involving arbitrary constants and the functions $f_{r,x}(l)$ and $g_{r,x}(l)$. Contrary to what most authors do, we will always give explicit expressions for the "periodic functions" appearing in our results.

2.3.5 Exponential Generating Functions

The *exponential generating function* (egf) of a sequence $\langle a_k\rangle$ is

$$A(z) \stackrel{\text{def}}{=} \sum_{k\geq 0} a_k \frac{z^k}{k!}.$$

Clearly,

$$A'(z) = \sum_{k\geq 0} a_{k+1}\frac{z^k}{k!} \tag{2.35}$$

is the egf of $\langle a_{k+1}\rangle$, and

$$\int_0^z A(t)dt = \sum_{k\geq 1} a_{k-1}\frac{z^k}{k!} \tag{2.36}$$

is the egf of $\langle a_{k-1}\rangle$.

If $A(z)$ and $B(z)$ are the egf's of $\langle a_k\rangle$ and $\langle b_k\rangle$ respectively, then

$$A(z)B(z) = \sum_{k\geq 0}\left(\sum_{j=0}^{k}\binom{k}{j}a_j b_{k-j}\right)\frac{z^k}{k!} \tag{2.37}$$

is the egf for the *binomial convolution* of $\langle a_k\rangle$ and $\langle b_k\rangle$.

2.3.6 Poisson Transform

The *Poisson transform* (also called [16, 25] *Poisson generating function*) of a sequence $\langle a_k\rangle$ is a real-to-real function, denoted by $\mathcal{P}(a_k; z)$ and defined as

$$\mathcal{P}(a_k;\, z) = A(z) \stackrel{\text{def}}{=} e^{-z}\sum_{k\geq 0} a_k\frac{z^k}{k!}. \tag{2.38}$$

If $A(z)$ has a Taylor series expansion around $z=0$, then the original sequence can be recovered by using the inversion formula

$$a_k = \sum_{j=0}^{k}\binom{k}{j}A^{(j)}(0), \qquad \text{for } k = 0, 1, \ldots,$$

where $A^{(j)}(0)$ denotes the jth derivative of $A(z)$ at $z = 0$. This formula may be derived by multiplying both sides of (2.38) by e^z, and by subsequently extracting the coefficients of $\frac{z^n}{n!}$ in both sides. The approximation theorem (first used in [22], but proven in detail in [37])

$$a_k = A(k) + \sum_{s\geq 1}\frac{1}{k^s}\left(\sum_{r=s+1}^{2s}\{c_{r,s}\;\; A^{(r)}(k)\;\; k^r\}\right), \qquad \text{for } k = 0, 1, \ldots, \tag{2.39}$$

where $c_{r,s}$ are constants given by

$$c_{r,s} = \frac{1}{r!} \sum_{l=s}^{r} \left\{ (-1)^{r-l+s} \binom{r}{l} \begin{bmatrix} l \\ l-s \end{bmatrix} \right\} \qquad \text{for } r, s = 0, 1, \ldots,$$

with $\begin{bmatrix} k \\ j \end{bmatrix}$ being the Stirling number of the first kind, is useful when the double sum in (2.39) is an asymptotic expansion. Is such a case, this sum gives the error made when a_k is approximated by its Poisson transform $A(k)$.

We are now in a position to verify our earlier claim (end of Section 2.2) that the Poisson model of randomness may be viewed as a transformation of the fixed population model. The next result will be sufficient for our purposes.

Lemma 2.1 *Let X_n be a random variable under the fixed population model with n keys, and let $X(z)$ be its Poissonized version. Then, the Poisson transforms of all moments and factorial moments of X_n are equal to the corresponding moments of the Poissonized version of X_n, that is,*

$$\mathcal{P}\Big(\mathbf{E}(X_n^r);\, z\Big) = \mathbf{E}\Big(X^r(z)\Big), \qquad \textit{integer } r \geq 1$$

and

$$\mathcal{P}\Big(\mathbf{E}(X_n^{\underline{r}});\, z\Big) = \mathbf{E}\Big(X^{\underline{r}}(z)\Big), \quad \textit{integer } r \geq 1.$$

Proof: Suppose that X_n is discrete, and let N be the (discrete) random variable, denoting the number of keys under the Poisson model. We have

$$\begin{aligned}
\mathbf{E}\Big(X^r(z)\Big) &= \sum_{k\geq 0} k^r \, \mathbf{Pr}[X(z)=k] && \text{dfn. of expected value} \\
&= \sum_{k\geq 0} k^r \sum_{n\geq 0} \mathbf{Pr}[X(z)=k \mid N=n]\, \mathbf{Pr}[N=n] && \text{condition on \# of keys} \\
&= \sum_{k\geq 0} k^r \sum_{n\geq 0} \mathbf{Pr}[X_n=k]\, e^{-z}\frac{z^n}{n!} && \text{by (2.5)}
\end{aligned}$$

$$
\begin{aligned}
&= e^{-z} \sum_{n\geq 0} \left(\sum_{k\geq 0} k^r \, \mathbf{Pr}[X_n = k] \right) \frac{z^n}{n!} \\
&= e^{-z} \sum_{n\geq 0} \mathbf{E}(X_n^r) \, \frac{z^n}{n!} && \text{dfn. of expected value} \\
&= \mathcal{P}\Big(\mathbf{E}(X_n^r);\, z\Big) && \text{by (2.38).}
\end{aligned}
$$

The proof for continuous X_n, as well as the proof of the second indentity of the lemma are similar. ■

We will use this lemma in Chapters 5 and 6, in order to derive expected values and variances under the Poisson model of randomness. We will first obtain recurrences for (factorial) moments under the fixed population model, and we will subsequently consider the Poisson transforms of these recurrences. The solution of the latter recurrences will give us the corresponding moments under the Poisson model.

It is also worth noting at this point that $\mathcal{P}\Big(\mathbf{Var}(X_n); z\Big) \neq \mathbf{Var}\Big(X(z)\Big)$ in general, that is, the Poisson transform of the variance of a random variable is *not* equal to the variance of the Poissonized version of the same random variable. For example, for the random variable $X_n = n$, one can easily verify that $0 = \mathcal{P}\Big(\mathbf{Var}(X_n); z\Big) \neq \mathbf{Var}\Big(X(z)\Big) = z$.

2.3.7 Binomial Transform

The *binomial transform* of a sequence $\langle a_k \rangle$ (first appeared in [26, ex. 5.2.2.36]) is a new sequence, denoted by $\langle \widehat{a_k} \rangle$, and defined as

$$\widehat{a_k} \stackrel{\text{def}}{=} \sum_{j=0}^{k} \binom{k}{j} (-1)^j a_j, \qquad \text{for } k = 0, 1, \ldots. \tag{2.40}$$

It can be shown (see [26, ex. 5.2.2.36], or [23, eq. (5.48)]) that

$$\langle \widehat{\widehat{a_k}} \rangle = \langle a_k \rangle, \tag{2.41}$$

or, equivalently,

$$\sum_{j=0}^{k} \binom{k}{j} (-1)^j a_j = b_k \iff a_k = \sum_{j=0}^{k} \binom{k}{j} (-1)^j b_j, \qquad \text{for } k = 0, 1, \ldots. \tag{2.42}$$

The next result relates the Poisson transform of a sequence and the egf of its binomial transform.

Lemma 2.2 *Let $\langle \widehat{a_k} \rangle$ be the binomial transform of an arbitrary sequence $\langle a_k \rangle$, and let $\widehat{A}(z)$ be the egf of $\langle \widehat{a_k} \rangle$. Then*

$$\mathcal{P}(a_k;\, z) = \widehat{A}(-z).$$

Proof: We have

$$\begin{aligned}
\mathcal{P}(a_k;\, z) &= \left(\sum_{k \geq 0} (-1)^k \frac{z^k}{k!} \right) \left(\sum_{k \geq 0} a_k \frac{z^k}{k!} \right) && \text{by (2.38)} \\
&= \sum_{k \geq 0} \left(\sum_{j=0}^{k} \binom{k}{j} a_j (-1)^{k-j} \right) \frac{z^k}{k!} && \text{by (2.37)} \\
&= \sum_{k \geq 0} \left(\sum_{j=0}^{k} \binom{k}{j} (-1)^j a_j \right) \frac{(-z)^k}{k!} \\
&= \sum_{k \geq 0} \widehat{a_k} \frac{(-z)^k}{k!} && \text{by (2.40)} \\
&= \widehat{A}(-z), && \text{dfn. of egf}
\end{aligned}$$

and the proof is complete. ■

The above mathematical machinery will be used in Chapters 5 and 6 to derive exact solutions of recurrences, in the following manner: we will consider the Poisson transform of a recurrence, we will use the last lemma to derive the binomial transform, and we will finally apply (2.42) to obtain the desired solution.

2.3.8 Mellin Transform

The *Mellin transform* (introduced by H. Mellin [32] in 1902, used by probabilists [9, 34] extensively since then, and by computer scientists [16] since early 1980's) of a real-to-real function f with respect to a complex variable s, is a complex-to-complex function, denoted by $\mathcal{M}\Big(f(z);s\Big)$ (or $f^*(s)$ for short), and defined in a region of the complex plane as

$$\mathcal{M}\Big(f(z);\, s\Big) = f^*(s) \stackrel{\text{def}}{=} \int_0^\infty f(z) z^{s-1} dz. \tag{2.43}$$

For example, the definition (2.1) of the Gamma function implies that $\mathcal{M}(e^{-z};s) = \Gamma(s)$, for $\Re(s) > 0$. The related Cauchy-Saalschütz identities

$$\mathcal{M}\left(e^{-z} - \sum_{j=0}^{k}\left\{(-1)^j \frac{z^j}{j!}\right\};\;\; s\right) \;\; = \;\; \Gamma(s), \qquad \text{for } -(k+1) < \Re(s) < -k \tag{2.44}$$

give a representation of the Gamma function in vertical strips where it is not defined by (2.1). An immediate and very useful property of the Mellin transform is

$$\mathcal{M}\Big(f(az);\, s\Big) = a^{-s}\mathcal{M}\Big(f(z);\, s\Big), \qquad \text{real } a. \tag{2.45}$$

Given the Mellin transform f^* of a function f, the original function f can be recovered by using the inverse Mellin transform

$$f(z) = \frac{1}{2\pi i}\int_{c-i\infty}^{c+i\infty} f^*(s) z^{-s} ds, \tag{2.46}$$

where c is a real number, such that $f^*(s)$ exists on the vertical line $\Re(s) = c$. One may prove without much effort that if $f(z) = O(z^a)$ as $z \to 0$, and if $f(z) = O(z^b)$ as $z \to \infty$, then $f^*(s)$ exists for $-a < \Re(s) < -b$.

In practice, if we know the Mellin transform f^* of a function f and we want to recover f by computing the line integral (2.46), we consider the last integral along

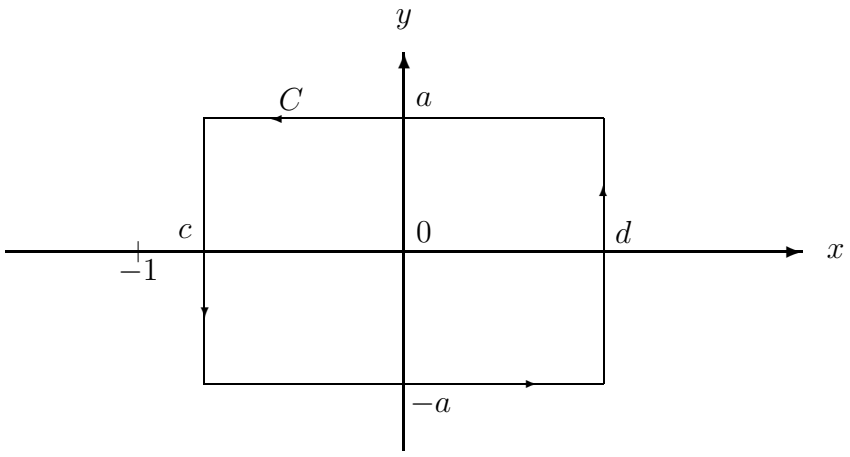


Figure 2.2: Line integration for inversion of Mellin's transform

a closed rectangle C, as shown in Figure 2.2. We then use the residue theorem from complex variable theory, which gives the integral of a function h (if h is single-valued and analytic inside and on C, except at its poles) along a closed curve as

$$\oint_C h(s)ds \;=\; 2\pi i \sum_{\alpha=\text{pole inside } C} \operatorname*{Res}_{s=\alpha}\{h(s)\}, \tag{2.47}$$

where $\operatorname*{Res}_{s=\alpha}\{h(s)\}$ is the residue of h at the pole $s=\alpha$. As we may recall from elementary complex variable theory, a function $h(s)$ has a pole of order k, for some $k>0$, at $s=\alpha$ (or a simple, double, triple, ... pole), if $h(s)=\frac{w(s)}{(s-\alpha)^k}$, where $w(a)\neq 0$ and $w(s)$ is analytic everywhere in a region including $s=\alpha$. For our purposes, we will use repeatedly the fact that $\Gamma(s)$ has simple poles at $s=0,-1,-2,\ldots$.

Having determined the poles of a function $h(s)$, the residues of $h(s)$ at each pole $s=\alpha$ of order k are given by

$$\operatorname*{Res}_{\substack{s=\alpha \\ \text{order}=k}} \{h(s)\} = \lim_{s\to\alpha}\left\{\frac{1}{(k-1)!}\;\frac{d^{k-1}}{ds^{k-1}}\left((s-\alpha)^k h(k)\right)\right\}. \tag{2.48}$$

In this thesis, we will write explicitly the order of the pole under "Res" only if it is *not* 1.

To make all the above more clear, we will now work out in detail the computation of a particular line integral. This integral will be the first one that we will encounter in this thesis (Chapter 5), and, at that point, we will quote the result derived here. Having showed such a computation once in detail, the presentation of the computations of subsequent line integrals will be sketchier.

Example 2.1 *Given the Mellin transform f^* of a function f as*

$$f^*(s) = \frac{1 + qp^{-s-1}}{1 - p^{-s}} \, \Gamma(s) \, , \qquad \textit{for } -1 < \Re(s) < 0,$$

for arbitrary $p = 1-q$ with $0 < p, q < 1$, find the original function f.

Solution: According to the inverse Mellin transform (2.46), f will be given by

$$f(z) = \frac{1}{2\pi i} \int_{c-i\infty}^{c+i\infty} h(s) ds, \qquad \text{for } -1 < c < 0, \tag{2.49}$$

where

$$h(s) \stackrel{\text{def}}{=} f^*(s) z^{-s}.$$

Since the integral of h along any rectangle equals the sum of the line integrals along each side of the rectangle, we get

$$\int_{c-i\infty}^{c+i\infty} h(s)ds = -\oint_C h(s)ds + \int_{c-i\infty}^{d-i\infty} h(s)ds + \int_{d-i\infty}^{d+i\infty} h(s)ds + \int_{d+i\infty}^{c+i\infty} h(s)ds, \text{ for any } d > 0 \tag{2.50}$$

if we let $a \to \infty$ in the rectangle of Figure 2.2. The integrals along the top and bottom lines are 0, that is,

$$\int_{c-i\infty}^{d-i\infty} h(s)ds = \int_{d+i\infty}^{c+i\infty} h(s)ds = 0,$$

since

$$\left| \int_{c \pm ia}^{d \pm ia} h(s)ds \right|$$

$$\begin{aligned}
&\leq \int_c^d \frac{|1+qp^{-t-1\mp ia}|}{|1-p^{-t\mp ia}|}\,|\Gamma(t\pm ia)|\,\left|z^{-t\mp ia}\right|\,dt && \text{by } s\leftarrow t\pm ia\\
&\leq \int_c^d \frac{1+qp^{-t-1}}{|1-p^{-t}|}\,|\Gamma(t\pm ia)|\,z^{-t}\,dt && \text{by } \Big||x|-|y|\Big|\leq|x\pm y|\leq|x|+|y|\\
&\leq O\left(\frac{1}{e^{\frac{\pi a}{2}}\sqrt{a}}\int_c^d \frac{p^t+\frac{q}{p}}{|p^t-1|}\left(\frac{a}{z}\right)^t dt\right) && \text{as } a\to\infty. \text{ by (2.24)}
\end{aligned}$$

We can also show that the integral along the right vertical line is superpolynomially small, or, more precisely,

$$\int_{d-i\infty}^{d+i\infty} h(s)ds = O\left(\frac{1}{z^d}\right) \quad \text{as } z\to\infty, \qquad \text{for any } d>0.$$

Indeed,

$$\begin{aligned}
&\left|\int_{d-i\infty}^{d+i\infty} h(s)ds\right|\\
&\leq \int_{-\infty}^{+\infty} \frac{\left|1+qp^{-d-1-it}\right|}{|1-p^{-d-it}|}\,|\Gamma(d+it)|\,\left|z^{-d-it}\right|\,dt && \text{by } s\leftarrow d+it\\
&\leq 2\,\frac{1+qp^{-d-1}}{|1-p^{-d}|}\,z^{-d}\int_0^{+\infty}|\Gamma(d+it)|\,dt && \text{by } \Big||x|-|y|\Big|\leq|x\pm y|\leq|x|+|y|\\
&= 2\,\frac{1+qp^{-d-1}}{|1-p^{-d}|}\,z^{-d}\,\sqrt{\pi}\int_0^{+\infty}\frac{(d-1+it)!}{(t\sinh(\pi t))^{1/2}}dt, && \text{by (2.16) and (2.21)}
\end{aligned}$$

and the improper integral appearing in the last expression exists and it depends only on d (since, by working as with the series (2.32), we can show that the series $\sum_{k\geq 0}\left(\frac{k^{2d-1}}{\sinh(\pi k)}\right)^{1/2}$ converges to a quantity depending on d). Therefore, (2.50) becomes

$$\int_{c-i\infty}^{c+i\infty} h(s)ds = -\oint_C h(s)ds + O\left(\frac{1}{z^d}\right) \quad \text{as } z\to\infty, \qquad \text{for any } d>0,$$

and application of the residue theorem (2.47) yields

$$\int_{c-i\infty}^{c+i\infty} h(s)ds = -2\pi i \sum_{\alpha=\text{pole inside } C} \operatorname*{Res}_{s=\alpha}\{h(s)\} + O\left(\frac{1}{z^d}\right) \quad \text{as } z\to\infty, \quad \text{for any } d>0,$$

or

$$f(z) = -\sum_{\alpha=\text{pole inside } C} \operatorname*{Res}_{s=\alpha}\{h(s)\} + O\left(\frac{1}{z^d}\right) \quad \text{as } z\to\infty, \quad \text{for any } d>0,$$

according to (2.49). We now determine the poles of $h(s)$ inside the infinitely tall rectangle C. The presence of $\Gamma(s)$ in the numerator of $h(s)$, introduces a simple pole at $s=0$, inside C. The presence of $1-p^{-s}$ in the denominator of $h(s)$, introduces additional simple poles at the points for which $1=p^{-s}$, or $e^{i2k\pi}=e^{-s\ln p}$, that is, at $s=s_k \stackrel{\text{def}}{=} i\frac{2k\pi}{\ln\frac{1}{p}}$ for $k=0,\pm1,\pm2,\ldots$. Hence, the last relation becomes

$$f(z) = -\operatorname*{Res}_{\substack{s=0\\ \text{order}=2}} \{h(s)\} - \sum_{k\in\mathbb{Z}-\{0\}} \operatorname*{Res}_{s=s_k}\{h(s)\} + O\left(\frac{1}{z^d}\right) \quad \text{as } z\to\infty, \quad \text{for any } d>0, \tag{2.51}$$

and the appearing residues can be computed from (2.48). Indeed, for the double pole at $s=0$, (2.48) gives

$$\operatorname*{Res}_{\substack{s=0\\ \text{order}=2}} \{h(s)\} = \lim_{s\to0}\left\{\frac{d}{ds}\left(\frac{1+qp^{-s-1}}{1-p^{-s}}\,\Gamma(s)\,z^{-s}s^2\right)\right\},$$

and after some algebra, we arrive at

$$\operatorname*{Res}_{\substack{s=0\\ \text{order}=2}} \{h(s)\} = \frac{1}{p}\log_{\frac{1}{p}} z - \frac{1}{p}\left(\frac{\gamma}{\ln p}+\frac{1}{2}\right)+1 \tag{2.52}$$

by using (2.16), (2.17) and (2.26). For the simple poles at $s=s_k$, we have

$$\begin{aligned}
\operatorname*{Res}_{s=s_k}\{h(s)\} &= \lim_{s\to s_k}\left\{\left(1+qp^{-s-1}\right)\Gamma(s)z^{-s}\frac{s-s_k}{1-p^{-s}}\right\} && \text{by (2.48)}\\
&= \left(1+qp^{-s_k-1}\right)\Gamma(s_k)z^{-s_k}\frac{1}{\ln p} && \text{by L'Hôpital's rule}\\
&= -\left(1+\frac{q}{p}\right)\Gamma\left(i\,\frac{2k\pi}{\ln\frac{1}{p}}\right) z^{-i\frac{2k\pi}{\ln\frac{1}{p}}}\frac{1}{\ln\frac{1}{p}} && \text{by } p^{-s_k}=1\\
&= -\frac{1}{p}\,\frac{1}{\ln\frac{1}{p}}\,\Gamma\left(i\,\frac{2k\pi}{\ln\frac{1}{p}}\right) e^{-i\,2k\pi\log_{\frac{1}{p}} z}, && \text{by } p+q=1 \text{ and } z^y=e^{y\ln z}
\end{aligned}$$

and therefore the sum of all residues at $s=s_k$ becomes

$$\sum_{k\in\mathbb{Z}-\{0\}} \operatorname*{Res}_{s=s_k}\{h(s)\} = -\frac{1}{p\ln\frac{1}{p}} \sum_{k\geq 1}\left\{\Gamma\left(i\,\frac{2k\pi}{\ln\frac{1}{p}}\right)e^{-i\,2k\pi\log_{\frac{1}{p}} z} + \Gamma\left(-i\,\frac{2k\pi}{\ln\frac{1}{p}}\right)e^{i\,2k\pi\log_{\frac{1}{p}} z}\right\}.$$

We may now use property (2.20) of the Gamma function, and if we recall that $e^{-ix}=\overline{e^{ix}}$ for any real x, and $s+\overline{s}=2\Re(s)$ for any complex s, we can simplify the last sum as

$$\sum_{k\in\mathbb{Z}-\{0\}} \operatorname*{Res}_{s=s_k}\{h(s)\} = -\frac{2}{p\ln\frac{1}{p}} \sum_{k\geq 1}\left\{\Re\left(\Gamma\left(-i\frac{2k\pi}{\ln\frac{1}{p}}\right)e^{i\,2k\pi\log_{\frac{1}{p}} z}\right)\right\},$$

or as

$$\sum_{k\in\mathbb{Z}-\{0\}} \operatorname*{Res}_{s=s_k}\{h(s)\} = -\frac{1}{p}\,f_{0,\frac{1}{p}}(z), \tag{2.53}$$

by the definition (2.3) of $f_{r,x}$. Therefore, we finally get

$$f(z) = -\frac{1}{p}\log_{\frac{1}{p}} z + \frac{1}{p}\left(\frac{\gamma}{\ln p}+\frac{1}{2}\right) - 1 + \frac{1}{p}f_{0,\frac{1}{p}}(z) + O\left(\frac{1}{z^d}\right) \quad \text{as } z\to\infty, \quad \text{for any } d>0$$

from (2.51), (2.52) and (2.53). ■

2.3.9 Alternating Sums

Alternating sums of the form $\sum_{k=r}^{m}\binom{m}{k}(-1)^k h(k)$, for arbitrary $h(k)$, will appear repeatedly in the analyses of our algorithms. Such sums not only give us no information on their asymptotic behaviour, but they are also expensive to compute numerically, due to the alternating signs of their terms.

In Chapter 3, we will be able to get away with the derivation of asymptotic expressions for such sums; we will express our formulae in terms of alternating sums that have been studied before in detail. However, in Chapters 5 and 6, we will be faced with new alternating sums. For the asymptotic evaluation of these sums, we will use a method introduced by Knuth in [26, Exercise 5.2.2.53], and attributed

to S. O. Rice. (A similar, more "Mellin-like" approach, has been proposed by Szpankowski in [48].) Rice's method is based on the following well-known result (for a proof, see, for example, [30, pp. 275–276]).

Lemma 2.3 *If C is a closed curve encompassing the points $r, r+1, \ldots, m$ and no other integer points, and $h(s)$ is a function analytic within C, then*

$$\sum_{k=r}^{m} \binom{m}{k} (-1)^k h(k) = -\frac{1}{2\pi i} \oint_C \mathrm{B}(m+1, -s)\, h(s)\, ds,$$

where

$$\mathrm{B}(x, y) \stackrel{\text{def}}{=} \frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)} \tag{2.54}$$

is the classical Beta function.

To evaluate an alternating sum, we change the rectangle C into an arbitrarily large rectangle C' surrounding C, around which the integral of $\mathrm{B}(m+1, -s)h(s)$ is very small. This allows us to approximate the original integral along C by the sum of the residues of $\mathrm{B}(m+1, -s)h(s)$ that lie in the area between C and C'.

To make the above more clear, we will now derive an asymptotic expression for an alternating sum. This sum will be encountered in Chapter 6, and, at that point, we will quote the result derived here. The alternating sums that we will encounter in Chapter 5 will be more complicated, and, for the reader's benefit, we treat them later, when they occur.

Example 2.2 *Derive an asymptotic expression (as $n \to \infty$) for the alternating sum*

$$S_n = \sum_{k=2}^{n} \binom{n}{k} (-1)^k \frac{1}{1-2^{1-k}} \sum_{r=2}^{b} \binom{k}{r} (-1)^r r! \sigma_r \,,$$

for arbitrary integer $b \geq 1$, and arbitrary reals $\sigma_2, \sigma_3, \ldots, \sigma_b$.

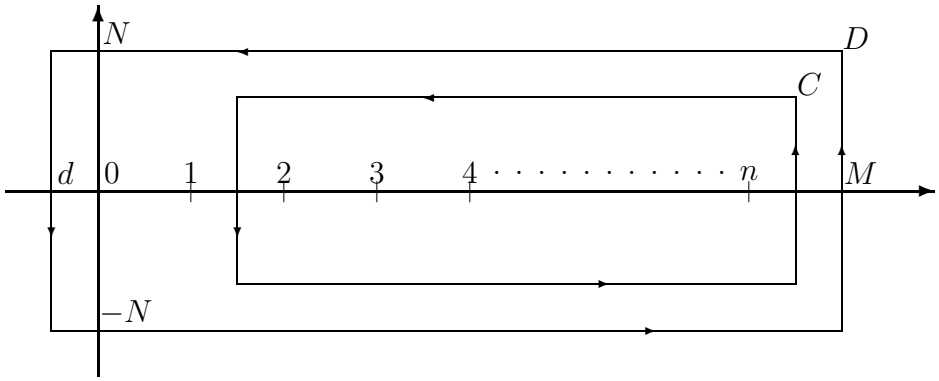


Figure 2.3: Rice's method for computation of alternating sums

Solution: According to Lemma 2.3, the given sum is

$$S_n = -\frac{1}{2\pi i}\oint_C \mathrm{B}(n+1,-s)\,h(s)\,ds, \tag{2.55}$$

where

$$h(s) \stackrel{\text{def}}{=} \frac{1}{1-2^{1-s}}\sum_{k=2}^{b}\binom{s}{k}(-1)^k k!\,\sigma_k\,, \tag{2.56}$$

and C is a rectangle encompassing only the integer points $2,3,\ldots,n$, as shown in Figure 2.3. Consider now a larger rectangle D, with corners the four points $d\pm iN$ and $M\pm iN$, where $d<1$ and $M>n$. The residue theorem (2.47) gives

$$\oint_D \mathrm{B}(n+1,-s)h(s)ds - \oint_C \mathrm{B}(n+1,-s)h(s)ds = 2\pi i \sum_{\substack{\alpha=\text{pole inside } D\\ \text{but outside } C}} \operatorname*{Res}_{s=\alpha}\Big\{\mathrm{B}(n+1,-s)h(s)\Big\}. \tag{2.57}$$

By using the asymptotic approximation (2.23) for the ratio of two Gamma functions, one may verify that the integral of $\mathrm{B}(n+1,-s)h(s)$ is 0 along the top and bottom lines of D as $N\to\infty$, and also 0 along the right line of D as $M\to\infty$. Along the left line of D, this integral is $O(n^d)$. Therefore, (2.55) and (2.57) imply

$$S_n = \sum_{\substack{\alpha=\text{pole inside } C'\\ \text{but outside } C}} \operatorname*{Res}_{s=\alpha}\Big\{\mathrm{B}(n+1,-s)h(s)\Big\} + O(n^d) \quad \text{as } n\to\infty, \text{ for any } d<1,$$

where C' is the rectangle resulting from D when we let $N \to \infty$ and $M \to \infty$. We now determine the poles of $\mathrm{B}(n+1,-s)h(s)$ in the area between C and C'. The term $h(s)$ from (2.56) may be rewritten as

$$h(s) = \frac{s(s-1)}{1-2^{1-s}} \sum_{k=2}^{b} \binom{s-2}{k-2} (-1)^k (k-2)!\, \sigma_k, \tag{2.58}$$

since $\binom{s}{k} = \frac{s}{k}\binom{s-1}{k-1}$ for non-zero k. The presence of $1-2^{1-s}$ in the denominator, introduces simple poles at $s = s_m \stackrel{\text{def}}{=} 1 - i\,\frac{2m\pi}{\ln 2}$ for $m = 0, \pm 1, \pm 2, \ldots$, all inside C' but outside C, of which the pole $s = s_0 = 1$ is removed by the term $s-1$ in the numerator. By the definition (2.54) of the Beta function, the term $\mathrm{B}(n+1,-s)$ has simple poles at $s = 0, 1, 2, \ldots, n$, of which only $s = 0$ and $s = 1$ are inside C' but outside C, and of which the pole $s = 0$ is removed by the term s in the numerator of $h(s)$. Thus, finally, inside C' but outside C, the expression $\mathrm{B}(n+1,-s)h(s)$ has simple poles at $s = 1$ and at $s = s_m$ for $m = \pm 1, \pm 2, \ldots$, and therefore

$$\begin{aligned} S_n &= \operatorname*{Res}_{s=1}\Big\{\mathrm{B}(n+1,-s)h(s)\Big\} + \sum_{m \in \mathbb{Z}-\{0\}} \operatorname*{Res}_{s=s_m}\Big\{\mathrm{B}(n+1,-s)h(s)\Big\} \\ &\quad + O(n^d) \quad \text{as } n \to \infty, \quad \text{for any } d < 1. \end{aligned} \tag{2.59}$$

The main contribution to S_n comes from the pole at $s = 1$. If we use the expression (2.58) for $h(s)$, (2.48) gives

$$\begin{aligned} &\operatorname*{Res}_{s=1}\Big\{\mathrm{B}(n+1,-s)\, h(s)\Big\} \\ &\quad = \lim_{s \to 1} \left\{ \frac{s-1}{1-2^{1-s}} \, \frac{\Gamma(n+1)\Gamma(-s)}{\Gamma(n+1-s)} \, s\,(s-1) \sum_{k=2}^{b} \binom{s-2}{k-2} (-1)^k (k-2)!\, \sigma_k \right\}. \end{aligned}$$

But $\lim_{s\to 1}\left\{\frac{s-1}{1-2^{1-s}}\right\} = \frac{1}{\ln 2}$ by L'Hôpital's rule, $\frac{\Gamma(n+1)}{\Gamma(n)} = n$ by the factorial representation (2.18) of the Gamma function, $\lim_{s\to 1}\{\Gamma(-s)s(s-1)\} = 1$ by properties (2.16) and (2.17), and $\binom{-1}{k-2} = (-1)^{k-2}$, as discussed after the definition (2.12) of the binomial coefficients. Therefore,

$$\operatorname*{Res}_{s=1}\Big\{\mathrm{B}(n+1,-s)\, h(s)\Big\} = \frac{n}{\ln 2} \sum_{k=2}^{b} (k-2)!\, \sigma_k. \tag{2.60}$$

The poles at $s=s_m$ contribute only oscillations (times n), since at each pole

$$\begin{aligned}
&\operatorname*{Res}_{s=s_m}\Big\{\mathrm{B}(n+1,-s)\,h(s)\Big\}\\
&\quad= \lim_{s\to s_m}\left\{\frac{s-s_m}{1-2^{1-s}}\,\frac{\Gamma(n+1)}{\Gamma(n+1-s)}\,\Gamma(-s)\sum_{k=2}^{b}\binom{s}{k}(-1)^k k!\,\sigma_k\right\} \quad \text{by (2.48) \& (2.56)}\\
&\quad= \frac{1}{\ln 2}\,\frac{\Gamma(n+1)}{\Gamma(n+1-s_m)}\,\Gamma(-s_m)\sum_{k=2}^{b}(-s_m)^{\underline{k}}\,\sigma_k \qquad \text{by L'Hôpital's rule}\\
&\quad= \frac{1}{\ln 2}\,n^{1-i\,\frac{2m\pi}{\ln 2}}\left(1+O\left(\frac{1}{n}\right)\right)\sum_{k=2}^{b}\{\Gamma(-s_m+k)\,\sigma_k\} \qquad \text{by (2.23) \& (2.16)}\\
&\quad= \frac{n}{\ln 2}\left(1+O\left(\frac{1}{n}\right)\right)e^{-i\,2m\pi\log_2 n}\sum_{k=2}^{b}\left\{\sigma_k\,\Gamma\left(k-1+i\,\frac{2m\pi}{\ln 2}\right)\right\},
\end{aligned}$$

and hence, the sum of all residues at $s=s_m$ is

$$\sum_{m\in\mathbb{Z}-\{0\}}\operatorname*{Res}_{s=s_m}\Big\{\mathrm{B}(n+1,-s)\,h(s)\Big\} = n\sum_{k=2}^{b}\Big\{\sigma_k\,f_{k-1,\,2}(n)\Big\}+O(1) \quad \text{as } n\to\infty, \tag{2.61}$$

by working as in the derivation of (2.53). Therefore, we finally get

$$S_n = \frac{n}{\ln 2}\sum_{k=2}^{b}\sigma_k(k-2)!+n\sum_{k=2}^{b}\sigma_k f_{k-1,\,2}(n)+O(1) \qquad \text{as } n\to\infty \tag{2.62}$$

from (2.59), (2.60) and (2.61). ■

Chapter 3

Probabilistic Skip Lists

The Probabilistic Skip List (PSL) was recently introduced by W. Pugh [40] as an alternative to search trees. Although Pugh established its good average behaviour by means of an *upper bound*, the precise analysis of its search cost had been elusive. In this chapter, we derive the *exact* average cost for the search of the mth key in a PSL of n keys, a result which confirms that Pugh's upper bound is fairly tight for the interesting cases. Assuming a uniform query distribution, we then derive the average cost for the search of the average key in a PSL of n keys. Finally, we also derive all insert and delete costs. Earlier versions of these results have appeared in [35, 36].

3.1 Review of the Structure

The main idea in the PSL is that each of its keys is stored in one or more of a set of sorted linear linked lists. All keys are stored in sorted order in a linked list denoted as *level* 1, and each key in the linked list at level k ($k = 1, 2, \ldots$) is included with

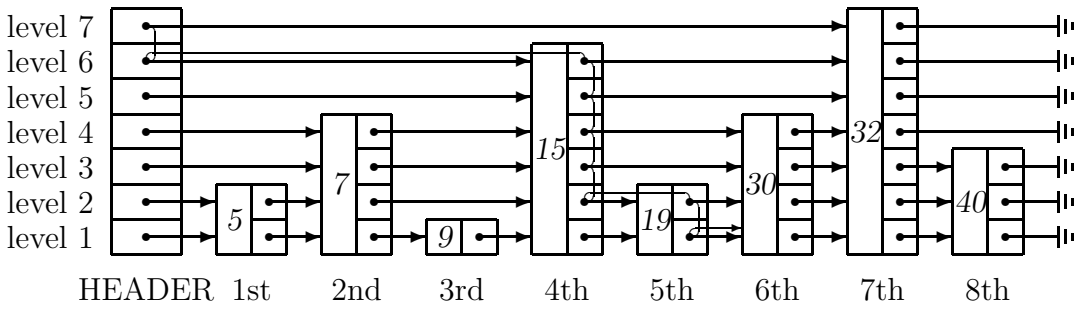


Figure 3.1: A PSL of 8 keys, and the search path for the 6th key

(independent) probability p $(0 < p < 1)$ in the linked list at level $k + 1$. A *header* contains the references to the first key in each linked list. The *height* of the data structure, that is, the number of linked lists, is also stored.

As will become apparent, the number of linear linked lists in which a key belongs, remains fixed as long as the key stays in the PSL. It therefore makes sense to store each key in the PSL once, together with an array of horizontal pointers, as shown in Figure 3.1. Each such "vertical" structure in a PSL will be called an *element* of the PSL.

A search for a key begins at the header of the highest numbered linked list. This linked list is scanned, until it is observed that its next key is *greater than or equal to* the one sought (or the reference is null). At that point, the search continues one level below until it terminates at level 1 (see the search for the 6th key in Figure 3.1). We have adopted the convention that an equality test is done only at level 1 as the last comparison. This is the usual choice in a standard binary search and avoids two tests (or a three-way branch) at each step.

Insertions and deletions (as defined by Pugh) are very straightforward. A new key is inserted where a search for it terminated at level 1. As it is put in linked list k

$(k = 1, 2, \ldots)$, it is inserted, with probability p, where its search terminated at level $k + 1$. This continues until, with probability $q = 1 - p$, the choice is not to insert. The counter for the height of the data structure is increased, if necessary. Deletions are completely analogous to insertions. A key to be deleted is removed from the linked lists in which it is found. The height of the data structure is updated by scanning the header's pointers by decreasing level until a non-null pointer is found.

At this point we can make an observation on the adopted "scan the list until the next key is *greater than or equal to* the one sought" criterion to drop down a level during the search. Had it been simply "... *greater than* the one sought ...", then the search path for the mth key in this case would have been identical to the search path for the $(m + 1)$st key in our case. This implies that the average search cost in this case would have been higher. Moreover, our approach makes the deletion a simple extension of the search.

An immediate property of the PSL is that the deletion of a key produces the structure that would have existed had it never been inserted. This implies that PSLs maintain the (average logarithmic) search and update cost, even after a long sequence of updates. This is in sharp contrast with the results such as those of Culberson and Munro [10, 11], indicating that the usual update algorithms in binary search trees lead to degeneration in behaviour, although the notions of Aragon and Seidel [3] show how to maintain the "history independent" property in (nearly) standard binary search trees.

Before we proceed with the formal definitions of the search costs, it should be "fairly obvious" at this early stage that the PSL should exhibit logarithmic behaviour. Consider the search cost for $+\infty$. Clearly, all elements will be at level 1, about pn elements will make it to level 2, about p^2n elements will make it to

level 3, etc. Therefore, the height of the PSL is expected to be about $\log_{\frac{1}{p}} n$. Since, among all elements that made it to a certain level, about every $\frac{1}{p}$th element will make it to the next higher level, one should expect to make $\frac{1}{p}$ key comparisons per level. Therefore, one should expect about $\frac{1}{p}\log_{\frac{1}{p}} n$ key comparisons in total, when searching for $+\infty$. As it will turn out (Theorem 3.3), this is exactly the leading term in the search cost for $+\infty$ in a PSL of n keys.

3.2 Notation and Previous Results

The *search cost*, for a particular key in a particular PSL, is the number of pointer inspections, excluding the last one for the equality test. Under the fixed population model, we define

$$\begin{aligned} C_n^{(m)} &\stackrel{\text{def}}{=} \text{cost for successful search of } m\text{th key in a PSL of } n \text{ keys}, \quad \text{for } 1 \le m \le n \\ &\stackrel{\text{def}}{=} \text{cost for unsuccessful search of a key between the } (m-1)\text{st and the } m\text{th} \\ &\quad \text{in a PSL of } n \text{ keys}, \qquad \text{for } 1 \le m \le n+1, \end{aligned}$$

assuming of course that the 0th key is $-\infty$ and the $(n+1)$st key is $+\infty$. For example, for the PSL of Figure 3.1 we have $C_8^{(1)} = 7$, $C_8^{(6)} = C_8^{(7)} = C_8^{(9)} = 9$ and $C_8^{(8)} = 8$. Clearly, $C_n^{(m)}$ is a random variable; for fixed n and m, its value depends on the outcomes of the random number generator used to create the PSL, or equivalently, on the shape of the PSL. We emphasize here again that $C_n^{(m)}$ is independent of the insertion sequence.

Two particular values of m yield random variables of special interest, and we have reserved special names for them. When $m = 1$,

$$T_n \stackrel{\text{def}}{=} C_n^{(1)} = \text{height of a PSL with } n \text{ keys}$$

and when $m = n + 1$,

$$F_n \stackrel{\text{def}}{=} C_n^{(n+1)} = \text{cost for (unsuccessful) search of } +\infty \text{ in a PSL of } n \text{ keys .}$$

Although it is *not* true that in every PSL the search for $+\infty$ is the most expensive one (consider, for example, a PSL with the last element of height 2 and all other elements of height 1), it *is* true that *on the average* the search for $+\infty$ is the most expensive. Hence the significance of F_n: its expected value, in addition of being mathematically more tractable, is also an upper bound of the average search cost for *any key* in the structure.

Two more random variables are of interest. The cost for a successful search for the "average key"

$$S_n \stackrel{\text{def}}{=} \frac{1}{n} \sum_{m=1}^{n} C_n^{(m)}$$

and the cost for an unsuccessful search for the "average key"

$$U_n \stackrel{\text{def}}{=} \frac{1}{n+1} \sum_{m=1}^{n+1} C_n^{(m)}.$$

For example, for the PSL of Figure 3.1, these two random variables have the values $S_8 = \frac{7+8+8+9+8+9+9+8}{8} = 8.25$ and $U_8 = \frac{7+8+8+9+8+9+9+8+9}{9} \simeq 8.33$. Unlike this particular PSL, it's not hard to see that there exist PSLs for which $U_n < S_n$. Consider, for example, a PSL with six elements, the first five of which are of height 1 and the last one of height 2. However, *on an average PSL*, unsuccessful searches for the average key must be more expensive than successful searches for the average key, that is, $\mathbf{E}(U_n) > \mathbf{E}(S_n)$ must hold.

As discussed near the end of the previous section, a PSL is expected to have about $\log_{\frac{1}{p}} n$ levels. Difficulties related to the difference between this expected height and the actual height of the structure, caused Pugh to propose [40, initial version] cutting the structure off at level $\lceil \log_{\frac{1}{p}} n \rceil$. This enabled him to give an

upper bound of $\frac{1}{p}\log_{\frac{1}{p}} n+1$ for the average cost for the search of $+\infty$ *in a PSL cut off* at level $\lceil \log_{\frac{1}{p}} n \rceil$. He was later [40, 39] able to prove that the expected height of an unboundly high PSL (like the one we are considering) is at most $\log_{\frac{1}{p}} n+\frac{1}{q}$, a fact which allowed him to establish [40] the bound

$$\mathbf{E}(F_n) \leq \frac{1}{p}\log_{\frac{1}{p}} n + \frac{1}{q} + 1 \tag{3.1}$$

for the average cost for the search of $+\infty$. Finally, in [39] he proved the bound[1]

$$\mathbf{E}(C_n^{(m)}) \;\leq\; \begin{cases} \log_{\frac{1}{p}} n + \dfrac{q}{p}\log_{\frac{1}{p}}(m-1) + \dfrac{1}{q} + 1 & \text{for } m = 2, 3, \ldots, \text{n+1} \\ \log_{\frac{1}{p}} n + \dfrac{1}{q} & \text{for } m = 1 \end{cases} . \tag{3.2}$$

for the average cost for the search of the mth key.

In this chapter, we consider the exact population model (see Section 2.2), and we derive exact and asymptotic expressions for $\mathbf{E}(C_n^{(m)})$ (and therefore, for $\mathbf{E}(F_n)$ as well), for $\mathbf{E}(S_n)$ and for $\mathbf{E}(U_n)$. Second moments of the above random variables will be considered in Chapter 5. Notice here that the random variables $C_n^{(1)}, C_n^{(2)}, \ldots, C_n^{(n+1)}$ are *not* independent, since if, for example, $C_n^{(1)} = 1$, then we must have $C_n^{(j)} = j$ for all $2 \leq j \leq n+1$. However, in this chapter we are concerned only with expected values, and therefore, the dependencies between $C_n^{(1)}, C_n^{(2)}, \ldots, C_n^{(n+1)}$ will not complicate our computation of $\mathbf{E}(S_n)$ and $\mathbf{E}(U_n)$.

[1] Pugh's results (as well as Devroye's [12, 13] — to be stated in Chapter 5) are derived for the length of the search path from the top of the header to the bottom of the mth element, which, according to our definitions, is the search cost for the $(m+1)$st element. Our references to their results have been translated into our terms.

3.3 Search Costs: Exact Values

In order to compute $\mathbf{E}(C_n^{(m)})$, we divide the steps on the search path into vertical steps (that is, one less than T_n) and horizontal steps. To this end, define the random variable

$L_{m-1} \stackrel{\text{def}}{=}$ number of full horizontal steps on the search path for the $(m-1)$st key,

which obviously takes non-negative integer values. We use the term *full* horizontal steps to emphasize actually moving to the next element in the list, rather than simply inspecting it and dropping down a level. For example, for the search in the PSL of Figure 3.1 we have $L_5 = 2$, while for any PSL we have $L_0 = 0$ and $L_1 = 1$. Clearly, $C_n^{(m)} = T_n + L_{m-1}$, and thus

$$\mathbf{E}(C_n^{(m)}) = \mathbf{E}(T_n) + \mathbf{E}(L_{m-1}). \tag{3.3}$$

In the next two lemmata we give exact expressions for $\mathbf{E}(T_n)$ and $\mathbf{E}(L_{m-1})$.

Lemma 3.1 *Under the fixed population model, the expected height of a PSL of n keys is*

$$\mathbf{E}(T_n) \;=\; \sum_{k\geq 1} \left\{ k \left((1-p^k)^n - (1-p^{k-1})^n \right) \right\} \;=\; \sum_{k\geq 1} \left\{ 1 - \left(1 - p^{k-1} \right)^n \right\}.$$

Proof: Follows from

$$\mathbf{E}(T_n) = \sum_{k\geq 1} k\, \mathbf{Pr}[T_n = k] = \sum_{k\geq 1} \mathbf{Pr}[T_n \geq k],$$

if we recall that the heights of the elements in a PSL are i.i.d. random variables following the geometric distribution. ■

Lemma 3.2 *Under the fixed population model, the expected number of full horizontal steps on the search path for the $(m-1)$st key is*

$$\mathbf{E}(L_{m-1}) = \begin{cases} 1 + \frac{q}{p} \sum_{j=1}^{m-2} \sum_{k\geq 1} p^k (1-p^k)^j & \text{if } m = 2, 3, \ldots, n+1 \\ 0 & \text{if } m = 1 \end{cases}$$

Proof: Consider the $m-2$ indicator random variables

$$I_j = \begin{cases} 1 & \text{if } j\text{th element is on path from header to } (m-1)\text{st key} \\ 0 & \text{otherwise} \end{cases} \qquad j = 1, \ldots, m-2.$$

(For example, for the search path of Figure 3.1, we have $I_1 = I_2 = I_3 = 0$ and $I_4 = 1$.) Clearly,

$$L_{m-1} = \begin{cases} 1 + \sum_{j=1}^{m-2} I_j & \text{if } m = 2, 3, \ldots, n+1 \\ 0 & \text{if } m = 1 \end{cases} . \tag{3.4}$$

If E_j is a random variable denoting the height of the jth element for $j = 1, 2, \ldots m$–2,

$$\mathbf{Pr}[I_j = 1] = \mathbf{Pr}[E_j \geq E_{j+1}, \ldots, E_{m-1}],$$

or, since $E_j, E_{j+1}, \ldots, E_{m-1}$ are independent random variables,

$$\begin{aligned} \mathbf{Pr}[I_j = 1] &= \sum_{k\geq 1} \{\mathbf{Pr}[E_j = k]\, \mathbf{Pr}[E_{j+1} \leq k] \cdots \mathbf{Pr}[E_{m-1} \leq k]\} \\ &= q \sum_{k\geq 1} \left\{ p^{k-1} (1-p^k)^{m-j-1} \right\}, \qquad \text{for } j = 1, 2, \ldots, m-2. \end{aligned}$$

The last, together with (3.4) and the fact that $\mathbf{E}(I_j) = \mathbf{Pr}[I_j = 1]$, completes the proof. ■

We may now observe that our random variable T_n is identical to the random variable D_{n+1}, denoting the depth of a key in a $\frac{1}{p}$-ary trie (i.e. a trie with keys over an alphabet of size $\frac{1}{p}$) with $n+1$ keys drawn from the uniform [0,1] distribution.[2]

[2]A binary trie with 7 keys in shown in Figure 6.1. For this trie, D_8 takes the values 1, 1, 2, 2, 3, 4, and 4.

Indeed, both T_n and D_{n+1} take the values $1, 2, 3, \ldots$ for $n \geq 1$, and moreover

$$\begin{aligned}
\mathbf{Pr}[D_{n+1} \geq k] &= 1 - \mathbf{Pr}[D_{n+1} \leq k-1] \\
&= 1 - \mathbf{Pr}[\text{key matches no other key's } (k-1) \text{ char prefix}] \\
&= 1 - (1 - \mathbf{Pr}[\text{the } (k-1) \text{ char prefixes of two keys match}])^n \\
&= 1 - (1 - p^{k-1})^n \\
&= 1 - \mathbf{Pr}[T_n < k] \\
&= \mathbf{Pr}[T_n \geq k].
\end{aligned}$$

For arbitrary p in (0,1), T_n is equal to the depth of the leftmost key in a biased (with probability p of having a "0" in a key) binary trie with $n+1$ keys.

The above observations lead us to try to express $\mathbf{E}(T_n)$ of Lemma 3.1 in terms of

$$W_p(l) \stackrel{\text{def}}{=} \sum_{k=2}^{l} \binom{l}{k} (-1)^k \frac{p^{k-1}}{1 - p^{k-1}}, \qquad \text{for } p \in (0,1), \quad \text{integer } l \geq 0 \tag{3.5}$$

and/or

$$V_p(l) \stackrel{\text{def}}{=} \frac{1}{l} \sum_{k=2}^{l} \binom{l}{k} (-1)^k \frac{k p^{k-1}}{1 - p^{k-1}}, \qquad \text{for } p \in (0,1), \quad \text{integer } l \geq 0 \tag{3.6}$$

(where, by convention, $W_p(0) = W_p(1) = V_p(0) = V_p(1) = 0$), which are slight modifications and/or extensions of functions first reported by Knuth [26, Ex. 5.2.2.38 and Ex. 5.2.2.50] in the context of analyses of generalizations of exchange sorts and of trie searching. The W_p and V_p look similar, and they are indeed related as

$$W_p(l) - W_p(l-1) = V_p(l), \tag{3.7}$$

since

$$W_p(l) - W_p(l-1) = \sum_{k=2}^{l} \binom{l}{k} (-1)^k \frac{p^{k-1}}{1 - p^{k-1}} - \sum_{k=2}^{l} \binom{l-1}{k} (-1)^k \frac{p^{k-1}}{1 - p^{k-1}}$$

$$\begin{aligned} &= \sum_{k=2}^{l} \binom{l-1}{k-1} (-1)^k \frac{p^{k-1}}{1-p^{k-1}} \\ &= \sum_{k=2}^{l} \frac{k}{l} \binom{l}{k} (-1)^k \frac{p^{k-1}}{1-p^{k-1}} \\ &= V_p(l) \qquad \text{for } l = 1, 2, \ldots. \end{aligned}$$

We now express $\mathbf{E}(T_n)$ in terms of V_p.

Lemma 3.3 *Under the fixed population model, the expected height of a PSL of n keys is*

$$\mathbf{E}(T_n) = 1 + V_p(n+1),$$

where $V_p(n+1)$ is the sum defined in (3.6).

Proof: We have:

$\mathbf{E}(T_n)$

$$\begin{aligned} &= \sum_{k\geq 1} \left\{ k(1-p^k)^n - (1-p^{k-1})^n \right\} && \text{by Lemma 3.1} \\ &= \sum_{k\geq 1} \left\{ k \left(\sum_{j=0}^{n} \left\{ \binom{n}{j} (-1)^j p^{kj} \right\} - \sum_{j=0}^{n} \left\{ \binom{n}{j} (-1)^j p^{(k-1)j} \right\} \right) \right\} && \text{by binomial theorem} \\ &= \sum_{j=1}^{n} \left\{ \binom{n}{j} (-1)^j \sum_{k\geq 1} k (p^j)^k \right\} - \sum_{j=1}^{n} \left\{ \binom{n}{j} (-1)^j \sum_{k\geq 1} k (p^j)^{k-1} \right\} && \text{“}j=0\text{” terms cancel out} \\ &= \sum_{j=1}^{n} \left\{ \binom{n}{j} (-1)^j \frac{p^j - 1}{(1-p^j)^2} \right\} && \text{by geometric series} \\ &= \sum_{j=1}^{n} \left\{ \binom{n}{j} (-1)^j \left(-1 + \frac{p^j}{p^j - 1} \right) \right\} \\ &= 1 + \sum_{j=1}^{n} \left\{ \binom{n}{j} (-1)^j \frac{p^j}{p^j - 1} \right\}. && \text{by binomial theorem} \end{aligned}$$

If we now rewrite $\binom{n}{j}$ as $\frac{j+1}{n+1}\binom{n+1}{j+1}$, the sum in the last expression becomes $V_p(n+1)$, and this completes the proof. ■

Our final lemma is the key result (although not the most technically difficult to prove) towards the expression of $\mathbf{E}(C_n^{(m)})$ in terms of V_p and W_p.

Lemma 3.4 *Under the fixed population model, the expected height of a PSL of l keys, and the expected number of horizontal steps on the search path for the lth key in any PSL, are related as*

$$\mathbf{E}(T_l) - \frac{p}{q}(\mathbf{E}(L_l) - 1) = \frac{1}{q}, \qquad \textit{integer } l \geq 1.$$

Proof: Starting with the expression for $\mathbf{E}(L_l)$ in Lemma 3.2, we can interchange the two summations and compute the sum $\sum_{j=1}^{l-1} \left(1 - p^k\right)^j$, to get

$$\mathbf{E}(L_l) = 1 + \frac{q}{p} \sum_{k \geq 1} \left\{1 - p^k - (1 - p^k)^l\right\}, \qquad \text{integer } l \geq 1.$$

Using now the second expression for $\mathbf{E}(T_l)$ in Lemma 3.1, we can verify that in order to prove the lemma, it suffices to prove

$$\sum_{k \geq 1} \left\{\left(1 - p^k\right)^l - \left(1 - p^{k-1}\right)^l\right\} + \sum_{k \geq 1} p^k = \frac{1}{1-p}. \qquad \text{integer } l \geq 1,$$

But this follows immediately, by applying the telescopic property on the first sum, and by rewriting $\sum_{k \geq 1} p^k$ as $\frac{p}{1-p}$. ■

We can now easily derive exact expressions, in terms of V_p and W_p, for the various expected search costs.

Theorem 3.1 *Under the fixed population model, in a PSL of n keys, the expected cost for a successful search of the mth key, or for an unsuccessful search for a key between the $(m-1)$st and the mth, is*

$$\mathbf{E}(C_n^{(m)}) \;=\; V_p(n+1) \;+\; \frac{q}{p}\, V_p(m) \;+\; 1,$$

where $V_p(n+1)$ and $V_p(m)$ are the sums defined in (3.6).

Proof: The theorem follows from relation (3.3), from the value of $\mathbf{E}(T_n)$ in Lemma 3.3, and from

$$\mathbf{E}(L_{m-1}) = \frac{q}{p} V_p(m), \qquad m = 1, 2, \ldots, n+1,$$

which is a consequence of Lemmata 3.3 and 3.4. ■

Theorem 3.2 *Under the fixed population model, in a PSL of n keys, the expected cost for a successful search for the average key is*

$$\mathbf{E}(S_n) \;=\; V_p(n+1) \;+\; \frac{q}{p}\,\frac{W_p(n)}{n} \;+\; 1,$$

and the expected cost for an unsuccessful search for the average key is

$$\mathbf{E}(U_n) \;=\; V_p(n+1) \;+\; \frac{q}{p}\,\frac{W_p(n+1)}{n+1} \;+\; 1,$$

where $W_p(n)$, $W_p(n+1)$ and $V_p(n+1)$ are the sums defined in (3.5) and (3.6).

Proof: Both results follow immediately from Theorem 3.1, if we use the relationship (3.7) between W_p and V_p and we recall $W_p(0) = 0$. ■

3.4 Search Costs: Asymptotic Values

Theorems 3.1 and 3.2 give the search costs in terms of functions that "someone" has studied. In this section, we provide an approximation to the search costs in terms of constants (e.g. γ) and functions (e.g. Gamma function) that "everyone" has studied.

Naturally, our starting point will be the asymptotic expressions

$$W_p(l) = l \log_{\frac{1}{p}} l + l \left(\frac{1-\gamma}{\ln p} - \frac{1}{2} + f_{-1,\frac{1}{p}}(l) \right) + O(1) \qquad \text{as } l \to \infty \tag{3.8}$$

and

$$V_p(l) = \log_{\frac{1}{p}} l - \frac{\gamma}{\ln p} - \frac{1}{2} - f_{0,\frac{1}{p}}(l-1) + O\Big(\frac{1}{l}\Big) \qquad \text{as } l \to \infty, \tag{3.9}$$

that Knuth derived in [26, Ex. 5.2.2.50 and Ex. 6.3.19] (when $\frac{1}{p}$ is an integer, but his results hold for arbitrary p as well), where $f_{0,\frac{1}{p}}(l)$ and $f_{-1,\frac{1}{p}}(l-1)$ are the oscillating functions defined in (2.3) and discussed in Section 2.3.4. Knuth's proofs, based on an approach suggested by N. G. de Bruijn [26, p. 131], are essentially the same as those using Mellin transforms. Flajolet [15, p. 293] and Vitter and Flajolet [50, p. 489] derive the asymptotic expression for $V_{\frac{1}{2}}(l)$ by using Mellin transforms, and Mahmoud [30, Theorem 5.5] derives the asymptotic expression for $V_{\frac{1}{m}}(l)$, where m is an arbitrary integer, also by using Mellin transforms. Alternatively, one may use Rice's method, discussed in Section 2.3.9, to derive the asymptotics (3.8) and (3.9). Finally, since W_p and V_p are related as in (3.7), one may derive an asymptotic expression for $V_p(l)$ by using [26, Ex.5.2.2.50]

$$W_p(l) = l\log_{\frac{1}{p}} l + l\left(\frac{1-\gamma}{\ln p} - \frac{1}{2} + f_{-1,\frac{1}{p}}(l)\right) + \frac{1}{1-p} + \frac{1}{2\ln p} - \frac{1}{2}f_{1,\frac{1}{p}}(l) + O\Big(\frac{1}{l}\Big) \quad \text{as } l \to \infty.$$

We note though, that the last approach [35] results in tedious manipulations of the periodic functions $f_{1,\frac{1}{p}}(l)$ and $f_{-1,\frac{1}{p}}(l)$ appearing in $W_p(l)$ above, and the end result is not as elegant as the asymptotic expression for $V_p(l)$ in (3.9).

To express our results in a nicer form, we will use repeatedly the formulae

$$\ln(l+1) - \ln l = O\Big(\frac{1}{l}\Big) \qquad \text{as } l \to \infty \tag{3.10}$$

and

$$f_{0,x}(l) - f_{0,x}(l-1) = O\Big(\frac{1}{l}\Big) \qquad \text{as } l \to \infty. \tag{3.11}$$

The first formula is true, as it follows immediately from (2.28). To see why the second formula is true, we recall (from the proof of (2.33)) that the series $f_{0,x}(l)$

converges absolutely, and we therefore have

$$|f_{0,x}(l) - f_{0,x}(l-1)| \;\le\; \frac{2}{\ln x} \sum_{k\ge 1} \left(\left| \Gamma\left(-i\frac{2k\pi}{\ln x} \right) \right| \; \left| e^{i\,2k\pi \log_x l} - e^{i\,2k\pi \log_x (l-1)} \right| \right).$$

Since now $|s_1 - s_2| \le |\arg(s_1) - \arg(s_2)|$ for all s_1, s_2 on the unit circle, the last inequality implies

$$|f_{0,x}(l) - f_{0,x}(l-1)| \;\le\; \frac{2}{\ln x} \sum_{k\ge 1} \left(2k\pi \left| \Gamma\left(-i\frac{2k\pi}{\ln x} \right) \right| \; \left| \log_x l - \log_x (l-1) \right| \right) \;,$$

and (3.11) follows from (2.19) and (3.10).

We can now derive asymptotic expressions for the various expected search costs.

Theorem 3.3 *Under the fixed population model, in a PSL of n keys, the expected cost for a successful search of the mth key, or for an unsuccessful search of a key between the $(m-1)$st and the mth, is*

$$\begin{aligned} \mathbf{E}(C_n^{(m)}) &= \log_{\frac{1}{p}} n + \frac{q}{p} V_p(m) - \frac{\gamma}{\ln p} + \frac{1}{2} - f_{0,\frac{1}{p}}(n) + O\Big(\frac{1}{n}\Big) \qquad \text{as } n \to \infty, \\ &= \log_{\frac{1}{p}} n + \frac{q}{p} \log_{\frac{1}{p}} m - \frac{1}{p}\left(\frac{\gamma}{\ln p} + \frac{1}{2}\right) + 1 - f_{0,\frac{1}{p}}(n) - \frac{q}{p} f_{0,\frac{1}{p}}(m) + O\Big(\frac{1}{m}\Big) \text{ as } m,n \to \infty, \end{aligned}$$

where $f_{0,\frac{1}{p}}(l)$ is the periodic function of $\log_{\frac{1}{p}} l$ defined in (2.3), with its range bounded as in (2.33).

Proof: When $n \to \infty$, the exact expression for $\mathbf{E}(C_n^{(m)})$ from Theorem 3.1 and the asymptotic value of V_p in (3.9), together with identity (3.10), establish the first asymptotic expression for $\mathbf{E}(C_n^{(m)})$. When $n \to \infty$ and $m \to \infty$, Theorem 3.1 and (3.9) again, together with identities (3.10) and (3.11), establish the second asymptotic expression for $\mathbf{E}(C_n^{(m)})$. ■

The oscillating terms $f_{0,\frac{1}{p}}(n)$ and $f_{0,\frac{1}{p}}(m)$ in the above asymptotic search costs, although not very nice, can be computed to an arbitrary accuracy for any value of

p, m and n. From Table 2.1, we see that for all interesting values of p, the range of $f_{0,\frac{1}{p}}(l)$ is small, and its bound $F_{0,\frac{1}{p}}$ is very tight. Moreover, the computation of $F_{0,\frac{1}{p}}$ is very fast, since this series converges very rapidly. These suggest that one may substitute the straightforward and exact computation of the oscillations in the search costs by a faster computation of the bound of the range of the oscillations (the bounds are tight), or even drop the oscillations altogether (for $p = .5$, for example, one will be off my at most $.3164 \times 10^{-5}$ from the average search cost).

Theorem 3.4 *Under the fixed population model, in a PSL of n keys, the expected cost for a successful or an unsuccessful search for the average key is*

$$\begin{aligned}\mathbf{E}(S_n) \;=\; \mathbf{E}(U_n) \;&=\; \frac{1}{p}\log_{\frac{1}{p}} n - \frac{1}{p}\left(\frac{\gamma}{\ln p} + \frac{1}{2}\right) + \frac{q}{p\ln p} + 1 \\ &\quad + \frac{q}{p}\, f_{-1,\frac{1}{p}}(n) - f_{0,\frac{1}{p}}(n) + O\Big(\frac{1}{n}\Big) \qquad \text{as } n \to \infty,\end{aligned}$$

where $f_{0,\frac{1}{p}}(n)$ and $f_{-1,\frac{1}{p}}(n)$ are the periodic functions of $\log_{\frac{1}{p}} n$ defined in (2.3), and the range of the entire periodic term is bounded as

$$\left|\frac{q}{p} f_{-1,\frac{1}{p}}(n) - f_{0,\frac{1}{p}}(n)\right| \;\le\; \frac{1}{p}\sqrt{-\frac{2}{\ln p}}\sum_{k\ge 1}\left(\frac{\ln^2 p + 4k^2\pi^2 p^2}{k(\ln^2 p + 4k^2\pi^2)\sinh\left(-\frac{2k\pi^2}{\ln p}\right)}\right)^{1/2}.$$

Proof: The asymptotic expression for $\mathbf{E}(S_n)$ follows immediately from the exact value of $\mathbf{E}(S_n)$ in Theorem 3.2, from the asymptotic values of $W_p(l)$ and $V_p(l)$ in (3.8) and (3.9), and from identity (3.10). Also from Theorem 3.2 and asymptotics (3.8), (3.9) and (3.10), follows that the asymptotic value of $\mathbf{E}(U_n)$ is the same, but with $f_{-1,\frac{1}{p}}(n)$ replaced by $f_{-1,\frac{1}{p}}(n+1)$. Therefore, it suffices to show

$$f_{-1,\frac{1}{p}}(n+1) - f_{-1,\frac{1}{p}}(n) = O\Big(\frac{1}{n}\Big) \qquad \text{as } n \to \infty.$$

The proof of this is similar to the proof of identity (3.11) if we use property (2.22) of the Gamma function. Finally, the bound on the oscillating term follows similarly, if we use property (2.16). ■

The remarks about $f_{0,\frac{1}{p}}(l)$, that we made after Theorem 3.3, hold for $f_{-1,\frac{1}{p}}(l)$ as well. Notice also, that although we could have simply said "$\frac{q}{p}f_{-1,\frac{1}{p}}(n) - f_{0,\frac{1}{p}}(n)$ is an oscillating term of small magnitude", we chose not only to actually state a bound of the range of this term, but also to derive a bound tighter than the obvious $\frac{q}{p}F_{-1,\frac{1}{p}} + F_{0,\frac{1}{p}}$, where $F_{-1,\frac{1}{p}}$ is the bound of $f_{-1,\frac{1}{p}}(l)$ defined in (2.34).

3.5 Update Costs

Our results on the search costs allow us to derive costs for insertions and deletions; these two operations were described in Section 3.1.

The insertion of a key between the $(m-1)$st and the mth requires on the average $\mathbf{E}(C_n^{(m)})$ key comparisons to search for it, plus $\frac{2}{q}$ pointer updates, since the average height of an element is $\frac{1}{q}$.

The deletion of the mth key requires on the average $\mathbf{E}(C_n^{(m)})$ key comparisons to search for it, plus $\frac{1}{q}$ pointer updates to "remove" it, plus $1+\mathbf{E}(T_n)-\mathbf{E}(T_{n-1})$ pointer inspections to discover a possible update in the height of the PSL. But, according to Lemma 3.3, $\mathbf{E}(T_n)-\mathbf{E}(T_{n-1}) = V_p(n+1)-V_p(n)$, and the asymptotic value of V_p in (3.9), together with identities (3.10) and (3.11), imply $V_p(n+1)-V_p(n) = O\left(\frac{1}{n}\right)$. Therefore, to discover a possible decrease in the height of a PSL after a deletion, $1+O\left(\frac{1}{n}\right)$ pointer inspections are required on the average.

The expected value for the cost to insert or delete the average key, follow easily from our previous results.

n	optimal p	n	optimal p	n	optimal p	n	optimal p	n	optimal p
≤ 8	0	40	.2600	90	.2827	500	.3101	1,000	.3164
9	.0861	50	.2684	100	.2851	600	.3117	2,000	.3214
10	.1386	60	.2736	200	.2982	700	.3130	3,000	.3243
20	.2289	70	.2772	300	.3036	800	.3142	4,000	.3260
30	.2472	80	.2801	400	.3076	900	.3154	∞	$\frac{1}{e} = .3679$

Table 3.1: Optimal p, minimizing exact successful search cost for the average key

3.6 Choosing p

In our discussions so far, p was an unspecified "design parameter". Having derived now all exact and asymptotic search and update costs, we may turn to the issue of picking the optimal p.

If we assume that our target is the minimization of the average cost for the average key, and if we also assume that all searches (updates) are equally likely, then Theorem 3.4 implies that *asymptotically* the optimal choice for p is $\frac{1}{e}$. This value of p was also proposed by Pugh [40], based on the fact that asymptotically it minimizes his upper bound (3.1) of the search cost for $+\infty$.

Clearly, the above "asymptotically" optimal p, is the optimal p "for large values of $\ln n$". This makes the choice of the *exact* value $p = \frac{1}{e}$ disputable. Indeed, the values of p minimizing the *exact* (rather than the asymptotic) values of $\mathbf{E}(S_n)$ (as given in Theorem 3.2) that are shown in Table 3.1, imply that making p a bit smaller than $\frac{1}{e}$ (say, $.3125 = 5/16$ that requires only 4 random bits, or $.34375 = 11/32$ that requires only 5 random bits) is a better choice for moderate n.

Having observed that, the next issue that may be worth investigating is how much $\mathbf{E}(S_n)$ is affected if we do *not* choose the exact optimal p from Table 3.1, but we choose $p = \frac{1}{e}$ instead. The answer is "not very much". For example, from Table 3.1 we see that for a PSL of $n = 1000$ keys the optimal p minimizing $\mathbf{E}(S_{1000})$ is $p = .3164$; in this case, we can compute $\mathbf{E}(S_{1000}) = 18.1029$. If we use $p = \frac{1}{e}$

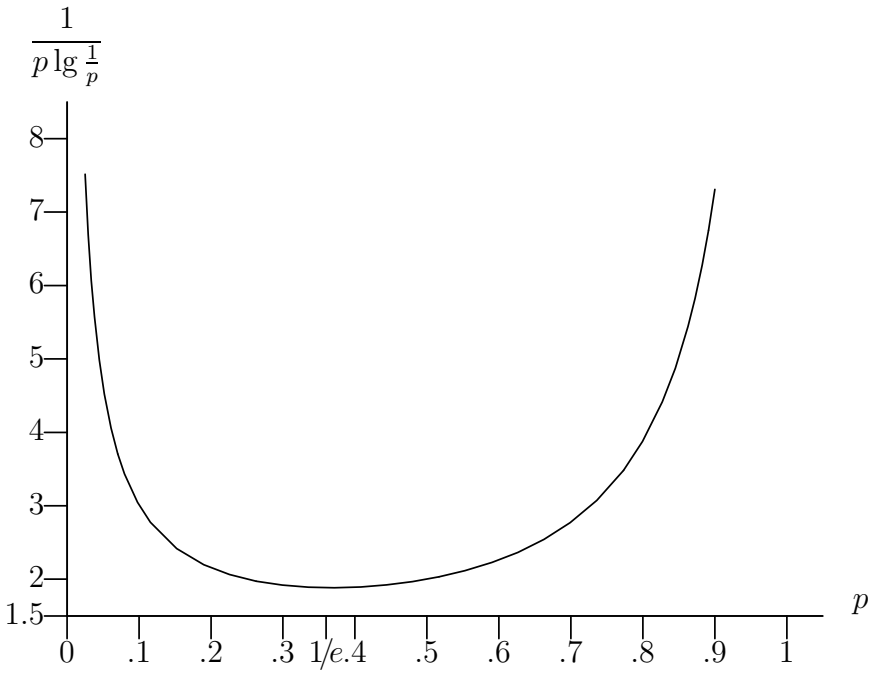


Figure 3.2: Coefficient of $\lg n$ in asymptotic search cost for the average key

instead, we get $\mathbf{E}(S_{1000}) = 18.2711$, a mere 1% slower average search time for the average key. There are basically two reasons for which this is happening. First, the asymptotic expression for $\mathbf{E}(S_n)$ is an excellent approximation of the exact value of $\mathbf{E}(S_n)$, even for moderate values of n. And second, the coefficient $\frac{1}{p \lg p}$ of $\lg n$ in the asymptotic expression of $\mathbf{E}(S_n)$ is very flat around $p = \frac{1}{e}$, as we may see from Figure 3.2.

To summarize, a value of p slightly smaller than $\frac{1}{e}$ may be a better choice for moderate n, but not by much. Notice here that the values of p minimizing the exact value of $\mathbf{E}(U_n)$ are different (although very close) from the values of p shown in Table 3.1. The values of p shown in Table 3.1 also suggest the idea of having p depend on the level; p would be relatively small at the bottom and tend to rise to $\frac{1}{e}$ in higher levels. We note however, that such ideas will impact only a constant in search costs.

3.7 Comparison with Previous Results

It is appropriate to ask how close our asymptotic expressions, or indeed Pugh's upper bounds, are to the exact values of the expected costs, in the range of n for which calculating the exact values is feasible.

We first consider $\mathbf{E}(F_n)$, a bound on the average cost for the search of any key, as discussed in Section 3.2. Exact values

$$\mathbf{E}(F_n) = \frac{1}{p} V_n(n+1) + 1 \tag{3.12}$$

of the average search cost for $+\infty$ (from Theorem 3.1), our approximation

$$\mathbf{E}(F_n) \simeq \frac{1}{p} \log_{\frac{1}{p}} n - \frac{1}{p} \left(\frac{\gamma}{\ln p} + \frac{1}{2} \right) + 1 - \frac{1}{p} f_{0,\frac{1}{p}}(n) \tag{3.13}$$

of this cost (from Theorem 3.3 and asymptotics (3.10) and (3.11)), and Pugh's upper bound

$$\frac{1}{p} \log_{\frac{1}{p}} n + \frac{1}{q} + 1$$

of it (stated in (3.1)), are shown in Table 3.2, for a PSL of $n = 50$, 500, 1000 and 5000 keys. The computation of the exact $\mathbf{E}(F_n)$ from (3.12) was very time consuming; these values, computed to an accuracy of 10^{-10}, took approximately 40 hours on a MIPS M2000 machine, with the computation of $\mathbf{E}(F_{5000})$ alone taking almost 39 hours. The computation of our approximation (3.13) on the other hand, was quite fast; had we been willing to slightly give up the accuracy of our approximation by computing $F_{0,\frac{1}{p}}$, the bound of $f_{0,\frac{1}{p}}(n)$ from (2.33), instead of the actual $f_{0,\frac{1}{p}}(n)$ appearing in (3.13), the computation of our approximation to $\mathbf{E}(F_n)$ would have been even faster. Finally the computation of Pugh's upper bound was extremely fast.

If we look carefully at the values in Table 3.2, we may observe that across the lines (that is, for fixed p) the differences between the exact value of the expected

p	our approximate	(our) exact	Pugh's bound of	our approximate	(our) exact	Pugh's bound of
	$\mathbf{E}(F_{50})$	$\mathbf{E}(F_{50})$	$\mathbf{E}(F_{50})$	$\mathbf{E}(F_{500})$	$\mathbf{E}(F_{500})$	$\mathbf{E}(F_{500})$
$1–10^{-4}$=.9999	44,895.1315	44,994.8031	49,123.1862	67,922.1338	67,932.1309	72,150.1885
$1–10^{-3}$=.9990	4,491.9847	4,501.9563	4,914.9806	6,795.7220	6,796.7222	7,218.7180
$1–10^{-2}$=.9900	451.6823	452.6840	494.1748	683.1018	683.2023	725.5942
9/10=.9000	47.7871	47.8922	52.2554	72.0697	72.0802	76.5380
4/5=.8000	25.5227	25.5785	27.9143	38.4213	38.4269	40.8129
3/4=.7500	21.1398	21.1860	23.1312	31.8117	31.8163	33.8031
7/10=.7000	18.2662	18.3061	20.0020	27.4886	27.4927	29.2244
2/3=.6667	16.8577	16.8946	18.4724	25.3760	25.3797	26.9907
$1–1/e$=.6321	15.6925	15.7268	17.2109	23.6341	23.6376	25.1525
3/5=.6000	14.8137	14.8462	16.2637	22.3263	22.3296	23.7764
1/2=.5000	12.9532	12.9820	14.2877	19.5971	19.5999	20.9316
2/5=.4000	11.9985	12.0257	13.3402	18.2807	18.2834	19.6226
$1/e$=.3679	11.8438	11.8709	13.2160	18.1032	18.1060	19.4750
1/3=.3333	11.7581	11.7851	13.1826	18.0460	18.0487	19.4703
3/10=.3000	11.7614	11.7887	13.2594	18.1373	18.1400	19.6344
1/4=.2500	11.9580	11.9877	13.6210	18.5968	18.5997	20.2649
1/5=.2000	12.4553	12.4884	14.4034	19.5984	19.6014	21.5568
1/10=.1000	15.4104	15.4418	19.1008	25.4104	25.4137	29.1008
10^{-2}=.0100	40.8507	41.0032	86.9586	105.2538	105.2707	136.9586
10^{-3}=.0010	49.8206	49.8444	568.3243	394.9697	395.1214	901.6577
10^{-4}=.0001	50.8802	50.8827	4,249.4251	488.7558	488.7795	6,749.4251
	$\mathbf{E}(F_{1000})$	$\mathbf{E}(F_{1000})$	$\mathbf{E}(F_{1000})$	$\mathbf{E}(F_{5000})$	$\mathbf{E}(F_{5000})$	$\mathbf{E}(F_{5000})$
$1–10^{-4}$=.9999	74,853.9522	74,858.9516	79,082.006955	90,949.1361	90,950.1361	95,177.1909
$1–10^{-3}$=.9990	7,489.2161	7,489.7162	7,912.2120	9,099.4594	9,099.5594	9,522.4553
$1–10^{-2}$=.9900	752.7660	752.8162	795.2584	914.5213	914.5313	957.0137
9/10=.9000	79.3795	79.3847	83.8478	96.3523	96.3533	100.8206
4/5=.8000	42.3041	42.3069	44.6957	51.3198	51.3204	53.7114
3/4=.7500	35.0243	35.0266	37.0157	42.4836	42.4841	44.4750
7/10=.7000	30.2649	30.2669	32.0006	36.7111	36.7115	38.4468
2/3=.6667	27.9403	27.9422	29.5549	33.8944	33.8947	35.5090
$1–1/e$=.6321	26.0248	26.0265	27.5432	31.5757	31.5761	33.0942
3/5=.6000	24.5878	24.5895	26.0379	29.8389	29.8392	31.2890
1/2=.5000	21.5971	21.5985	22.9316	26.2409	26.2412	27.5754
2/5=.4000	20.1720	20.1734	21.5137	24.5632	24.5635	25.9049
$1/e$=.3679	19.9870	19.9884	21.3592	24.3619	24.3622	25.7341
1/3=.3333	19.9394	19.9408	21.3631	24.3341	24.3343	25.7581
3/10=.3000	20.0571	20.0585	21.5535	24.5131	24.5134	26.0094
1/4=.2500	20.5973	20.5988	22.2649	25.2364	25.2367	26.9088
1/5=.2000	21.7476	21.7492	23.7101	26.7476	26.7480	28.7101
1/10=.1000	28.3830	28.3849	32.1111	35.4105	35.4108	39.1008
10^{-2}=.0100	110.6127	110.6134	152.0101	140.8507	140.8523	186.9586
10^{-3}=.0010	634.1211	634.3051	1,002.0010	999.2546	999.2714	1,234.9910
10^{-4}=.0001	952.7258	952.7711	7,502.0001	3,936.1934	3,936.3451	9,249.4251

Table 3.2: Search costs for $+\infty$

search cost and the approximation of it do not seem to be $O(\frac{1}{n})$. In fact, these differences are not even decreasing (the differences, for example, for $p=10^{-4}$ form an increasing sequence). This should not surprise us, if we recall that the approximation of the expected search cost contains an oscillating function $f_{0,\frac{1}{p}}(l)$. This oscillation causes our approximation to move close to the exact value, then to move away, then to move close again, etc. It is the *maximum difference* between the exact and the approximate value that is $O(\frac{1}{n})$; as we showed in (2.31), these maximum differences occur at n, $\frac{1}{p}n$, $\frac{1}{p^2}n$, $\frac{1}{p^3}n$, ..., not the values of n shown in Table 3.2.

We now turn to the comparison of the average costs for the successful search of the average key in a PSL; the results for unsuccessful searches don't differ by much. Exact values

$$\mathbf{E}(S_n) = V_p(n+1) + \frac{q}{p}\frac{W_p(n)}{n} + 1$$

(from Theorem 3.2), our approximation

$$\mathbf{E}(S_n) \simeq \frac{1}{p}\log_{\frac{1}{p}} n - \frac{1}{p}\left(\frac{\gamma}{\ln p} + \frac{1}{2}\right) + \frac{q}{p\ln p} + 1 + \frac{q}{p}f_{-1,\frac{1}{p}}(n) - f_{0,\frac{1}{p}}(n)$$

of it (from Theorem 3.4), and Pugh's upper bound

$$\log_{\frac{1}{p}} n + \frac{q}{pn}\sum_{m=2}^{n}\log_{\frac{1}{p}}(m-1) + \frac{1}{q} + \frac{n-1}{n}$$

of it from (3.2), are shown in Table 3.3. The remarks made on the difficulty of the computation of the numbers of Table 3.2 apply here as well.

A comparison now of the exact values and Pugh's upper bound indicates that Pugh's upper bound is a reasonably good approximation for $\mathbf{E}(S_n)$, for the interesting values of p. A comparison of our exact and approximate values indicates that our approximate costs (are not only much faster than the exact costs to compute, but also) approach the exact costs quite rapidly. The fact that the convergence for $\mathbf{E}(S_n)$ is slower that the convergence for $\mathbf{E}(F_n)$, should not be surprising; for

p	our approximate	(our) exact	Pugh's bound of	our approximate	(our) exact	Pugh's bound of
	$\mathbf{E}(S_{50})$	$\mathbf{E}(S_{50})$	$\mathbf{E}(S_{50})$	$\mathbf{E}(S_{500})$	$\mathbf{E}(S_{500})$	$\mathbf{E}(S_{500})$
$1-10^{-4}=.9999$	44,894.1314	44,993.8031	49,122.1455	67,921.1337	67,931.1309	72,149.1821
$1-10^{-3}=.9990$	4,490.9842	4,500.9558	4,913.9394	6,794.7215	6,795.7217	7,217.7111
$1-10^{-2}=.9900$	450.6773	451.6791	493.1289	682.0968	682.1972	724.5828
$9/10=.9000$	46.7325	46.8387	51.1590	71.0151	71.0257	75.4768
$4/5=.8000$	24.4024	24.4608	26.7507	37.3009	37.3068	39.6856
$3/4=.7500$	19.9811	20.0308	21.9286	30.6530	30.6580	32.6374
$7/10=.7000$	17.0647	17.1091	18.7555	26.2871	26.2915	28.0155
$2/3=.6667$	15.6246	15.6668	17.1937	24.1429	24.1471	25.7501
$1-1/e=.6321$	14.4236	14.4643	15.8958	22.3653	22.3694	23.8762
$3/5=.6000$	13.5086	13.5483	14.9116	21.0212	21.0252	22.4636
$1/2=.5000$	11.5105	11.5504	12.7951	18.1544	18.1584	19.4806
$2/5=.4000$	10.3614	10.4059	11.6493	16.6437	16.6481	17.9764
$1/e=.3679$	10.1255	10.1726	11.4421	16.3848	16.3895	17.7472
$1/3=.3333$	9.9381	9.9888	11.3045	16.2258	16.2309	17.6399
$3/10=.3000$	9.8242	9.8796	11.2613	16.1995	16.2050	17.6859
$1/4=.2500$	9.7910	9.8568	11.3922	16.4321	16.4386	18.0894
$1/5=.2000$	9.9606	10.0418	11.8466	17.1175	17.1257	19.0585
$1/10=.1000$	11.6170	11.7850	15.0912	21.6170	21.6338	25.1731
$10^{-2}=.0100$	20.7412	22.4358	64.9959	83.6352	83.8273	115.3649
$10^{-3}=.0010$	5.6377	25.1618	420.6895	212.4922	214.1887	756.4027
$10^{-4}=.0001$	−174.0366	25.4658	3,141.2962	226.9341	246.4584	5,659.0443
	$\mathbf{E}(S_{1000})$	$\mathbf{E}(S_{1000})$	$\mathbf{E}(S_{1000})$	$\mathbf{E}(S_{5000})$	$\mathbf{E}(S_{5000})$	$\mathbf{E}(S_{5000})$
$1-10^{-4}=.9999$	74,852.9521	74,857.9516	79,081.0034	90,948.1360	90,949.1360	95,176.1899
$1-10^{-3}=.9990$	7,488.2156	7,488.7157	7,911.2080	9,098.4589	9,098.5589	9,521.4540
$1-10^{-2}=.9900$	751.7610	751.8112	794.2499	913.5162	913.5263	956.0078
$9/10=.9000$	78.3249	78.3302	82.7896	95.2977	95.2988	99.7651
$4/5=.8000$	41.1838	41.1867	43.5715	50.1995	50.2001	52.5901
$3/4=.7500$	33.8656	33.8681	35.8531	41.3249	41.3254	43.3157
$7/10=.7000$	29.0633	29.0655	30.7950	35.5095	35.5099	37.2442
$2/3=.6667$	26.7072	26.7093	28.3177	32.6612	32.6616	34.2748
$1-1/e=.6321$	24.7560	24.7580	26.2702	30.3069	30.3073	31.8243
$3/5=.6000$	23.2827	23.2847	24.7285	28.5338	28.5342	29.9828
$1/2=.5000$	20.1544	20.1564	21.4842	24.7982	24.7986	26.1316
$2/5=.4000$	18.5349	18.5372	19.8715	22.9261	22.9266	24.2666
$1/e=.3679$	18.2688	18.2711	19.6356	22.6437	22.6442	24.0145
$1/3=.3333$	18.1190	18.1215	19.5370	22.5136	22.5141	23.9362
$3/10=.3000$	18.1183	18.1211	19.6095	22.5747	22.5752	24.0699
$1/4=.2500$	18.4340	18.4372	20.0944	23.0761	23.0767	24.7431
$1/5=.2000$	19.2635	19.2675	21.2175	24.2635	24.2643	26.2229
$1/10=.1000$	24.5345	24.5425	28.1916	31.6170	31.6187	35.1894
$10^{-2}=.0100$	95.9356	96.0311	130.4570	120.7412	120.7581	165.4464
$10^{-3}=.0010$	368.6445	369.4604	857.0133	804.8403	805.0365	1,090.2741
$10^{-4}=.0001$	474.8386	484.3861	6,431.6196	2,130.0436	2,131.7404	8,163.0722

Table 3.3: Successful search costs for the average key

"small" n, "many" m — over which $\mathbf{E}(S_n)$ is taken — are "small", resulting in "bad" approximations for $\mathbf{E}(S_n)$. The negative values of the approximate $\mathbf{E}(S_n)$ should not be surprising either; what is important is that the differences between the exact and the approximate values of $\mathbf{E}(S_n)$ are $O(\frac{1}{n})$.

3.8 Summary

In this chapter, we derived expected values for various costs in the PSL, under the fixed population model. We gave exact (Theorem 3.1) and asymptotic (Theorem 3.3) expressions for the cost of the search of the mth key in an average PSL of n keys, as well as exact (Theorem 3.2) and asymptotic (Theorem 3.4) expressions for the cost of the search of an average key in an average PSL of n keys. We also observed (Tables 3.2 and 3.3) that the derived asymptotic expressions are excellent approximations to the corresponding exact values, and that Pugh's bounds of the various costs are fairly good for all interesting values of p. Although our asymptotic expressions contain some oscillating terms, these terms can be computed fast to an arbitrary accuracy, and, for all interesting values of p, they take small values, and their ranges are tightly bound by an easily computable function of p alone (i.e. the bound of their ranges is independent of m and n). This leads to the observation that the expected search costs can be expressed as falling within a small constant of a very simple form.

To derive the above results, we split the search cost into two (*non*-independent) parts, and we made two key observations. The first observation was that one of these parts is related to some well-studied parameters of tries (discussed after Lemma 3.2). The second observation was that the expected values of these two parts are related (Lemma 3.4).

As a consequence of the computation of the search costs, we derived the insert and delete costs (Section 3.5). We also discussed the choice of the parameter p, used to build a PSL; although the value of $p = \frac{1}{e}$ was known to be asymptotically the optimal choice, we observed (Table 3.1) that for PSLs of reasonable sizes, a somewhat lower value of p is a slightly better choice.

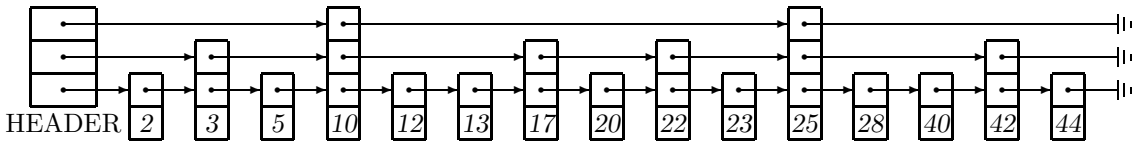
Finally, we remind the reader that the number of comparisons performed in a PSL is heavily dependent on coding details; e.g. when we test for equality, if we note that a comparison has already been made, if we count tests for null pointers (that is, if we count comparisons with the $+\infty$ key), etc. We have chosen one particular coding scheme for our analysis, but other variants easily follow.

Chapter 4

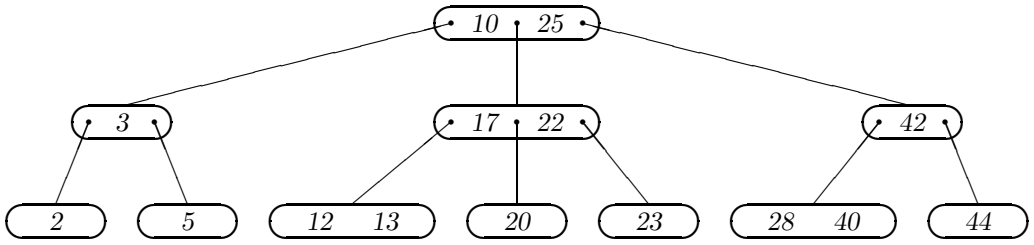
Deterministic Skip Lists

The good *average case* performance of the PSL discussed in the previous chapter, although independent of the input, does depend on the random number generator behaving "as expected". Should this not be the case at a particular instance (if, for example, the random number generator creates elements of equal heights, or of decreasing heights left-to-right), the PSL may degenerate into a structure worse than a linear linked list. It is therefore natural to try to explore techniques based on the skip list notion, that will lead to good *worst case* performance.

As discussed in Section 1.1, there exist several balanced search tree schemes (AVL, 2-3, red-black) which have guaranteed logarithmic worst case search and update costs. A general problem though with these schemes is that they are above the threshold of difficulty for most programmers to implement in virtually any case, except of course in B-tree packages. The solutions that we propose in this chapter are competitive in terms of space and time with balanced search trees, and, we feel, inherently simpler when taken from first principles. They also give rise to a hybrid data structure, halfway between the standard search tree and the original skip list.



(a)



(b)

Figure 4.1: A 1-2 DSL and its corresponding 2-3 tree

An earlier version of our results appeared in [33].

4.1 The Starting Point

A natural starting point is the perfectly balanced skip list, a skip list in which every kth element of height at least h is also of height at least $h+1$ (for some fixed k). Although the search cost in balanced skip lists is logarithmic, the insert and delete costs are prohibitive. We should therefore examine the consequences of relaxing a bit the strict requirement "*every* kth element".

Assuming that a skip list of n keys has a 0th and a $(n+1)$st element of height equal to the height of the skip list, we will say that two elements are *linked* when there exists (at least) one pointer going from one to the other. Given two linked elements, one of height exactly h $(h>1)$ and another of height h or higher, their

gap size will be the number of elements of height $h-1$ that exist between them. For example, in the skip list of Figure 3.1 the gap size between *19* and *30* is 0, and in the skip list of Figure 4.1(a) the gap size between *10* and *25* is 2, whereas the gap size between $-\infty$ and *3* is 1. A skip list with the property that *every gap is of size either 1 or 2* will be called a *1-2 Deterministic Skip List* (DSL). As we see from Figure 4.1, there exists a one-to-one correspondence between 1-2 DSLs and 2-3 trees.

A search for a key in a 1-2 DSL is performed in the same manner as in Pugh's PSL. One may of course observe that after 2 key comparisons at the same level, the only next legal step is to drop down a level. Therefore, it is possible to throw in an extra line of code and save up to 1 key comparison per level.

An insertion of a key in a 1-2 DSL is made by initially searching for the key to be inserted, and by subsequently inserting the key as an element of height 1 in the appropriate spot. This may, of course, trigger the invalid configuration of 3 elements of height 1 "in a row". This is easily rectified by letting the middle of these 3 elements grow to height 2. If this now results in 3 elements of height 2 in a row, we let the middle of these 3 elements grow to height 3, etc. For example, the insertion of *15* in the skip list of Figure 4.1(a) will cause *13* to grow from height 1 to height 2, that will cause *17* to grow from 2 to 3, and that will cause *17* again to grow from 3 to 4.

A deletion of a key from a 1-2 DSL is done in a way completely analogous to the way it is done in the corresponding 2-3 tree. The deletion of *5* for example from the 1-2 DSL of Figure 4.1(a), will cause *3* to shrink, and this will cause *10* to shrink and *17* to grow.

In general, one may verify that the above described insertion algorithm may

cause up to $\lfloor \lg(n+2) \rfloor - 1$ elements to grow in the worst case (except when $n = 3$ and $n = 9$, in which case this number is 1 less). In the 2-3 tree analogy, this is not a great problem, since each increase of an element's height takes constant time. However, in our setting, if we insist on implementing the set of horizontal pointers of each DSL element as an array of pointers, then, when an element grows, we have to allocate space for a higher element, and we have to copy all pointers in and out of the old element into the new one. Therefore, the growth of a single element from height h to height $h+1$ requires the change of $2h+2$ pointers. Since in the worst case all elements of heights $1, 2, \ldots, \lfloor \lg(n+2) \rfloor - 1$ will have to grow, an insertion may take up to $\Theta(\lg^2 n)$ time. Similarly, a deletion may take up to $\Theta(\lg^2 n)$ time, too.

Although the 1-2 DSL does not exhibit the desired logarithmic behaviour, it is interesting to note that its worst case space complexity is not worse than that of other balanced search tree schemes. The next lemma states this result more precisely.

Lemma 4.1 *In the worst case, the total number of pointers (including the header and the null pointers) in a 1-2 DSL of n keys is exactly*

$$2n - \nu_2(n+1) + 1, \qquad \textit{integer } n \geq 1,$$

where $\nu_2(n+1)$ is the numbers of 1's in the binary representation of $n+1$.

Proof: The number of pointers at level 1 is always $n+1$. The number of pointers at level 2 is at most $\left\lfloor \frac{n-1}{2} \right\rfloor + 1$; this occurs when every second element is raised to level 2. Similarly, the number of pointers at level 3 is at most $\left\lfloor \frac{\left\lfloor \frac{n-1}{2} \right\rfloor - 1}{2} \right\rfloor + 1 = \left\lfloor \frac{n-(1+2)}{4} \right\rfloor + 1$, and in general, the number of pointers at level k ($k = 1, 2, \ldots$)

is at most $\left\lfloor \frac{n-(1+2+4+\cdots+2^{k-2})}{2^{k-1}} \right\rfloor + 1 = \left\lfloor \frac{n+1}{2^{k-1}} \right\rfloor$. When all of the above occur, the height of the structure is exactly $\lfloor \lg(n+1) \rfloor$. Therefore, in such a case, the number of pointers is $\sum_{k=1}^{\lfloor \lg(n+1) \rfloor} \left\lfloor \frac{n+1}{2^{k-1}} \right\rfloor$. But this is a well known sum (see, for example, [23, pp. 113–114]), which evaluates to the expression of the lemma. ■

4.2 Achieving logarithmic worst case costs

One solution to the problem of the $\Theta(\lg^2 n)$ cost encountered in the preceding section is to implement the set of horizontal pointers of each element as a linked list of pointers. This way, raising or lowering an element takes $\Theta(1)$ time, resulting thus in a $\Theta(\lg n)$ worst case update cost. However, such a solution will double the space requirement of our data structure; each horizontal pointer will be substituted by a right pointer, a down pointer, and a pointer to the key (or the key itself). Therefore, according to Lemma 4.1, this version of the 1-2 DSL will require in the worst case up to $6n$ pointers and n keys, or up to $4n$ pointers and $2n$ keys. We will return to this notion in the following sections, because of its simplicity.

If the above space overhead is considered unacceptable, we may implement the set of horizontal pointers of each element as an array of pointers, if we choose these arrays to have *exponentially increasing physical heights*. For simplicity, let's say that all elements will have physical heights 2^k, for $k = 0, 1, 2, 3, \ldots$. The logical heights of the elements will never exceed their physical heights, and they will be equal to the heights of the elements in the 1-2 DSL considered in the last section. We reserve the term *array implementation* of the 1-2 DSL for this variant of deterministic skip list. For example, if we insert *15* into the skip list of Figure 4.1(a), the resulting skip list under the just described scheme will be the one shown in Figure 4.2.

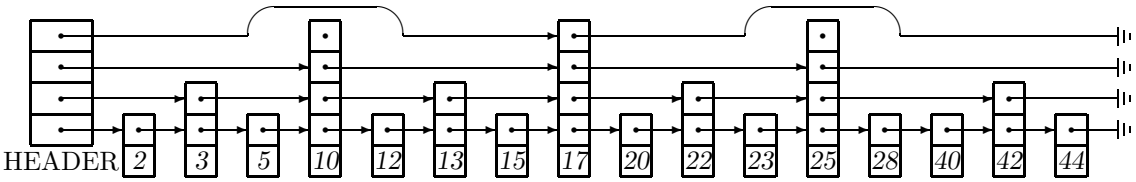


Figure 4.2: Array implementation of 1-2 DSL

When, in performing an insertion, an element has to grow, we use the higher level pointer field, if one exists. If not, a new higher element has to be created, and all pointers in and out of the old element have to be copied into the new one. At this point, we achieve the $\Theta(\lg n)$ worst case insertion cost. Consider the 2^k ($k \geq 1$) elements of logical heights $2^k+1, \ldots, 2^{k+1}$. All of them are of physical height 2^{k+1}. When the heights of all of them have to be increased by 1, only the element of logical height 2^{k+1} has to be copied into a new element; the remaining elements will merely increase their logical heights, and their physical heights allow them to do so in constant time. Hence, the time required for an insertion, is proportional to the sum of the heights of the elements that will be copied, that is, $\sum_{k=0}^{\Theta(\lg \lg n)} 2^k = \Theta(\lg n)$. To delete a key, we apply the same technique, achieving thus a worst case $\Theta(\lg n)$ cost as well.

As it turns out, our new skip list variant does not increase the storage requirement by very much.

Lemma 4.2 *In the worst case, the total number of pointers (including the header and the null pointers) in the array implementation of a 1-2 DSL of n keys is exactly*

$$\begin{cases} n+1 & \text{for } n = 1, 2 \\ n+1+\sum_{k=0}^{\lfloor \lg(\lg(n+1)-1) \rfloor} 2^k \left\lfloor \frac{n+1}{2^{2^k}} \right\rfloor & \text{for } n \geq 3 \end{cases}, \tag{4.1}$$

which never exceeds

$$(\alpha_n + 1)n + \alpha_n + 1 \;\sim\; 2.2814941\ldots n, \tag{4.2}$$

where

$$\alpha_n \stackrel{\text{def}}{=} \begin{cases} 0 & \textit{for} \quad n = 1, 2 \\ \displaystyle\sum_{k=0}^{\lfloor \lg(\lg(n+1)-1)\rfloor} 2^{k-2^k} & \textit{for} \quad n \geq 3 \end{cases} .$$

Proof: The number of pointers at level 1 is always $n+1$. The number of pointers at level 2 is at most $\left\lfloor \frac{n-1}{2} \right\rfloor + 1$; this occurs when every second element is raised to level 2. The number of pointers at levels 3 and 4 is at most $2\left\lfloor \frac{n-(1+2)}{2^2} \right\rfloor + 2$; this occurs when every second element at level 2 is raised to level 3. Similarly, the number of pointers at levels 5, 6, 7 and 8 is at most $4\left\lfloor \frac{n-(1+2+2^2+2^3)}{2^4} \right\rfloor + 4$, and in general, the number of pointers at levels $2^k+1, \ldots, 2^{k+1}$ $(k = 0, 1, 2, \ldots)$ is at most $2^k\left\lfloor \frac{n-(1+2+2^2+\cdots+2^{2^k-1})}{2^{2^k}} \right\rfloor + 2^k = 2^k\left\lfloor \frac{n+1}{2^{2^k}} \right\rfloor$. When all the above occur, it is not hard to see that the physical height of the structure is exactly $\lfloor \lg(\lg(n+1) - 1)\rfloor + 1$, and the result follows easily. ■

We note that α_n can be computed quickly to a high accuracy, since its few terms drop off doubly exponentially. We have $\alpha_3 = \alpha_4 = \alpha_5 = \alpha_6 = .5$, $\alpha_7 = \cdots \alpha_{30} = 1$, $\alpha_{31} = \cdots \alpha_{510} = 1.25$, and $\alpha_n \simeq 1.281$ for $n \geq 511$. Furthermore, the bound (4.2) is very tight, since from (4.1) we may compute that in the worst case the total numbers of pointers in skip lists with $10, 10^2, 10^3, 10^4, 10^5$ and 10^6 keys are exactly 20, 225, 2,273, 22,813, 228,121 and 2,281,489 respectively.

The linked list and the array implementation of the elements of the 1-2 DSL, although both achieving logarithmic worst case costs, have different merits. The second uses fewer extra pointers; up to $2.282n$, versus $6n$. The first is simpler to code; for example, in order to to raise a skip list element we only have to change 3

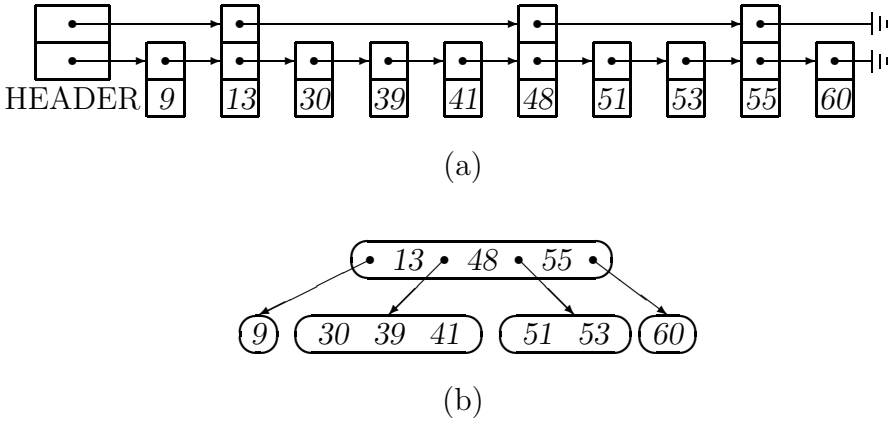


Figure 4.3: A 1-3 DSL and its corresponding 2-3-4 tree

pointers, whereas in the array implementation we have to initiate a search for the key in the element to be raised, so that we can splice at the proper place and insert the new element.

Finally, we observe that the correspondence between 2-3 trees and 1-2 DSLs can be easily generalized. For any B-tree of order m, $(m \geq 3)$, we can define a deterministic skip list with gaps of sizes $\lceil \frac{m}{2} \rceil - 1$, $\lceil \frac{m}{2} \rceil$, $\ldots, m-2$, or $m-1$; an example is shown in Figure 4.3. Clearly, any such 1-2, 1-3, 2-4, 2-5, 3-6, ... DSL achieves logarithmic worst case costs by having its elements implemented either as linked lists or as arrays of exponential heights.

4.3 Top-down Deterministic Skip Lists

As noted in [24], insertions in a 2-3-4 tree can be performed top-down, eliminating thus the need to maintain a stack with the search path. Adopting this approach, we may chose to perform an insertion in a 1-3 DSL by splitting any gap of size 3 on our way to the bottom level into two gaps of size 1. We ensure in this way that the

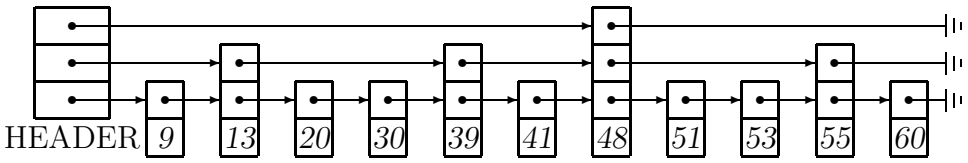


Figure 4.4: Top-down 1-3 DSL insertion

structure retains the gap invariant with or without the inserted key. To be more precise, we start our search at the header, and at level 1 higher than the height of the skip list. If the gap that we are going to drop in is of size 1 or 2, we simply drop. If the gap is of size 3, first we raise the middle element in this gap, creating thus two gaps of size 1 each, and then we drop. When we reach the bottom level, we simply insert a new element of height 1. Since our algorithm allowed only gaps of sizes 1 and 2, the newly inserted element leaves us with a valid 1-3 DSL.

As an example, consider the case of inserting *20* in the skip list of Figure 4.3(a). We start at level 3 of the header, we look at level 2 and we raise *48*, then we drop to level 2 of the header, we move to level 2 of *13*, we look at level 1 and we raise *39*, then we drop to level 1 of *13*, and finally we insert *20* as a new node of height 1. The resulting skip list is shown in Figure 4.4.

To delete a key from a 1-3 DSL, we may work in a top-down manner as well. We want the search preceding the actual removal of the element to have the side-effect that each gap is of legal — but above minimum — size as we pass through it. This is handled by either merging with a neighbour, or borrowing from a neighbour. More precisely, we start our search at the header and at level equal to the height of the skip list. If the gap G that we are going to drop in is of size 2 or 3, we simply drop. If the gap G is of size 1, we proceed as follows. If G is not the last gap in the current level, then if the following gap G' is of size 1 we "merge" G and G' (by lowering the element separating G and G'), whereas if G' is of size 2 or 3 we

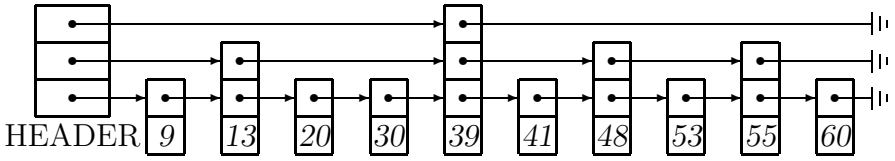


Figure 4.5: Top-down 1-3 DSL deletion

"borrow" from it (by lowering the element separating G and G' and raising the first element in G'). If G is the last gap, we merge with, or borrow from, its preceding gap. After we borrow/merge from the following/preceding gap, we then drop down a level. We continue in this way, until we reach the bottom level, where we remove the element of height 1 (if the key to be deleted is not in an element of height 1, we swap it with its predecessor of height 1, and we remove its predecessor). Since our algorithm did not allow any gaps to be of size 1, what we are left with after the removal of the element of height 1, is a valid 1-3 DSL.

As an example, consider the case of deleting *51* from the structure of Figure 4.4. We start at level 3 of the header, we move to level 3 of *48*, we look at level 2 and we lower *48* and raise *39*, we drop to level 2 of *48*, we look at level 1, we drop to level 1 of *48*, and we finally remove *51*. The resulting 1-3 DSL is shown in Figure 4.5.

Clearly, the top-down 1-3 DSL achieves logarithmic worst case search and update costs if its elements are implemented either as linked lists, or as arrays of exponential heights. Both of these implementations are simpler than their counterparts for the 1-2 DSL, because all of the work is done in one pass down through the structure. Furthermore, the linked list version of the top-down 1-3 DSL is simpler that the red-black implementation of the corresponding 2-3-4 tree and other balanced search tree algorithms; such algorithms are notoriously complicated because of the numerous cases that arise involving single and double rotations to the left and to the right.

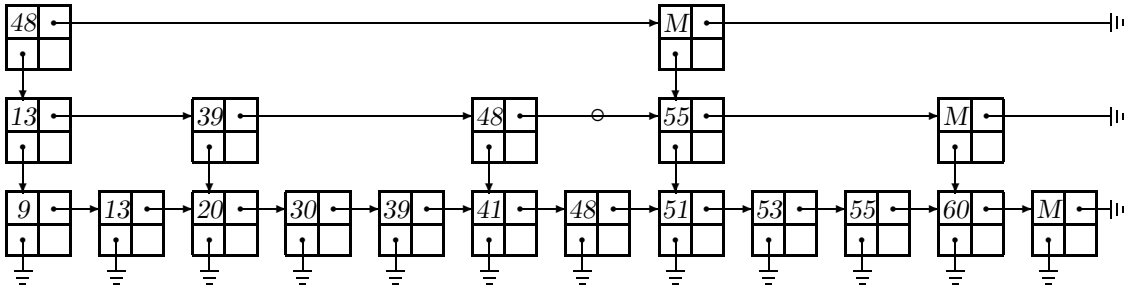


Figure 4.6: Linked list implementation of 1-3 DSL

If we chose to implement the skip list elements as linked lists, we may simplify and speed up our code by "pulling the keys back", as shown in Figure 4.6 for the 1-3 DSL of Figure 4.4 (M there stands for the maximum key). What we gain by doing so is that, during a search, instead of going right to examine the key there and to decide whether we are actually going to move there or we are going to drop down a level, we pull that key back to the *node* we are currently in. We reserve the term *linked list implementation* of a skip list for such a structure

Admittedly, this last twist of the skip list creates a structure between what one might view as a "standard" skip list (i.e. a skip list each element of which has an array of pointers) and as a binary search tree. Unlike skip lists and like binary search trees, each node has one key and two pointers, and we are now branching according to the key in the current node. On the other hand, unlike binary search trees and like skip lists, we still have "extra" pointers, that is, pointers that we are never going to follow (like the one marked with a "$\circ$" in Figure 4.6). What these extra pointers do, is that they "thread" the structure, making borrowing from a gap and merging of two gaps easier. For example, if we were going to delete *53* from the structure of Figure 4.6, the "$\circ$" pointer which is redundant before the deletion, is not redundant after the operation is complete.

Another way of looking at the linked list implementation of the skip list is the

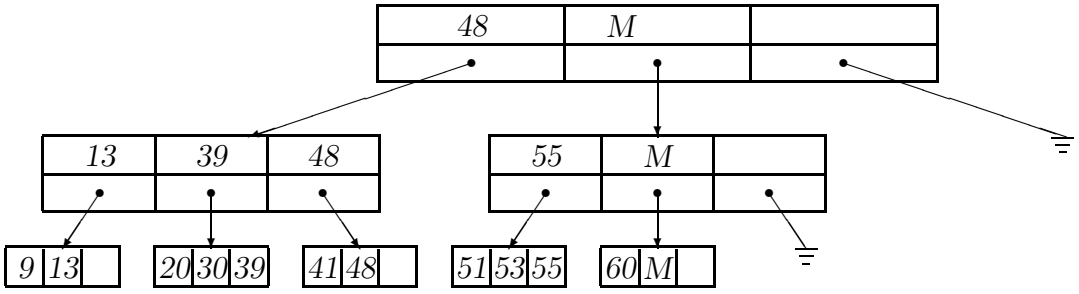


Figure 4.7: Horizontal array implementation of 1-3 DSL

following: a standard skip list is a binary search tree (having redundant pointers and duplicate keys) whose left pointers have been collapsed into array indices. This suggests that we may as well consider collapsing the right pointers of a binary search tree into array indices. The structure that we get in this way will be called *horizontal array implementation* of a skip list, and as an example, we give in Figure 4.7 the structure resulting from the skip list of Figure 4.4 (or from the search tree of Figure 4.6). This form of the skip list is very similar to the B^+ tree[1] , which is most appropriately used when the number of disk accesses, rather than the number of key comparisons, is of major concern.

Finally, we may observe that the top-down insertion and deletion are easily generalizable to any k-$(2k+1)$ skip list, for $k = 1, 2, \ldots$. When inserting a key in a 2-5, 3-7, 4-9, ... DSL, we split a gap of size 5, 7, 9, ... into two gaps of (legal) sizes 2, 3, 4, ... before we drop down a level. When deleting a key from such a DSL, we merge/borrow if we are going to drop in a gap of size 2, 3, 4, ... Obviously, searches and top-down updates have logarithmic worst case costs if we use either a linked list, or an array (of exponential size), or a horizontal array implementation.

[1]A B^+ tree is like a B tree, but with all keys placed in leaf nodes. The internal nodes contain values (not necessarily appearing as keys) used to guide the search in the following way: each value is greater than or equal to all keys contained in the subtree rooted immediately to its left.

4.4 Implementation and Experimental Results

To illustrate the simplicity of our deterministic skip lists, we now give working C code for the linked list implementation of the top-down 1-3 DSL.

To avoid having special cases in our code, we introduce a dummy `head` node, and two sentinel nodes `bottom` and `tail`. Assuming the declarations of Figure 4.8, the initialization of an empty 1-3 DSL is given in Figure 4.9. Two versions of the search code for a key `v` are given in Figure 4.10. Both versions return a pointer to the node containing `v` if the search is successful, or a pointer to `bottom` if the search is unsuccessful. Observe that the first version of our search code uses 3-way branches in each node and is identical to the search procedure in a binary search

```
#define maxkey 2147483647       /*  maxkey = (2^31)-1  */
#define newnode (nodePtr)malloc(sizeof(struct node))

typedef struct node * nodePtr;

struct node{
   int      key;
   nodePtr  r, d;
};

nodePtr  head, bottom, tail;
```

Figure 4.8: Declarations for linked list implementation of top-down 1-3 DSL

```
void EmptyList()
{
   head = newnode;
   bottom = newnode;
   tail = newnode;
   head->key = maxkey;
   head->d = bottom;
   head->r = tail;
   bottom->r = bottom;
   bottom->d = bottom;
   tail->key = maxkey;
   tail->r = tail;
}
```

Figure 4.9: Initialization in linked list implementation of top-down 1-3 DSL

```
nodePtr Search(v)
int  v;
{
  nodePtr  x;

  x = head;
  bottom->key = v;
  while (v != x->key)
    x = (v < x->key) ? x->d : x->r;
  return(x);
}
```

```
nodePtr Search(v)
int  v;
{
  nodePtr  x;

  x = head;
  while  (x != bottom)  {
    while (v > x->key)
     x = x->r;
    if  (x->d == bottom)
      return((v == x->key) ? x : bottom );
    x = x->d;
  }
}
```

Figure 4.10: Search in linked list implementation of top-down 1-3 DSL

tree, whereas the second version tests for equality only at the lowest level and it is skip list in flavour. If we choose the second version for the search code, then the insertion code shown in Figure 4.11 is just a matter of filling in a few gaps; we simply have to add a node in a (horizontal) linked link whenever necessary. Finally, the deletion code is given in Figure 4.12. Notice here that we have assumed that no duplicate keys are allowed in our structures, and that a call to `Insert` (`Delete`) is not necessarily preceded by a call to `Search`. Therefore our insertion (deletion) code for a key `v` returns `1` if the insertion (deletion) is successful and `0` is the insertion (deletion) is unsuccessful, that is, if `v` is (is not) in the structure.

As it is the case with almost every dictionary implementation (with Pugh's PSL being a noticeable exception), the `Delete` code is more complicated than the `Insert` one. This shouldn't be surprising, since, for example, not all gaps can be handled by the same piece of code; the first gap can only merge with (borrow from) its successor, whereas the last gap can only merge with (borrow from) its predecessor. This implies that we must detect at each level whether we drop at the first (or last) gap. The fact also that we may merge with (borrow from) the preceding gap has a couple of implications. First, in addition of keeping track of the

```
int Insert(v)
int  v;
{
   nodePtr  t, x;
   int      success;

   x = head;        /* x = current elm in search path */
   bottom->key = v;
   success = 1;
   while (x != bottom)  {     /* do at each level */
      while (v > x->key)   /* find where you drop */
         x = x->r;
      /* if gap_size=3, or at bottom level & must insert ==> raise middle elm */
      if  (x->key > x->d->r->r->key) {
         t = newnode;
         t->r = x->r;
         t->d = x->d->r->r;
         x->r = t;
         t->key = x->key;
         x->key = x->d->r->key;
      }
      else  if (x->d == bottom)  /* if insert_Key already in DSL */
         success = 0;
      x = x->d;
   }
   if (head->r != tail)  {   /* raise height of DSL if necessary */
      t = newnode;
      t->d = head;
      t->r = tail;
      t->key = maxkey;
      head = t;
   }
   return(success);
}
```

Figure 4.11: Insertion in linked list implementation of top-down 1-3 DSL

current gap, we also have to keep track of the preceding gap. Second, we cannot simply "merge/borrow and then drop", since we may lower the element we are currently in (for example, when we are level 3 of *48*, during the deletion of *51* from the structure of Figure 4.4); we must instead "*mark the gap to drop*, merge/borrow and then drop". Of course, none of the above is conceptually hard, but putting all these together, results in a deletion code quite longer than the insertion one.

The simplicity and performance of the given C code for the top-down 1-3 DSL was compared with implementations of other versions of our deterministic skip lists and with other competing data structures. More precisely, we considered:

```
int Delete(v)
int v;
{
   nodePtr  x, px, nx, t;
   int      pred, lastAbove, success;

   x = head->d;          /* x = current elm in search path */
   success = (x != bottom);   bottom->key = v;
   lastAbove = head->key;  /* last key at level above */
   while  (x != bottom)  {   /* do at every level */
      while  (v > x->key) {   /* find where you drop */
         px = x;   x = x->r;    /* keeping track of the previous gap */
      }
      nx = x->d;   /* mark where to drop at level below */
      /* if {only 1 elm in gap to drop}, or {at bottom level & must delete} */
      if  (x->key == nx->r->key)
         if  (x->key != lastAbove) { /*** if does NOT drop in last gap ***/
            t = x->r;
            /* if 1 elm in next gap, or at bottom level */
            if ((t->key==t->d->r->key) || (nx==bottom)) {
               x->r = t->r; /* lower separator of current+next gap */
               x->key = t->key;  free(t);
             }
             else {      /* if >=2 elms in next gap */
                x->key = t->d->key;  /* raise 1st elm in next gap & lower... */
                t->d = t->d->r;      /* ... separator of current+next gap */
             }
         }
         else                        /*** if DOES drop in last gap ***/
            if (px->key <= px->d->r->key) { /* if only 1 elm in previous gap */
               if  (nx == bottom)   /* if del_Key is in elm of height>1 */
                  pred = px->key;   /* predecessor of del_key at bottom level*/
               px->r = x->r;     /* lower separator of previous+current gap */
               px->key = x->key;   free(x);   x = px;
            }
            else  {        /* if >=2 elms in previous gap */
               /*  t = last elm in previous gap */
               t = (px->key == px->d->r->r->key ?  px->d->r : px->d->r->r);
               px->key = t->key; /* raise last elm in previous gap & lower... */
               x->d = t->r;      /* ... separator of previous+current gap */
            }
      else  if (nx == bottom)  /* if del_Key not in DSL */i
         success = 0;
      lastAbove = x->key;   x = nx;
   }  /* while */
   x = head->d;  /* Do a 2nd pass; del_key might have been in elm of height>1 */
   while  (x != bottom) {
      while  (v > x->key)   x = x->r;
      if  (v == x->key)   x->key = pred;
      x = x->d;
   }
   if  (head->d->r == tail)  {   /* lower header of DSL, if necessary */
      x = head;   head = x->d;   free(x);
   }
   return(success);
}
```

Figure 4.12: Deletion in linked list implementation of top-down 1-3 DSL

1. Pugh's implementation of the PSL. This is longer than the straightforward PSL code, since it is optimized in a couple of ways. First, the parameter p has the value .25; this reduces the number of calls to the random number generator by a factor of 16 (assuming 32-bits words), since one may strip off 2 bits from the outcome of the random number generator every time a new random number is required. Second, the PSL is never allowed to grow higher than 1 plus its current height.

2. A naive, recursive implementation of the AVL tree, based on the code in [21].

3. A highly optimized, non-recursive implementation of the AVL tree, written by J. Macropol of Contel (also used in the timings in [40]). In this code, each tree node, in addition to its 2 balance bits, has also a pointer to its parent node. Furthermore, the number of `malloc` calls is reduced by a factor of 100, since each call to `malloc` returns space sufficient for 100 tree nodes.

4. A red-black tree implementation of the top-down 2-3-4 tree, based on the code given in [42] and [24].

5. The linked list implementation of the top-down 1-3 DSL, given in Figures 4.8 to 4.12. Recall that this is another implementation of top-down 2-3-4 tree.

6. The linked list implementation of the top-down k-$(2k+1)$ DSL, for $k = 4$. This is an extension of the given 1-3 DSL code, based on storing in each node the gap size to the right of the node. Although this extra field is clearly redundant for the nodes at level 1, we chose, for simplicity, to have all nodes of the same size.

7. The horizontal array implementation of the top-down k-$(2k+1)$ DSL, for $k=2$ and for $k=6$. Each node in this structure is as shown in Figure 4.7, with the

addition of a "counter" field per node for the number of keys stored in that node.

To get a feeling about how (un)complicated the above algorithms are, we report in Table 4.1 the number of lines of their C implementations. For each algorithm, we give two numbers: the first number is referring to the various declarations, the initializations, the search and the insertion procedure; the second number is referring to all the above, plus the deletion procedure. For example, for the 1-3 DSL code given in Figures 4.8, 4.9, 4.10, 4.11 and 4.12, we need 8, 13, 10, 32 and 62 lines respectively, resulting in a total of 125 lines. Notice here that we do not count the empty or comment lines, but we do count lines containing a single "{" or "}". In all of *our* implementations we used the same coding style and layout; we adjusted other people's implementations to match our layout, but we did not modify their coding style to match ours.

As an indication of how well these algorithms may perform in practice, we considered the space and time requirements for n random insertions in an initially empty structure. The experiments were run on a MIPS M2000 machine, running UMIPS 4.51 Unix, during a period of very low work-load. The given CPU times include both "user" and "system" times. As far as the space requirements are concerned, UMIPS 4.51 is identical to 4.3BSD Unix, an environment many of us may find ourselves into. This version of Unix is a (binary) buddy memory allocation system, in which requests for memory of size up to 508 bytes result in the allocation of a number of bytes equal to the smallest power of 2 adequate for 4 bytes (used for administrative purposes) more than the requested space. For example, a call for 5, 12, 13, or 29 bytes will return 16, 16, 32 and 64 bytes respectively. In our system, the pointers and our integer keys were 4 bytes each. We can now see why, for example, an AVL tree of 1,000,000 keys took 32M of space. Each node, containing 1 key, 2

n	PSL	AVL (naive)	AVL (opt.)	red-black	l.l. 1-3	l.l. 4-9	h.a. 2-5	h.a. 6-13
100	.0015	.0025	.0016	.0025	.0018	.0018	.0015	.0014
250	.0041	.0068	.0042	.0068	.0049	.0048	.0038	.0039
500	.0088	.014	.0088	.014	.010	.011	.0082	.0080
750	.014	.023	.014	.023	.016	.016	.013	.013
1,000	.019	.032	.019	.029	.022	.021	.017	.017
2,500	.052	.091	.049	.087	.061	.061	.047	.047
5,000	.12	.20	.11	.19	.14	.14	.098	.096
7,500	.20	.32	.17	.30	.24	.22	.16	.15
10,000	.28	.46	.23	.43	.35	.32	.22	.21
25,000	.98	1.3	.73	1.3	1.2	1.1	.65	.60
50,000	2.3	2.9	1.7	3.2	3.0	2.4	1.4	1.3
75,000	3.8	4.7	2.7	5.1	5.1	4.1	2.3	2.1
100,000	5.5	6.6	4.0	7.4	7.4	5.6	3.2	3.0
250,000	16.0	18.1	11.6	19.8	22.4	17.3	9.4	8.3
500,000	38.7	42.7	25.4	47.5	53.3	48.5	20.6	18.2
750,000	73.3	79.7	45.1	94.8	119.7	103.5	35.7	29.1
1,000,000	124.2	117.2	67.0	135.6	176.4	163.7	51.0	41.8
C lines	95/120	108/166	248/316	96/171	63/125	82/153	84/167	
`malloc`'s	n	n	$.01n$	n	$1.570n$	$1.167n$	$.313n$	$.113n$
bytes	$20n$	$32n$	$32n$	$32n$	$25n$	$37n$	$20n$	$14n$

Table 4.1: Timings for n random insertions

pointers and (normally) 2 balance bits, requires more than 12 bytes, and it will thus be allocated 32 bytes; the addition of an extra "parent" pointer will not alter the space requirements at all. Similarly, each node in the linked list implementation of a DSL with and without counters will be allocated 32 and 16 bytes respectively. The reader should now be able to verify that, given the number of performed `malloc`'s in Table 4.1, the space requirements for all considered structures (except for the PSL) are indeed as stated.

We would like to emphasize at this point that Table 4.1 does not tell us which algorithm takes less space, or runs faster, or even is simpler; it just gives us a very rough idea. The figures of Table 4.1 should be only interpreted as an indication that our new proposed algorithms are indeed competitive with existing schemes; whether or not they are the best choice for a particular application will depend on

the specific needs of the application (average/worst space/time minimization), on the particular system they will run on, and on coding details.

To illustrate the significance of coding details, consider the change of the

```
if (x->key > x->d->r->r->key)
```

line in our insertion code of Figure 4.11 into the equivalent

```
if (x->key == x->d->r->r->r->key)
```

line. Our timings revealed that such a change results in an average slow-down of about 20%, due to the extra pointer that we have to follow in the second case.

Other coding optimizations may be pertinent to the particular system the DSL programs are running on. For example, the array implementation of any DSL is especially suited for (binary) buddy systems, like 4.3BSD Unix. We can tune our structure by choosing the physical heights of its elements to be such that the entire element will fit exactly into the space allocated by the memory manager. For example, if our keys and pointers are 4 bytes each, and if we also store the logical height of each DSL element in the element itself, then we should choose elements of physical heights $2^k - 3$, for $k = 2, 3, \ldots$. By doing so, the array implementation of any DSL does not require more space than its naive counterpart described in Section 4.1, and it improves the $\Theta(\lg^2 n)$ worst case update cost of the latter to the desired $\Theta(\lg n)$. It's also important to observe here that such a fine tuning of our structure will not affect more than a handful of lines in our code; all we need is a function "`int PH(k)`", computing the physical height of an element of logical height `k`, since, for example, we can then write

```
if (PH(elm->h) == elm->h)
   newNodeOfHeight(PH(elm->h+1)}
```

to raise an element if necessary.

It is also worth noting that the array implementation of our bottom-up DSLs described in Section 4.2, can be streamlined somewhat. If the $\Theta(\lg \lg n)$ (`malloc`) calls to the storage manager are of major concern, we can sidestep most of them by simply keeping, along with the structure, $\Theta(\lg \lg n)$ free element nodes, one of each physical size. When an insertion takes place, we initially request a free element of the smallest physical size from our set of free nodes, and if some elements have to grow, we repeatedly request a new element one size larger and we release the old one. This leaves our set of free nodes with one node of each size except for the final node requested. A single call now to the memory manager rectifies the situation.

The $\Theta(\lg \lg n)$ calls to the storage manager can also be avoided if we choose to raise the elements' heights top-down, rather than bottom-up. We can do so, if, during our search, we record the gap we are dropping into. We can then determine which element has to grow to which height. Suppose that we find out that the highest element that has to grow is of height k and that it has to grow to height $k+j$. Then, the next lowest element that will grow, will grow to height exactly k. Thus, we can make only one call to the memory manager for an element of height $k+j$, since we may subsequently reuse the freed element of height k for the new element to grow to height k, etc.

All the above optimizations are by no means must's or even necessarily worth doing. We simply wanted to highlight certain points in our algorithms, which, if fine-tuned properly and according to the system used, they may lead to significant improvements, both in space and in running time.

4.5 Summary

In this chapter, we introduced several versions of deterministic skip lists, simple data structures, based on the notion of Pugh's skip list, but with *guaranteed* logarithmic search and update costs.

We defined k-$2k$ and k-$(2k{+}1)$ DSLs, for $k = 1, 2, 3, \ldots$, and we presented bottom-up algorithms for both of them. In particular for the k-$(2k+1)$ DSLs, top-down update algorithms also exist, and they are simpler and faster than their bottom-up counterparts. The logarithmic worst case costs are achieved by using either an array, or a linked list, or a horizontal array implementation. The linked list implementation is a binary search tree with special properties, combining features of both the standard binary search tree and Pugh's original skip list. The linked list implementation may also be viewed as the starting point for deriving the array implementation (or Pugh's PSL) on one hand, and the horizontal array implementation (or the B^+ tree) on the other hand. Finally, our experimental results showed that our algorithms compare favorably in terms of simplicity, space requirements and running time with other balanced search tree schemes.

Chapter 5

Probabilistic Skip Lists Revisited

The derivation of the PSL average search cost in Chapter 3 was based on three key remarks. First, the search cost can be split into the height of the PSL and the number of horizontal steps; second, the average height and the average number of horizontal steps are related; and third, the height is related to some well-studied quantities of tries. Unfortunately, this method cannot be extended (easily) to the computation of the variance of the search cost, for a couple of reasons. First, there is no (obvious) relationship between the variance of the height and the variance of the number of horizontal steps; and second, the height and the number of horizontal steps are *not* independent random variables, and therefore, their covariance should be computed as well.

In this chapter, we consider the variance of the search cost for $+\infty$ in a PSL, and we follow a totally different approach towards its derivation. By using probability generating functions, exponential generating functions, as well as binomial, Poisson and Mellin transforms, and by applying Rice's method, we obtain exact and asymptotic expressions for this variance under the fixed population model (that is, when

the PSL has n keys), and an asymptotic expression for the same variance under the Poisson model (that is, when the PSL has a variable number of keys following the Poisson distribution). In the course of our work, we also derive an asymptotic expression for the expected value of the search cost for $+\infty$ under the Poisson model. This chapter is after Poblete's preliminary results [38] for the non-oscillatory part of the asymptotic costs under the Poisson model.

5.1 Notation and Related Results

The notation used in this chapter is that of Chapter 3, stated in Section 3.2; the reader is thus advised to review it, before proceeding further.

Several attempts have been made to show that the search cost "usually" stays close to its expected value. Pugh showed [39] that

$$C_n^{(m)} \preceq Y_n^{(m)}, \tag{5.1}$$

where "$\preceq$" means *stochastically smaller*,[1] and $Y_n^{(m)}$ is a random variable he defined precisely. As an immediate consequence of his expression for $Y_n^{(m)}$, Pugh derived

$$\mathbf{E}(Y_n^{(m)}) = \begin{cases} \log_{\frac{1}{p}} n + \dfrac{q}{p}\log_{\frac{1}{p}}(m-1) + \dfrac{1}{q} + 1 & \text{for } m = 2, 3, \ldots, \text{n+1} \\ \log_{\frac{1}{p}} n + \dfrac{1}{q} & \text{for } m = 1 \end{cases} \tag{5.2}$$

and

$$\mathbf{Var}(Y_n^{(m)}) = \begin{cases} \dfrac{q}{p^2}\log_{\frac{1}{p}}(m-1) + \dfrac{p}{q^2} + \dfrac{1}{p} - \dfrac{1}{p^2}\left(q + \dfrac{1}{m-1}\right) & \text{for } m = 2, 3, \ldots, n+1 \\ \dfrac{p}{q^2} & \text{for } m = 1 \end{cases} \tag{5.3}$$

[1] Given two random variables X and Y, we have $X \preceq Y$ if and only if $\mathbf{Pr}[X \geq t] \leq \mathbf{Pr}[Y \geq t]$ for all t.

Although stochastical ordering of two random variables implies ordering of their expected values (and this is precisely how Pugh derived, from (5.1) and (5.2), his bound (3.2) of $\mathbf{E}(C_n^{(m)})$), it does *not* imply ordering of their variances.[2] Therefore, (5.1) does *not* imply that the expression in (5.3) is an upper bound of $\mathbf{Var}(C_n^{(m)})$. However, we may use (5.3) to bound the probability that the search cost will exceed a certain value, since for any $\epsilon > 0$ we have

$$\begin{aligned}
\mathbf{Pr}[C_n^{(m)} \geq \mathbf{E}(Y_n^{(m)}) + \epsilon] &\leq \mathbf{Pr}[Y_n^{(m)} \geq \mathbf{E}(Y_n^{(m)}) + \epsilon] && \text{by (5.1)} \\
&\leq \mathbf{Pr}[\,|Y_n^{(m)} - \mathbf{E}(Y_n^{(m)})| \geq \epsilon] \\
&\leq \frac{\mathbf{Var}(Y_n^{(m)})}{\epsilon^2} && \text{by Chebyshev's inequality}
\end{aligned}$$

So, for example, for the search cost for $+\infty$ we get (for $n \geq 2$)

$$\mathbf{Pr}\left[F_n \geq \frac{1}{p}\log_{\frac{1}{p}} n + \frac{1}{q} + 1 + \epsilon\right] \leq \frac{1}{\epsilon^2}\left(\frac{q}{p^2}\log_{\frac{1}{p}} n + \frac{p}{q^2} + \frac{1}{p} - \frac{1}{p^2}\left(q + \frac{1}{n}\right)\right), \quad \text{for any } \epsilon > 0.$$

Sen also showed [43] that

$$\mathbf{Pr}[C_n^{(m)} \geq c \log_{\frac{1}{p}} n] = O\left(\frac{1}{n^2}\right), \qquad \text{for some constant } c.$$

Finally, Devroye proved [12, 13] the more refined results

$$\frac{C_n^{(m)} - \log_{\frac{1}{p}} n + \log_{\frac{1}{p}}(m-1)}{\log_{\frac{1}{p}}(m-1)} \xrightarrow{\mathcal{P}} \frac{1}{p}$$

and

$$\frac{C_n^{(m)} - \log_{\frac{1}{p}} n - \frac{q}{p}\log_{\frac{1}{p}}(m-1)}{\sqrt{\frac{q}{p^2}\log_{\frac{1}{p}}(m-1)}} \xrightarrow{\mathcal{D}} \mathcal{N}(0,1),$$

where $\xrightarrow{\mathcal{P}}$ and $\xrightarrow{\mathcal{D}}$ denote convergence in probability and in distribution respectively, and $\mathcal{N}(0,1)$ is the standard normal distribution. These results suggest that $\mathbf{Var}(C_n^{(m)}) \sim \frac{q}{p^2}\log_{\frac{1}{p}} m$.

[2] If $\mathbf{Pr}[X=1] = \mathbf{Pr}[X=5] = .5$ and $\mathbf{Pr}[Y=4] = \mathbf{Pr}[Y=6] = .5$, then we have $X \preceq Y$, but $4 = \mathbf{Var}(X) > \mathbf{Var}(Y) = 1$.

In this chapter, we derive an exact expression for $\mathbf{Var}(C_n^{(n+1)}) = \mathbf{Var}(F_n)$, and asymptotic expressions for $\mathbf{Var}(F_n)$ and $\mathbf{Var}\Big(F(z)\Big)$, where $F(z)$ is the Poissonized version of F_n with mean z. On our way there, we also derive an asymptotic expression for $\mathbf{E}\Big(F(z)\Big)$.

As stated in the beginning of Chapter 2, the understanding of our *results* requires only the knowledge of Sections 2.1 and 2.2, but the understanding of the *derivations* of our results presumes the knowledge of the entire Section 2.3. For brevity, in our proofs we will use the notation

$$a_n \stackrel{\text{def}}{=} \mathbf{E}(F_n) \qquad \text{and} \qquad b_n \stackrel{\text{def}}{=} \mathbf{E}\Big(F_n(F_n-1)\Big)$$

and

$$A(z) \stackrel{\text{def}}{=} \mathbf{E}\Big(F(z)\Big) = \mathcal{P}(a_n;\, z) \quad \text{and} \quad B(z) \stackrel{\text{def}}{=} \mathbf{E}\Big(F(z)(F(z)-1)\Big) = \mathcal{P}(b_n;\, z);$$

notice that the last two equalities follow from Lemma 2.1.

5.2 The Set-up

To derive moments of F_n, we will use a fairly standard — in principle — method: we will consider the pgf of F_n

$$P_n(z) \stackrel{\text{def}}{=} \sum_{k\geq 0} \mathbf{Pr}[F_n = k] z^k, \qquad \text{integer } n \geq 0,$$

we will obtain a recurrence for it, and will differentiate the latter at $z=1$.

Lemma 5.1 *Under the fixed population model, the pgf of F_n satisfies the recurrence*

$$P_n(z) = (1-z)\delta_{n,0} + \sum_{\substack{l+j+k=n \\ l,j,k\geq 0}} \left\{ \binom{l+j-1}{j} p^l q^{j+k} z^{k+1} P_l(z) \right\}, \quad \textit{for } n \geq 0.$$

Proof: Clearly, for any PSL with at least 1 element, the search cost for $+\infty$ is ≥ 2, that is,

$$\begin{cases} \mathbf{Pr}[F_0=0]=1 \\ \mathbf{Pr}[F_n=0]=\mathbf{Pr}[F_n=1]=0 \quad \text{for } n\geq 1 \end{cases} . \tag{5.4}$$

F_n takes a value r ($r\geq 2$) in the following two cases. First, if the PSL has exactly $r-1$ elements, all of height 1. Second, if the PSL has at least one element of height ≥ 2, and some additional conditions hold. Let R be the rightmost element of height ≥ 2 in this case. R splits the elements of the PSL into a group of k (≥ 0) elements of height 1 to its right, and a group of $n-k-1$ elements of various heights to its left. Suppose that l (≥ 0) of these $n-k-1$ elements are of height ≥ 2, and consider furthermore the PSL S' resulting from the given PSL S by removing level 1 from all n elements of S. Clearly, S' has $l+1$ elements, and the search cost for $+\infty$ in it will be $r-k-1$ if and only if $F_n=r$. (For example, for the PSL of Figure 3.1, we have $F_9=9$, $k=0$ and $l=5$.) Putting now all these together, we get

$$\mathbf{Pr}[F_n=r]=q^n\delta_{n,r-1}+\sum_{\substack{l+j+k=n-1\\ l,j,k\geq 0}}\left\{\binom{l+j}{j}p^lq^jpq^k\mathbf{Pr}[F_{l+1}=r-k-1]\right\}, \text{for } n\geq 1,\ r\geq 2,$$

and if we multiply both sides of this by z^r, and add the resulting equalities for all $r\geq 2$, we get

$$P_n(z)=q^nz^{n+1}+\sum_{\substack{l+j+k=n-1\\ l,j,k\geq 0}}\left\{\binom{l+j}{j}p^{l+1}q^{j+k}z^{k+1}P_{l+1}(z)\right\}, \quad \text{for } n\geq 1$$

by using the initial conditions (5.4), or

$$P_n(z)=q^nz^{n+1}+(1-z)\delta_{n,0}+\sum_{\substack{l+j+k=n-1\\ l,j,k\geq 0}}\left\{\binom{l+j}{j}p^{l+1}q^{j+k}z^{k+1}P_{l+1}(z)\right\}, \quad \text{for } n\geq 0$$

since $P_0(z)=1$ from (5.4). The result now follows, if we observe that for $l=-1$ the sum in the last expression reduces to q^nz^{n+1}, since its only non-zero term occurs when $j=0$. ■

5.3 Expected Values

Before we plunge into the derivation of the variance of the search cost, it is instructive to look at the derivation of the expected value of it. In both cases, we are going to perform the same steps and use the same techniques, but — as expected — the derivation of the expected value will be simpler and easier to understand.

Lemma 5.2 *Under the fixed population model, the expected cost for the search of* $+\infty$ *in a PSL of* n *keys satisfies the recurrence*

$$\begin{cases} a_0 = 0 \\ a_n = \dfrac{1-q^{n+1}}{p} - \delta_{n,0} + \displaystyle\sum_{k=0}^{n} \left\{ \binom{n}{k} p^k q^{n-k} a_k \right\}, \quad \textit{for } n \geq 0 \end{cases} . \tag{5.5}$$

Proof: If we differentiate the recurrence of Lemma 5.1 with respect to z, and we evaluate the resulting expression at $z=1$, we get

$$a_n = -\delta_{n,0} + \sum_{\substack{l+j+k=n \\ l,j,k\geq 0}} \left\{ \binom{l+j-1}{j} p^l q^{j+k} (a_l+k+1) \right\}, \qquad \text{for } n \geq 0,$$

if we recall that $P_r(1)=1$ from (2.6), and $a_r = P_r'(1)$ from (2.7). We now simplify the sum in the last expression, by splitting it into two sums. The first sum is

$$\sum_{\substack{l+j+k=n \\ l,j,k\geq 0}} \left\{ \binom{l+j-1}{j} p^l q^{j+k} a_l \right\} = \sum_{l=0}^{n} p^l q^{n-l} a_l \sum_{j=0}^{n-l} \binom{l-1+j}{j},$$

which gives

$$\sum_{\substack{l+j+k=n \\ l,j,k\geq 0}} \left\{ \binom{l+j-1}{j} p^l q^{j+k} a_l \right\} = \sum_{l=0}^{n} \binom{n}{l} p^l q^{n-l} a_l \tag{5.6}$$

by applying (2.14) in the inner sum. The second sum is

$$\sum_{\substack{l+j+k=n \\ l,j,k\geq 0}} \left\{ \binom{l+j-1}{j} p^l q^{j+k} (k+1) \right\} = \sum_{l=0}^{n} p^l q^{n-l} \sum_{j=0}^{n-l} \binom{l+j-1}{j} (n+1-l-j),$$

and since the inner sum in this case is

$$\begin{aligned}\sum_{j=0}^{n-l}\binom{l-1+j}{j}(n+1-l-j) &= (n+1)\sum_{j=0}^{n-l}\binom{l-1+j}{j} - \sum_{j=0}^{n-l}\binom{l+j-1}{l-1}(l+j)\\ &= (n+1)\binom{n}{l} - l\sum_{j=0}^{n-l}\binom{l+j}{l} \qquad \text{by (2.14)}\\ &= (l+1)\binom{n+1}{l+1} - l\binom{n+1}{l+1}, \qquad \text{by (2.14)}\\ &= \binom{n+1}{l+1}, \qquad (5.7)\end{aligned}$$

the second sum becomes

$$\sum_{\substack{l+j+k=n\\ l,j,k\geq 0}} \left\{\binom{l+j-1}{j}p^l q^{j+k}(k+1)\right\} = \frac{1-q^{n+1}}{p}$$

by applying the binomial theorem. The result now follows. ■

Recurrences similar to the one of the last lemma have appeared repeatedly in the analyses of trie-related algorithms. Knuth, for example, considers the recurrence

$$\begin{cases} x_0 = x_1 = 0\\ x_n = c_n + \displaystyle\sum_{k=0}^{n}\binom{n}{k}p^k q^{n-k}(x_k + x_{n-k}), \quad \text{for } n\geq 2\end{cases}$$

in the context of radix exchange sort, and the recurrence

$$\begin{cases} x_0 = x_1 = 0\\ x_n = c_n + m^{1-n}\displaystyle\sum_{k=0}^{n}\binom{n}{k}(m-1)^{n-k}x_k, \quad \text{for } n\geq 2\end{cases}$$

in the context of m-ary tries. He solves both of them exactly for arbitrary $\langle c_n\rangle$ [26, Ex. 5.2.2.53 and Ex. 6.3.17] by using binomial transforms, and he derives asymptotic expressions for the first recurrence when, for example, $c_n = (n-1)pq$ [26, Ex. 5.2.2.53] (x_n is the average number of key exchanges), and for the second recurrence [26, Ex. 6.3.18, Ex. 6.3.19 and Ex. 6.3.31] when $c_n = 1$ (x_n is the average

number of nodes in an m-ary trie), when $c_n = n$ (x_n is the average external path length in an m-ary trie), and when $c_n = n(1-2^{1-n})$ and $m = 2$ (x_n is the average external path length in a Patricia trie). Szpankowski considers [47, Theorem 1] a generalization of the recurrence

$$\begin{cases} x_0,\, x_1 = \text{arbitrary} \\ x_n = c_n + \displaystyle\sum_{k=0}^{n} \binom{n}{k} p^k q^{n-k} x_k, \quad \text{for } n \geq 2 \end{cases},$$

which appears in a variety of algorithms; he solves this recurrence exactly for arbitrary $\langle c_n \rangle$, and asymptotically when c_n is a linear combination of $\binom{n}{r} v^n$.

Our recurrence (5.5) is precisely in the form of the last recurrence. This should come as no surprise, since, as discussed in Chapter 3, a_n is a well-studied quantity appearing in the analyses of tries. Although we already have an exact expression for a_n (from Theorem 3.1), and/or we could simply quote an equivalent expression for it from Szpankowski's solution, we choose to re-solve (5.5) exactly, since half of the work required for it has to be done anyhow towards the derivation of the average Poissonized cost.

Consider the generalization of (5.5)

$$\begin{cases} a_0 = 0 \\ a_n = c_n + \displaystyle\sum_{k=0}^{n} \left\{ \binom{n}{k} p^k q^{n-k} a_k \right\}, \quad \text{for } n \geq 0 \end{cases}, \tag{5.8}$$

where $\langle c_n \rangle$ is an arbitrary sequence. Let $A(z) = \mathcal{P}(a_n; z)$ be the Poisson transform of $\langle a_n \rangle$. By definition, we have

$$\mathcal{P}\left(\sum_{k=0}^{n} \left\{ \binom{n}{k} p^k q^{n-k} a_k \right\};\, z \right) = e^{-z} \sum_{n \geq 0} \left(\sum_{k=0}^{n} \binom{n}{k} (p^k a_k) q^{n-k} \right) \frac{z^n}{n!},$$

and since $\displaystyle\sum_{j \geq 0} a_j p^j \frac{z^j}{j!} = e^{pz} A(pz)$ and $\displaystyle\sum_{j \geq 0} q^j \frac{z^j}{j!} = e^{qz}$, the last equality implies

$$\mathcal{P}\left(\sum_{k=0}^{n} \left\{ \binom{n}{k} p^k q^{n-k} a_k \right\};\, z \right) = A(pz), \tag{5.9}$$

according to the definition of the binomial convolution in (2.37). Therefore, if we take the Poisson transform of both sides of (5.8), we get

$$A(z) - A(pz) = \mathcal{P}(c_n;\, z). \tag{5.10}$$

This equation will be the starting point for the derivation of an *exact expression for* $a_n = \mathbf{E}(F_n)$ and an *asymptotic expression for* $A(z) = \mathbf{E}\Big(F(z)\Big)$. We first work towards that derivation of a_n.

Lemma 5.3 *The exact solution of recurrence (5.8) is*

$$a_n = \sum_{k=1}^{n} \binom{n}{k} (-1)^k \frac{\widehat{c_k}}{1-p^k}, \qquad \textit{integer } n \geq 0,$$

where $\widehat{c_k}$ *is the binomial transform of* c_k.

Proof: Let $\widehat{E}(z) \stackrel{\text{def}}{=} \sum_{k\geq 0} \widehat{c_n} \dfrac{z^n}{n!}$ be the egf of $\langle \widehat{c_n} \rangle$. We rewrite (5.10) as

$$A(z) - A(pz) = \widehat{E}(-z) \tag{5.11}$$

by using Lemma 2.2. Since

$$\begin{aligned} A(z) &= e^{-z} \sum_{n\geq 0} a_n \frac{z^n}{n!} \\ &= \sum_{k\geq 0} (-1)^k \frac{z^k}{k!} \sum_{n\geq 0} a_n \frac{z^n}{n!} \\ &= \sum_{n\geq 0} \sum_{k=0}^{n} \binom{n}{k} (-1)^{n-k} a_k \frac{z^n}{n!}, \qquad \text{by (2.37)} \end{aligned}$$

if we extract the coefficients of $\frac{z^n}{n!}$ from both sides of (5.11), we get

$$\sum_{k=0}^{n} \binom{n}{k} (-1)^{n-k} a_k - p^n \sum_{k=0}^{n} \binom{n}{k} (-1)^{n-k} a_k = \widehat{c_n} (-1)^n, \qquad \text{integer } n \geq 0.$$

We now rearrange the terms of the last as

$$\sum_{k=0}^{n} \binom{n}{k} (-1)^k a_k = \frac{\widehat{c_n}}{1-p^n}, \qquad \text{integer } n \geq 1,$$

and the result follows from property (2.42). ■

As an application of the last lemma, we may re-derive the exact value of $\mathbf{E}(F_n)$. Indeed, for

$$c_n = \frac{1-q^{n+1}}{p} - \delta_{n,0} \tag{5.12}$$

we have

$$\begin{aligned}
\widehat{c}_k &= \sum_{j=0}^{k} \binom{k}{j} (-1)^j \left(\frac{1}{p} - \frac{q}{p} q^j - \delta_{j,0} \right) && \text{by dfn. of binomial transform} \\
&= 0 - \frac{q}{p} \sum_{j=0}^{k} \left\{ \binom{k}{j} (-q)^j \right\} - \binom{k}{0} (-1)^0 && \text{by binomial theorem} \\
&= -\frac{q}{p} p^k - 1, && \text{by binomial theorem \& } 1-q=p
\end{aligned}$$

and therefore, Lemmata 5.2 and 5.3 yield

$$\mathbf{E}(F_n) = a_n = \sum_{k=1}^{n} \left\{ \binom{n}{k} (-1)^k \frac{-qp^{k-1}-1}{1-p^k} \right\}. \tag{5.13}$$

It can of course be verified that this expression is equivalent to the $\frac{1}{p}V_p(n+1)+1$ of Theorem 3.1.

We now go back to equation (5.10) as promised; we will work towards the derivation of $\mathbf{E}\Big(F(x)\Big)$. Since, in our case, c_n is given by (5.12), we can easily verify that (5.10) becomes

$$A(z) - A(pz) = -\left(e^{-z}-1\right) - \frac{q}{p}\left(e^{-pz}-1\right). \tag{5.14}$$

Theorem 5.1 *Under the Poisson model with mean z, the expected cost for the search of $+\infty$ in a PSL is*

$$\mathbf{E}\Big(F(z)\Big) = \frac{1}{p}\log_{\frac{1}{p}} z - \frac{1}{p}\left(\frac{\gamma}{\ln p} + \frac{1}{2}\right) + 1 - \frac{1}{p} f_{0,\frac{1}{p}}(z) + O\left(\frac{1}{z^d}\right) \quad \textit{as } z \to \infty, \quad \textit{for any } d > 0,$$

where the defined in (2.3) $f_{0,\frac{1}{p}}(z)$ is a periodic function of $\log_{\frac{1}{p}} z$, with small magnitude. ■

Proof: We are taking the Mellin transforms of both sides of (5.14). Property (2.45), together with the special case

$$\mathcal{M}(e^{-z}-1;\ s)=\Gamma(s), \qquad \text{for} \quad -1<\Re(s)<0$$

of the Cauchy-Saalschütz identity (2.44), and

$$\mathcal{M}(e^{-pz}-1;\ s)=\frac{1}{p^s}\Gamma(s), \quad \text{for} \ -1<\Re(s)<0, \tag{5.15}$$

which follows from the previous and (2.45), give

$$A^*(s)-p^{-s}A^*(s)=-\Gamma(s)-\frac{q}{p^{s+1}}\Gamma(s), \qquad \text{for} \ -1<\Re(s)<0,$$

where $A^*(s)$ is the Mellin transform of $A(z)$. The last can be rearranged as

$$A^*(s)=-\frac{1+qp^{-s-1}}{1-p^{-s}}\Gamma(s), \qquad \text{for} \ -1<\Re(s)<0, \tag{5.16}$$

and the right-hand side of this is the Mellin transform that we inverted in Example 2.1. This completes the proof. ■

A quick look at Theorems 3.3 and 5.1 reveals that the asymptotic expected costs under the fixed population model and under the Poisson model are the same. Although — to the best of our knowledge — this is indeed the case for all algorithms studied so far under the two models, one should be cautious, since the above is not always true. One may easily find algorithms for which the expected values under the two models are not even of the same order; if, for example, X_n denotes the space complexity for writing down a truth table, then $\mathbf{E}(X_n)=2^n$, but $\mathbf{E}\Big(X(z)\Big)=e^z$ from Lemma 2.1.

5.4 Variances

In order to derive the variance of the search cost, we first derive the second factorial moment of it. The derivation of the latter is done in a manner completely analo-

gous to the derivation of the expected value in the preceding section: we obtain a recurrence for the second factorial moment, and we subsequently take its Poisson transform, from which we get exact and asymptotic values.

Lemma 5.4 *Under the fixed population model, the second factorial moment of the search cost for $+\infty$ in a PSL of n keys satisfies the recurrence*

$$\begin{cases} b_0 = 0 \\ b_n = -\dfrac{2q^{n+1}}{p}n + \dfrac{2q(1-q^n)}{p^2} + 2\displaystyle\sum_{k=0}^{n}\left\{\binom{n+1}{k+1}p^k q^{n-k}a_k\right\} + \sum_{k=0}^{n}\left\{\binom{n}{k}p^k q^{n-k}b_k\right\}, \text{ for } n \geq 0 \end{cases} \tag{5.17}$$

Proof: If we differentiate the recurrence of Lemma 5.1 twice with respect to z, and we evaluate the resulting expression at $z=1$, we get

$$b_n = \sum_{\substack{l+j+k=n \\ l,j,k\geq 0}} \left\{\binom{l+j-1}{j}p^l q^{j+k}\{k(k+1) + 2(k+1)a_l + b_l\}\right\}, \quad \text{for } n \geq 0,$$

if we recall that $P_r(1) = 1$ from (2.6), $a_r = P_r'(1)$ from (2.7), and $b_r = P_r''(1)$ from (2.8). The result now follows by splitting the sum in the last expression into three sums, and by simplifying each of them separately. The first sum becomes

$$\begin{aligned} &\sum_{\substack{l+j+k=n \\ l,j,k\geq 0}} \left\{\binom{l+j-1}{j}p^l q^{j+k}k(k+1)\right\} \\ &= pq^{n-1}\sum_{k=0}^{n} k(k+1)\sum_{l=0}^{n-k}\binom{n-k-1}{n-k-l}\left(\frac{p}{q}\right)^{l-1} \\ &= pq^{n-1}n(n+1)\binom{-1}{0}\left(\frac{p}{q}\right)^{-1} && \text{separate ``}k=n\text{'' term} \\ &\quad + pq^{n-1}\sum_{k=0}^{n-1} k(k+1)\sum_{l=0}^{n-k}\binom{n-k-1}{l-1}\left(\frac{p}{q}\right)^{l-1} && \text{by } \tbinom{r}{s} = \tbinom{r}{r-s} \text{ for } r \geq 0 \\ &= n(n+1)q^n + pq^{n-1}\sum_{k=0}^{n-1} k(k+1)\left(1+\frac{p}{q}\right)^{n-k-1} && \text{by binomial theorem} \end{aligned}$$

$$
\begin{aligned}
&= n(n+1)q^n + p\sum_{k=0}^{n-1} k(k+1)q^k && \text{by } p+q=1\\
&= n(n+1)q^n + \left(-q^n n^2 - \frac{1+q}{p}q^n n + \frac{2q(1-q^n)}{p^2}\right) && \text{by geometric series}\\
&= -\frac{2q^{n+1}}{p}\,n + \frac{2q(1-q^n)}{p^2},
\end{aligned}
$$

the second sum (divided by 2) becomes

$$
\begin{aligned}
&\sum_{\substack{l+j+k=n\\ l,j,k\geq 0}} \left\{ \binom{l+j-1}{j} p^l q^{j+k}(k+1)a_l \right\}\\
&\quad = \sum_{l=0}^{n} p^l q^{n-l} a_l \sum_{j=0}^{n-l} \binom{l-1+j}{j}(n+1-l-j)\\
&\quad = \sum_{l=0}^{n} \binom{n+1}{l+1} p^l q^{n-l} a_l, && \text{by (5.7)}
\end{aligned}
$$

and the third sum is

$$
\sum_{\substack{l+j+k=n\\ l,j,k\geq 0}} \left\{ \binom{l+j-1}{j} p^l q^{j+k} b_l \right\} = \sum_{k=0}^{n} \binom{n}{k} p^k q^{n-k} b_k,
$$

as we have already shown in (5.6). ■

We may now use Lemma 5.3 to derive an exact expression for $b_n = \mathbf{E}\Big(F_n(F_n-1)\Big)$; $\mathbf{Var}(F_n)$ will follow immediately.

Theorem 5.2 *Under the fixed population model, the second factorial moment of the search cost for $+\infty$ in a PSL of n keys is*

$$
\begin{aligned}
\mathbf{E}\Big(F_n(F_n-1)\Big) &= \frac{2q}{p^2}\sum_{k=1}^{n}\left\{\binom{n}{k}(-1)^k \frac{p^k}{1-p^k}\,(k-1)\right\}\\
&\quad - \frac{2}{p}\sum_{k=1}^{n}\left\{\binom{n}{k}(-1)^k \frac{p^k}{(1-p^k)^2}\,(1+qp^{k-1})\right\}\\
&\quad + \frac{2q}{p^2}\sum_{k=1}^{n}\left\{\binom{n}{k}(-1)^k \frac{p^k}{1-p^k}\sum_{j=1}^{k}\frac{p^j}{1-p^j}\right\}.
\end{aligned}
$$

Proof: Lemma 5.3 gives us

$$b_n = \sum_{k=1}^{n} \binom{n}{k} (-1)^k \frac{\widehat{c_k}}{1-p^k}, \qquad \text{integer } n \geq 0 \tag{5.18}$$

as the solution of the recurrence of Lemma 5.4, where $\widehat{c_k}$ is the binomial transform of c_k, that is,

$$\widehat{c_k} = \sum_{r=0}^{k} \left\{ \binom{k}{r} (-1)^r \left(-\frac{2q^{r+1}}{p} r + \frac{2q(1-q^r)}{p^2} + 2 \sum_{l=0}^{r} \left\{ \binom{r+1}{l+1} p^l q^{r-l} a_l \right\} \right) \right\}, \text{ for } k \geq 0.$$

We now proceed by splitting the last sum into three sums in the obvious way. The first sum is

$$\frac{2q^2}{p} \sum_{r=0}^{k} \left\{ \binom{k}{r} r (-1)^{r-1} q^{r-1} \right\} = 2q^2 k p^{k-2}, \tag{5.19}$$

since differentiation of the binomial theorem (2.13) with respect to x, when $y=1$, gives

$$\sum_{r=0}^{k} \binom{k}{r} r x^{r-1} = k(1+x)^{k-1}, \qquad \text{real } x,$$

which for $x=-q$ establishes (5.19). The second sum is

$$\frac{2q}{p^2} \sum_{r=0}^{k} \left\{ \binom{k}{r} (-1)^r (1-q^r) \right\} = -2qp^{k-2}$$

as it follows easily from the binomial theorem. We therefore have

$$\widehat{c_k} = 2qp^{k-2}(qk-1) - s_k, \tag{5.20}$$

where, by using the exact expression for a_l in (5.13),

$$s_k \stackrel{\text{def}}{=} 2 \sum_{r=1}^{k} \binom{k}{r} (-1)^r \sum_{l=0}^{r} \binom{r+1}{l+1} p^l q^{r-l} \sum_{j=1}^{l} \binom{l}{j} (-1)^j \frac{1+qp^{j-1}}{1-p^j},$$

and we now simplify the above triple sum. First, we rearrange the order of summation, to get

$$s_k = 2 \sum_{l=1}^{k} \sum_{j=1}^{k} \binom{l}{j} (-1)^j \left(\frac{p}{q} \right)^l \frac{1+qp^{j-1}}{1-p^j} \sum_{r=l}^{k} \binom{k}{r} \binom{r+1}{l+1} (-q)^r, \tag{5.21}$$

and we proceed by computing the rightmost sum in the last expression. If we differentiate the identity $\sum_{r\geq 0}\binom{k}{r}x^{r+1} = x(1+x)^k$ (following from the binomial theorem) $l+1$ times with respect to x, we can verify that we get

$$\sum_{r\geq 0}\binom{k}{r}(r+1)^{\underline{l+1}}x^{r-l} = k^{\underline{l}}(1+x)^{k-l-1}(l+1+kx+x), \quad \text{integer } l\geq 0,\ \text{real } x.$$

We now multiply both sides of the last by $\frac{x^l}{(l+1)!}$ to get

$$\sum_{r\geq 0}\binom{k}{r}\binom{r+1}{l+1}x^r = \binom{k}{l}\frac{(1+x)^{k-1}}{l+1}\left(\frac{x}{1+x}\right)^l(l+1+kx+x), \quad \text{integer } l\geq 0,\ \text{real } x,$$

and by setting $x=-q$, we can rewrite (5.21) as

$$\begin{aligned} s_k &= 2\sum_{l=1}^{k}\sum_{j=1}^{k}\binom{l}{j}(-1)^j\left(\frac{p}{q}\right)^l\frac{1+qp^{j-1}}{1-p^j}\binom{k}{l}\frac{p^{k-1}}{l+1}\left(-\frac{q}{p}\right)^l\Big((l+1)-(k+1)q\Big) \\ &= 2p^{k-1}\left(\sum_{l=1}^{k}\sum_{j=1}^{k}\binom{k}{l}\binom{l}{j}(-1)^{l+j}\frac{1+qp^{j-1}}{1-p^j}\right. \\ &\qquad \left. -q(k+1)\sum_{l=1}^{k}\sum_{j=1}^{l}\binom{k}{l}\binom{l}{j}(-1)^{l+j}\frac{1+qp^{j-1}}{1-p^j}\frac{1}{l+1}\right). \end{aligned}$$

Since now the first double sum above is

$$\sum_{l=1}^{k}\sum_{j=1}^{k}\binom{k}{l}\binom{l}{j}(-1)^{l+j}\frac{1+qp^{j-1}}{1-p^j} = \frac{1+qp^{k-1}}{1-p^k}$$

by (2.41), and the second double sum is

$$\begin{aligned} &\sum_{l=1}^{k}\sum_{j=1}^{l}\binom{k}{l}\binom{l}{j}(-1)^{l+j}\frac{1+qp^{j-1}}{1-p^j}\frac{1}{l+1} \\ &= \frac{1}{k+1}\sum_{j=1}^{k}(-1)^j\frac{1+qp^{j-1}}{1-p^j}\sum_{l=0}^{k}\binom{k+1}{1+l}\binom{l}{j}(-1)^l && \text{by } \tbinom{0}{j}=0 \text{ for } j\geq 1 \\ &= \frac{1}{k+1}\sum_{j=1}^{k}(-1)^j\frac{1+qp^{j-1}}{1-p^j}\left((-1)^{k+2}\binom{-1}{j-k-1}+(-1)^j\right) && \text{by (2.15)} \\ &= \frac{1}{k+1}\sum_{j=1}^{k}\frac{1+(1-p)p^{j-1}}{1-p^j} && \text{by } \tbinom{-1}{m}=0 \text{ for } m<0 \end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{k+1}\sum_{j=1}^{k}\left\{1+\frac{p^{j-1}}{1-p^j}\right\} \\
&= \frac{1}{k+1}\left(k+\sum_{j=1}^{k}\frac{p^{j-1}}{1-p^j}\right),
\end{aligned}
$$

we get

$$
s_k = \frac{2}{p}\,p^k\,\frac{1+qp^{k-1}}{1-p^k} - \frac{2q}{p^2}\,p^k kp - \frac{2q}{p^2}p^k\sum_{j=1}^{k}\frac{p^j}{1-p^j}. \tag{5.22}
$$

The result now follows from (5.18), (5.20) and (5.22). ■

Corollary 5.1 *Under the fixed population model, the variance of the search cost for $+\infty$ in a PSL of n keys is*

$$
\begin{aligned}
\mathbf{Var}(F_n) \;=\; & \frac{2q}{p^2}\sum_{k=1}^{n}\left\{\binom{n}{k}(-1)^k\frac{p^k}{1-p^k}\,(k-1)\right\} \\
& -\frac{2}{p}\sum_{k=1}^{n}\left\{\binom{n}{k}(-1)^k\frac{p^k}{(1-p^k)^2}\,(1+qp^{k-1})\right\} \\
& +\frac{2q}{p^2}\sum_{k=1}^{n}\left\{\binom{n}{k}(-1)^k\frac{p^k}{1-p^k}\sum_{j=1}^{k}\left\{\frac{p^j}{1-p^j}\right\}\right\} \\
& -\sum_{k=1}^{n}\left\{\binom{n}{k}(-1)^k\frac{1+qp^{k-1}}{1-p^k}\right\} \\
& -\left(\sum_{k=1}^{n}\left\{\binom{n}{k}(-1)^k\frac{1+qp^{k-1}}{1-p^k}\right\}\right)^2.
\end{aligned}
$$

Proof: Immediate from (5.13) and Theorem 5.2, since $\mathbf{Var}(F_n) = \mathbf{E}\Big(F_n(F_n-1)\Big) + \mathbf{E}(F_n) - \{\mathbf{E}(F_n)\}^2$. ■

As promised in the beginning of this section, we continue with the derivation of asymptotics for the second factorial moment of $F(z)$. The variance of $F(z)$ will follow immediately.

Theorem 5.3 *Under the Poisson model with mean z, the second factorial moment of the search cost for $+\infty$ in a PSL is*

$$\begin{aligned}\mathbf{E}\Big(F(z)(F(z)-1)\Big) &= \frac{1}{p^2}\log^2_{\frac{1}{p}} z - \left(\frac{2\gamma}{p^2\ln p}+\frac{2}{p^2}\,f_{0,\frac{1}{p}}(z)\right)\log_{\frac{1}{p}} z\\ &\quad+\frac{2q}{p^2\ln p}+\frac{\gamma^2}{p^2\ln^2 p}+\frac{\pi^2}{6p^2\ln^2 p}-\frac{1}{6p^2}-\frac{2q}{p^2}\sum_{k\ge1}\frac{kp^k}{1-p^k}\\ &\quad-\frac{2q}{p^2}\,f_{1,\frac{1}{p}}(z)-\frac{2}{p^2\ln p}\,g_{0,\frac{1}{p}}(z)\\ &\quad+O\left(\frac{1}{z^d}\right)\qquad \text{as } z\to\infty,\qquad \text{for any } d>0,\end{aligned}$$

where the defined in (2.3) and (2.4) $f_{0,\frac{1}{p}}(z)$, $f_{1,\frac{1}{p}}(z)$ and $g_{0,\frac{1}{p}}(z)$ are periodic functions of $\log_{\frac{1}{p}} z$, with small magnitude.

Proof: We start by taking the Poisson transform of the recurrence of Lemma 5.4. Recall that $A(z)=\mathcal{P}(a_n;z)$ and $B(z)=\mathcal{P}(b_n;z)=\mathbf{E}\Big(F(z)(F(z)-1)\Big)$. We have

$$\mathcal{P}\left(\sum_{k=0}^{n}\left\{\binom{n}{k}p^kq^{n-k}b_k\right\};\ z\right)=B(pz)$$

as in (5.9), and we can verify

$$\mathcal{P}\left(-\frac{2q^{n+1}}{p}n+\frac{2q(1-q^n)}{p^2};\ z\right)=-\frac{2q^2}{p}\,ze^{-pz}-\frac{2q}{p^2}\left(e^{-pz}-1\right).$$

To find the Poisson transform of the remaining term in (5.17), we observe that the egf of $\langle p^n a_n\rangle$ is $e^{pz}A(pz)$, and thus the egf of $\langle p^{n-1}a_{n-1}\rangle$ is $\int_0^z e^{pt}A(pt)dt$ by (2.36). Since also the egf of $\langle q^n\rangle$ is e^{qz}, it follows from (2.37) that the binomial convolution $\left\langle\sum_{j=0}^{n}\binom{n}{j}p^{j-1}a_{j-1}q^{n-j}\right\rangle$ of the last two sequences has egf $e^{qz}\int_0^z e^{pt}A(pt)dt$, and (2.35) now implies that the egf of $\left\langle\sum_{j=0}^{n+1}\binom{n+1}{j}p^{j-1}q^{n+1-j}a_{j-1}\right\rangle$ is $e^{qz}e^{pz}A(pz)+qe^{qz}\int_0^z e^{pt}A(pt)dt$ since $A(0)=0$. Therefore,

$$\mathcal{P}\left(\sum_{k=0}^{n}\left\{\binom{n+1}{k+1}p^kq^{n-k}a_k\right\};\ z\right)=A(pz)+qe^{-pz}\int_0^z e^{pt}A(pt)dt,$$

and thus, the Poisson transform of (5.17) results in

$$B(z)-B(pz)=-\frac{2q^2}{p}\,ze^{-pz}-\frac{2q}{p^2}\,(e^{-pz}-1)+2A(pz)+2qe^{-pz}\int_0^z e^{pt}A(pt)dt. \quad (5.23)$$

This is similar to (5.14) — which we solved by taking its Mellin transform — but the presence of $A(pt)$ in the integrand makes it more complicated; so we still have some more work to do. The asymptotic expression for $A(x)$ from Theorem 5.1 is not appropriate for our purposes, because it contains a $O(x^{-d})$ term. Instead, we may solve (5.14) iteratively, to get

$$A(x)=\lim_{k\to\infty}\left\{A(p^kx)\right\}-\sum_{k\geq 0}\left\{e^{-p^kx}-1\right\}-\frac{q}{p}\sum_{k\geq 1}\left\{e^{-p^kx}-1\right\},$$

or

$$A(x)=-\left(\frac{1}{p}\sum_{k\geq 1}\left\{e^{-p^kx}-1\right\}+e^{-x}-1\right), \quad (5.24)$$

since $\lim_{k\to\infty}\left\{A(p^kx)\right\}=0$. We may now use this expression for $A(x)$ to compute the integral in (5.23) term-by-term. Indeed,

$$\int_0^z e^{pt}A(pt)dt=-\frac{1}{p}\sum_{k\geq 1}\int_0^z e^{pt}\left(e^{-p^{k+1}t}-1\right)dt\;-\;\int_0^z e^{pt}\left(e^{-pt}-1\right)dt,$$

and routine integration yields

$$\int_0^z e^{pt}\left(e^{-p^{k+1}t}-1\right)dt\;=\;\frac{-e^{pz}+e^{pz}p^k+e^{pz-p^{k+1}z}-p^k}{p(1-p^k)}$$

and

$$\int_0^z e^{pt}\left(e^{-pt}-1\right)dt\;=\;z-\frac{1}{p}\,(e^{pz}-1).$$

Therefore, after some algebra, we get

$$\int_0^z e^{pt}A(pt)dt=-\frac{1}{p^2}\,(e^{pz}-1)\sum_{k\geq 1}\frac{p^k}{1-p^k}-\frac{1}{p^2}e^{pz}\sum_{k\geq 1}\frac{e^{-p^{k+1}z}-1}{1-p^k}-z+\frac{1}{p}\,(e^{pz}-1),$$

and substitution of the last into (5.23) yields

$$B(z)-B(pz) = -\frac{2q}{p}\,ze^{-pz}-\frac{2q(p+1)}{p^2}\left(e^{-pz}-1\right)+2A(pz) + \frac{2q}{p^2}\left(e^{-pz}-1\right)\sum_{k\geq 1}\frac{p^k}{1-p^k}-\frac{2q}{p^2}\sum_{k\geq 1}\frac{e^{-p^{k+1}z}-1}{1-p^k}. \quad (5.25)$$

Now, we are in "known territory"; we continue, as in (5.14), by taking the Mellin transform of the last. We have $\mathcal{M}\left(e^{-\alpha z}-1;\, s\right)=\alpha^{-s}\Gamma(s)$ for $-1<\Re(s)<0$ as in (5.15), $\mathcal{M}(ze^{-pz};\, s)=p^{-s-1}\Gamma(s+1)$, and $\mathcal{M}(A(pz);\, s)=p^{-s}A^*(s)$ by (2.45), which implies $\mathcal{M}(A(pz);\, s)=-\frac{1+qp^{-s-1}}{p^s(1-p^{-s})}\Gamma(s)$ for $-1<\Re(s)<0$ by (5.16). So, the Mellin transform of (5.25) yields

$$B^*(s)-\frac{1}{p^s}\,B^*(s) = -\frac{2q}{p^2}\,\frac{\Gamma(s+1)}{p^s} - \frac{2q(p+1)}{p^2}\,\frac{\Gamma(s)}{p^s} - \frac{2(1+qp^{-s-1})}{p^s(1-p^{-s})}\,\Gamma(s) + \frac{2q}{p^2}\,\frac{\Gamma(s)}{p^s}\sum_{k\geq 1}\frac{p^k}{1-p^k} - \frac{2q}{p^2}\,\frac{\Gamma(s)}{p^s}\sum_{k\geq 1}\frac{1}{p^{ks}(1-p^k)} \quad \text{for } -1<\Re(s)<0.$$

Observe now that the sum $\sum_{k\geq 1}\frac{1}{p^{ks}(1-p^k)}$ in the last expression may be rewritten as $\sum_{k\geq 1}\left(\frac{1}{p^s}\right)^k+\sum_{k\geq 1}\frac{1}{p^{k(s-1)}(1-p^k)}$, and that the first of these two new sums converges to $\frac{p^{-s}}{1-p^{-s}}$ since $\Re(s)<0$. Therefore, the last equation may be rewritten as

$$B^*(s) = -\frac{2q}{p^2}\,\frac{\Gamma(s+1)}{p^s-1} + \frac{2}{p^2}\,\frac{\Gamma(s)}{(p^s-1)(p^{-s}-1)} + \frac{2q}{p^2}\,\frac{\Gamma(s)}{p^s-1}\sum_{k\geq 1}\frac{p^k}{1-p^k} = -\frac{2q}{p^2}\sum_{k\geq 1}\frac{\Gamma(s)p^k}{p^{ks}(1-p^k)(p^s-1)} \quad \text{for } -1<\Re(s)<0 \quad (5.26)$$

by doing some elementary algebra. We now recover $B(z)$ by using the inverse Mellin transform

$$B(z)=\frac{1}{2\pi i}\int_{c-i\infty}^{c+i\infty}B^*(s)z^{-s}ds, \qquad \text{for } -1<c<0.$$

To evaluate the above line integral, we work as in the example of Section 2.3.8: we consider an infinitely tall rectangle C, we verify that the integral along the right vertical line is $O(z^{-d})$, and along the top and bottom lines are 0, and by using the residue theorem we arrive at

$$B(z) = -\sum_{\alpha=\text{pole inside } C} \operatorname*{Res}_{s=\alpha}\left\{B^*(s)z^{-s}\right\} + O\left(\frac{1}{z^d}\right) \quad \text{as } z\to\infty, \qquad \text{for any } d>0. \tag{5.27}$$

We now use (2.48) to routinely compute the residues of each of the four terms of $B^*(s)z^{-s}$. The first term has simple poles at $s=s_k \stackrel{\text{def}}{=} i\frac{2k\pi}{\ln p}$, for $k=0,\pm1,\pm2,\ldots$, and we may compute

$$\sum_{\alpha=\text{pole inside } C} \operatorname*{Res}_{s=\alpha}\left\{\frac{\Gamma(s+1)}{p^s-1}\, z^{-s}\right\} = \frac{1}{\ln p} - f_{1,\frac{1}{p}}(z). \tag{5.28}$$

The second term is the most complicated to handle, since it has a triple pole at $s=0$, and double poles at $s=s_k$, for $k=\pm1,\pm2,\ldots$. After some lengthy calculations, we arrive at

$$\begin{aligned}&\sum_{\alpha=\text{pole inside } C} \operatorname*{Res}_{s=\alpha}\left\{\frac{\Gamma(s)}{(p^s-1)(p^{-s}-1)}\, z^{-s}\right\}\\ &= -\frac{1}{2}\log^2_{\frac{1}{p}} z + \left(\frac{\gamma}{\ln p} + f_{0,\frac{1}{p}}(z)\right)\log_{\frac{1}{p}} z + \frac{1}{12} - \frac{\gamma^2}{2\ln^2 p} - \frac{\pi^2}{12\ln^2 p} + \frac{1}{\ln p} g_{0,\frac{1}{p}}(z).\end{aligned} \tag{5.29}$$

The third and fourth terms have double poles at $s=0$, and simple poles at $s=s_k$, for $k=\pm1,\pm2,\ldots$, yielding

$$\sum_{\alpha=\text{pole inside } C} \operatorname*{Res}_{s=\alpha}\left\{\frac{\Gamma(s)}{p^s-1}\, z^{-s}\right\} = \log_{\frac{1}{p}} z - \frac{1}{2} - \frac{\gamma}{\ln p} - f_{0,\frac{1}{p}}(z) \tag{5.30}$$

and

$$\sum_{\alpha=\text{pole inside } C} \operatorname*{Res}_{s=\alpha}\left\{\frac{\Gamma(s)}{p^{ks}(p^s-1)}\, z^{-s}\right\} = -k + \log_{\frac{1}{p}} z - \frac{1}{2} - \frac{\gamma}{\ln p} - f_{0,\frac{1}{p}}(z),$$

which implies

$$\sum_{\alpha=\text{pole inside } C} \operatorname*{Res}_{s=\alpha} \left\{ \sum_{k\geq 1} \frac{\Gamma(s)p^k}{p^{ks}(1-p^k)(p^s-1)}\, z^{-s} \right\}$$
$$= \; -\sum_{k\geq 1} \frac{kp^k}{1-p^k} \; + \; \left(\log_{\frac{1}{p}} z - \frac{1}{2} - \frac{\gamma}{\ln p} - f_{0,\frac{1}{p}}(z) \right) \sum_{k\geq 1} \frac{p^k}{1-p^k}. \qquad (5.31)$$

The result now follows from (5.26) through (5.31). ■

Corollary 5.2 *Under the Poisson model with mean z, the variance of the search cost for $+\infty$ in a PSL is*

$$\begin{aligned} \mathbf{Var}\Big(F(z)\Big) \;=\; & \frac{q}{p^2}\, \log_{\frac{1}{p}} z \\ & - \frac{5}{12p^2} + \frac{1}{2p} + \frac{q(2-\gamma)}{p^2 \ln p} + \frac{\pi}{6p^2 \ln^2 p} - \frac{2q}{p^2} \sum_{k\geq 1} \frac{kp^k}{1-p^k} \\ & - \frac{2q}{p^2}\, f_{1,\frac{1}{p}}(z) - \frac{2}{p^2 \ln p}\, g_{0,\frac{1}{p}}(z) - \left(\frac{2\gamma}{p^2 \ln p} + \frac{q}{p^2} \right) f_{0,\frac{1}{p}}(z) - \frac{1}{p^2} f^2_{0,\frac{1}{p}}(z) \\ & + O\left(\frac{\ln z}{z^d} \right) \qquad \textit{as } z\to\infty, \qquad \textit{for any } d>0, \end{aligned}$$

where the defined in (2.3) and (2.4) $f_{1,\frac{1}{p}}(z)$, $f_{0,\frac{1}{p}}(z)$ and $g_{0,\frac{1}{p}}(z)$ are periodic functions of $\log_{\frac{1}{p}} z$, with small magnitude.

Proof: Follows from Theorems 5.1 and 5.3, after doing some algebra. ■

The derivation of an asymptotic expression for $\mathbf{Var}(F_n)$ is, clearly, of great interest. Having already derived $\mathbf{E}\Big(F(z)(F(z)-1)\Big) = B(z)$, and having shown that $\mathbf{E}\Big(F(z)\Big)$ and $\mathbf{E}(F_n)$ are "the same", one might be tempted to use approximation theorem (2.39) to show that $b_n = \mathbf{Var}(F_n)$ is "the same" as $\mathbf{Var}\Big(F(z)\Big)$. However, such a claim is not well-justified, since the $\frac{\partial^r}{\partial z^r} B(z)$ appearing in b_n are — in theory — unbounded, due to the presence of the $O(z^{-d})$ term in $B(z)$; for example, the very small term $\frac{\cos(e^z)}{e^z}$ has very large derivatives. In practice though, such "nasty"

functions do not arise naturally in the analyses of algorithms. Hence, it is reasonable to expect that $\mathbf{Var}(F_n) \sim \mathbf{Var}\Big(F(n)\Big)$, although the above reasoning is not a rigorous proof of that.

We diverge at this point, to remind the reader that there do exist data structures for which the variances under the fixed population model and under the Poisson model are different. For example [41], the variance of the number of internal nodes in a binary symmetric trie is $c(z) = (k + \text{oscillations})z + O(1)$ under the Poisson model (where $k = 2.9272\ldots$), but $c_n = \left(k - \frac{1}{\ln^2 2} + \text{oscillations}\right)n + O(1)$ under the fixed population model. This result is *not* an example in which the error term $O(1)$ in the Poissonized variance $c(z)$ is "nasty"; assuming that the $O(1)$ term has small derivatives, de-Poissonization of the second moment of the number of nodes, by means of (2.39), will eventually yield the given expression for c_n. The above discrepancy of the two variances, is not counter-intuitive either; the "extra degree of freedom" introduced in the Poisson model (variable number of keys) is reasonable to increase the variability of the number of nodes in the trie. One should be careful though, not to assume that the last reasoning leads to valid results in all cases; if, for example, $X_n = a(=\text{constant})$, then $\mathbf{Var}(X_n) = \mathbf{Var}\Big(X(z)\Big) = 0$.

Back to our problem now: derive asymptotics for $\mathbf{Var}(F_n)$, or for $\mathbf{E}\Big(F_n(F_n - 1)\Big)$. We have repeatedly stated so far, that our work for the variance (in this section) parallels our work for the expected value (in the preceding section). However, the problem of deriving asymptotics for $\mathbf{E}(F_n)$ wasn't even our concern in the preceding section; we had already derived $\mathbf{E}(F_n)$ by other means in Chapter 3. So, now, we have to resort to different techniques. A quick look at the exact value of $\mathbf{E}\Big(F_n(F_n - 1)\Big)$, given in Theorem 5.2, reveals that it consists of three alternating sums. We can therefore apply Rice's method, as explained in Section 2.3.9, to evaluate these sums. As one might expect, our calculations this time will be more

complicated, and — for the case of the third alternating sum — less straightforward.

Theorem 5.4 *Under the fixed population model, the second factorial moment of the search cost for $+\infty$ in a PSL of n keys is*

$$\begin{aligned}\mathbf{E}\Big(F_n(F_n-1)\Big) &= \frac{1}{p^2}\log^2_{\frac{1}{p}} n - \left(\frac{2\gamma}{p^2\ln p}+\frac{2}{p^2}\,f_{0,\frac{1}{p}}(n)\right)\log_{\frac{1}{p}} n \\ &\quad + \frac{2q}{p^2\ln p}+\frac{\gamma^2}{p^2\ln^2 p}+\frac{\pi^2}{6p^2\ln^2 p}-\frac{1}{6p^2}-\frac{2q}{p^2}\sum_{k\geq 1}\frac{kp^k}{1-p^k} \\ &\quad - \frac{2q}{p^2}\,f_{1,\frac{1}{p}}(n)-\frac{2}{p^2\ln p}\,g_{0,\frac{1}{p}}(n) \\ &\quad + O\left(\frac{\ln n}{n}\right) \qquad as\ n\to\infty,\end{aligned}$$

where the defined in (2.3) and (2.4) $f_{0,\frac{1}{p}}(n)$, $f_{1,\frac{1}{p}}(n)$ and $g_{0,\frac{1}{p}}(n)$ are periodic functions of $\log_{\frac{1}{p}} n$, with small magnitude.

Proof: According to Lemma 2.3, the first alternating sum in the exact expression for $\mathbf{E}\Big(F_n(F_n-1)\Big)$ is

$$S_1 \stackrel{\text{def}}{=} \sum_{k=1}^{n}\left\{\binom{n}{k}(-1)^k\frac{p^k}{1-p^k}\,(k-1)\right\} = -\frac{1}{2\pi i}\oint_C \mathrm{B}(n+1,-s)\,h_1(s)\,ds,$$

where

$$h_1(s) \stackrel{\text{def}}{=} \frac{p^s}{1-p^s}(s-1),$$

and C is a rectangle encompassing only the integer points $1,2,3,\ldots,n$. Working now as in Example 2.2, we consider an infinitely large rectangle C' surrounding C, and we verify that the integral of $\mathrm{B}(n+1,-s)h_1(s)$ along C' is $O(n^{-d})$ for any $d>0$. Inside C' but outside C, the term $\mathrm{B}(n+1,-s)$ has a simple pole at $s=0$, and $h_1(s)$ has simple poles at $s=s_k \stackrel{\text{def}}{=} i\frac{2k\pi}{\ln p}$, for $k=0,\pm1,\pm2,\ldots$. Therefore, the last expression for S_1 gives

$$S_1 = \operatorname*{Res}_{\substack{s=0\\ \text{order}=2}}\Big\{\mathrm{B}(n+1,-s)h_1(s)\Big\} + \sum_{k\in\mathbb{Z}-\{0\}}\operatorname*{Res}_{s=s_k}\Big\{\mathrm{B}(n+1,-s)h_1(s)\Big\}$$

$$+ O\left(\frac{1}{n^d}\right) \quad \text{as } n\to\infty, \quad \text{for any } d>0.$$

We now use (2.48) to compute these residues. For the residue at the double pole we get

$$\operatorname*{Res}_{\substack{s=0\\ \text{order}=2}} \Big\{\mathrm{B}(n+1,-s)h_1(s)\Big\} = \frac{1}{\ln p} - \frac{1}{2} - \frac{\gamma}{\ln p} - \frac{\Psi(n+1)}{\ln p}$$

after some algebra, or

$$\operatorname*{Res}_{\substack{s=0\\ \text{order}=2}} \Big\{\mathrm{B}(n+1,-s)h_1(s)\Big\} = \frac{1}{\ln p} - \frac{1}{2} - \frac{\gamma}{\ln p} + \log_{\frac{1}{p}} n + O\left(\frac{1}{n}\right) \quad \text{as } n\to\infty$$

by using property (2.29) of the Psi function. The residues at each of the simple poles are

$$\begin{aligned}
&\operatorname*{Res}_{s=s_k}\Big\{\mathrm{B}(n+1,-s)\,h_1(s)\Big\}\\
&\quad=\lim_{s\to s_k}\left\{\frac{\Gamma(n+1)\Gamma(-s)}{\Gamma(n+1-s)}\frac{p^s}{1-p^s}(s-1)(s-s_k)\right\} \qquad \text{by (2.48)}\\
&\quad=-\frac{\Gamma(n+1)}{\Gamma(n+1-s_k)}\Gamma(-s_k)(-s_k+1)p^{s_k}\lim_{s\to s_k}\left\{\frac{s-s_k}{1-p^{s_k}}\right\}\\
&\quad=-n^{i\,\frac{2k\pi}{\ln p}}\left(1+O\left(\frac{1}{n}\right)\right)\Gamma(-s_k)\,(-s_k+1)\left(-\frac{1}{\ln p}\right) \quad \text{by (2.23), } p^{s_k}=1 \text{ \& L'Hôpital's}\\
&\quad=\left(1+O\left(\frac{1}{n}\right)\right)\frac{1}{\ln p}n^{i\,\frac{2k\pi}{\ln p}}\Big(\Gamma(-s_k+1)+\Gamma(-s_k)\Big), \quad \text{by (2.16)}
\end{aligned}$$

and therefore, the sum of these residues is

$$\sum_{k\in\mathbb{Z}-\{0\}}\operatorname*{Res}_{s=s_k}\Big\{\mathrm{B}(n+1,-s)h_1(s)\Big\} = -f_{1,\frac{1}{p}}(n) - f_{0,\frac{1}{p}}(n) + O\left(\frac{1}{n}\right) \quad \text{as } n\to\infty.$$

Thus, the first sum becomes

$$S_1 = \log_{\frac{1}{p}} n + \frac{1}{\ln p} - \frac{1}{2} - \frac{\gamma}{\ln p} - f_{1,\frac{1}{p}}(n) - f_{0,\frac{1}{p}}(n) + O\left(\frac{1}{n}\right) \quad \text{as } n\to\infty. \quad (5.32)$$

The second sum in the exact expression for $\mathbf{E}\Big(F_n(F_n-1)\Big)$ is

$$S_2 \stackrel{\text{def}}{=} \sum_{k=1}^{n}\left\{\binom{n}{k}(-1)^k\frac{p^k}{(1-p^k)^2}\,(1+qp^{k-1})\right\} = -\frac{1}{2\pi i}\oint_C \mathrm{B}(n+1,-s)\,h_2(s)\,ds,$$

where

$$h_2(s) \stackrel{\text{def}}{=} \frac{p^s}{(1-p^s)^2}\,(1+qp^{s-1}),$$

and C is a rectangle encompassing only the integer points $1, 2, 3, \ldots, n$. Working as with S_1, we get

$$\begin{aligned} S_2 &= \operatorname*{Res}_{\substack{s=0\\ \text{order}=3}} \Big\{\mathrm{B}(n+1,-s)h_2(s)\Big\} + \sum_{k\in\mathbb{Z}-\{0\}} \operatorname*{Res}_{\substack{s=s_k\\ \text{order}=2}} \Big\{\mathrm{B}(n+1,-s)h_2(s)\Big\} \\ &\quad + O\left(\frac{1}{n^d}\right) \quad \text{as } n\to\infty, \quad \text{for any } d>0. \end{aligned}$$

After some lengthy algebra, we compute the residue at the triple pole to be

$$\begin{aligned} \operatorname*{Res}_{\substack{s=0\\ \text{order}=3}} \Big\{\mathrm{B}(n+1,-s)h_2(s)\Big\} &= \frac{\Psi'(n+1)}{2p\ln^2 p} - \frac{\Psi^2(n+1)}{2p\ln^2 p} - \frac{\ln p - p\ln p + \gamma}{p\ln^2 p}\,\Psi(n+1) \\ &\quad - \frac{\pi^2}{12p\ln^2 p} - \frac{q\gamma}{p\ln p} - \frac{\gamma^2}{2p\ln^2 p} - \frac{5}{12p} + \frac{1}{2}, \end{aligned}$$

and by using properties (2.29) and (2.30) of the Psi function, we can simplify it to

$$\begin{aligned} \operatorname*{Res}_{\substack{s=0\\ \text{order}=3}} \Big\{\mathrm{B}(n+1,-s)h_2(s)\Big\} &= -\frac{1}{2p}\log^2_{\frac{1}{p}} n + \left(\frac{q}{p} + \frac{\gamma}{p\ln p}\right)\log_{\frac{1}{p}} n \\ &\quad - \frac{\pi^2}{12p\ln^2 p} - \frac{q\gamma}{p\ln p} - \frac{\gamma^2}{2p\ln^2 p} - \frac{5}{12p} + \frac{1}{2} \\ &\quad + O\left(\frac{\ln n}{n}\right) \quad \text{as } n\to\infty. \end{aligned}$$

The residues at the double poles are

$$\begin{aligned} &\operatorname*{Res}_{\substack{s=s_k\\ \text{order}=2}} \Big\{\mathrm{B}(n+1,-s)h_2(s)\Big\} \\ &= \lim_{s\to s_k}\left\{\frac{d}{ds}\left(\frac{\Gamma(n+1)\Gamma(-s)}{\Gamma(n+1-s)}\frac{p^s}{(1-p^s)^2}(1+qp^{s-1})(s-s_k)^2\right)\right\} \qquad \text{by (2.48)} \\ &= -\frac{\Gamma(n+1)}{\Gamma(n+1-s_k)}\frac{1}{p\ln^2 p}\Gamma(-s_k)\Big(\Psi(-s_k) - \Psi(n+1-s_k) - q\ln p\Big) \quad \text{do algebra} \\ &= -n^{i\frac{2k\pi}{\ln p}}\left(1+O\left(\frac{1}{n}\right)\right)\frac{1}{p\ln^2 p} \\ &\quad \left(\Gamma(-s_k)\Psi(-s_k) - \Gamma(-s_k)\left(\ln n + O\Big(\frac{1}{n}\Big)\right) - q\ln p\,\Gamma(-s_k)\right) \quad \text{by (2.23) \& (2.29)} \end{aligned}$$

and therefore, their sum becomes

$$\sum_{k\in\mathbb{Z}-\{0\}} \operatorname*{Res}_{\substack{s=s_k \\ \text{order}=2}} \Big\{\mathrm{B}(n+1,-s)h_2(s)\Big\} = \frac{1}{p\ln p}\, g_{0,\frac{1}{p}}(n) + \frac{1}{p}\, f_{0,\frac{1}{p}}(n)\log_{\frac{1}{p}} n - \frac{q}{p}\, f_{0,\frac{1}{p}}(n) + O\left(\frac{\ln n}{n}\right) \quad \text{as } n\to\infty.$$

Thus, the second sum is

$$\begin{aligned} S_2 &= -\frac{1}{2p}\log^2_{\frac{1}{p}} n + \left(\frac{q}{p} + \frac{\gamma}{p\ln p} + \frac{1}{p}\, f_{0,\frac{1}{p}}(n)\right)\log_{\frac{1}{p}} n \\ &\quad - \frac{\pi^2}{12p\ln^2 p} - \frac{q\gamma}{p\ln p} - \frac{\gamma^2}{2p\ln^2 p} - \frac{5}{12p} + \frac{1}{2} \\ &\quad + \frac{1}{p\ln p}\, g_{0,\frac{1}{p}}(n) - \frac{q}{p} f_{0,\frac{1}{p}}(n) + O\left(\frac{\ln n}{n}\right) \quad \text{as } n\to\infty. \end{aligned} \tag{5.33}$$

The third sum in the exact expression for $\mathbf{E}\Big(F_n(F_n-1)\Big)$ is

$$S_3 \stackrel{\text{def}}{=} \sum_{k=1}^{n} \binom{n}{k} (-1)^k h_3(k),$$

where

$$h_3(k) \stackrel{\text{def}}{=} \frac{p^k}{1-p^k} \sum_{j=1}^{k} \frac{p^j}{1-p^j},$$

but its evaluation poses a problem, since $h_3(s)$ is not analytic on the complex plane, and hence its integral along the rectangle C is not defined. A similar problem had been encountered in [17], and we will follow an analogous approach for its solution here. If

$$w(s) \stackrel{\text{def}}{=} \sum_{j\geq 1} \frac{p^j s}{1-p^j s},$$

then, $h_3(s) = \frac{p^s}{1-p^s}\,(w(1) - w(p^s))$, and hence

$$S_3 = -\frac{1}{2\pi i} \oint_C \mathrm{B}(n+1,-s)\, \frac{p^s}{1-p^s} \Big(w(1) - w(p^s)\Big) ds,$$

where C is a rectangle encompassing the points $1, 2, 3, \ldots, n$. Working as with S_1 and S_2, the last gives

$$S_3 = \operatorname*{Res}_{s=0} \left\{ \mathrm{B}(n+1,-s) \frac{p^s}{1-p^s} \Big(w(1) - w(p^s)\Big) \right\} + O\Big(\frac{1}{n^d}\Big) \quad \text{as } n\to\infty, \quad \text{for any } d>0;$$

notice here that all poles of $\frac{1}{1-p^s}$ are removed by $w(1)-w(p^s)$. We now proceed with the computation of the last residue. We have

$$\begin{aligned}
&\operatorname*{Res}_{s=0}\left\{\mathrm{B}(n+1,-s)\frac{p^s}{1-p^s}\Big(w(1)-w(p^s)\Big)\right\}\\
&\quad= \lim_{s\to 0}\left\{\frac{\Gamma(n+1)\Gamma(-s)}{\Gamma(n+1-s)}\frac{p^s}{1-p^s}\Big(w(1)-w(p^s)\Big)s\right\} && \text{by (2.48)}\\
&\quad= -\lim_{s\to 0}\left\{\frac{w(1)-w(p^s)}{1-p^s}\right\} && \text{by (2.18) \& (2.16)}\\
&\quad= -\lim_{s\to 0}\left\{\frac{\frac{d}{ds}\left(w(1)-w(p^s)\right)}{\frac{d}{ds}(1-p^s)}\right\} && \text{by L'Hôpital's}\\
&\quad= \lim_{s\to 0}\left\{\frac{\sum_{j\geq 1}\frac{d}{ds}\left(\frac{p^{j+s}}{1-p^{j+s}}\right)}{-p^s\ln p}\right\}\\
&\quad= -\sum_{j\geq 1}\frac{p^j}{(1-p^j)^2} && \text{by doing algebra}
\end{aligned}$$

and since $\sum_{k\geq 1} kx^k = \frac{x}{(1-x)^2}$, the last sum can be rewritten as

$$\sum_{j\geq 1}\frac{p^j}{(1-p^j)^2} = \sum_{j\geq 1}\sum_{k\geq 1} kp^{jk} = \sum_{k\geq 1} k\sum_{j\geq 1}(p^k)^j = \sum_{k\geq 1}\frac{kp^k}{1-p^k},$$

and we therefore have

$$S_3 = -\sum_{k\geq 1}\frac{kp^k}{1-p^k} + O\left(\frac{1}{n^d}\right) \quad \text{as } n\to\infty, \quad \text{for any } d>0. \tag{5.34}$$

The result now follows from Theorem 5.2, and expressions (5.32), (5.33) and (5.34) for the three sums appearing in it. ■

Corollary 5.3 *Under the fixed population model, the variance of the search cost for $+\infty$ in a PSL of n keys is*

$$\begin{aligned}
\mathbf{Var}(F_n) &= \frac{q}{p^2}\log_{\frac{1}{p}} n\\
&\quad -\frac{5}{12p^2}+\frac{1}{2p}+\frac{q(2-\gamma)}{p^2\ln p}+\frac{\pi}{6p^2\ln^2 p}-\frac{2q}{p^2}\sum_{k\geq 1}\frac{kp^k}{1-p^k}
\end{aligned}$$

$$- \frac{2q}{p^2} f_{1,\frac{1}{p}}(n) - \frac{2}{p^2 \ln p} g_{0,\frac{1}{p}}(n) - \left(\frac{2\gamma}{p^2 \ln p} + \frac{q}{p^2} \right) f_{0,\frac{1}{p}}(n) - \frac{1}{p^2} f^2_{0,\frac{1}{p}}(n)$$
$$+ O\left(\frac{\ln n}{n} \right) \qquad \text{as } n \to \infty,$$

where the defined in (2.3) and (2.4) $f_{1,\frac{1}{p}}(n)$, $f_{0,\frac{1}{p}}(n)$ *and* $g_{0,\frac{1}{p}}(n)$ *are periodic functions of* $\log_{\frac{1}{p}} n$, *with small magnitude.*

Proof: Follows from the expression for $\mathbf{E}\Big(F_n(F_n-1)\Big)$ given in Theorem 5.4, and from

$$\mathbf{E}(F_n) = \frac{1}{p} \log_{\frac{1}{p}} n - \frac{1}{p} \left(\frac{\gamma}{\ln p} + \frac{1}{2} \right) + 1 - \frac{1}{p} f_{0,\frac{1}{p}}(n) + O\Big(\frac{1}{n}\Big) \qquad \text{as } n \to \infty,$$

which is an immediate consequence of Theorem 3.3 and of asymptotics (3.10) and (3.11). ■

5.5 Summary

In this chapter, we re-examined the search cost for $+\infty$ in a PSL, aiming primarily at the derivation of its variance. First, we derived an exact expression for the variance of this cost under the fixed population model (Corollary 5.1). Next, we derived asymptotic expressions for the same variance under the Poisson model (Corollary 5.2) and under the fixed population model (Corollary 5.3), and we observed that these two variances are the same. On our way towards the derivation of these variances, we also obtained an asymptotic expression for the expected value of the same cost under the Poisson model (Theorem 5.1), and we observed that this is also equal to its corresponding search cost under the fixed population model.

The techniques that we used in this chapter, were more general and powerful than those of Chapter 3. We set up a recurrence for the probability generating

function of the search cost, and we differentiated it once and twice to derive a recurrence for the first and second factorial moment. We then considered the Poisson transform of the last recurrence, and we used it to achieve two things. First, by using Mellin transforms, we obtained asymptotic expressions for the first and second factorial moments under the Poisson model. Second, by using binomial transforms, we obtained exact expressions for these moments under the fixed population model, and we subsequently used Rice's method to derive asymptotics of these expressions.

Chapter 6

Elastic-Bucket Tries

The methods used to derive moments of the PSL search cost may also be used to derive moments of the trie space complexity, despite the fact that the trie is a dictionary implementation very different in flavour from the skip lists. Each operation performed on a trie, instead of being a comparison of two entire keys, is a comparison of two bits (or characters), one from each key.

In this chapter, we consider the "elastic bucket" trie, a variant of the standard trie, in which each external node holding j keys has exactly j key slots. We are interested in the number of external nodes of each size, and we examine these quantities under the fixed population model and under the Poisson model. To derive our results, we work as in the previous chapter. We start by setting up a recurrence for the (moment) generating function of the space complexity, we differentiate this recurrence once and twice, and we solve the resulting new recurrences by using binomial, Poisson and Mellin transforms, as well as Rice's method. Working in this way, we obtain asymptotics for the expected values under both models of randomness, and for the variances and covariances under the Poisson model. We

also derive exact expressions for the expected values under the fixed population model. A preliminary version of these results has appeared in [31].

6.1 Review of the Structure

The *trie*, introduced in late 1950's by de la Briandais [8] and Fredkin [19], is a well-known data structure that allows efficient update and re*trie*val of information. Unlike the skip lists and the search trees, the trie is based on the representation of the keys stored in it, rather than on the order statistics of the keys. In its simplest form (see Figure 6.1(a)), a trie is a binary tree having internal and external nodes. The internal nodes are used to guide the search for a key; the external nodes are used to store the keys. The keys are (encoded) strings over the alphabet $\{0, 1\}$.

To search for a key, we start at the root and we branch to the left or to the right depending on whether the first character of the search key is 0 or 1. At the new node, we branch in the same way to the left or to the right according to the second character of the key, etc. When we reach an external node, we compare the search key with the key (if any) stored in that external node.

To insert a new key, we first search for it. If our unsuccessful search terminates in an empty external node, we simply insert the new key in that node. If our search terminates in a non-empty external node, we extend the search path by adding an appropriate number of internal nodes, and we finally store the new key in a new external node. For example, insertion of key *110111* into the trie of Figure 6.1(a) gives the trie of Figure 6.1(b). Observe here that tries are independent of the insertion sequences of their keys.

A natural generalization of the trie is the (*fixed-*)*bucket trie*, first proposed by

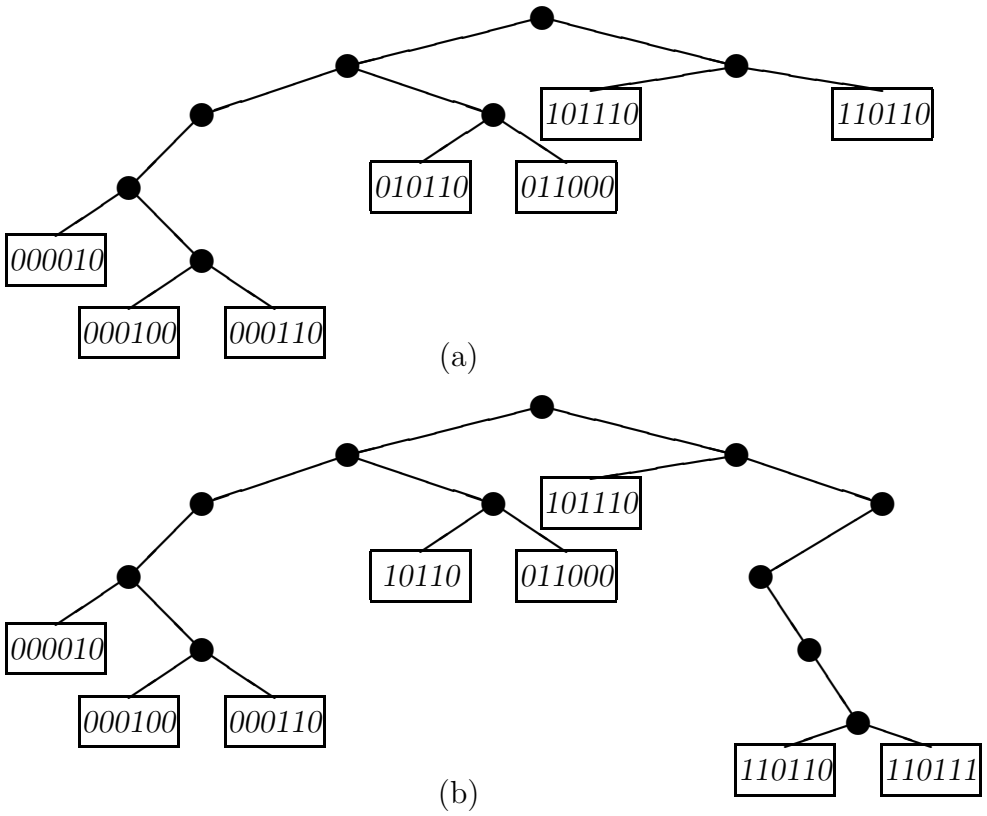


Figure 6.1: Insertion of *110111* into a trie

Sussenguth in [45], and studied by Knuth in [26, Ex. 6.3.20]. In this structure, we store up to b keys in each external node (*bucket*), and each bucket has fixed capacity b, for some $b > 1$. For example, if $b = 2$, the two tries of Figure 6.1 will form the two bucket tries of Figure 6.2. Obviously, a bucket trie is shorter than a non-bucket one, a fact which has a couple of implications. First, the search paths are shorter, and this affects the search time; it will be cheaper to reach an external node, but more expensive to search within it. Second, the number of internal nodes will be smaller, and this will affect the space complexity; less space will be needed for the internal nodes, although, on the other hand, more space will be needed for

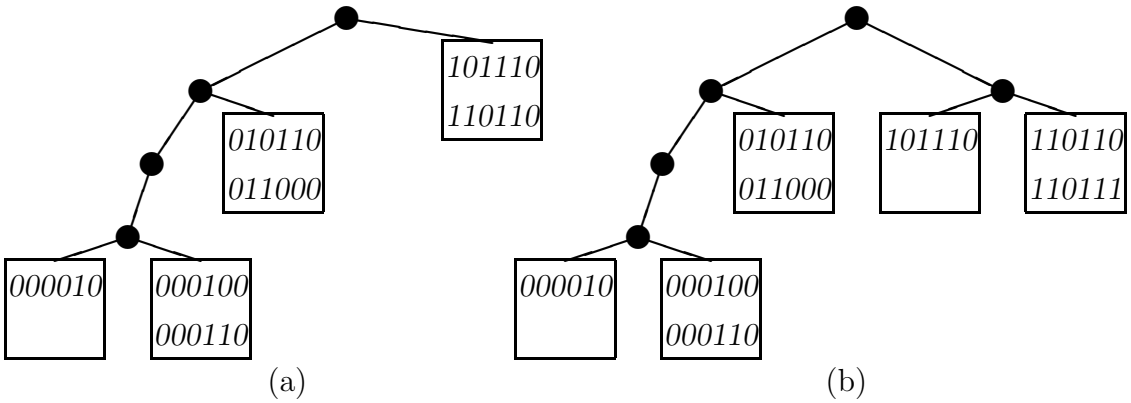


Figure 6.2: Insertion of *110111* into a fixed-bucket trie

the external nodes, since not all buckets will be full.

A remedy for the problem of the wasted space in the fixed-bucket trie, is the *elastic-bucket trie*. If an external node in the fixed-bucket trie contains j keys ($1 \leq j \leq b$), it will be allocated space for exactly j keys in the elastic-bucket trie, and it will be called a *type j bucket*. In other words, we maintain b different types of external nodes, reducing thus the space requirements, at the expense of a slight complication in the update algorithms and in the storage management.

The notion of "elastic" (also called "exact-fit") nodes just introduced is not new. It has been studied before in the context of B-trees [18, 29], B^+-trees [5], and m-ary search trees [4, 27]. In this chapter, we present the first study on the effect of "elastic" nodes (buckets) in tries.

Since there exists a one-to-one correspondence between fixed- and elastic-bucket tries, all known results on the number of internal nodes, the height, and the search cost in a fixed-bucket trie carry over to elastic-bucket tries. These results include not only expected values of the above mentioned quantities, but also higher moments and convergences of these random variables (see [21, Section 3.4.4] for an extensive list of 46 papers, and [30, Chapter 5] for a textbook-like presentation of the basic

results). The only new and interesting quantities introduced in the elastic-bucket trie are the numbers of type j buckets, for $1 \le j \le b$.

6.2 Notation and Related Results

Given the number of keys stored in a bucket trie, the shape of the trie — which uniquely determines the number of buckets of each size — will depend on the (representation of the) keys themselves. Our assumption will be that the n keys of the trie are infinite strings of i.i.d. 0/1 bits. This model can be shown to be equivalent to drawing the n keys from the [0,1]-uniform distribution, and, in the context of tries, it is also called the *Bernoulli model* of randomness. We will continue using the term "fixed population" model for it.

Under the fixed population model, we define the random variable

$$X_n^{(j)} \stackrel{\text{def}}{=} \text{number of type } j \text{ buckets in a trie of } n \text{ keys}, \qquad \text{for } 0 \le j \le b.$$

For example, for the trie of Figure 6.2(a), we have $X_7^{(0)} = 1$, $X_7^{(1)} = 1$, and $X_7^{(2)} = 3$. As usual, $X^{(j)}(z)$ will denote the Poissonized version of $X_n^{(j)}$.

To add flexibility to our results so that they cover more cases, we will study the linear combination

$$X_n \stackrel{\text{def}}{=} \lambda_0 X_n^{(0)} + \lambda_1 X_n^{(1)} + \cdots + \lambda_b X_n^{(b)}, \qquad \text{for } \lambda_0 = 0, \quad \text{arbitrary reals } \lambda_1, \lambda_2, \ldots, \lambda_b$$

for the fixed population model,[1] as well as its Poissonized counterpart $X(z)$. Clearly, if we compute $\mathbf{E}(X_n)$, we can then compute all $\mathbf{E}\left(X_n^{(j)}\right)$ for $1 \le j \le b$, and if we also compute $\mathbf{Var}(X_n)$, we can then compute all $\mathbf{Cov}\left(X_n^{(j)}, X_n^{(l)}\right)$ for $1 \le j, l \le b$.

[1] The reader should not confuse X_n^j and $X_n^{(j)}$; the former is the product $\overbrace{X_n X_n \cdots X_n}^{j \text{ times}}$, whereas the latter is the number of type j buckets.

As mentioned earlier, the $X_n^{(j)}$ have not been studied before. A related quantity though, the number of internal nodes in a fixed-bucket trie, has been studied extensively. Knuth, for example, showed [26, Ex. 6.3.20] that the average number of internal nodes is

$$n\left(\frac{1}{b\ln 2} + \text{oscillations}\right) + O(1), \tag{6.1}$$

and Jacquet and Régnier [25, 41] obtained additional results on the variance and asymptotic distribution of the same quantity.

In this chapter, we derive exact and asymptotic expressions for $\mathbf{E}(X_n)$, and an asymptotic expression for $\mathbf{E}\Big(X(z)\Big)$; $\mathbf{E}\Big(X_n^{(j)}\Big)$ and $\mathbf{E}\Big(X^{(j)}(z)\Big)$ follow easily. From our expression for $\mathbf{E}(X_n)$, we also obtain the expected total number of *non-empty* buckets. Finally, we obtain an asymptotic expression for $\mathbf{Var}\Big(X(z)\Big)$, which allows us to compute the entire variance-covariance matrix of all $X^{(j)}(z)$ for $1 \le j \le b$.

For brevity, in our proofs, we will use the notation

$$a_n \stackrel{\text{def}}{=} \mathbf{E}(X_n) \qquad \text{and} \qquad b_n \stackrel{\text{def}}{=} \mathbf{E}(X_n^2)$$

and

$$A(z) \stackrel{\text{def}}{=} \mathbf{E}\Big(F(z)\Big) = \mathcal{P}(a_n;\, z), \qquad \text{and} \qquad B(z) \stackrel{\text{def}}{=} \mathbf{E}\Big(X^2(z)\Big) = \mathcal{P}(b_n;\, z),$$

analogous to the notation of Chapter 5, as well as

$$V(z) \stackrel{\text{def}}{=} \mathbf{Var}\Big(X(z)\Big).$$

6.3 The Set-up

As mentioned in the beginning of this chapter, we are going to work in a manner completely analogous to that of Chapter 5. Although in that chapter we started by

considering the *pgf* of the random variable of interest, in this chapter, we are going to consider the *mgf*

$$M_n(z) \stackrel{\text{def}}{=} \mathbf{E}\left(e^{zX_n}\right), \qquad \text{integer } n \geq 0$$

of X_n, since X_n does *not* take integer values, and hence, its pgf is not defined.

Lemma 6.1 *Under the fixed population model, the mgf of X_n satisfies the recurrence*

$$M_n(z) = \begin{cases} e^{z\lambda_n} & \text{for } 0 \leq n \leq b \\ \dfrac{1}{2^n} \displaystyle\sum_{k=0}^{n} \binom{n}{k} M_k(z) M_{n-k}(z) & \text{for } n > b \end{cases}$$

Proof: Clearly, when $1 \leq n \leq b$, the trie has only one bucket (at its root), which contains n keys. Therefore,

$$X_n = \lambda_n, \qquad \text{for} \quad 0 \leq n \leq b,$$

since also $X_0 = \lambda_0 = 0$. This establishes the initial values of our recurrence.
When $n > b$, the trie has more than 1 bucket, and we may therefore condition on L_n, the number of keys in the left subtree, to get

$$M_n(z) = \sum_{k=0}^{n} \left\{ \mathbf{E}\left(e^{zX_n} \mid L_n = k\right) \mathbf{Pr}[L_n = k] \right\}, \qquad \text{for } n > b.$$

According to our probabilistic assumptions (beginning of Section 6.2), the first bit of each key may be either a 0 or a 1, each with probability 1/2. Therefore, L_n follows the binomial distribution with parameter 1/2, and thus the last equation may be rewritten as

$$M_n(z) = \sum_{k=0}^{n} \left\{ \mathbf{E}\left(e^{z(X_k + X_{n-k})}\right) \binom{n}{k} \frac{1}{2^n} \right\}, \qquad \text{for } n > b,$$

which completes the proof, since $\mathbf{E}\left(e^{z(X_k+X_{n-k})}\right) = \mathbf{E}\left(e^{zX_k}\right) \mathbf{E}\left(e^{zX_{n-k}}\right)$ by independence. ■

We now continue by differentiating the recurrence of the above lemma once (in Section 6.4) and twice (in Section 6.5) at $z=0$; according to (2.10) and (2.11), such differentiations will yield recurrences for $a_n=\mathbf{E}(X_n)$ and $b_n=\mathbf{E}(X_n^2)$ respectively.

6.4 Expected Values

We start by deriving a recurrence for $\mathbf{E}(X_n)$.

Lemma 6.2 *Under the fixed population model, the expected value of the linear combination of the numbers of all bucket types, in a trie of n keys, satisfies the recurrence*

$$\begin{cases} a_n = \lambda_n & \text{for} \quad 0 \le n \le b \\ a_n = \dfrac{1}{2^{n-1}} \displaystyle\sum_{k=0}^{n} \binom{n}{k} a_k & \text{for} \quad n > b \end{cases} . \tag{6.2}$$

Proof: If we differentiate the recurrence of Lemma 6.1 with respect to z, and we evaluate the resulting expression at $z=0$, we get

$$a_n = \lambda_n, \qquad \text{for } 0 \le n \le b$$

and

$$a_n = \frac{1}{2^n} \sum_{k=0}^{n} \binom{n}{k} (a_k + a_{n-k}), \qquad \text{for } n > b,$$

since $M_r(0) = 1$ from (2.9), and $M_r'(0) = a_r$ from (2.10). The result now follows, since $\sum_{k=0}^{n} \binom{n}{k} a_k = \sum_{k=0}^{n} \binom{n}{k} a_{n-k}$. ■

Recurrence (6.2) is not in the form of any of the four recurrences discussed after Lemma 5.2 (their initial values are different). Although we could appeal to a solution of a generalization of (6.2), given by Szpankowski in [49] — as we did in [31] — we choose instead to (re)solve (6.2), for a couple of reasons. First, de-generalizing Szpankowski's solution requires some effort, and second, half of

the work done towards the solution of (6.2) has to be done anyway towards the derivation of the solution's asymptotic Poissonized value.

We work as with the recurrence of Lemma 5.2. If we recall that $A(z)=\mathcal{P}(a_n;z)$, multiplication of both sides of (6.2) by $e^{-z}\frac{z^n}{n!}$, and addition of the resulting equalities gives

$$A(z) = e^{-z}\sum_{n=0}^{b}\lambda_n\frac{z^n}{n!} + 2A\left(\frac{z}{2}\right) - e^{-z}\sum_{n=0}^{b}\sum_{k=0}^{n}\frac{1}{2^{n-1}}\binom{n}{k}\lambda_k\frac{z^n}{n!}$$

by using (5.9). But the "$n=0$" terms of both sums in the above expression are 0, and the "$n=1$" terms are λ_1. Therefore, the last equation may be rewritten as

$$A(z) - 2A\left(\frac{z}{2}\right) = e^{-z}y_b(z), \tag{6.3}$$

where

$$y_b(z) \stackrel{\text{def}}{=} \sum_{k=2}^{b}\sigma_k z^k, \tag{6.4}$$

with

$$\sigma_k \stackrel{\text{def}}{=} \frac{\lambda_k}{k!} - \frac{1}{k!\,2^{k-1}}\sum_{r=1}^{k}\binom{k}{r}\lambda_r, \qquad \text{for } 2\le k\le b. \tag{6.5}$$

Recurrence (6.3) is the analogous of (5.10), and it will thus be the starting point for the derivation of an *exact expression for* $a_n = \mathbf{E}(X_n)$, and of an *asymptotic expression for* $A(z)=\mathbf{E}\Big(X(z)\Big)$. We now proceed with the derivation of a_n.

Theorem 6.1 *Under the fixed population model, the expected value of the linear combination of the numbers of all bucket types, in a trie of n keys, is*

$$\mathbf{E}(X_n) = \lambda_1 n + \sum_{k=2}^{n}\left\{\binom{n}{k}(-1)^k\frac{1}{1-2^{1-k}}\left(\sum_{r=2}^{b}\binom{k}{r}(-1)^r r!\,\sigma_r\right)\right\}, \quad \textit{for } n\ge 1,$$

where the terms σ_r are defined in (6.5).

Proof: Working as in the proof of Lemma 5.3, we get

$$A(z) = \sum_{n\ge 0}\sum_{k=0}^{n}\binom{n}{k}(-1)^{n-k}a_k\frac{z^n}{n!}.$$

We also have

$$\begin{aligned} e^{-z}y_b(z) &= e^{-z}\sum_{n\geq 2}(n!\,\sigma_n)\frac{z^n}{n!} && \text{by (6.4)} \\ &= \sum_{n\geq 2}(-1)^n\widehat{n!\,\sigma_n}\frac{z^n}{n!} && \text{by Lemma 2.2} \\ &= \sum_{n\geq 2}\sum_{k=2}^{b}\binom{n}{k}(-1)^{n-k}k!\,\sigma_k\frac{z^n}{n!}, && \text{by (2.40)} \end{aligned}$$

and therefore, if we extract the coefficients of $\frac{z^n}{n!}$ from both sides of (6.3), we get

$$\sum_{k=0}^{n}\binom{n}{k}(-1)^{n-k}a_k - \frac{1}{2^{n-1}}\sum_{k=0}^{n}\binom{n}{k}(-1)^{n-k}a_k = \sum_{k=2}^{b}\binom{n}{k}(-1)^{n-k}k!\,\sigma_k, \quad \text{for } n\geq 2,$$

which we may rearrange as

$$\sum_{k=0}^{n}\binom{n}{k}(-1)^{k}a_k = \frac{1}{1-2^{1-n}}\sum_{k=2}^{b}\binom{n}{k}(-1)^{k}k!\,\sigma_k, \quad \text{for } n\geq 2.$$

The result now follows from property (2.42), since $\sum_{k=0}^{0}\binom{0}{k}(-1)^k a_k = 0$ and $\sum_{k=0}^{1}\binom{1}{k}(-1)^k a_k = -\lambda_1$. ■

To derive the (more interesting) $\mathbf{E}\big(X_n^{(j)}\big)$, we have to simplify the expressions for σ_k, when only one of the λ_j's is 1. To be precise, if $\sigma_k^{(j)}$ denotes the value of σ_k when $\lambda_j = 1$ for some $1\leq j\leq b$ and $\lambda_r = 0$ for all $r\neq j$, (6.5) gives

$$\sigma_k^{(j)} = \frac{\delta_{j,k}}{j!} - \frac{\binom{k}{j}}{k!\,2^{k-1}}, \qquad \text{for } \; 2\leq k\leq b, \quad 1\leq j\leq b. \tag{6.6}$$

Corollary 6.1 *Under the fixed population model, the expected number of type j buckets in a trie on n keys is*

$$\mathbf{E}\big(X_n^{(j)}\big) = \begin{cases} n - \displaystyle\sum_{k=2}^{n}\binom{n}{k}(-1)^k G_k^{(1)} & \text{if} \quad j=1 \\ (-1)^j\displaystyle\sum_{k=2}^{n}\binom{n}{k}(-1)^k G_k^{(j)} & \text{if} \quad 2\leq j\leq b \end{cases}$$

where

$$G_k^{(j)} \stackrel{\text{def}}{=} \frac{\binom{k}{j}}{1-2^{1-k}} \left(1 - \frac{1}{2^{j-1}} \sum_{r=0}^{b-j} \binom{k-j}{r} (-1)^r \frac{1}{2^r} \right), \qquad \textit{for } 1 \le j \le b.$$

Proof: Follows easily from Theorem 6.1, by using (6.6). For example, when $2 \le j \le b$, we have

$$\begin{aligned}
&\mathbf{E}\Big(X_n^{(j)}\Big) \\
&= \sum_{k=2}^{n} \binom{n}{k} (-1)^k \frac{1}{1-2^{1-k}} \left(\sum_{m=2}^{b} \binom{k}{m} (-1)^m m! \, \frac{\delta_{m,j}}{j!} - \sum_{m=2}^{b} \binom{k}{m} (-1)^m m! \, \frac{\binom{m}{j}}{m!\, 2^{m-1}} \right) \\
&= \sum_{k=2}^{n} \binom{n}{k} (-1)^k \frac{1}{1-2^{1-k}} \left(\binom{k}{j} (-1)^j - \binom{k}{j} \sum_{m=j}^{b} \binom{k-j}{m-j} (-1)^m \frac{1}{2^{m-1}} \right) \\
&= \sum_{k=2}^{n} \binom{n}{k} (-1)^k \frac{\binom{k}{j}}{1-2^{1-k}} \left((-1)^j - \sum_{r=0}^{b-j} \binom{k-j}{r} (-1)^{r+j} \frac{1}{2^{r+j-1}} \right) \\
&= (-1)^j \sum_{k=2}^{n} \binom{n}{k} (-1)^k \frac{\binom{k}{j}}{1-2^{1-k}} \left(1 - \frac{1}{2^{j-1}} \sum_{r=0}^{b-j} \binom{k-j}{r} (-1)^r \frac{1}{2^r} \right).
\end{aligned}$$

The proof for $\mathbf{E}\Big(X_n^{(1)}\Big)$ is similar. ■

We now go back to recurrence (6.3), as promised; we will work towards the derivation of $A(z)$. Our method will be slightly different from the one followed in Theorem 5.1. We will not take Mellin transforms of both sides of (6.3), because the existence of a *non-zero* λ_1 poses certain problems to the existence of the Mellin transform of $A(z)$. Instead, we will solve (6.3) iteratively, a solution that will be useful anyhow in the derivation of $\mathbf{Var}\Big(X(z)\Big)$ (compare with (5.24), in the derivation of the variance of the PSL search cost).

Theorem 6.2 *Under the Poisson model with mean z, the expected value of the linear combination of the numbers of all bucket types in a trie is*

$$\mathbf{E}\Big(X(z)\Big) = z \left(\lambda_1 + \frac{1}{\ln 2} \sum_{k=2}^{b} \sigma_k \, (k-2)! \right) + z \left(\sum_{k=2}^{b} \sigma_k \, f_{k-1,\,2}(z) \right)$$

$$+ O\left(\frac{1}{z^d}\right) \quad \textit{as } z \to \infty, \quad \textit{for any } d > -1,$$

where the terms σ_k are defined in (6.5), and the periodic functions $f_{m,2}(z)$ are defined in (2.3).

Proof: An iterative solution of recurrence (6.3) yields

$$A(z) = \lim_{k\to\infty} \left\{ 2^k A\left(\frac{z}{2^k}\right) \right\} + \sum_{k\geq 0} \left\{ 2^k e^{-\frac{z}{2^k}} y_b\left(\frac{z}{2^k}\right) \right\}.$$

If we recall that $A(z)$ is the Poisson transform of $a_n = \mathbf{E}(X_n)$, the limit appearing above is

$$\begin{aligned} \lim_{k\to\infty} \left\{ 2^k A\left(\frac{z}{2^k}\right) \right\} &= \lim_{k\to\infty} \left\{ 2^k e^{-\frac{z}{2^k}} \sum_{r\geq 1} \mathbf{E}(X_r) \frac{\left(\frac{z}{2^k}\right)^r}{r!} \right\} \\ &= \lim_{k\to\infty} \left\{ \mathbf{E}(X_1) e^{-\frac{z}{2^k}} z + e^{-\frac{z}{2^k}} \sum_{r\geq 2} \mathbf{E}(X_r) \frac{z^r}{2^{(r-1)k}\, r!} \right\} \\ &= \lambda_1 z. \end{aligned} \tag{6.7}$$

Therefore, the iterative solution of (6.3) becomes

$$A(z) = \lambda_1 z + Q_b(z), \tag{6.8}$$

where

$$Q_b(z) \stackrel{\text{def}}{=} \sum_{k\geq 0} \left\{ 2^k e^{-\frac{z}{2^k}} y_b\left(\frac{z}{2^k}\right) \right\}, \tag{6.9}$$

and we now have to find a closed form for $Q_b(z)$. But $Q_b(z)$ is a "harmonic sum", which can be handled by taking its Mellin transform

$$Q_b^*(s) = \sum_{k\geq 0} \left\{ 2^k \mathcal{M}\left(e^{-\frac{z}{2^k}} y_b\left(\frac{z}{2^k}\right);\ s \right) \right\},$$

or, according to (2.45),

$$Q_b^*(s) = \mathcal{M}\left(e^{-z} y_b(z);\ s\right) \sum_{k\geq 0} \left(2^{s+1}\right)^k.$$

The series $\sum_{k\geq 0}\left(2^{s+1}\right)^k$ in the last expression converges when $\Re(s)<-1$, in which case it is $\frac{1}{1-2^{s+1}}$. The Mellin transform appearing in the above expression is

$$\begin{aligned}\mathcal{M}\Big(e^{-z}y_b(z);\ s\Big) &= \int_0^\infty e^{-z}\left(\sum_{k=2}^{b}\sigma_k z^k\right)z^{s-1}\,dz \qquad \text{by (2.43) \& (6.4)}\\ &= \sum_{k=2}^{b}\left\{\sigma_k\int_0^\infty e^{-z}z^{k+s-1}\,dz\right\}\\ &= \sum_{k=2}^{b}\sigma_k\,\Gamma(k+s), \qquad\qquad \text{by (2.1)}\end{aligned}$$

which exists when $\Re(s)>-k$ for all $2\leq k\leq b$. Therefore,

$$Q_b^*(s)=\frac{1}{1-2^{s+1}}\sum_{k=2}^{b}\sigma_k\,\Gamma(k+s), \qquad \text{for } -2<\Re(s)<-1,$$

and hence the inverse Mellin transform (2.46) gives

$$Q_b(z)=\frac{1}{2\pi i}\int_{c-i\infty}^{c+i\infty}Q_b^*(s)\,z^{-s}\,ds, \qquad \text{for } -2<c<-1.$$

We now use contour integration along an infinitely tall rectangle C, whose left vertical line is $\Re(s)=c$, and we get

$$Q_b(z)=-\sum_{\alpha=\text{pole inside } C}\operatorname*{Res}_{s=\alpha}\Big\{Q_b^*(s)z^{-s}\Big\}\ +\ O\left(\frac{1}{z^d}\right)\quad \text{as } z\to\infty,\quad \text{for any } d>-1,$$

as we have done already a couple of times before (Example 2.1 and Theorem 5.3). Inside C, $Q_b^*(s)z^{-s}$ has simple poles at $s=s_m\stackrel{\text{def}}{=}-1+i\frac{2m\pi}{\ln 2}$ for $m=0,\pm1,\pm2,\ldots$, and thus

$$\begin{aligned}Q_b(z) &= -\operatorname*{Res}_{s=-1}\Big\{Q_b^*(s)z^{-s}\Big\}\ -\sum_{m\in\mathbb{Z}-\{0\}}\operatorname*{Res}_{s=s_m}\Big\{Q_b^*(s)z^{-s}\Big\}\\ &\quad +O\left(\frac{1}{z^d}\right)\quad \text{as } z\to\infty,\quad \text{for any } d>-1. \qquad (6.10)\end{aligned}$$

The appearing residues can be computed routinely by using (2.48). We get

$$\operatorname*{Res}_{s=-1}\Big\{Q_b^*(s)z^{-s}\Big\}=-\frac{z}{\ln 2}\sum_{k=2}^{b}\sigma_k\,\Gamma(k-1) \qquad (6.11)$$

and

$$\operatorname*{Res}_{s=s_m}\Big\{Q_b^*(s)z^{-s}\Big\}=-\frac{z}{\ln 2}\sum_{k=2}^{b}\Big\{\sigma_k\,\Gamma\Big(k-1+i\,\frac{2m\pi}{\ln 2}\Big)\,e^{-i\,2m\pi\log_2 z}\Big\},\ \text{for } m=\pm1,\pm2,\ldots,$$

which implies

$$\sum_{m\in\mathbb{Z}-\{0\}}\operatorname*{Res}_{s=s_m}\Big\{Q_b^*(s)z^{-s}\Big\}=-z\sum_{k=2}^{b}\sigma_k\,f_{k-1,\,2}(z), \tag{6.12}$$

by working as in the derivation of (2.53). The result now follows from (6.8), (6.10), (6.11), and (6.12). ■

Corollary 6.2 *Under the Poisson model with mean z, the expected number of type 1 buckets in a trie is*

$$\begin{aligned}\mathbf{E}\left(X^{(1)}(z)\right) &= z\left(1-\frac{1}{\ln 2}\sum_{k=2}^{b}\frac{1}{(k-1)2^{k-1}}\right)\;-\;z\left(\sum_{k=2}^{b}\frac{f_{k-1,\,2}(z)}{(k-1)!\,2^{k-1}}\right)\\ &\quad+O\left(\frac{1}{z^d}\right)\quad \textit{as } z\to\infty,\quad \textit{for all } d>-1,\end{aligned}$$

and the expected numbers of type j buckets, for all $2\le j\le b$, are

$$\begin{aligned}\mathbf{E}\left(X^{(j)}(z)\right) &= \frac{z}{\ln 2}\left(\frac{1}{j(j-1)}-\sum_{k=2}^{b}\frac{\binom{k}{j}}{k(k-1)2^{k-1}}\right)+z\left(\frac{f_{j-1,\,2}(z)}{j!}-\sum_{k=2}^{b}\frac{\binom{k}{j}f_{k-1,\,2}(z)}{k!\,2^{k-1}}\right)\\ &\quad+O\left(\frac{1}{z^d}\right)\quad \textit{as } z\to\infty,\quad \textit{for all } d>-1,\end{aligned}$$

where the periodic functions $f_{m,2}(z)$ are defined in (2.3).

Proof: Follows from Theorem 6.2, and the special expressions for σ_k in (6.6). ■

An asymptotic expression for $\mathbf{E}(X_n)$, the expected number of buckets under the fixed population model, is clearly of great interest. As discussed after Corollary 5.2, a "sloppy" application of the approximation theorem (2.39) on $\mathbf{E}\Big(X(z)\Big)$ could yield the desired $\mathbf{E}(X_n)$, but such a derivation would not have been a rigorous proof. Instead, we work as in the previous chapter: we go back to the exact expression for $\mathbf{E}(X_n)$, and we evaluate its alternating sum by using Rice's method.

Theorem 6.3 *Under the fixed population model, the expected value of the linear combination of the numbers of all bucket types, in a trie of n keys, is*

$$\mathbf{E}(X_n) = n\left(\lambda_1 + \frac{1}{\ln 2}\sum_{k=2}^{b}\sigma_k\,(k-2)!\right) + n\left(\sum_{k=2}^{b}\sigma_k\,f_{k-1,\,2}(n)\right) + O(1) \qquad \textit{as } n\to\infty,$$

where the terms σ_k are defined in (6.5), and the periodic functions $f_{m,2}(z)$ are defined in (2.3).

Proof: We start with the exact expression for $\mathbf{E}(X_n)$, given in Theorem 6.1. The result follows, if we recall that the alternating sum appearing in $\mathbf{E}(X_n)$ was evaluated in Example 2.2, and its asymptotic value is given in (2.62). ■

As one may immediately see from Theorems 6.2 and 6.3, the expected value of X_n and of its Poissonized version $X(z)$ are the same. The counterpart of Corollary 6.2 is thus the following.

Corollary 6.3 *Under the fixed population model, the expected number of type 1 buckets in a trie of n keys is*

$$\begin{aligned}\mathbf{E}\left(X^{(1)}(n)\right) \;&=\; n\left(1 - \frac{1}{\ln 2}\sum_{k=2}^{b}\frac{1}{(k-1)2^{k-1}}\right) \;-\; n\left(\sum_{k=2}^{b}\frac{f_{k-1,\,2}(n)}{(k-1)!\,2^{k-1}}\right)\\ &\quad + O(1) \quad \textit{as } n\to\infty,\end{aligned}$$

and the expected numbers of type j buckets, for all $2\le j\le b$, are

$$\begin{aligned}\mathbf{E}\left(X^{(j)}(n)\right) = \frac{n}{\ln 2}&\left(\frac{1}{j(j-1)} - \sum_{k=2}^{b}\frac{\binom{k}{j}}{k(k-1)2^{k-1}}\right) + n\left(\frac{f_{j-1,\,2}(n)}{j!} - \sum_{k=2}^{b}\frac{\binom{k}{j}f_{k-1,\,2}(n)}{k!\,2^{k-1}}\right)\\ &+ O(1) \quad \textit{as } n\to\infty,\end{aligned}$$

where the periodic functions $f_{m,2}(z)$ are defined in (2.3).

b \ j	1	2	3	4	5	6	7	8	9	10
2	.2787	.361								
3	.0983	.180	.180							
4	.0382	.090	.120	.105						
5	.0157	.045	.075	.083	.068					
6	.0066	.023	.045	.060	.059	.047				
7	.0029	.011	.026	.041	.047	.043	.034			
8	.0013	.006	.015	.027	.036	.037	.032	.026		
9	.0006	.003	.009	.017	.026	.031	.029	.025	.020	
10	.0003	.001	.005	.011	.018	.024	.026	.023	.020	.016

Table 6.1: Asymptotic ratio of number of type j buckets, over number of keys

To get a feeling of the average number of different buckets, we give in Table 6.1 several values of the proportionality ratio (ignoring the oscillations)

$$\frac{\mathbf{E}\left(X_n^{(j)}\right)}{n} \sim \begin{cases} 1 - \dfrac{1}{\ln 2} \displaystyle\sum_{k=2}^{b} \frac{1}{(k-1)2^{k-1}} & \text{if } j=1 \\ \dfrac{1}{\ln 2}\left(\dfrac{1}{j(j-1)} - \displaystyle\sum_{k=j}^{b} \frac{\binom{k}{j}}{k(k-1)2^{k-1}}\right) & \text{if } 2\le j\le b \end{cases}$$

of the average number of type j buckets over the number of keys in a trie.

6.5 Variances

Continuing to work as in the PSL search cost case, first we derive a recurrence for $b_n = \mathbf{E}(X_n^2)$.

Lemma 6.3 *Under the fixed population model, the second moment of the linear combination of the numbers of all bucket types, in a trie of n keys, satisfies the recurrence*

$$\begin{cases} b_n = \lambda_n^2 & \text{for} \quad 0\le n\le b \\ b_n = \dfrac{1}{2^{n-1}} \displaystyle\sum_{k=0}^{n} \binom{n}{k} a_k a_{n-k} + \frac{1}{2^{n-1}} \sum_{k=0}^{n} \binom{n}{k} b_k & \text{for} \quad n>b \end{cases} . \qquad (6.13)$$

Proof: If we differentiate the recurrence of Lemma 6.1 twice with respect to z, and we evaluate the resulting expression at $z=0$, we get

$$b_n = \lambda_n^2, \qquad \text{for } 0 \le n \le b$$

and

$$b_n = \frac{1}{2^n} \sum_{k=0}^{n} \binom{n}{k} (2a_k a_{n-k} + b_k + b_{n-k}), \qquad \text{for } n > b,$$

since $M(0)=1$ from (2.9), $M_r'(0)=a_r$ from (2.10), and $M_r''(0)=b_r$ from (2.11). The result now follows, since $\sum_{k=0}^{n} \binom{n}{k} b_k = \sum_{k=0}^{n} \binom{n}{k} b_{n-k}$. ■

We may now derive a recurrence for $B(z)=\mathbf{E}\left(X^2(z)\right)=\mathcal{P}(b_n;z)$. Multiplication of both sides of (6.13) by $e^{-z}\frac{z^n}{n!}$, and addition of the resulting equalities gives

$$\begin{aligned} B(z) &= e^{-z} \sum_{n=0}^{b} \left\{ \lambda_n^2 \frac{z^n}{n!} \right\} + e^{-z} \sum_{n \ge 0} \left\{ \frac{1}{2^{n-1}} \sum_{k=0}^{n} \binom{n}{k} a_k a_{n-k} \frac{z^n}{n!} \right\} \\ &\quad - e^{-z} \sum_{n=0}^{b} \left\{ \frac{1}{2^{n-1}} \sum_{k=0}^{n} \binom{n}{k} a_k a_{n-k} \frac{z^n}{n!} \right\} + 2B\left(\frac{z}{2}\right) - e^{-z} \sum_{n=0}^{b} \left\{ \frac{1}{2^{n-1}} \sum_{k=0}^{n} \binom{n}{k} b_k \frac{z^n}{n!} \right\} \end{aligned}$$

by using (5.9). We now simplify this recurrence. First, we observe that the "$n=0$" and "$n=1$" terms of the three finite sums in the above recurrence cancel out, and we recall that, for $0 \le n \le b$, we have $a_n = \lambda_n$ and $b_n = \lambda_n^2$ from (6.2) and (6.13). Second, since the egf of $\langle a_n \rangle$ is $e^z A(z)$, (2.37) implies that the egf of the binomial convolution of $\langle a_n \rangle$ with itself is $e^{2z}A^2(z)$, and thus

$$\sum_{n \ge 0} \frac{1}{2^n} \sum_{k=0}^{n} \binom{n}{k} a_k a_{n-k} \frac{z^n}{n!} = e^z A^2\left(\frac{z}{2}\right).$$

Therefore, the last recurrence may be simplified as

$$B(z) - 2B\left(\frac{z}{2}\right) = 2A^2\left(\frac{z}{2}\right) + e^{-z} \sum_{k=2}^{b} \tau_k z^k, \tag{6.14}$$

where

$$\tau_k \stackrel{\text{def}}{=} \frac{\lambda_k^2}{k!} - \frac{1}{k!\, 2^{k-1}} \left(\sum_{r=0}^{k} \binom{k}{r} \lambda_r \lambda_{k-r} + \sum_{r=0}^{k} \binom{k}{r} \lambda_r^2 \right), \qquad \text{for } 2 \le k \le b. \tag{6.15}$$

At this point, we could solve (6.14) to get $\mathbf{E}\Big(X^2(z)\Big)$, and we could then derive $\mathbf{Var}\Big(X(z)\Big)$ by using the expression for $\mathbf{E}\Big(X(z)\Big)$ from Theorem 6.2. Instead, we choose to change (6.14) into a recurrence for $V(z)=\mathbf{Var}\Big(X(z)\Big)$, and to solve the newly derived recurrence, obtaining thus $\mathbf{Var}\Big(X(z)\Big)$ directly.

If we rewrite (6.14) as

$$\Big(B(z)-A^2(z)\Big)-2\left(B\left(\frac{z}{2}\right)-A^2\left(\frac{z}{2}\right)\right)=4A^2\left(\frac{z}{2}\right)-A^2(z)+e^{-z}\sum_{k=2}^{b}\tau_k z^k,$$

we may use relations $V(z)=B(z)-A^2(z)$ and (6.3) to rewrite the above as

$$V(z)-2V\left(\frac{z}{2}\right)=e^{-z}w_b(z), \tag{6.16}$$

where

$$w_b(z) \stackrel{\text{def}}{=} e^{-z}y_b^2(z)-2y_b(z)A(z)+\sum_{k=2}^{b}\tau_k z^k. \tag{6.17}$$

We will now solve recurrence (6.16). As we did in the derivation of the variance of the PSL search cost in Theorem 5.3, we will start by deriving an iterative solution of it. The calculations involved are lengthy, but since the techniques that we will use have been used in our thesis a number of times already (Example 2.1, Example 2.2, Theorem 5.3, and Theorem 6.2) we will merely give enough details for the reader to check our steps.

Theorem 6.4 *Under the Poisson model with mean z, the variance of the linear combination of the numbers of all bucket types in a trie is*

$$\begin{aligned}\mathbf{Var}\Big(X(z)\Big) \;=\; & z\left(\lambda_1^2+\frac{1}{\ln 2}\sum_{k=2}^{b}\{\tau_k(k-2)!-2\lambda_1\sigma_k(k-1)!\}\right.\\ & \left.+\frac{2}{\ln 2}\sum_{k=2}^{b}\sum_{r=2}^{b}\left\{\sigma_k\sigma_r(k+r-2)!\left(\frac{1}{2^{k+r}}-\sum_{m\geq 0}\frac{2^{mk}}{(1+2^m)^{k+r-1}}\right)\right\}\right)\\ & +z\left(\sum_{k=2}^{b}\{\tau_k f_{k-1,\,2}(z)-2\lambda_1\sigma_k f_{k,2}(z)\}\right.\end{aligned}$$

$$+2\sum_{k=2}^{b}\sum_{r=2}^{b}\left\{\sigma_k\sigma_r\left(\frac{f_{k+r-1,\,2}(z)}{2^{k+r}}-\sum_{m\geq 0}\frac{2^{mk}\,f_{k+r-1,\,2}(z+2^m z)}{(1+2^m)^{k+r-1}}\right)\right\}\Bigg)$$
$$+O\left(\frac{1}{z^d}\right)\quad \textit{as } z\to\infty,\qquad \textit{for any } d>-1,$$

where the terms σ_m and τ_m are defined in (6.5) and (6.15) respectively, and the periodic functions $f_{m,2}(z)$ are defined in (2.3).

Proof: One may easily verify that an iterative solution of recurrence (6.16) yields

$$V(z)=\lim_{k\to\infty}\left\{2^k V\Big(\frac{z}{2^k}\Big)\right\}+\sum_{k\geq 0}\left\{2^k e^{-\frac{z}{2^k}}w_b\Big(\frac{z}{2^k}\Big)\right\}. \tag{6.18}$$

To compute the appearing limit, we recall that $B(z)=\mathbf{E}\Big(X^2(z)\Big)=\mathcal{P}\Big(\mathbf{E}(X_n^2);z\Big)$, and we may thus rewrite $V(z)=\mathbf{E}\Big(X^2(z)\Big)-\Big\{\mathbf{E}\Big(X(z)\Big)\Big\}^2$ as

$$V(z)=e^{-z}\sum_{n\geq 0}\left\{\mathbf{E}(X_n^2)\frac{z^n}{n!}\right\}-A^2(z).$$

Therefore, the limit in (6.18) may be expressed as

$$\begin{aligned}\lim_{k\to\infty}\left\{2^k V\Big(\frac{z}{2^k}\Big)\right\} &= \lim_{k\to\infty}\left\{2^k e^{-\frac{z}{2^k}}\left(\mathbf{E}(X_1^2)\frac{z}{2^k}+\sum_{n\geq 2}\mathbf{E}(X_n^2)\frac{z^n}{2^{kn}\,n!}\right)\right\}\\ &\quad-\lim_{k\to\infty}\left\{\frac{1}{2^k}\left(2^k A\Big(\frac{z}{2^k}\Big)\right)^2\right\},\end{aligned}$$

and the first of these two new limits is $\lambda_1^2 z$, whereas the second one is 0 by (6.7). Hence, (6.18) gives

$$V(z)=\lambda_1^2 z+R_b(z), \tag{6.19}$$

where

$$R_b(z)\ \stackrel{\text{def}}{=}\ \sum_{k\geq 0}\left\{2^k\,e^{-\frac{z}{2^k}}\,w_b\Big(\frac{z}{2^k}\Big)\right\},$$

and we now have to find a closed form for $R_b(z)$. We do so, by working as with $Q_b(z)$ of (6.9); we should expect though much more complicated computations,

since the $w_b(z)$ of $R_b(z)$ is more complicated than its counterpart $y_b(z)$ of $Q_b(z)$. The Mellin transform of $R_b(z)$ is

$$R_b^*(s) = \mathcal{M}\Big(e^{-z} w_b(z);\ s\Big) \sum_{k\geq 0} (2^{s+1})^k,$$

which, according to the definition (6.17) of w_b, and the convergence of the series above, gives

$$\begin{aligned} R_b^*(s) \ &= \ \frac{1}{1-2^{s+1}}\bigg(\mathcal{M}\Big(e^{-2z}\, y_b^2(z);\ s\Big) - 2\mathcal{M}\Big(e^{-z}\, y_b(z)\, A(z);\ s\Big) \\ &\qquad + \mathcal{M}\Big(e^{-z}\sum_{k=2}^{b} \tau_k z^k;\ s\Big)\bigg), \qquad \text{for } \Re(s) < -1. \end{aligned} \tag{6.20}$$

The first of the three Mellin transforms appearing above, may be computed as

$$\begin{aligned} \mathcal{M}\Big(e^{-2z} y_b^2(z);\ s\Big) \ &= \ \int_0^\infty e^{-2z}\, y_b^2(z)\, z^{s-1}\, dz && \text{by (2.43)} \\ &= \ \int_0^\infty e^{-2z}\bigg(\sum_{k=2}^{b} \sigma_k z^k\bigg)^2 z^{s-1}\, dz && \text{by (6.4)} \\ &= \ \int_0^\infty e^{-2z}\bigg(\sum_{k=2}^{b}\sum_{r=2}^{b} \sigma_k\sigma_r z^{k+r}\bigg) z^{s-1}\, dz \\ &= \ \frac{1}{2}\sum_{k=2}^{b}\sum_{r=2}^{b} \sigma_k\sigma_r \int_0^\infty e^{-x}\left(\frac{x}{2}\right)^{k+r+s-1} dx, && \text{by } x \leftarrow 2z \end{aligned}$$

which, according to the definition (2.1) of the Gamma function gives

$$\mathcal{M}\Big(e^{-2z} y_b^2(z);\ s\Big) = \sum_{k=2}^{b}\sum_{r=2}^{b} \frac{\sigma_k\sigma_r}{2^{k+r+s}} \Gamma(k+r+s), \qquad \text{for } \Re(s) > -4. \tag{6.21}$$

The second Mellin transform in (6.20) is the toughest one to compute, since it contains $A(z)$. We will compute it term-by-term, as in (5.23). If we multiply $y_b(z)$ and $A(z)$, as given in (6.4) and (6.8) respectively, we get

$$\begin{aligned} &\mathcal{M}\Big(e^{-z} y_b(z) A(z);\ s\Big) \\ &= \ \lambda_1 \int_0^\infty e^{-z} \sum_{k=2}^{b} \sigma_k z^{k+s}\, dz \ + \ \int_0^\infty e^{-z} \sum_{k=2}^{b}\sum_{r=2}^{2}\sum_{m\geq 0} \sigma_k\sigma_r 2^{m(1-r)} e^{-\frac{z}{2^m}}\, z^{r+k+s-1}\, dz. \end{aligned}$$

But the first integral in this expression is

$$\begin{aligned}\int_0^\infty e^{-z}\sum_{k=2}^{b}\sigma_k z^{k+s}\,dz &= \sum_{k=2}^{b}\sigma_k\int_0^\infty e^{-z}z^{k+s}\,dz\\ &= \sum_{k=2}^{b}\sigma_k\,\Gamma(k+s+1), \qquad \text{for } \Re(s)>-3,\end{aligned}$$

and the second integral is

$$\begin{aligned}&\int_0^\infty e^{-z}\sum_{k=2}^{b}\sum_{r=2}^{b}\sum_{m\ge 0}\sigma_k\sigma_r 2^{m(1-r)}e^{-\frac{z}{2^m}}z^{r+k+s-1}\,dz\\ &= \sum_{k=2}^{b}\sum_{r=2}^{b}\sum_{m\ge 0}\sigma_k\sigma_r 2^{m(1-r)}\int_0^\infty e^{-z(1+2^{-m})}z^{r+k+s-1}\,dz\\ &= \sum_{k=2}^{b}\sum_{r=2}^{b}\sum_{m\ge 0}\sigma_k\sigma_r\frac{2^{m(1-r)}}{(1+2^{-m})^{k+r+s}}\int_0^\infty e^{-x}x^{k+r+s-1}\,dx\\ &= \sum_{k=2}^{b}\sum_{r=2}^{b}\sum_{m\ge 0}\sigma_k\sigma_r\frac{2^{m(k+s+1)}}{(1+2^{m})^{k+r+s}}\Gamma(k+r+s), \qquad \text{for } \Re(s)>-4,\end{aligned}$$

and thus

$$\begin{aligned}\mathcal{M}\Big(e^{-z}y_b(z)A(z);\ s\Big) &= \sum_{k=2}^{b}\sum_{r=2}^{b}\sigma_k\sigma_r\Gamma(k+r+s)\sum_{m\ge 0}\frac{2^{m(k+s+1)}}{(1+2^{m})^{k+r+s}}\\ &\quad+\lambda_1\sum_{k=2}^{b}\sigma_k\,\Gamma(k+s+1), \qquad \text{for } \Re(s)>-3. \quad (6.22)\end{aligned}$$

The third Mellin transform in (6.20) may be easily seen to be

$$\mathcal{M}\left(e^{-z}\sum_{k=2}^{b}\tau_k z^k;\ s\right)=\sum_{k=2}^{b}\tau_k\Gamma(k+s), \qquad \text{for } \Re(s)>-2. \quad (6.23)$$

Putting now together (6.20), (6.21), (6.22), and (6.23), we get

$$\begin{aligned}R_b^*(s) &= \frac{1}{1-2^{s+1}}\Bigg(\sum_{k=2}^{b}\tau_k\,\Gamma(k+s)\ -\ 2\lambda_1\sum_{k=2}^{b}\sigma_k\,\Gamma(k+s+1)\\ &\qquad -2\sum_{k=2}^{b}\sum_{r=2}^{b}\sigma_k\sigma_r\Gamma(k+r+s)\sum_{m\ge 0}\frac{2^{m(k+s+1)}}{(1+2^{m})^{k+r+s}}\\ &\qquad +\sum_{k=2}^{b}\sum_{r=2}^{b}\frac{\sigma_k\sigma_r}{2^{k+r+s}}\Gamma(k+r+s)\Bigg), \text{ for } -2<\Re(s)<-1. (6.24)\end{aligned}$$

As we have done a number of times before, from the inverse Mellin transform (2.46) we now get

$$R_b(z) = -\sum_{\alpha=\text{pole inside } C} \operatorname*{Res}_{s=\alpha}\Big\{R_b^*(s)z^{-s}\Big\} \;+\; O\Big(\frac{1}{z^d}\Big) \quad \text{as } z\to\infty, \quad \text{for any } d>-1,$$

where C is an infinitely tall rectangle to the right of the vertical line $\Re(s)=c$, for some $-2<c<-1$. Inside C, the only poles of $R_b^*(s)z^{-s}$ are at $s=s_v \stackrel{\text{def}}{=} -1+i\,\frac{2v\pi}{\ln 2}$, for $v=0,\pm1,\pm2,\ldots$, and therefore, the last relation becomes

$$\begin{aligned} R_b(z) \;\;=\;\; -\operatorname*{Res}_{s=-1}\Big\{R_b^*(s)z^{-s}\Big\} \;-\sum_{v\in\mathbb{Z}-\{0\}} \operatorname*{Res}_{s=s_v}\Big\{R_b^*(s)z^{-s}\Big\} + O\Big(\frac{1}{z^d}\Big) \quad \text{as } z\to\infty, \\ \text{for any } d>-1. \end{aligned} \tag{6.25}$$

We now use (2.48) to routinely compute the above residues. For the simple pole $s=-1$, we immediately get

$$\begin{aligned} &\operatorname*{Res}_{s=-1}\Big\{R_b^*(s)z^{-s}\Big\} \\ &\quad= -\frac{z}{\ln 2}\Bigg(\sum_{k=2}^{b}\Big\{\tau_k(k-2)! - 2\lambda_1\sigma_k(k-1)!\Big\} \\ &\qquad\qquad + 2\sum_{k=2}^{b}\sum_{r=2}^{b}\Bigg\{\sigma_k\sigma_r(k+r-2)!\Big(\frac{1}{2^{k+r}} - \sum_{m\ge 0}\frac{2^{mk}}{(1+2^m)^{k+r-1}}\Big)\Bigg\}\Bigg), \end{aligned} \tag{6.26}$$

by using (2.18). At the simple poles $s=s_v$, for $v=\pm1,\pm2,\ldots$, we compute the residues of each of the four summands of $R_b^*(s)z^{-s}$ separately. For the first summand we have

$$\begin{aligned} \sum_{v\in\mathbb{Z}-\{0\}}\operatorname*{Res}_{s=s_v}\Bigg\{\frac{1}{1-2^{s+1}}\sum_{k=2}^{b}\tau_k\Gamma(k+s)z^{-s}\Bigg\} \;&=\; -\frac{z}{\ln 2}\sum_{k=2}^{b}\tau_k\sum_{v\in\mathbb{Z}-\{0\}}\Gamma\Big(k-1+i\frac{2v\pi}{\ln 2}\Big)\,z^{-i\,\frac{2v\pi}{\ln 2}} \\ &=\; -z\sum_{k=2}^{b}\tau_k f_{k-1,\,2}(z), \end{aligned}$$

for the second summand we have

$$\sum_{v\in\mathbb{Z}-\{0\}}\operatorname*{Res}_{s=s_v}\Bigg\{\frac{1}{1-2^{s+1}}\sum_{k=2}^{b}\sigma_k\Gamma(k+s+1)z^{-s}\Bigg\} \;=\; -\frac{z}{\ln 2}\sum_{k=2}^{b}\sigma_k\sum_{v\in\mathbb{Z}-\{0\}}\Gamma\Big(k+i\frac{2v\pi}{\ln 2}\Big)\,z^{-i\,\frac{2v\pi}{\ln 2}}$$

$$= -z\sum_{k=2}^{b}\sigma_k f_{k,2}(z),$$

for the third summand we have

$$\begin{aligned}
&\sum_{v\in\mathbb{Z}-\{0\}}\operatorname*{Res}_{s=s_v}\left\{\frac{1}{1-2^{s+1}}\sum_{k=2}^{b}\sum_{r=2}^{b}\sigma_k\sigma_r\Gamma(k+r+s)\sum_{m\geq 0}\frac{2^{m(k+s+1)}}{(1+2^m)^{k+r+s}}\,z^{-s}\right\}\\
&= -\frac{z}{\ln 2}\sum_{k=2}^{b}\sum_{r=2}^{b}\sigma_k\sigma_r\sum_{m\geq 0}\frac{2^{mk}}{(1+2^m)^{k+r-1}}\sum_{v\in\mathbb{Z}-\{0\}}\Gamma\Big(k+r-1+i\frac{2v\pi}{\ln 2}\Big)\frac{\left(2^{i\frac{2v\pi}{\ln 2}}\right)^m}{(1+2^m)^{i\frac{2v\pi}{\ln 2}}}\,z^{-i\frac{2v\pi}{\ln 2}}\\
&= -z\sum_{k=2}^{b}\sum_{r=2}^{b}\sigma_k\sigma_r\sum_{m\geq 0}\frac{2^{mk}}{(1+2^m)^{k+r-1}}\,f_{k+r-1,\,2}(z+2^m z),
\end{aligned}$$

since $2^{i\frac{2v\pi}{\ln 2}}=1$ (by $1-2^{s_v+1}=0$), and, finally, for the fourth summand we have

$$\begin{aligned}
&\sum_{v\in\mathbb{Z}-\{0\}}\operatorname*{Res}_{s=s_v}\left\{\frac{1}{1-2^{s+1}}\sum_{k=2}^{b}\sum_{r=2}^{b}\frac{\sigma_k\sigma_r}{2^{k+r+s}}\Gamma(k+r+s)z^{-s}\right\}\\
&= -\frac{z}{\ln 2}\sum_{k=2}^{b}\sum_{r=2}^{b}\frac{\sigma_k\sigma_r}{2^{k+r-1}}\sum_{v\in\mathbb{Z}-\{0\}}\left\{\frac{1}{2^{i\frac{2v\pi}{\ln 2}}}\Gamma\Big(k+r-1+i\,\frac{2v\pi}{\ln 2}\Big)\,z^{-i\frac{2v\pi}{\ln 2}}\right\}\\
&= -z\sum_{k=2}^{b}\sum_{r=2}^{b}\left\{\frac{\sigma_k\sigma_r}{2^{k+r-1}}f_{k+r-1,\,2}(z)\right\}.
\end{aligned}$$

Hence, the total contribution of all poles at $s=s_v$ is

$$\begin{aligned}
&\sum_{v\in\mathbb{Z}-\{0\}}\operatorname*{Res}_{s=s_v}\Big\{R_b^*(s)z^{-s}\Big\}\\
&= -z\Bigg(\sum_{k=2}^{b}\Big\{\tau_k f_{k-1,\,2}(z)-2\lambda_1\sigma_k f_{k,2}(z)\Big\}\\
&\qquad +2\sum_{k=2}^{b}\sum_{r=2}^{b}\sigma_k\sigma_r\Big(\frac{f_{k+r-1,\,2}(z)}{2^{k+r}}-\sum_{m\geq 0}\frac{2^{mk}\,f_{k+r-1,\,2}(z+2^m z)}{(1+2^m)^{k+r-1}}\Big)\Bigg). \qquad (6.27)
\end{aligned}$$

The result now follows from (6.19), (6.25), (6.26) and (6.27). ■

To derive the (more interesting) $\mathbf{Var}\Big(X^{(j)}(z)\Big)$, for all $1\leq j\leq b$, we have to simplify the expressions for τ_k, when only one of the λ_j's is 1. To be precise, let $\tau_k^{(j)}$ be the value of τ_k when $\lambda_j=1$ for some $1\leq j\leq b$ and $\lambda_r=0$ for all $r\neq j$. Then,

from the definition (6.15) of τ_k, we have

$$\tau_k^{(j)} = \frac{\delta_{j,k}}{k!} - \frac{1+\delta_{k,2j}}{k!\,2^{k-1}}\binom{k}{j} \qquad \text{for } 2\le k\le b, \quad 1\le j\le b, \tag{6.28}$$

since the only non-zero term of the first sum in (6.15) occurs when $j=r$ and $k-j=r$, that is, when $k=2j$. To present our formulae for the variances (and covariances) in a more compact way, we will use the terms

$$\Lambda_{k,r} \stackrel{\text{def}}{=} (k+r-2)!\left(\frac{1}{2^{k+r}} - \sum_{m\ge 0}\frac{2^{mk}}{(1+2^m)^{k+r-1}}\right), \qquad \text{integers } k,r\ge 1 \tag{6.29}$$

and

$$\begin{aligned}\tilde{\Lambda}_{k,r} &\stackrel{\text{def}}{=} \Lambda_{k,r} + \Lambda_{r,k} \\ &= (k+r-2)!\left(\frac{1}{2^{k+r-1}} - \sum_{m\ge 0}\frac{2^{mk}+2^{mr}}{(1+2^m)^{k+r-1}}\right), \quad \text{integers } k,r\ge 1,\end{aligned} \tag{6.30}$$

and the oscillating functions

$$\Phi_{k,r}(z) \stackrel{\text{def}}{=} \frac{f_{k+r-1,\,2}(z)}{2^{k+r}} - \sum_{m\ge 0}\frac{2^{mk}\,f_{k+r-1,\,2}(z+2^m z)}{(1+2^m)^{k+r-1}}, \quad \text{integers } k,r\ge 1 \tag{6.31}$$

and

$$\begin{aligned}\tilde{\Phi}_{k,r}(z) &\stackrel{\text{def}}{=} \Phi_{k,r}(z) + \Phi_{r,k}(z) \\ &= \frac{f_{k+r-1,\,2}(z)}{2^{k+r-1}} - \sum_{m\ge 0}\frac{(2^{mk}+2^{mr})f_{k+r-1,2}(z+2^m z)}{(1+2^m)^{k+r-1}}, \quad \text{integers } k,r\ge 1\end{aligned} \tag{6.32}$$

in them.

Corollary 6.4 *Under the Poisson model with mean z, the variance of the number of type 1 buckets in a trie is*

$$\begin{aligned}&\mathbf{Var}\Big(X^{(1)}(z)\Big) \\ &= z\left(1 + \frac{1}{\ln 2}\left(\frac{3}{2} - \frac{1}{2^{b-2}} - \sum_{k=2}^{b}\frac{1}{(k-1)2^{k-1}} + \sum_{k=2}^{b}\sum_{r=2}^{b}\frac{\Lambda_{k,r}}{(k-1)!\,(r-1)!\,2^{k+r-3}}\right)\right) \\ &\quad + z\left(-\frac{f_{1,2}(z)}{2} + \sum_{k=2}^{b}\frac{2f_{k,2}(z)-f_{k-1,\,2}(z)}{(k-1)!\,2^{k-1}} + \sum_{k=2}^{b}\sum_{r=2}^{b}\frac{\Phi_{k,r}(z)}{(k-1)!\,(r-1)!\,2^{k+r-3}}\right) \\ &\quad + O\Big(\frac{1}{z^d}\Big) \qquad \textit{as } z\to\infty, \quad \textit{for any } d>-1,\end{aligned}$$

and the variances of the numbers of type j buckets, for all $2 \leq j \leq b$, are

$$\begin{aligned}
&\mathbf{Var}\Big(X^{(j)}(z)\Big) \\
&= \frac{z}{\ln 2}\Bigg(\frac{1}{j(j-1)} + \frac{2\Lambda_{j,j}}{j!\,j!} - \frac{\binom{2j}{j}}{j(2j-1)4^j}\,[\,b \geq 2j\,] - \sum_{k=2}^{b} \frac{\binom{k}{j}}{k(k-1)2^{k-1}} \\
&\qquad - \frac{1}{j!}\sum_{k=2}^{b} \frac{\binom{k}{j}\tilde{\Lambda}_{k,j}}{k!\,2^{k-2}} + \sum_{k=2}^{b}\sum_{r=2}^{b} \frac{\binom{k}{j}\binom{r}{j}\Lambda_{k,r}}{k!\,r!\,2^{k+r-3}}\Bigg) \\
&\quad + z\Bigg(\frac{f_{j-1,\,2}(z)}{j!} + \frac{2\Phi_{j,j}(z)}{j!\,j!} - \frac{2\binom{2j}{j} f_{2j-1,\,2}(z)}{(2j)!\,4^j}\,[\,b \geq 2j\,] - \sum_{k=2}^{b} \frac{\binom{k}{j} f_{k-1,\,2}(z)}{k!\,2^{k-1}} \\
&\qquad - \frac{1}{j!}\sum_{k=2}^{b} \frac{\binom{k}{j}\tilde{\Phi}_{k,j}(z)}{k!\,2^{k-2}} + \sum_{k=2}^{b}\sum_{r=2}^{b} \frac{\binom{k}{j}\binom{r}{j}\Phi_{k,r}(z)}{k!\,r!\,2^{k+r-3}}\Bigg) \\
&\quad + O\Big(\frac{1}{z^d}\Big) \qquad \textit{as } z \to \infty, \quad \textit{for any } d > -1,
\end{aligned}$$

where the terms $\Lambda_{k,r}$ and $\tilde{\Lambda}_{k,r}$ are defined in (6.29) and (6.30), and the oscillating functions $\Phi_{k,r}(z)$, $\tilde{\Phi}_{k,r}(z)$ and $f_{m,2}(z)$ are defined in (6.31), (6.32) and (2.3).

Proof: Both expressions are derived from $\mathbf{Var}\Big(X(z)\Big)$, given in Theorem 6.4, by setting $\tau_k \leftarrow \tau_k^{(j)}$ from (6.28), and $\sigma_k \leftarrow \sigma_k^{(j)}$ from (6.6). Recall that we must also set $\lambda_1 \leftarrow 1$ and $\lambda_1 \leftarrow 0$ for the derivation of $\mathbf{Var}\Big(X^{(1)}(z)\Big)$ and $\mathbf{Var}\Big(X^{(j)}(z)\Big)$ respectively. The results follow after some algebraic simplification. ■

Finally, we may derive the covariances between the numbers of any two bucket types, from

$$\mathbf{Cov}\Big(X^{(j)}(z),\, X^{(l)}(z)\Big) = \frac{1}{2}\Big(\mathbf{Var}\Big(X^{(j)}(z) + X^{(l)}(z)\Big) - \mathbf{Var}\Big(X^{(j)}(z)\Big) - \mathbf{Var}\Big(X^{(l)}(z)\Big)\Big). \tag{6.33}$$

To make use of that, we have to compute $\mathbf{Var}\Big(X^{(j)}(z) + X^{(l)}(z)\Big)$, for all $1 \leq j \neq l \leq b$. But this can be easily done, by setting $\lambda_j = \lambda_l = 1$, and $\lambda_r = 0$ for all $r \neq j, l$, in $\mathbf{Var}\Big(X(z)\Big)$. If $\sigma_k^{(j,l)}$ and $\tau_k^{(j,l)}$ are the values of σ_k and τ_k for the just mentioned

values of $\lambda_1, \lambda_2, \ldots, \lambda_b$, we may derive without much effort

$$\sigma_k^{(j,l)} = \frac{\delta_{k,j} + \delta_{k,l}}{k!} - \frac{\binom{k}{j} + \binom{k}{l}}{k!\,2^{k-1}}, \qquad \text{integers } 2 \le k \le b, \quad 1 \le j \ne l \le b \tag{6.34}$$

from the definition (6.5) of σ_k, and

$$\tau_k^{(j,l)} = \frac{\delta_{k,j} + \delta_{k,l}}{k!} - \frac{\binom{k}{j} + \binom{k}{l} + 2\binom{k}{j}\delta_{k,\,j+l} + \binom{k}{j}\delta_{k,\,2j} + \binom{k}{l}\delta_{k,\,2l}}{k!\,2^{k-1}}, \qquad \text{integers } 2 \le k \le b, \quad 1 \le j \ne l \le b \tag{6.35}$$

from the definition (6.15) of τ_k.

Corollary 6.5 *Under the Poisson model with mean z, the covariances between the numbers of buckets of types 1 and j, for all $2 \le j \le b$, in a trie are*

$$\begin{aligned}
&\mathbf{Cov}\left(X^{(1)}(z), X^{(j)}(z)\right) \\
&= \frac{z}{\ln 2}\Bigg(- \frac{1}{j2^j}\,[\,b \ge j{+}1\,] \;-\; \frac{1}{j} \;+\; \sum_{k=2}^{b} \frac{\binom{k}{j}}{k2^{k-1}} \\
&\qquad - \frac{1}{j!}\sum_{k=2}^{b} \frac{\tilde{\Lambda}_{k,j}}{(k-1)!\,2^{k-1}} \;+\; \sum_{k=2}^{b}\sum_{r=2}^{b} \frac{\binom{k}{j}\tilde{\Lambda}_{k,r}}{k!\,(r-1)!\,2^{k+r-2}}\Bigg) \\
&\quad + z\Bigg(- \frac{f_{j,2}(z)}{j!\,2^j}\,[\,b \ge j{+}1\,] \;-\; \frac{f_{j,2}(z)}{j!} \;+\; \sum_{k=2}^{b} \frac{\binom{k}{j} f_{k,2}(z)}{k!\,2^{k-1}} \\
&\qquad - \frac{1}{j!}\sum_{k=2}^{b} \frac{\tilde{\Phi}_{k,j}(z)}{(k-1)!\,2^{k-1}} \;+\; \sum_{k=2}^{b}\sum_{r=2}^{b} \frac{\binom{k}{j}\tilde{\Phi}_{k,r}(z)}{k!\,(r-1)!\,2^{k+r-2}}\Bigg) \\
&\quad + O\left(\frac{1}{z^d}\right) \qquad \textit{as } z \to \infty, \quad \textit{for any } d > -1,
\end{aligned}$$

and the covariances between the numbers of buckets of types j and l, for all $2 \le j \ne l \le b$, are

$$\begin{aligned}
&\mathbf{Cov}\left(X^{(j)}(z), X^{(l)}(z)\right) \\
&= \frac{z}{\ln 2}\Bigg(- \frac{\binom{l+j}{j}}{(l+j)(l+j-1)2^{l+j-1}}\,[\,b \ge j{+}l\,] \;+\; \frac{\tilde{\Lambda}_{j,l}}{j!\,l!} \;-\; \frac{1}{j!}\sum_{k=2}^{b} \frac{\binom{k}{l}\tilde{\Lambda}_{k,j}}{k!\,2^{k-1}}
\end{aligned}$$

$$
\begin{aligned}
&- \frac{1}{l!}\sum_{k=2}^{b}\frac{\binom{k}{j}\tilde{\Lambda}_{k,l}}{k!\,2^{k-1}} + \sum_{k=2}^{b}\sum_{r=2}^{b}\frac{\binom{k}{j}\binom{r}{l}\tilde{\Lambda}_{k,r}}{k!\,r!\,2^{k+r-2}}\Big) \\
&+ z\Big(- \frac{\binom{l+j}{j} f_{l+j-1,2}(z)}{(l+j)!\,2^{l+j-1}}\,[\,b \geq j+l\,] + \frac{\tilde{\Phi}_{j,l}(z)}{j!\,l!} - \frac{1}{j!}\sum_{k=2}^{b}\frac{\binom{k}{l}\tilde{\Phi}_{k,j}(z)}{k!\,2^{k-1}} \\
&\quad - \frac{1}{l!}\sum_{k=2}^{b}\frac{\binom{k}{j}\tilde{\Phi}_{k,l}(z)}{k!\,2^{k-1}} + \sum_{k=2}^{b}\sum_{r=2}^{b}\frac{\binom{k}{j}\binom{r}{l}\tilde{\Phi}_{k,r}(z)}{k!\,r!\,2^{k+r-2}}\Big) \\
&+ O\Big(\frac{1}{z^d}\Big) \qquad \textit{as } z\to\infty, \quad \textit{for any } d>-1,
\end{aligned}
$$

where the terms $\tilde{\Lambda}_{k,r}$ are defined in (6.30), and the oscillating functions $\tilde{\Phi}_{k,r}(z)$ and $f_{m,2}(z)$ are defined in (6.32) and (2.3) respectively.

Proof: To derive $\mathbf{Cov}\left(X^{(1)}(z), X^{(j)}(z)\right)$, we compute $\mathbf{Var}\left(X^{(1)}(z)+X^{(j)}(z)\right)$ from $\mathbf{Var}\big(X(z)\big)$ given in Theorem 6.4, by setting $\lambda_1 \leftarrow 1$, $\sigma_k \leftarrow \sigma_k^{(1,j)}$ from (6.34), and $\tau_k \leftarrow \tau_k^{(1,j)}$ from (6.35). Next, we use identity (6.33), in which the appearing $\mathbf{Var}\left(X^{(1)}(z)\right)$ and $\mathbf{Var}\left(X^{(j)}(z)\right)$ are given by Corollary 6.4. The involved calculations are very lengthy, but nevertheless straightforward. The derivation of $\mathbf{Cov}\left(X^{(j)}(z), X^{(l)}(z)\right)$ is done similarly. ■

6.6 Remarks on Fixed-Bucket Tries

Under the fixed population model, let

$$Y_n \stackrel{\text{def}}{=} X_n^{(1)} + X_n^{(2)} + \cdots + X_n^{(b)}$$

be the total number of non-empty buckets in a trie of n keys. Since

$$\mathbf{E}(Y_n) = \sum_{j=1}^{b}\mathbf{E}\Big(X_n^{(j)}\Big)\,,$$

we may derive $\mathbf{E}(Y_n)$, without much effort, from our results of Section 6.4. Notice here that $\mathbf{E}(Y_n)$ does *not* follow from the expected number of internal nodes (6.1) in a trie, since Y_n does not include the empty buckets.

Theorem 6.5 *Under the fixed population model, the expected number of non-empty fixed-size buckets, in a trie of n keys, is*

$$\mathbf{E}(Y_n) \;=\; n \;-\; \sum_{k=2}^{n}\binom{n}{k}(-1)^k\frac{1}{1-2^{1-k}}\sum_{r=2}^{b}\binom{k}{r}(-1)^r\left(1-\frac{1}{2^{r-1}}\right), \qquad \textit{for } n\geq 1$$

or

$$\begin{aligned}\mathbf{E}(Y_n) \;&=\; n\left(1-\frac{1}{\ln 2}\left(1-\frac{1}{b}-\sum_{k=2}^{b}\frac{1}{k(k-1)2^{k-1}}\right)\right) \;-\; n\sum_{k=2}^{b}\left(1-\frac{1}{2^{k-1}}\right)\frac{f_{k-1,\,2}(n)}{k!}\\ &\quad + O(1) \quad \textit{as } n\to\infty,\end{aligned}$$

where the periodic functions $f_{m,2}(z)$ are defined in (2.3).

Proof: The exact expression for $\mathbf{E}(Y_n)$ follows immediately from Theorem 6.1, since, when $\lambda_1=\lambda_2=\cdots=\lambda_b=1$, the σ_k's become

$$\begin{aligned}\sigma_k \;&=\; \frac{1}{k!}-\frac{1}{k!\,2^{k-1}}\sum_{r=1}^{k}\binom{k}{r} \qquad \text{by (6.5)}\\ &=\; \frac{1}{k!}\left(1-\frac{1}{2^{k-1}}(2^k-1)\right), \quad \text{by binomial theorem}\end{aligned}$$

that is,

$$\sigma_k = -\frac{1}{k!}\left(1-\frac{1}{2^{k-1}}\right), \qquad \text{for } \lambda_1=\lambda_2=\cdots=\lambda_b=1.$$

The asymptotic expression for $\mathbf{E}(Y_n)$ follows similarly from Theorem 6.3, since the σ_k's above give

$$\begin{aligned}\mathbf{E}(Y_n) \;&=\; n\left(1-\frac{1}{\ln 2}\sum_{k=2}^{b}\frac{1}{k(k-1)}\left(1-\frac{1}{2^{k-1}}\right)\right) \;-\; n\sum_{k=2}^{b}\frac{1}{k!}\left(1-\frac{1}{2^{k-1}}\right)f_{k-1,\,2}(n)\\ &\quad + O(1) \quad \text{as } n\to\infty,\end{aligned}$$

and the first sum appearing above becomes

$$\begin{aligned}\sum_{k=2}^{b}\frac{1}{k(k-1)}\left(1-\frac{1}{2^{k-1}}\right) \;&=\; \sum_{k=2}^{b}\frac{1}{k(k-1)}-\sum_{k=2}^{b}\frac{1}{k(k-1)2^{k-1}}\\ &=\; \sum_{k=2}^{b}\left\{\frac{1}{k-1}-\frac{1}{k}\right\}-\sum_{k=2}^{b}\frac{1}{k(k-1)2^{k-1}}\\ &=\; 1-\frac{1}{b}-\sum_{k=2}^{b}\frac{1}{k(k-1)2^{k-1}},\end{aligned}$$

which proves the result. ■

At this point, we may note that although no closed formula exists for the sum appearing in

$$x_b \stackrel{\text{def}}{=} 1 - \frac{1}{\ln 2}\left(1 - \frac{1}{b} - \sum_{k=2}^{b} \frac{1}{k(k-1)2^{k-1}}\right),$$

the non-oscillatory part of the coefficient of n in the expected number of fixed buckets, this sum converges very rapidly; for example, for all $b \geq 8$, it is .3068 when computed to a 4-digit accuracy.

An interesting question that we are now in a position to answer is the following: how much bucket space is unused in a fixed-bucket trie? We know that the used space (number of filled key slots) is exactly n. Therefore, to get an approximate answer, it suffices to compute nbx_b. The next corollary shows that the bucket space utilization is about 69%.

Corollary 6.6 *Under the fixed population model, the expected space taken by all non-empty fixed-size buckets in a trie of n keys, is proportional to n as $n \to \infty$, with an approximate proportionality factor of*

$$bx_b = \frac{1}{\ln 2} + O\left(\frac{1}{b2^b}\right) \approx 1.44269 + O\left(\frac{1}{b2^b}\right) \qquad as \quad b \to \infty.$$

Proof: If we rewrite the finite sum appearing in x_b as the difference of two infinite sums, we get

$$bx_b = \frac{1}{\ln 2} + b\left(1 - \frac{1}{\ln 2} + \frac{1}{\ln 2}\left(\sum_{k\geq 2} \frac{1}{k(k-1)2^{k-1}} - \sum_{k\geq b+1} \frac{1}{k(k-1)2^{k-1}}\right)\right).$$

But the first sum in the last expression is $1-\ln 2$ by the geometric series, and the second sum is $O\left(\frac{1}{b2^b}\right)$ as $b \to \infty$. The result now follows. ■

6.7 Summary

In this chapter, we examined the number of buckets of sizes $1, 2, \ldots, b$ in an elastic-bucket trie, by using the mathematical techniques of the previous chapter. We fully analyzed the expected values; we gave an exact (Corollary 6.1) and an asymptotic (Theorem 6.3 and Corollary 6.2) expression for the expected number of buckets of each size under the fixed population model, and we observed (Theorems 6.2 and 6.3) that the asymptotic expected values under the fixed population model and under the Poisson model are the same. As a byproduct of these results, we obtained (Theorem 6.5) the expected number of non-empty buckets in a (fixed- or elastic-) bucket trie, and we noted that about 69% of the bucket space is occupied by keys. Finally, we derived (Theorem 6.4) asymptotic expressions for the variances and covariances of the number of buckets of all sizes under the Poisson model. All expected values, variances and covariances are linear in the number of keys in the trie.

In view of our results, we may conclude that, under the Poisson model, the number of buckets of each size stays close to its expected value with high probability. This is probably true under the fixed population model as well, since known results on the number of internal nodes of tries (end of Section 5.4) suggest that the variances and covariances under the fixed population model will most probably be linear as well, but with a smaller constant in the leading term.

Finally, we may observe that our results hold for (fixed- and elastic-) bucket Patricia tries as well. To see why this is true, it suffices to recall that the Patricia trie is the same as a trie, but with all nodes of outdegree 1 eliminated. This implies that the pattern of external nodes in both schemes is identical.

Chapter 7

Concluding Remarks

The main objective of this thesis was to investigate the PSL dictionary implementation in two ways. First, to obtain additional results on its average case logarithmic time complexity, and second, to modify it into a structure with worst case logarithmic time complexity. The techniques used to achieve the former, allowed us to derive new results on another dictionary implementation, the elastic-bucket trie.

7.1 Summary of Results

All our probabilistic analyses were done under the (common) fixed population model of randomness and/or the Poisson model of randomness. The main contributions of our thesis are the following.

First, we showed (Theorem 3.3) that the expected value of the cost for the search of the mth key in a PSL of n keys is *exactly* $\log_{\frac{1}{p}} n + \frac{q}{p}\log_{\frac{1}{p}} m + \text{constant} + O(\frac{1}{m})$, and we gave a precise expression of the appearing "constant" term. This is an improvement over Pugh's [39] previously known *upper bound* of the same average

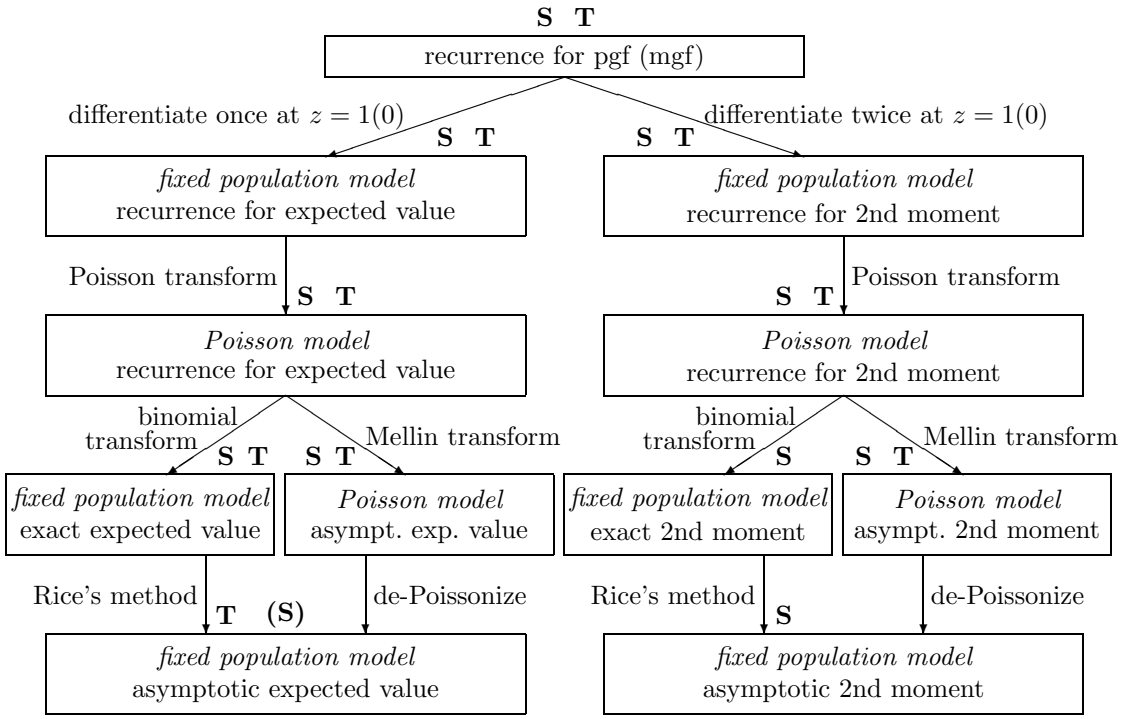


Figure 7.1: Derivation of our probabilistic results

search cost. Our results were derived, initially (Chapter 3) by appealing to some known results on tries, and subsequently (Chapter 5) by using more advanced techniques. As a consequence of our results, we also derived (Theorem 3.4) the expected value of the cost for the search of the average key in a PSL.

Second, we showed (Corollary 5.3) that the variance of the cost for the search of $+\infty$ in a PSL of n keys is $\frac{q}{p^2}\log_{\frac{1}{p}} n + \text{constant} + O\left(\frac{\ln n}{n}\right)$, and we gave a precise expression for the appearing "constant" term.

Third, we presented (Chapter 4) several versions of DSLs, having guaranteed worst case logarithmic search and update costs. These new structures lead to a "bridge" between the original PSL and the standard search trees, and — from a

practical point of view — they are simpler to implement than existing balanced search trees, and very competitive in terms of time and space.

Fourth, we analyzed (Chapter 6) the space complexity in the elastic-bucket trie. More precisely, we considered the number of buckets of each size in this structure, and we showed that each such number has a linear expected value under the fixed population model (Corollary 6.3), and a linear variance under the Poisson model (Corollary 6.4). Our conjecture is that the variance under the fixed population model will remain linear, but with a smaller proportionality factor.

The above probabilistic results, both for the PSL and for the bucket trie, were obtained not only by using the same analytical techniques (exponential, probability and moment generating functions; binomial, Poisson and Mellin transforms; Rice's method for evaluating alternating sums), but also by working in a completely analogous manner. The followed sequence of steps is summarized in Figure 7.1. The derived results for the PSL and the trie are marked by **S** and **T** respectively; a **(S)** denotes a PSL derived result by different means.

7.2 Open Problems

The DSLs of Chapter 4, open up a number of interesting research topics, stemming from analogous questions posed for standard search trees. For example, how can the skip list be extended to handle other dictionary operations, like deletion of minimum key, or splitting, concatenation, intersection and difference of two skip lists? How can the skip list be changed into a simpler structure with amortized logarithmic costs? How can it be modified to handle multidimensional data? How can it become a persistent data structure? How expensive may a search get, if insertions are to be faster? The latter may translate into determining, for example,

the height of the skip list if only $O(1)$ (as opposed to $O(\ln n)$ in our proposed solution) elements are allowed to be raised per insertion.

Other questions on DSLs are pertinent to our proposed versions. For example, can the extra pointers (in our linked list implementation) be used more advantageously than simply "threading" the structure, making it thus simpler to understand and implement?

For the probabilistic analysis of the PSL, the derivation of the variance of the search cost *for the mth key, for all* $1 \leq m \leq n$ is another very interesting, but still open, problem; the recurrences we derived for this variance, were too complicated to allow us to proceed. Assuming that the last problem has been solved, one may subsequently attempt to work as in Chapter 6, in order to derive the variance of a linear combination of all search costs, obtaining thus the complete variance-covariance matrix for all search costs. Such a result, if derived, would shed more light into the performance characteristics of the PSL. The $n+1$ different search costs do not assume values in isolation; they are related, yet, their relationship is totally unknown.

For the probabilistic analysis of the elastic-bucket trie, one open problem is evident from Figure 7.1: derive asymptotics for the variances and covariances of all bucket sizes under the fixed population model. This problem is particularly interesting in view of our earlier remark that we expect the variances under the fixed population model to be different from the (derived) variances under the Poisson model.

For the sake of completeness, in the study of the elastic-bucket trie, we would also like to include the number of empty buckets (i.e. nil pointers) in our calculations. Although the expected number of empty buckets follows immediately from

our results, their variance and covariances are not known. We note that our computations of Chapter 6 cannot be immediately extended to handle this problem, due to the non-existence of certain Mellin transforms.

Finally, another open problem for the elastic-bucket trie, completely different in flavour from the previous ones, but very important from a practical point of view, is the following. Suppose that only two different bucket sizes are allowed: b, and, say, x. What should x be, so that the average bucket space utilization is maximized? Our experimental results showed that the optimal choice for x is around $.7b$, but we were unable to prove that. For the $.7b$ choice of x, one may compute that about 84% of the bucket space is utilized; this is almost half way between the 69% space utilization when only 1 bucket size is allowed, and the 100% space utilization when all b different bucket sizes are allowed.

Bibliography

[1] G. M. Adel'son-Vel'skii and E. M. Landis. "An Algorithm for the Organization of Information." *Doklady Akademia Nauk SSSR*, vol. 146, 1962, pp. 263–266. English translation in *Soviet Mathematics Doklady*, vol. 3, pp. 1259–1263.

[2] A. Andersson. "Binary B-Trees are Simpler than Deterministic Skip Lists." *Unpublished Manuscript*, 1992.

[3] C. R. Aragon and R. G. Seidel. "Randomized Search Trees." *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science*, Research Triangle Park NC, Oct. 1989, pp. 540–545.

[4] R. A. Baeza-Yates. "Some Average Measures in m-ary Search Trees." *Information Processing Letters*, vol. 25, no. 6, July 1987, pp. 375–381.

[5] R. A. Baeza-Yates. "A Dynamic Storage Allocation Algorithm Suitable for File Structures." *Information Systems*, vol. 15, no. 5, 1990, pp. 515–521.

[6] R. Bayer and E. McCreight. "Organization and Maintenance of Large Ordered Indexes." *Acta Informatica*, vol. 1, 1972, pp. 173–189.

[7] J. Bentley. "Programming Pearls: Writing Correct Programs." *Communications of the ACM*, vol. 26, no. 12, Dec. 1983, pp. 1040–1045.

[8] R. de la Briandais. "File Searching Using Variable Length Keys." *AFIPS Western Joint Computer Conference*, San Francisco, CA, Mar. 1959, pp. 295–298.

[9] P. C. Consul. "The Exact Distribution of Certain Likelihood Criteria Useful in Multivariate Analysis." *Académie Royale de Belgique — Bulletin de la Classe des Sciences*, vol. 51, June 1965, pp. 683–691.

[10] J. Culberson and J. I. Munro. "Explaining the Behaviour of Binary Search Trees Under Prolonged Updates: A Model and Simulations." *The Computer Journal*, vol. 32, no. 1, Feb. 1989, pp. 68–75.

[11] J. Culberson and J. I. Munro. "Analysis of the Standard Deletion Algorithms in Exact Fit Domain for Binary Search Trees." *Algorithmica*, vol. 5, no. 3, 1990, pp. 295–311.

[12] L. Devroye. "Expected Time Analysis of Skip Lists." *Technical Report, School of Computer Science, McGill University*, Montreal, 1990.

[13] L. Devroye. "A Limit Theory for Random Skip Lists." *Annals of Applied Probability*, vol. 2, no. 3, Aug. 1992, pp. 597–609.

[14] R. Fagin, J. Nievergelt, N. Pippenger and H. R. Strong. "Extendible Hashing — A Fast Access Method for Dynamic Files." *ACM Transactions on Database Systems*, vol. 4, no. 3, Sep. 1979, pp. 315–344.

[15] P. Flajolet. "Mathematical Methods in the Analysis of Algorithms and Data Structures." In *Trends in Theoretical Computer Science*, Börger, E., ed., Computer Science Press: Rockville, MD, 1988, pp. 225–304.

[16] P. Flajolet, M. Régnier and R. Sedgewick. "Some Uses of the Mellin Integral Transform in the Analysis of Algorithms." *NATO Advanced Research Work-*

shop on Combinatorial Algorithms on Words, Eds. A. Apostolico and Z. Galil, Maratea, Italy, June 1984, pp. 241–254.

[17] P. Flajolet and R. Sedgewick. "Digital Search Trees Revisited." *SIAM Journal of Computing*, vol. 15, no. 3, Aug. 1986, pp. 748–767.

[18] G. N. Frederickson. "Improving Storage Utilization in Balanced Trees." *17th Annual Allerton Conference on Communication, Control and Computing*, Monticello, IL, Oct. 1979, pp. 255–264.

[19] E. Fredkin. "Trie Memory." *Communications of the ACM*, vol. 3, no. 9, Sep. 1960, pp. 490–499.

[20] P. W. Goetz. *The New Encyclopædia Britannica : Macropedia*, 15th edition, Chicago, 1985. S.v. "Encyclopedias and Dictionaries," written by R. L. Collison and W. E. Preece.

[21] G. H. Gonnet and R. A. Baeza-Yates. *Handbook of Algorithms and Data Structures in Pascal and C*, 2nd edition. Addison-Wesley: Reading, MA, 1991.

[22] G. H. Gonnet and J. I. Munro. "The Analysis of a Linear Probing Sort by the Use of a New Mathematical Transform." *Journal of Algorithms*, vol. 5, no. 4, Dec. 1984, pp. 451–470.

[23] R. L. Graham, D. E. Knuth and O. Patashnik. *Concrete Mathematics*. Addison-Wesley: Reading, MA, 1989.

[24] L. Guibas and R. Sedgewick. "A dichromatic framework for balanced trees." *19th Annual Symposium in Foundations of Computer Science* IEEE Computer Society. Ann Arbor, Michigan, Oct. 1978, pp. 8–21.

[25] P. Jacquet and M. Régnier. "Trie Partitioning Process: Limiting Distributions." *Lecture Notes in Computer Science, no. 214, Proceedings of the 11th Colloquium on Trees in Algebra and Programming (CAAP)*. Nice, France, Mar. 1986, pp. 196–210.

[26] D. E. Knuth. *The Art of Computer Programming, Volume 3: Sorting and Searching*. Addison-Wesley: Reading, MA, 1973.

[27] W. Lew and H. M. Mahmoud. "The Joint Distribution of Elastic Buckets in a Data Structure." *SIAM Journal on Computing*, to appear.

[28] H. R. Lewis and L. Denenberg. *Data Structures and their Algorithms*. Harper Collins: New York, NY, 1991.

[29] D. Lomet. "Partial Expansions for File Organizations with an Index." *ACM Transactions of Database Systems*, vol. 12, no. 1, Mar. 1987, pp. 65–84.

[30] H. Mahmoud. *Evolution of Random Search Trees*, John Wiley & Sons: New York, NY, 1992.

[31] H. Mahmoud and T. Papadakis. "A Probabilistic Analysis of Fixed and Elastic Buckets in Tries and Patricia Trees." *30th Annual Allerton Conference on Communication, Control and Computing*, Monticello, IL, Oct. 1992, pp. 874–883.

[32] H. Mellin. "Über den Zusammenhang Zwischen den Linearen Differential- und Differenzengleichungen" *Acta Mathematica*, vol. 25, 1902, pp. 139–164.

[33] J. I. Munro, T. Papadakis and R. Sedgewick. "Deterministic Skip Lists." *3rd Annual ACM-SIAM Symposium on Discrete Algorithms*, Orlando, FL, Jan. 1992, pp. 367–375.

[34] U. S. Nair. "The Application of the Moment Function in the Study of Distribution Laws in Statistics." *Biometrika*, vol. 30, 1938, pp.274–294.

[35] T. Papadakis, J. I. Munro and P. Poblete. "Analysis of the Expected Search Cost in Skip Lists." *Lecture Notes in Computer Science, no. 447, Proceedings of the 2nd Scandinavian Workshop of Algorithm Theory*. Bergen, Norway, July 1990, pp. 160–172.

[36] T. Papadakis, J. I. Munro and P. Poblete. "Average Search and Update Costs in Skip Lists." *BIT*, vol. 32, no. 2, 1992, pp. 316–322.

[37] P. Poblete. "Approximating Functions by their Poisson Transform." *Information Processing Letters*, vol. 23, no. 3, Oct. 1986, pp. 127–130.

[38] P. Poblete. Personal communication, 1991–1992.

[39] W. Pugh. "A Skip List CookBook". *Technical Report CS-TR-2286.1, Department of Computer Science, University of Maryland*, College Park, July 1989.

[40] W. Pugh. "Skip Lists: A Probabilistic Alternative to Balanced Trees." *Communications of the ACM*, vol. 33, no. 6, June 1990, pp. 668–676. (Initial version appeared in *Lecture Notes in Computer Science, no. 382, Proceedings of the Workshop of Algorithms and Data Structures*. Ottawa, Canada, August 1989, pp. 437–449.)

[41] M. Régnier and P. Jacquet. "New Results on the Size of Tries." *IEEE Transactions on Information Theory*, vol. 35, no. 1, Jan. 1989, pp.203–205.

[42] R. Sedgewick. *Algorithms in C*. Addison-Wesley: Reading, MA, 1990.

[43] S. Sen. "Some Observations on Skip Lists." *Information Processing Letters*, vol. 39, no. 4, Aug. 1991, pp. 173–176.

[44] D. D. Sleator and R. E. Tarjan. "Self-Adjusting Binary Search Trees." *Journal of the ACM*, vol. 32, no. 3, July 1985, pp. 652–686.

[45] E. H. Sussenguth, Jr. "Use of Tree Structures for Processing Files." *Communications of the ACM*, vol. 6, no. 5, May 1963, pp. 272–279.

[46] J. Sussman. *Instructor's Manual to Accompany "Introduction to Algorithms" by T. H. Cormen, C. E. Leiserson and D. L. Rivest*. The MIT Press: Cambridge, MA, 1991.

[47] W. Szpankowski. "On a Recurrence Equation Arising in the Analysis of Conflict Resolution Algorithms." *Communications in Statistics: Stochastic Models*, vol. 3, no. 1, 1987, pp. 89–114.

[48] W. Szpankowski. "The Evaluation of an Alternative Sum with Applications to the Analysis of Some Data Structures." *Information Processing Letters*, vol. 28, no. 1, May 1988, pp. 13–19.

[49] W. Szpankowski. "Some Results on V-ary Asymmetric Tries." *Journal of Algorithms*, vol. 9, no. 2, June 1988, pp. 224–244.

[50] J. S. Vitter and P. Flajolet. "Average-Case Analysis of Algorithms and Data Structures." In *Handbook of Theoretical Computer Science*, The MIT Press/Elsevier: Cambridge, MA, 1990, pp. 431-524.

[51] M. A. Weiss. *Data Structures and Algorithm Analysis*. Benjamin/Cummings: Redwood City, CA, 1992.