JOURNAL OF ALGORITHMS 1, 111–141 (1980)

Sequence of Operations Analysis for Dynamic Data Structures*

P. FLAJOLET

Iria-Laboria, 78150, Rocquencourt, France

J. FRANÇON

Centre de Calcul du CNRS, BP 20 CR0,67037 Strasbourg, France

AND

J. VUILLEMIN

Université de Paris-Sud, Batiment 490,91405 Orsay, France

Received October 8, 1979

This paper introduces a rather general technique for computing the average-case performance of dynamic data structures, subjected to arbitrary sequences of insert, delete, and search operations. The method allows us effectively to evaluate the integrated cost of various interesting data structure implementations, for stacks, dictionaries, symbol tables, priority queues, and linear lists; it can thus be used as a basis for measuring the efficiency of each proposed implementation. For each data type, a specific continued fraction and a family of orthogonal polynomials are associated with sequences of operations: Tchebycheff for stacks, Laguerre for dictionaries, Charlier for symbol tables, Hermite for priority queues, and Meixner for linear lists. Our main result is an explicit expression, for each of the above data types, of the generating function for integrated costs, as a linear integral transform of the generating functions for individual operation costs. We use the result to compute explicitly integrated costs of various implementations of dictionaries and priority queues.

1. INTRODUCTION

The purpose of this paper is to describe a rather general technique for computing the average cost of a sequence of operations, which is applicable to many of the interesting known implementations of data structures in computer science.

*A preliminary version of this paper was presented at the 20th IEEE Symposium on Foundations of Computer Science, at Puerto Rico, October 1979.

0196-6774/80/020111-31$02.00/0

We provide both a methodological framework and the corresponding computational techniques, allowing one to perform *quantitative* comparisons among various *data organizations*, i.e., classes of data structures together with associated algorithms for operating on these structures.

In comparing the costs (with respect to some measure such as time, space, OS charge, ...) of two data organizations A and B for the same operations, we cannot merely compare the costs of individual operations for data of given sizes: A may be better than B on some data, and conversely on others; operation 1 may be more efficient in A whereas operation 2 is more efficient in B; etc.

A reasonable way to measure the efficiency of a data organization is to consider *sequences of operations* on the structure; the cost of each such sequence is the sum of the costs of individual operations. Given a finite set of sequences of fixed length, we can define the maximal, minimal, and average costs over the set in an obvious way.

This general problem has already generated interesting work pertaining to the cost of sequences of operations for partition structures (disjoint set union); both maximal cost (see Tarjan [29]) and average cost (Doyle and Rivest [6]; Yao [33]; Knuth and Shönhage [26]) have been studied.

For dictionary structures, where insertion, deletions, and searches are allowed, the "Knott phenomenon" (Knott [22]) for *binary search trees* (Knuth [24]; Hibbard [19]) has stimulated a general methodological reflection by Knuth [25], and a difficult mathematical analysis by Jonassen and Knuth [21]. Interesting results for maximal cost have been obtained by Snyder [35] under hypotheses very similar to ours.

Our work is a continuation of that of Flajolet *et al.* [11], in which dictionaries are analyzed under sequences of operations. It also extends previous results by Françon [13] based on the combinatorial methods of Françon and Viennot [15]. Some of the techniques relative to the use of continued fractions appear in Flajolet [7, 8].

We provide here both a methodological framework and the corresponding computational techniques allowing one to perform quantitative comparisons between data structures belonging to the same type. The method applies to structures having a "randomness preservation property," which includes many of the classical implementations used in computer science; lists, search trees, tournament trees, binomial queues, and position trees are a sample of those amenable to our methods.

The plan of the paper is as follows: Section 2 provides the set of necessary definitions for data types, data structures, and integrated costs. Sections 3 and 4 describe the use of continued fractions and orthogonal polynomials to derive enumeration results relative to sequences of operations. Section 5 presents the integral transforms associated with each of the basic data types: these transforms map generating functions for individual

costs of operations to generating functions for integrated costs of sequences of operations. The results are then applied in Section 6 to provide explicit evaluations and comparisons of classical data structures. Finally, Section 7 discusses some of the implications of our work and outlines possible extensions.

2. Data Types, Histories, and Integrated Costs

2.1. *Data Types, Data Organizations*

We consider here data structures subject to the following operations:

A for *adjunction*, i.e., insertion;
S for *suppression*, i.e., deletion;
Q^+ for *positive query*, i.e., successful search
Q^- for *negative query*, i.e., negative search.

Keys are accessed either by value or by position, and additional constraints may be imposed on the set of keys accessible at each stage.

A *data type* is a specification of the basic operations allowed together with its set of possible restrictions. The five data types to be studied here are

Dictionary	Keys belonging to a totally ordered set are accessed by value; all four operations A, S, Q^+, Q^- are allowed without any restriction.
Priority queue	Keys belonging to a totally ordered set are accessed by value; the basic operations are A and S; deletion (S) is performed only on the key of minimal value (of "highest priority").
Linear list	Keys are accessed by position; operations are A and S without access restrictions (linear lists make it possible to maintain dynamically changing arrays).
Stack	Keys are accessed by position; operations are insertion (A) and deletion (S) but are restricted to operate on the key positioned first in the structure (the "top" of the stack).
Symbol table	This type is a particular case of dictionary, where deletion always operates on the key last inserted in the structure; only positive queries are performed (think of the management of the symbol table of an Algol-like language, where insert is

performed upon entering a block, while the corresponding delete occurs upon exit from that block).

A *data organization* is a machine implementation of a data type. It consists of a *data structure*, which specifies the way objects are internally represented in the machine, together with a collection of *algorithms* implementing the operations of the data type.

Stacks	They are almost universally represented by arrays, or linked lists.
Dictionaries	The most straightforward implementation is by sorted or unsorted lists; binary search trees have a faster execution time and several balancing schemes have been proposed: AVL and 2–3 trees; bichromatic trees by Guibas and Sedgewick [18]. Other alternatives are h-tables and digital trees (see Knuth [24] for many interesting implementations).
Priority queues	They can be represented by any of the search trees used for dictionaries; more interesting are *heaps* (see Knuth [24]), *P-tournaments* (Jonassen and Dahl [20]), *leftist tournaments* (Crane [34], *binomial tournaments* (Vuillemin [30]), *binary tournaments* and *pagodas* (Françon *et al.* [16]). One can also use *sorted lists*, and any of the *balanced tree* structures (used as a tournament rather than a search tree) for implementing priority queues (see Aho *et al.* [1]).
Linear lists	The most straightforward implementation is by linked lists and arrays. Position tournaments (Vuillemin [31]) are a more efficient implementation to which balancing schemes can be applied (Brown and Tarjan [3]).
Symbol tables	These are special cases of dictionaries. All the known implementations of dictionaries are applicable here, and the authors are not aware of interesting specific implementations for symbol tables, although a rather nice deletion algorithm can be programmed for binary search trees, if the keys arrival times are (implicitly) present in the structure.

Of course there are other interesting data types: queues and dequeues (see Knuth [23]) are closely related to stacks; partition structures (also called disjoint set union) involve the operation of union, which is not considered here. One could also allow for more operations: split and merge for dictionaries; extract and union for priority queues; search, cut, concatenate, and reverse for linear lists.

2.2 *Sequences of Operations*

Before proceeding further, we need to state precise definitions concerning sequences of operations for each of our basic data types.

A data type can be formally described by the universe of keys, the set of files, and the specification of the way operations perform on files.

(a) The *universe* $\mathcal{K}$ from which keys are drawn is the set of real numbers (in practice $\mathcal{K}$ is more likely to be some very large but finite set).

(b) A *file status*, or simply *file*, for a given data type is a structured finite set of keys: for dictionaries and priority queues, the set of files is the set of all finite subsets of $\mathcal{K}$ (i.e., a file can be any finite set of keys); for linear lists, stacks and symbol tables, the set of files is the set of all sequences on $\mathcal{K}$.

(c) For each input k, operation O, and file $\mathcal{F}$, we need to describe in each case the way $\mathcal{F}$ is transformed when operation $O \in \{A, S, Q^+, Q^-\}$ is performed on key k:

Stack — If $\mathcal{F} = \langle k_1, k_2, \ldots, k_s \rangle$, performing $A(k)$ leads to $\langle k_1, k_2, \ldots, k_s, k \rangle$; performing S leads to $\langle k_1, k_2, \ldots, k_{s-1} \rangle$ with output k_s, provided $s \geq 1$.

Dictionary — If $\mathcal{F} = \{k_1, k_2, \ldots, k_d\}$, performing $O(k)$ leads to a new file $\mathcal{F}'$ with

$$\begin{aligned} \mathcal{F}' &= \mathcal{F} && \text{if } O = Q^+ \text{ and } k \in \mathcal{F} \\ & && \text{or } O = Q^- \text{ and } k \notin \mathcal{F}; \\ \mathcal{F}' &= \mathcal{F} \cup \{k\} && \text{if } O = A \text{ and } k \notin \mathcal{F}; \\ \mathcal{F}' &= \mathcal{F} \setminus \{k\} && \text{if } O = S \text{ and } k \in \mathcal{F} \end{aligned}$$

(other cases undefined).

Priority queue — With $\mathcal{F} = \{k_1, k_2, \ldots, k_p\}$, $A(k)$ with $k \notin \mathcal{F}$ leads to $\mathcal{F}' = \mathcal{F} \cup \{k\}$; suppression S leads to $\mathcal{F}' = \mathcal{F} \setminus \{a\}$, where $a = \min_{1 \leq i \leq p}\{k_i\}$, and is meaningless if $p = 0$.

Linear list — With $\mathcal{F}$ a sequence of keys $\langle k_1, k_2, \ldots, k_l \rangle$, $A(p; k)$ is defined iff $1 \leq p \leq l + 1$ and the resulting file is $\mathcal{F}' = \langle k_1, \ldots, k_{p-1}, k, k_p, \ldots, k_l \rangle$; on the other hand, $S(p)$ leads to $\mathcal{F}' = \langle k_1, \ldots, k_{p-1}, k_{p+1}, \ldots, k_l \rangle$.

Symbol tables — With $\mathcal{F} = \langle k_1, k_2, \ldots, k_m \rangle$, performing $O(k)$ leads to a new file $\mathcal{F}'$ such that

$$\begin{aligned} \mathcal{F}' &= \mathcal{F} && \text{if } O = Q^+ \text{ and } k \in \mathcal{F}; \\ \mathcal{F}' &= \langle k_1, \ldots, k_m, k \rangle && \text{if } O = A \text{ and } k \in \mathcal{F}; \\ \mathcal{F}' &= \langle k_1, \ldots, k_{m-1} \rangle && \text{if } O = S. \end{aligned}$$

A *sequence of operations* is a sequence of the form $O_1(k_1); O_2(k_2); \ldots; O_n(k_n)$, where for $1 \leq i \leq n$, $k_i \in K$ is a key and $O_i \in O = \{A, S, Q^+, Q^-\}$ is an operation.

For each of the data types considered above, we define the content F_i of the file at stage i by the rules:

$$\begin{cases} F_0 = \varnothing \\ F_i \text{ is the result of performing } O_i(k_i) \text{ on } F_{i-1} \text{ for } 1 \leq i \leq n. \end{cases}$$

With each data organization and sequence of operations $O_1(k_1); \ldots; O_n(k_n)$ there is naturally associated a *cost* (execution time, number of comparisons, storage utilisation, etc.) which is the sum, for $1 \leq i \leq n$, of the cost of executing the algorithm implementing O_i, with input key k_i, on the data structure representing F_{i-1}.

2.3. *Histories*

In the remaining part of this section, we introduce a notion of *equivalence* between sequences of operations, serving two purposes: first, in a wide class of data organizations implementing each given data type, the sequences of operations physically performed *within the machine are the same*, for two equivalent sequences of operations (if we think in terms of machine language implementations, the sequences of contents of the *ordinal counter* are the same); second, the number of *equivalence classes of operation sequences is finite*, and thus we can define a notion of average cost over such sequences.

Of course, we have to define equivalence of operation sequences for each of our five data types. We shall proceed by defining canonical representations—called *histories*—for sequences of operations, with two sequences equivalent if they share the same underlying history. Canonical sequences for linear lists and stacks, where access to keys is by position only, are obtained by retaining only the information relative to the position of keys (forgetting about key values); canonical sequences for dictionaries, priority queues, and symbol tables are obtained by retaining at each stage only the relative rank (w.r.t. to the order relation on $\mathcal{K}$) of the key which is operated upon; in other words, two sequences of operations are equivalent iff they are order-isomorphic.

We can unify all cases by providing the following general definition of a *rank* function:

DEFINITION. Let $\mathcal{F}$ be a nonempty file content, for any of our five data types, and let k be an arbitrary key. We define $\text{rank}(k, \mathcal{F})$ as follows:

(1) For stacks, $\text{rank}(k, \mathcal{F}) = 0$;

(2) For dictionaries, $\mathcal{F} = \{k_1, \ldots, k_d\}$ with $k_1 < k_2 < \ldots < k_d$ and we let $\text{rank}(k, \mathcal{F}) = i - 1$ if $k \in \mathcal{F}$ and $k = k_i$, and $\text{rank}(k, \mathcal{F}) = i$ if $k \notin \mathcal{F}$ and $k_i < k < k_{i+1}$, with the convention $k_0 < k < k_{d+1}$.

(3) For priority queues, $\mathcal{F} = \{k_1, \ldots, k_p\}$ with $k_1 < k_2 < \ldots < k_p$ and we let $\text{rank}(k, \mathcal{F}) = i$ if $k_i < k \leq k_{i+1}$ with the convention $k_0 \leq k < k_{p+1}$.

(4) For linear lists, $\mathcal{F} = \langle k_1, \ldots, k_l \rangle$ and $\text{rank}(k, \mathcal{F}) = k - 1$ for $1 \leq k \leq l + 1$.

(5) For symbol tables, $\text{rank}(k, \mathcal{F})$ is defined as in case 2, for dictionaries.

With each legal sequence of operations $L = O_1(k_1); \ldots; O_n(k_n)$, we associate its *history*, which is a pair (S, V), where $S = \langle O_1, O_2, \ldots, O_n \rangle$ is the *schema* of L and $V = \langle \text{rank}(k_1, F_0), \ldots, \text{rank}(k_i, F_{i-1}), \ldots, \text{rank}(k_n, F_{n-1}) \rangle$ is the *valuation* of L; the number n is the *length* of the history.

Two sequences of operations are said to be equivalent if they have the same underlying history.

Figure 1 gives for each data type an instance of a sequence of operations together with its corresponding history.

Histories can be defined independently as combinatorial objects. For $s = O_1 O_2 \ldots O_n$ a schema, the *height* of i at stage h_i is defined by $h_i = |O_1 \ldots O_i|_A - |O_1 \ldots O_i|_s$, the excess of adjunctions over suppressions. Quantity h_i is also the size of the file resulting from any sequence

Dictionary							
	$A(2.7)$	$A(3.1)$	$A(0.57)$	$Q^-(0.69)$	$S(3.1)$	$Q^+(2.7)$	$S(0.57)$
	A_0	A_1	A_0	Q_1^-	S_2	Q_1^+	S_0
Priority queue							
	$A(2.7)$	$A(3.1)$	$A(0.57)$	$S_{\min}$	$A(0.69)$	$S_{\min}$	$S_{\min}$
	A_0	A_1	A_0	S_0	A_0	S_0	S_0
Stack							
	$A(2.7)$	$A(3.1)$	$A(0.57)$	S_{top}	$A(0.69)$	S_{top}	S_{top}
	A_0	A_0	A_0	S_0	A_0	S_0	S_0
Linear list							
	$A(1; 2.7)$	$A(1; 3.1)$	$A(3; 0.57)$	$S(2)$	$A(2; 0.69)$	$S(3)$	$S(1)$
	A_0	A_0	A_2	S_1	A_1	S_2	S_0
Symbol table							
	$A(2.7)$	$A(3.1)$	$A(0.57)$	S_{last}	$Q^+(3.1)$	$A(0.69)$	S_{last}
	A_0	A_1	A_0	S_0	Q_1^+	A_0	S_0

FIG. 1. Sequences of operations with corresponding histories; ranks are indicated as subscripts to the operators.

	Stack	Dictionary	Priority queues	Linear lists	Symbol table
$\text{pos}(A, k)$	1	$k+1$	$k+1$	$k+1$	$k+1$
$\text{pos}(S, k)$	1	k	1	k	1
$\text{pos}(Q^+, k)$	0	k	0	0	k
$\text{pos}(Q^-, k)$	0	$k+1$	0	0	0

FIG. 2. The five possibility functions considered here.

of operations with schema $O_1O_2 \ldots O_i$. A pair $(S; V)$ where $S = O_1O_2 \ldots O_N \in \{A, S, Q^+, Q^-\}^*$ and $V = V_1V_2 \ldots V_n \in N^*$ represents a history for a data type Σ iff

—each of the heights in $O_1 \ldots O_n$ is nonnegative,

—each of the V_i satisfies $0 \leq V_i < \text{pos}^\Sigma(O_i, h_{i-1})$ where $\text{pos}^\Sigma : \{A, S, Q^+, Q^-\} \times N \to N$ is the *possibility function* relative to each data type (see Fig. 2).

Histories are thus *combinatorial objects*, and there are *finitely many* histories of a given length. Figure 3 enumerates all histories of length 2 for each data type.

2.4. *Integrated Costs for Stationary Structures*

Restricting attention to histories is justified for data structures satisfying:

Relevance of histories hypothesis [RHH]: *The costs of two equivalent sequences of operations are identical.*

All of the data organizations mentioned in Section 2.1 satisfy RHH except h-code and digital search structures that perform arithmetic on their keys; roughly speaking, structures satisfying RHH access their keys by a *decision tree*, whose only primitive operation is comparison between keys.

Data type	Histories of length 2			
Stack	AS	AA		
	00	00		
Dictionary	Q^-Q^-	AS	Q^-A	AQ^+
	0 0	00	0 0	00
	$A\ Q^-$	AQ^-	$A\ A$	AA
	0 0	01	0 0	01
Priority queues	AS	AA	AA	
	00	00	01	
Linear lists	AS	AA	AA	
	00	00	01	
Symbol table	AS	AQ^+	AA	AA
	00	00	01	10

FIGURE 3

Other data organizations which violate RHH are *garbage collection structures*, where deletion S is performed by marking a node rather than by physically removing it from the structure.

These are the only data organizations violating our hypothesis, since RHH can be established, in a rather general setting, under the following hypotheses:

(1) *Decision Tree Hypothesis*. The only way in which the data organization can access the keys is by performing comparisons between keys.

(2) *Oblivion Hypothesis*. Only keys which are present in the structure (i.e., have *not* been deleted) can be compared.

Given a data organization satisfying RHH and a history h, we define the *cost of* h as the cost of any operation sequence having h for history.

DEFINITION. Let $\mathcal{H}_n$ be a finite set of histories of length n; the *integrated cost* K_n over $\mathcal{H}_n$ is defined as

$$K_n = \frac{1}{|\mathcal{H}_n|} \sum_{h \in \mathcal{H}_n} \text{cost}(h).$$

In most of our applications, we choose for $\mathcal{H}_n$ the set of all possible histories of length n, starting and finishing with an empty file. Comparing data organizations over this set of histories is analogous to what we do when we reduce the analysis of sorting algorithms over unbounded sets of keys, to an analysis over the $n!$ permutations of $[n] = \{1, 2, \ldots, n\}$.

Given a possibility function associated with a data type, a set of histories is *complete* if, for each schema $O_1; \ldots; O_n$ associated with a history in $\mathcal{H}$, any history g made of the same schema and of *any* valuation $V_1; \ldots; V_n$ (compatible with the schema by the possibility rule) is also in $\mathcal{H}$.

Thus, in a complete set of histories, the number of histories with a given schema $O_1; \ldots; O_n$ is $\Pi_{1 \le i \le n} \text{pos}(O_i, h_{i-1})$.

To define a complete set of histories, we thus need only describe the schemata of this set.

For example, the complete set of histories of length 2 contains six schemata Q^-Q^-, Q^-A, AS, AQ^-, AQ^+, AA corresponding to two stack histories, eight dictionary histories, etc. (cf. Fig. 3).

We denote by:

(1) $\mathcal{H}_n$ the complete set of histories of length n, and of initial and final height $h_0 = h_n = 0$; our main interest lies in these histories.

(2) $\mathcal{H}_{k,l,n}$ the complete set of histories of length n, of initial height $h_0 = k$ and final height $h_n = l$.

Of course, $\mathcal{H}_n = \mathcal{H}_{0,0,n}$ and we denote the cardinalities of these sets by $H_n = |\mathcal{H}_n|$ and $H_{k,l,n} = |\mathcal{H}_{k,l,n}|$.

Our goal in this paper is to compute explicitly integrated costs of data organizations over $\mathcal{H}$. This proves possible for data organizations which, in addition to satisfying RHH, have a "randomness" or "stationarity" property, which we now define.

For each data organization satisfying RHH, we call the *state* of the structure at a given time the state of a machine implementing it. Thus, algorithms representing the various operations A, S, and Q map states into states; we assume that the cost of such an algorithm depends only upon input and output states (it is time independent). The *size* of a state is the number of keys present in memory.

DEFINITION. Two states are *equivalent*, if they are attainable from the empty state (size 0) through two equivalent sequences of operations.

From now on, we use the word *state* instead of *equivalence class of states*; we let E_k denote the set of states of size k.

For binary search trees, E_k can be identified by the set of binary trees with k nodes; for sorted lists, E_k can be identified by the unique sequence $(1, 2, \ldots, k)$ while, for unsorted lists, the $k!$ states E_k correspond to permutations of $1, 2, \ldots, k$. Figure 4 describes the number of states of size k, for various data organizations.

DEFINITION. The probability distribution induced on the set E_k by all the possible histories consisting of k insertions (i.e., with schema

$$A^k = \underbrace{AA \ldots A}_{k \text{ times}}),$$

is called the *standard probability* distribution on E_k.

We let $p_s(e)$ denote the standard probability of state e.

DEFINITION. A data organization is *stationary* if for all k the three probability distributions induced over E_k by all possible histories of schemata $A^{k+1}S$, A^kQ^-, and A^kQ^+ coincide with the standard probability distribution (defined by histories of schema A^k). This stationary condition coincides with the notion of *randomness* under hypothesis (I_0, D_0) in Knuth [25].

As shown by Knott's phenomenon, randomness under our hypotheses is not equivalent for binary search trees to randomness under the condition that keys be independently drawn from a uniform distribution. Among the data organizations mentioned in Fig. 4, only balanced tree structures are *not* stationary:

PROPOSITION 1. *The implementation of stacks by lists, dictionaries by unsorted lists, sorted lists, binary search tree, priority queues by sorted lists,*

binary search trees, binary tournaments, pagodas, binomial queues, and linear lists by unsorted lists and position tournaments are all stationary data organizations. Except for binary search tree, the standard frequencies of states $e \in E_k$ *are all equal; such structures are called* uniform.

Proof. Stationarity of binary search trees resides in Hibbard's theorem for deletions (Hibbard [19]; Knuth [24]; Knott [22]).

The other cases are treated by the following result, due to Knuth [25] and Françon [14]:

PROPOSITION 2. *A data organization is uniform iff* $\forall\ k, \forall\ O \in \{A, S, Q^+, Q^-\}, \forall\ e' \in E_{k'}$:

$$|E_{k'}| \cdot |\{e \in E_k | O(e) = e'\}| = \text{pos}(O, k) \cdot |E_k|$$

where $k' = k$ *for* $O \in \{Q^+, Q^-\}$, $k' = k + 1$ *for* $O = A$ *and* $k' = k - 1$ *for* $O = S$.

Data organization	Data structure	Number of states of size k	Standard probability	Stationary
Stack	Lists	1	Uniform	Yes
Dictionary and symbol table	Unsorted lists UL	$k!$	Uniform	Yes
	Sorted lists SL	1	Uniform	Yes
	Binary search tree BST	$\frac{1}{k+1}\binom{2k}{k}$	cf. Knuth [24, p. 671]	Yes
	Binomial list BL	$\binom{k}{b_p 2^p, \ldots, b_0 2^0}$	?	No
	Balanced Tree BT	?	?	No
Priority queue	Sorted list	1	Uniform	Yes
	Heaps	cf. Knuth, [24, p. 154]	?	No
	Leftists and balanced tournaments	?	?	No
	Binary search tree	$\frac{1}{k+1}\binom{2k}{k}$	cf. Knuth [24, p. 671]	Yes
	Binary tournaments	$k!$	Uniform	Yes
	Pagodas	$k!$	Uniform	Yes
	Binomial queues	$k!/2^{k-\nu(k)}$	Uniform	Yes
Linear lists	Unsorted lists	$k!$	Uniform	Yes
	Position tournaments	$k!$	Uniform	Yes
	Balanced tree	?	?	No

FIG. 4. Data structures and their standard probabilities; here $k = (b_p \ldots b_0)_2$ and $\nu(k) = \Sigma_{0 \le i \le p} b_i$.

Proposition 1 follows by inspection, the nontrivial case of binomial queues being treated by Brown [2]. □

DEFINITION. For a stationary data organization, we define the *individual* cost CO_k *of operation* O *on a file of size* k by the formula $CO_k = \sum_{e \in E_k} p_s(e)$ (cost of O on state e), where $p_s(e)$ is the standard frequency of e.

Knuth [24] likes to express such costs, measured in time units on a MIX implementation of the algorithm realizing the operation. Simpler measures are also realistic, and Fig. 5 provides an expression of the average number of key comparisons for each of the stationary data structures of Fig. 4, in the author's implementations.

Starting from the average costs CO_k of operations on files of size k in stationary structures, we can express the integrated cost K_n over $\mathcal{H}_n$ as a *linear combination* of the CO_k, whose coefficients, *the level crossing numbers*, depend only upon $\mathcal{H}_n$.

DEFINITION. The *level crossing number* $NO_{k,n}$ is the number of operations of type O performed on a file of size k in the course of all histories in $\mathcal{H}_n$.

Our interest in stationary data organizations is now justified by the following formula for computing integral costs:

PROPOSITION 3. *For stationary data organizations, the individual costs* CO_k *of operations and the integrated cost* $K_n = (1/H_n) \cdot \sum_{h \in \mathcal{H}_n} \text{cost}(h)$ *are related by* $K_n = (1/H_n) \cdot \sum_{O,k} NO_{k,n} \cdot CO_k$, *where the* NO_k *are the level crossing numbers associated with* $\mathcal{H}_n$.

Proof. Apply the definitions and one inversion of summation. □

Data type	Data organization	CA_k	CS_k	CQ_k^+	CQ_k^-
Dictionary	Sorted list	$(k+2)/2$	$(k+1)/2$	$(k+1)/2$	$(k+2)/2$
	Unsorted list	0	$(k+1)/2$	$(k+1)/2$	k
	Binary search tree	$2(H_{k+1}-1)$	$2(1+1/k)H_k - 3$	$2(1+1/k)H_k - 3$	$2(H_{k+1}-1)$
Priority queue	Sorted list	$(k+2)/2$	0		
	Binary search tree	$2(H_{k+1}-1)$	0		
	Binary tournament	$H_{k+1} - 1/2$	$2(H_k - 2 + 1/k)$		
	Pagodas	$2(1 - 1/(k+1))$	$2(H_k - 2 + 1/k)$		
	Binomial queues	$1 + \nu(k) - \nu(k+1)$	$\sigma(k) + \nu(k) - 1 - \nu(k-1)$		
Linear lists	List	$(k+2)/2$	$(k+1)/2$		
	Position tournament	$2(H_{k+1}-1)$	$2(1+1/k)H_k - 3$		

FIG. 5. Average number of comparisons for operations in various stationary data organizations. Here, $H_k = 1 + \frac{1}{2} + \ldots + 1/k$, $k = \sum_{i \geq 0} b_i 2^i$, $\nu(k) = \sum_{i \geq 0} b_i$ and $\sigma(k) = (1/k)\sum_{i \geq 0} i b_i 2^i$.

At this stage, the problem of estimating integrated costs has been reduced to:

(1) evaluating individual costs, which itself combines a counting of instructions (or simply comparisons) with the weighting that comes from the standard probability distribution;
(2) evaluating the level crossing numbers $NO_{k,n}$ and plugging them into the formula for integrated costs.

The second aspect is to be discussed in the next two sections.

3. Histories and Continued Fractions

In Section 2, we reduced the computation of integrated costs to the evaluation of certain combinatorial sums in which there appear the quantities H_n, $H_{k,l,n}$, $NO_{k,n}$, etc. In this section we express the generating functions relative to these quantities in terms of continued fractions.

3.1. *The Continued Fraction Theorem*

We consider schemata as defined in Section 2 and we introduce an arbitrary possibility set π : $\text{pos}(A; k) = a_k$; $\text{pos}(Q^+; k) = q_k^+$; $\text{pos}(Q^-; k) = q_k^-$; $\text{pos}(S; k) = s_k$. We also let $q_k = q_k^+ + q_k^-$. Dictionaries thus correspond to the particular case: $a_k = k + 1$; $q_k = 2k + 1$; $s_k = k$. The following result is from Flajolet [7, 8]:

Theorem F (the continued fraction expansion theorem). *Let H_n be the number of histories ending at zero, relative to the possibility set* Π, *and let $H(z) = \sum_{n \geq 0} H_n z^n$ be the corresponding generating function. Then $H(z)$ has the following continued fraction expansion*:

$$H(z) = \cfrac{1}{1 - q_0 z - a_0 s_1 z^2 \Big/ \left(1 - q_1 z - \cfrac{a_1 s_2 z^2}{\cdots}\right)}.$$

Proof. (sketch). Define the alphabet $X = \{A_0, A_1, \ldots, Q_0, Q_1, \ldots, S_1, S_2, \ldots\}$, where O_j ($O = A$, Q, or S) denotes operation O on a file of size j. Let $S^{[h]}$ denote the set of schemata represented by words over X having height $\leq h$. The $S^{[h]}$ have the following regular expression descriptions:

$$S^{[0]} = (Q_0)^*; \; S^{[1]} = (Q_0 + A_0(Q_1)^* S_1)^*;$$
$$S^{[2]} = (Q_0 + A_0(Q_1 + A_1(Q_2)^* S_2)^* S_1)^* \ldots,$$

and in general $S^{[h+1]}$ is obtained by substituting $(Q_h + A_h(Q_{h+1})^* S_{h+1})$ for Q_h in the expression for $S^{[h]}$. If we let $H_n^{[h]}$ denote the number of histories of height $\leq h$, length n, and $H^{[h]}(z) = \Sigma_{n\geq 0} H_n^{[h]} \cdot z^n$, we have: $H^{[0]}(z) = 1 + q_0 z + q_0^2 z^2 + \ldots = 1/(1 - q_0 z)$,

$$H^{[1]}(z) = \frac{1}{1 - q_0 z - a_0 s_1 z^2/(1 - q_1 z)},$$

etc.; in general, $H^{[h+1]}(z)$ is obtained by substituting

$$q_h + \frac{a_h s_{h+1} z}{1 - q_{h+1} z}$$

for q_h in $H^{[h]}(z)$. The theorem follows by letting h go to infinity. □

Using the more economical notation

$$1/1 - q_0 z - a_0 s_1 z^2/ \ldots /1 - q_h z - a_h s_{h+1} z^2/ \ldots$$

for $H(z)$, we apply Theorem F to our five data types and obtain continued fraction expressions for the corresponding generating functions $H(z) = \Sigma_{n\geq 0} H_n z^n$:

Stacks	${}^{S}H(z) = 1/1 - z^2/1 - z^2/ \ldots /1 - z^2/ \ldots ;$
Dictionary	${}^{D}H(z) = 1/1 - 1z - 1^2z^2/1 - 3z - 2^2z^2 / \ldots /1 - (2k-1)z - k^2z^2/ \ldots ;$
Priority queue	${}^{PQ}H(z) = 1/1 - 1z^2/1 - 2z^2/ \ldots /1 - kz^2/ \ldots ;$
Linear list	${}^{LL}H(z) = 1/1 - 1^2z^2/1 - 2^2z^2/ \ldots /1 - k^2z^2/ \ldots ;$
Symbol table	${}^{ST}H(z) = 1/1 - 0z - 1z^2/1 - 1z - 2z^2 / \ldots /1 - (k-1)z - kz^2/ \ldots .$

Theorem F provides a means of obtaining expressions for the H_n in our five cases, by identifying the continued fraction with expansions derived from Gauss' continued fraction expression for hypergeometric series or from the Stieltjes–Rogers addition theorem (see Flajolet [8]); an alternative derivation is given below.

3.2. *Histories of Bounded Height*

With $H_n^{[h]}$ the number of histories of height $\leq h$, length n, and $H^{[h]}(z)$ the corresponding generating function, we have, as direct consequences of Theorem F (for proofs, see [8]):

PROPOSITION 4. *Histories of height $\leq h$ have a rational generating function given by $H^{[h]}(s) = P_h(z)/Q_h(z)$, where P_h and Q_h are polynomials*

that satisfy the recurrences

$$P_{-1}(z) = 0; \qquad P_0(z) = 1; \qquad P_h(z) = (1 - q_h z)P_{h-1}(z) - a_{h-1}s_h z^2 P_{h-2}(z);$$

$$Q_{-1}(z) = 1; \qquad Q_0(z) = 1 - q_0 z; \qquad Q_h(z) = (1 - q_h z)Q_{h-1}(z) - a_{h-1}s_h z^2 Q_{h-2}(z).$$

Hence $\deg P_h = \deg Q_{h-1} \leq h$ *for all* h.

PROPOSITION 5. *Let* $H_{k,l}(z) = \Sigma_{h \geq 0} H_{k,l,n} z^n$; *we have*

$$H_{k,l}(z) = \frac{Q_{\mu-1}(z)}{a_0 a_1 \cdots a_{k-1} s_1 s_2 \cdots s_l z^{k+l}} (Q_{\lambda-1}(z)H(z) - P_{\lambda-1}(z)),$$

where $\mu = \min(k, l)$ *and* $\lambda = \max(k, l)$.

In particular this gives expressions for $H_{0,k}(z)$ *and* $H_{k,0}(z)$, *namely*,

$$H_{0,k}(z) = \frac{1}{s_1 s_2 \cdots s_k z^k} (Q_{k-1}(z)H(z) - P_{k-1}(z))$$

and

$$H_{k,0}(z) = \frac{1}{a_0 a_1 \cdots a_{k-1} z^k} (Q_{k-1}(z)H(z) - P_{k-1}(z)).$$

An alternative way of looking at the relations between the formal series $H(z)$ and the polynomials $Q_h(z)$ which appear in the convergents is by means of orthogonality relations. Starting from the numbers H_n, $n \geq 0$, we introduce the linear form $\langle P(x) \rangle$ over polynomials $P(x) = \Sigma_{0 \leq i \leq k} p_i x^i$, defined by $\langle P(x) \rangle = \Sigma_{1 \leq i \leq k} p_i H_i$. This induces a *scalar product* $\langle P | Q \rangle = \langle P \cdot Q \rangle$, and a classical result (cf., for instance, Wall [32]) states:

PROPOSITION 6. *Let* $\overline{Q}_k(z) = z^{k+1} Q_k(1/z)$ *be the reciprocal polynomial of* $Q_k(z)$, *introduced in Proposition* 4. *The following orthogonality relations hold*:

$$\langle x^i | \overline{Q}_{k-1}(x) \rangle = \langle \overline{Q}_{i-1} | \overline{Q}_{k-1} \rangle = 0 \qquad \text{for } 0 \leq i < k;$$

$$\langle x^k | \overline{Q}_{k-1}(x) \rangle = \langle \overline{Q}_{k-1} | \overline{Q}_{k-1} \rangle = a_0 a_1 \cdots a_{k-1} s_1 s_2 \cdots s_k.$$

In other words, the $\overline{Q}_k$ form a basis for polynomials which is orthogonal with respect to the scalar product associated with the sequence $\{H_n | n \geq 0\}$. Proposition 5 follows from a more general result, expressing histories

by orthogonality relations:

PROPOSITION 7. *The number $H_{k,l,n}$ of histories of length n, starting at level k and finishing at level l, is given by*

$$H_{k,l,n} = \frac{1}{\psi(k,l)} \langle \bar{Q}_{k-1}(x) \bar{Q}_{l-1}(x) x^n \rangle,$$

with $\psi(k,l) = a_0 a_1 \ldots a_{k-1} s_1 s_2 \ldots s_l$.

Proof. Proposition 4 yields

$$H_{k,l}(z) = \frac{1}{\psi(k,l) z^{k+l}} \left[Q_{k-1}(z) Q_{l-1}(z) H(z) - P_{\lambda-1}(z) Q_{\mu-1}(z) \right].$$

Since the degree of $P_{\lambda-1} Q_{\mu-1}$ is strictly less than $k + l + 1$, it follows that $\psi(k,l) \cdot H_{k,l,n}$ is the coefficient of z^{k+l+n} in $Q_{k-1}(z) Q_{l-1}(z) H(z)$. Using $\langle x^p \rangle = H_p$ and elementary substitutions yields Proposition 7.

Proposition 6 follows by setting $k = 0$, and noting that $H_{0,l,n} = 0$ for $n < l$, since l steps are necessary to reach level l. □

4. Data Types and the Classical Orthogonal Polynomials

The preceding section provides formulas for the number of histories $H_{k,l,n}$, which are expressed only in terms of the orthogonal polynomials associated with the corresponding continued fraction.

Each data type, defined by its possibility set $\pi = \{\text{pos}(O,k) | O \in \Omega, k \geq 0\}$ is thus characterized by a family of orthogonal polynomials $\{\bar{Q}_{k-1} | k \geq 0\}$. Let us first recognize the polynomials associated with each of our five data types.

4.1. *Stacks and Tchebycheff Polynomials*

Polynomials associated with stacks satisfy $\bar{Q}_{-1} = 1$, $\bar{Q}_0 = z$, $\bar{Q}_k = z\bar{Q}_{k-1} - \bar{Q}_{k-2}$; thus $\bar{Q}(z,t) = \Sigma_{k \geq 0} \bar{Q}_{k-1}(z) t^k = 1/(1 - zt + t^2)$. Elementary manipulations lead to the explicit form

$$\bar{Q}_{k-1}(z) = \Sigma_{i \geq 0} (-1)^i \binom{k-i}{i} z^{k-2i},$$

a Tchebycheff polynomial.

THEOREM 1,S. *The Tchebycheff polynomials associated with stacks have a generating function* $\bar{Q}(z,t) = 1/(1 - zt + t^2)$. *Stack histories admit*

$$\sum_{n \geq 0} H_n z^n = \frac{1 - (1 - 4z^2)^{1/2}}{2z^2} = \sum_{n \geq 0} \frac{1}{n+1} \binom{2n}{n} z^{2n}$$

and

$$\sum_{n,k\geq 0} H_{0,k,n}t^k z^n = \sum_{n,k\geq 0} H_{k,0,n}t^k z^n = \frac{1-(1-4z^2)^{1/2}}{2z^2 - zt\left(1-(1-4z^2)^{1/2}\right)}$$

for generating functions.

Proof. Computing $\langle \bar{Q}(xt)\rangle$ in two different ways yields

$$\left\langle \frac{1}{1-xt+t^2}\right\rangle = \frac{1}{1+t^2}\left\langle \frac{1}{1-xt/(1+t^2)}\right\rangle = \frac{1}{1+t^2}\sum_{n\geq 0} H_n\left(\frac{t}{1+t^2}\right)^n$$

and

$$\left\langle \sum_{k\geq 0} \bar{Q}_{k-1}(x)t^k\right\rangle = \sum_{k\geq 0}\langle \bar{Q}_{-1}(x)|\bar{Q}_{k-1}(x)\rangle t^k = 1;$$

setting $t = (1-(1-4z^2)^{1/2})/2z$ leads to

$$\sum_{n\geq 0} H_n z^n = \left(1-(1-4z^2)^{1/2}\right)/2z^2.$$

Proceeding similarly with $\langle \bar{Q}(x,u)\bar{Q}(x,t)\rangle$ leads to

$$\sum_{k,l}\langle \bar{Q}_{k-1}(x)\bar{Q}_{l-1}(x)\rangle u^k t^l = \frac{1}{1-ut}$$

on the one hand, and to

$$\frac{1}{(1+t^2)}\sum_{k,n}\langle \bar{Q}_{k-1}(x)x^n\rangle u^k\left(\frac{t}{1+t^2}\right) = \frac{1}{(1+t^2)}\sum_{k,n} H_{0,k,n}u^k\left(\frac{t}{1+t^2}\right)^n$$

on the other. Identifying these two expressions and changing t to $(1-(1-4z^2)^{1/2})/2z$ yields the result. □

4.2. *Dictionaries and Laguerre Polynomials*

The polynomials associated with dictionaries satisfy $\bar{Q}_{-1}=1$, $\bar{Q}_0 = z-1$, $\bar{Q}_k = (z-2k-1)\bar{Q}_{k-1} - k^2\cdot\bar{Q}_{k-2}$. This recurrence translates, over the generating function

$$\bar{Q}(z,t) = \sum_{k\geq 0}\bar{Q}_{k-1}(z)\frac{t^k}{k!}$$

into the differential equation

$$\frac{\partial}{\partial t}\bar{Q}(z,t) = \bar{Q}(z,t)\cdot\frac{z-1-t}{(1+t)^2},$$

whose solution is

$$\bar{Q}(z, t) = \frac{1}{1+t} \exp z \frac{t}{1+t};$$

expanding leads to the explicit form

$$\bar{Q}_{k-1}(z) = \sum_{i \geq 0} (-1)^i \binom{k}{i} \frac{k!}{i!} z^i,$$

a Laguerre polynomial.

THEOREM 1,D. *The Laguerre polynomials associated with dictionaries admit*

$$\sum_{k \geq 0} \bar{Q}_{k-1}(z) \frac{t^k}{k!} = \frac{1}{1+t} \exp z \frac{t}{1+t}$$

for exponential generating function. As for dictionary histories:

$$H(z) = \sum_{n \geq 0} H_n \frac{z^n}{n!} = \frac{1}{1-z},$$

thus $H_n = n!$ *and*

$$H(u, v, z) = \sum_{k, l, n \geq 0} H_{k, l, n} u^k v^l \frac{z^n}{n!} = \frac{1}{1 - z(1+u)(1+v) - uv}.$$

Proof. As for stacks,

$$\langle \bar{Q}(x, t) \rangle = \left\langle \sum_{k \geq 0} \bar{Q}_{k-1}(x) \frac{t^k}{k!} \right\rangle = 1$$

on the one hand, and

$$\begin{aligned} \langle \bar{Q}(x, t) \rangle &= \frac{1}{1+t} \left\langle \sum_{n \geq 0} \frac{x^n}{n!} \left(\frac{t}{1+t} \right)^n \right\rangle \\ &= \frac{1}{1+t} \sum_{n \geq 0} \frac{1}{h!} H_n \left(\frac{t}{1+t} \right)^n, \end{aligned}$$

on the other; letting $t/(1+t) = z$ leads to

$$\sum_{n \geq 0} H_n \frac{z^n}{n!} = \frac{1}{1-z} = H(z).$$

Let $A(u, v, z) = \langle \bar{Q}(x, u)\bar{Q}(x, v)\bar{Q}(x, z/(1-z))\rangle$; we compute

$$A(u, v, z) = \sum_{n\geq 0} \left\langle \bar{Q}(x, u)\bar{Q}(x, v)x^n \frac{z^n}{n!} \right\rangle (1-z)$$

$$= (1-z) \sum_{k,l,n} \frac{u^k}{k!} \cdot \frac{v^l}{l!} \langle \bar{Q}_{k-1}(x)\bar{Q}_{l-1}(x)x^n \rangle \frac{z^n}{n!}.$$

By Proposition 7,

$$H_{k,l,n} = \frac{1}{k!l!} \langle \bar{Q}_{k-1}(x)\bar{Q}_{l-1}(x)x^n \rangle;$$

thus $A(u, v, z) = (1-z)H(u, v, z)$. Replacing $\bar{Q}$ by its expression in A gives

$$A(u, v, z) = \frac{1-z}{1+u} \cdot \frac{1}{1+v} \left\langle \exp x\left(z + \frac{u}{1+u} + \frac{v}{1+v}\right) \right\rangle$$

$$= \frac{1-z}{(1+u)(1+v)} H\left(z + \frac{u}{1+u} + \frac{v}{1+v}\right)$$

$$= \frac{1-z}{1-(1+u)(1+v)z+uv} = (1-z)H(u, v, z). \qquad \square$$

This treatment applies mutatis mutandis to the remaining data types, and we merely state the results.

4.3. *Priority Queues and Hermite Polynomials*

The relevant polynomials here are Hermite polynomials

$$\bar{Q}_{k-1}(z) = \sum_{0\leq i\leq k/2} (-1)^i \frac{k!}{2^i i!\,(k-2i)!} z^{k-2i}.$$

THEOREM 1,PQ. *The Hermite polynomials associated with priority queues admit*

$$\sum_{k\geq 0} \bar{Q}_{k-1}(z) \frac{t^k}{k!} = \exp\left(zt - \frac{t^2}{2}\right)$$

for an exponential generating function. As for priority queue histories:

$$H(z) = \sum_{n\geq 0} H_n \frac{z^n}{n!} = \exp\left(\frac{z^2}{2}\right),$$

$$H(u, v, z) = \sum_{k,l,n\geq 0} H_{k,l,n} u^k \frac{v^l}{l!} \frac{z^n}{n!} = \exp\left(\frac{z^2}{2} + zu + uv + vz\right).$$

4.4. *Linear Lists and Meixner Polynomials*

THEOREM 1,LL. *The Meixner polynomials associated with linear lists admit*

$$\sum_{k\geq 0} \bar{Q}_{k-1}(z)\frac{t^k}{k!} = \frac{1}{(1+t^2)^{1/2}}\exp(z \text{ arc tg } t)$$

for an exponential generating function. As for histories:

$$H(z) = \sum_{n\geq 0} H_n \frac{z^n}{n!} = \frac{1}{\cos z},$$

and

$$H(u, v, z) = \sum_{k,l,n\geq 0} H_{k,l,n} u^k v^l \frac{z^n}{n!} = \frac{1}{(1-uv)\cos z - (u+v)\sin z}.$$

4.5. *Symbol Tables and Charlier Polynomials*

THEOREM 1,ST. *The Charlier polynomials associated with symbol tables admit*

$$\sum_{k\geq 0} \bar{Q}_{k-1}(z)\frac{t^k}{k!} = (1+t)^{z+1}e^{-t}$$

for an exponential generating function. As for histories,

$$H(z) = \sum_{n\geq 0} H_n \frac{z^n}{n!} = \exp(e^z - z - 1)$$

and

$$H(u, v, z) = \sum_{n\geq 0} H_{k,l,n} u^k \frac{v^l}{l!}\frac{z^n}{n!}$$

$$= \exp(e^z(1+u)(1+v) - 1 - z - u - v).$$

4.6. *Other Data Types*

The reader might be curious, at this point, to know how many "classical" orthogonal polynomials there are, and what are the polynomials associated with other data structures, such as dequeue ($\text{pos}(A, k) = \text{pos}(S, k) = 2$), or linear lists with interrogations ($\text{pos}(A, k-1) = \text{pos}(S, k) = k$, $\text{pos}(Q, k) = k$), etc.

A partial answer to these questions can be given, provided we restrict ourselves to possibilities $\text{pos}(O, k) = \alpha k + \beta$ which are linear functions of k. Note, in this case, that the continued fraction expressing histories is the quotient of two diverging hypergeometric functions, as shown by Perron [28, Vol. 2, p. 288]. If we further restrict the product $\text{pos}(A, k) \cdot \text{pos}(S, k)$ to be of the form $k(\alpha k + \beta)$ with $\alpha + \beta > 0$, then the associated orthogonal polynomials are within the class of Meixner polynomials (Meixner [27]; see also Chihara [5, pp. 163–166]), which comprise only five generic families of polynomials. Data types that fall in that category are thus amenable to a treatment similar to that applied in one of the cases considered in this paper.

5. The Integral Cost Theorem

The preceding section provides expressions for the number $H_{k,l,n}$ of histories of length n, starting at level k and finishing at level l. From the definition of level crossing numbers (Section 2.4), we infer the formulas

$$NA_{k,n} = \sum_{0 \le i < n} H_{0,k,i} \cdot a_k \cdot H_{k+1,0,n-i-1},$$

$$NQ_{k,n} = \sum_{0 \le i < n} H_{0,k,i} \cdot q_k \cdot H_{k,0,n-i-1},$$

$$NS_{k,n} = \sum_{0 \le i < n} H_{0,k,i} \cdot s_k \cdot H_{k-1,0,n-i-1}.$$

We are thus in possession of all the quantities needed in order to apply the integral cost formula:

$$KA_n = \sum_{k \ge 0} CA_k \cdot NA_{k,n}, \qquad KQ_n = \sum_{k \ge 0} CQ_k \cdot NQ_{k,n},$$

$$KS_n = \sum_{k \ge 1} CS_k \cdot NS_{k,n} \qquad \text{and} \qquad K_n = KA_n + KQ_n + KS_n.$$

5.1. *Integrated Cost for Stacks*

In the case of stacks, $a_k = s_k = 1$ and $q_k = 0$; thus $NA_k(z) = \sum_{n \ge 0} NA_{k,n} z^n = zH_{0,k}(z) \cdot H_{k+1,0}(z)$, where $H_{a,b}(z) = \sum_{n \ge 0} H_{a,b,n} z^n$. By Theorem 1,S, $H_{0,k}(z) = H_{k,0}(z) = z^k B(z)^{k+1}$, where

$$B(z) = \frac{1 - (1 - 4z^2)^{1/2}}{2z^2}.$$

Substituting in the generating function $KA(z) = \sum_{n\geq 0} KA_n z^n$ leads to

$$KA(z) = \sum_{k\geq 0} CA_k \cdot z^{2k+2}(B(z))^{2k+3}$$
$$= z^2 B(z)^3 CA(z^2 B^2(z)),$$

where we let $CA(t) = \sum_{k\geq 0} CA_k \cdot t^k$ represent the generating function of unitary adjunction costs for the stack implementation. A similar treatment can be applied to KS, and we find:

THEOREM 2,S. *The generating functions of unitary costs $CA(t)$ and $CS(t)$ and integrated costs $KA(z)$ and $KS(z)$ for stacks are related by the linear transform $KA(z) = z^2 B^3(z) CA(z^2 B^2(z))$ and $KS(z) = B(z) CS(z^2 B^2(z))$, where*

$$B(z) = \frac{1 - (1 - 4z^2)^{1/2}}{2z^2}.$$

5.2. *Integrated Costs for Dictionaries*

The formula $NA_k(z) = a_k z H_{0,k}(z) \cdot H_{k+1,0}(z)$ is of no direct use here, since the ordinary generating functions $H_{0,k}(z)$ diverge for all real z. Theorem 1,D, however, provides an analytic expression for the exponential generating function

$$\hat{H}_{a,b}(z) = \sum_{n\geq 0} H_{a,b,n} \frac{z^n}{n!}.$$

This leads to an expression for the exponential generating function of level crossings

$$NA_k(z) = \sum_{n\geq 0} NA_{k,n} \frac{z^n}{n!},$$

through the classical convolution theorem for Laplace transforms:

PROPOSITION 8. *The exponential generating functions for level crossing numbers $\widehat{NO}_k(z)$ and paths $\hat{\hat{H}}_{a,b}(z)$ are related by*

$$\widehat{NA}_k(z) = a_k \hat{H}_{0,k}(z) * \hat{H}_{k+1,0}(z),$$
$$\widehat{NQ}_k(z) = q_k \hat{H}_{0,k}(z) * \hat{H}_{k,0}(z),$$
$$\widehat{NS}_k(z) = s_k \hat{H}_{0,k}(z) * \hat{H}_{k-1,0}(z),$$

where $*$ denotes (Laplace) convolution

$$(\hat{A} * \hat{B})(x) = \int_0^x \hat{A}(x - \tau)\hat{B}(\tau)\, d\tau.$$

Proof. We start with the classical lemma expressing the fact that the Laplace transform maps a convolution product into an ordinary Cauchy product: let

$$\hat{A}(x) = \sum_{n \geq 0} a_n \frac{x^n}{n!} \qquad \text{and} \qquad \hat{B}(x) = \sum_{n \geq 0} b_n \frac{x^n}{n!};$$

the product $(\hat{A} * \hat{B})(x) = \int_0^x \hat{A}(x - \tau)\hat{B}(\tau)\, d\tau$ is equal to

$$(\hat{A} * \hat{B})(x) = \sum_{n \geq 0} \left(\sum_i a_i b_{n-i} \right) \frac{x^{n+1}}{(n+1)!}.$$

The (purely algebraic) proof starts with

$$P = \int_0^x \hat{A}(x - \tau)\hat{B}(\tau)\, d\tau = \sum_{n,\, m \geq 0} \int_0^x \frac{a_n b_m}{n!m!} (x - \tau)^n \tau^m\, d\tau$$

$$= \sum_{n,\, m,\, k \geq 0} \int_0^x (-1)^k \frac{a_n b_m}{n!m!} \binom{n}{k} x^{n-k} \tau^{m+k}\, d\tau$$

$$= \sum_{n,\, m \geq 0} \frac{a_n b_m}{n!m!} x^{n+m+1} \sum_k \frac{(-1)^k}{m + k + 1} \binom{n}{k}.$$

Using the well-known inversion formula for binomial coefficients,

$$\sum_k \frac{(-1)^k}{m + k + 1} \binom{n}{k} = 1 \Big/ \binom{n + m + 1}{n},$$

we get

$$P = \sum_{n,\, m \geq 0} a_n b_m \frac{x^{n+m+1}}{(n + m + 1)!}.$$

The expressions for $\widehat{NO}_k(z)$ then follow directly from the formulas given at the beginning of Section 5. □

Theorem 1,D yields the expression

$$\hat{H}_{0,k}(z) = \hat{H}_{k,0}(z) = z^k / (1 - z)^{k+1}.$$

Substituting in the formula for $\widehat{KA}(z)$:

$$\widehat{KA}(z) = \sum_{k \geq 0} CA_k \cdot \widehat{NA}_k(z)$$

$$= \sum_{k \geq 0} (k + 1) CA_k \int_0^z \frac{(z - \tau)^k}{(1 - z + \tau)^{k+1}} \frac{\tau^{k+1}}{(1 - \tau)^{k-2}}\, d\tau$$

$$= \int_0^z \mathcal{C}_A \left(\frac{\tau(z - \tau)}{(1 - z + \tau)(1 - \tau)} \right) \frac{\tau\, d\tau}{(1 - z - \tau)(1 - \tau)^2};$$

where $\mathcal{C}_A(x) = \Sigma_{k\geq 0}(k+1)CA_k x^k$. Splitting the integral $\int_0^z = \int_0^{z/2} + \int_{z/2}^z$ and performing the respective changes of variable $\tau = z/2 - \sigma$ and $\tau = z/2 + \sigma$ lead to

$$\widehat{KA}(z) = 2\int_0^{z/2} \mathcal{C}_A\left(\frac{\alpha^2 - \sigma^2}{\beta^2 - \sigma^2}\right) \frac{\alpha\beta + \sigma^2}{(\beta^2 - \sigma^2)^2}\, d\sigma$$

with $\alpha = z/2$, $\beta = 1 - \alpha$. Setting $u = (\alpha^2 - \sigma^2)/(\beta^2 - \sigma^2)$ yields, after simplifications,

$$\widehat{KA}(z) = \int_0^{t^2} \mathcal{C}_A(u)\left(\frac{t-u}{1-u}\right)^{1/2} du \qquad \text{with } t = \frac{z}{2-z}.$$

A similar treatment is applied to $\widehat{KQ}^+$, $\widehat{KQ}^-$, and $\widehat{KS}$ and we find:

THEOREM 2,D. *Exponential generating functions*

$$\widehat{KO}(z) = \sum_{n\geq 0} KO_n \cdot \frac{z^n}{n!}$$

of the integrated cost of dictionaries are related, for each operation O, to the generating functions of unitary costs, by the following linear integral transforms:

$$\widehat{KA}(z) = \int_0^{t^2} \mathcal{C}_A(u)\left(\frac{t-u}{1-u}\right)^{1/2} du,$$

$$\mathcal{C}_A(x) = \sum_{k\geq 0} (k+1)CA_k \cdot x^k, \qquad t = \frac{z}{2-z};$$

$$\widehat{KQ}(z) = \frac{2}{2-z}\int_0^{t^2} \mathcal{C}_Q(u) \frac{du}{((1-u)(t-u))^{1/2}},$$

$$\mathcal{C}_Q(x) = \sum_{k\geq 0} (kCQ_k^+ + (k+1)CQ_k^-)x^k;$$

$$\widehat{KS}(z) = \int_0^{t^2} \mathcal{C}_S(u)\left(\frac{t-u}{1-u}\right)^{1/2} du,$$

$$\mathcal{C}_S(x) = \sum_{k\geq 0} (k+1)CS_{k+1} \cdot x^k$$

Of course, $\hat{K}(z) = \widehat{KA}(z) + \widehat{KQ}(z) + \widehat{KS}(z)$ can be expressed in terms of $CO(x) = \Sigma_{k\geq 0} CO_k \cdot x^k$ rather than in terms of the modified $\mathcal{C}_0$ above;

for example, integration by parts provides

$$\widehat{KA}(z) = t^2 CA(t^2)\left(\frac{t}{1+t}\right)^{1/2} - \frac{1-t}{2}\int_0^{t^2} CA(u)\frac{u\,du}{(1-u)((t-u)(1-u))^{1/2}},$$

an expression which is less convenient to work with.

5.3. *Integrated Costs for Priority Queues*

For priority queues, Theorem 1,PQ gives us $\hat{H}_{0,k}(z) = k!\cdot\hat{H}_{k,0}(z) = z^k \exp(z^2/2)$. Following the same computation as that for dictionaries leads to

$$\hat{K}(z) = e^{z^2/2}\int_0^z \tau\cdot C(\tau(z-\tau))\exp((\tau - z)\tau)\,d\tau$$

with

$$C(x) = \sum_{k\geq 0}(CA_k + CS_{k+1})\frac{x^k}{k!}.$$

In order to simplify the expression for K, let us formally set $C(x) = \exp(ux)$, where u^k should be identified with $CA_k + CS_{k+1}$. Changing the variable to $\mu = \tau - z/2$ in the integral,

$$\hat{K}(z) = e^{z^2/2}\int_{-z/2}^{z/2}(\mu + s/2)e^{(-\mu^2+z^2/4(u-1))}\,d\mu$$

$$= ze^{z^2/4(u+1)}\int_0^{z/2} e^{\mu^2(1-u)}\,d\mu,$$

thus expressing K in terms of the *Erf* function of probability theory. We check that $\hat{K}(z)$ is an even function of z, as expected since $H_{0,0,2n+1} = 0$. In

$$\hat{K}(z)/z = \sum_{\substack{k\geq 0\\ n\geq 0}} N_{k,2n}u^k s^{2n-1}/(2n)!,$$

set $z = 2s^{1/2}$ and apply a Borel–Laplace transform $\mathcal{B}(f;t) = \int_0^\infty e^{-s}f(st)\,ds$. We obtain, on the one hand,

$$E = \mathcal{B}(\lambda z\cdot K(z)/z;t) = \sum_{k,n\geq 0} N_{k,2n}u^k\int_0^\infty e^{-s}(st)^{n-1/2}\,ds$$

$$= \frac{\Gamma(1/2)}{2t^{1/2}}\sum_{\substack{k\geq 0\\ n\geq 0}} N_{k,2n}u^k\frac{t^n}{n!}.$$

Applying the transform to the integral expression above yields, on the other hand,

$$E = \int_0^\infty e^{-s} \cdot e^{st(u+1)} \int_0^{(st)^{1/2}} e^{\mu^2(1-u)}\, d\mu\, ds;$$

integrating by parts leads to

$$E = \frac{t^{1/2}}{1 - t(u+1)} \int_0^\infty e^{-s(1-2t)} \frac{ds}{2s^{1/2}} = \frac{t^{1/2}}{2} \cdot \frac{\Gamma(1/2)}{(1-2t)^{1/2}} \cdot \frac{1}{1 - t(u+1)},$$

and, expanding back as a power series in u, we identify both terms to obtain a remarkably simple expression:

THEOREM 2,PQ. *The generating function*

$$\tilde{K}(z) = \sum_{n \geq 0} K_{2n} \frac{z^n}{n!}$$

of the integrated cost of priority queues is related to $C(x) = \Sigma_{k \geq 0}(CA_k + CS_{k+1})x^{k+1}$ *by*

$$K(z) = \frac{1}{(1-2z)^{1/2}} \cdot C\left(\frac{z}{1-z}\right).$$

5.4. *Integrated Cost for Linear Lists and Symbol Tables*

Computation of integrated costs for linear lists, symbol tables, and in fact any data structure whose associated polynomial falls within the Meixner class (see Section 4.6) can be carried out along the lines followed for dictionaries. Simplifications such as those found for priority queues, however, have not been apparent to the authors.

THEOREM 2,LL. *The exponential generating function* $K(z)$ *of integrated costs for linear lists is given by*

$$\hat{K}(z) = 2 \int_0^{\operatorname{tg}^2 z/2} C(u) \frac{du}{u\left((1-u)^2 - 4u \cot^2 z\right)^{1/2}},$$

where

$$C(x) = \sum_{k \geq 0} (k+1)(CA_k + Cs_{k+1})x^{k+1}.$$

THEOREM 2,ST. *The exponential generating function $K(z)$ of integrated costs for symbol tables is given by*

$$\hat{K}(z) = 2e^{e^z - z - 1} \int_0^{(e^{z/2}-1)^2} ((e^z + 1 - u)\mathcal{C}_{AS}(u) + \mathcal{C}_Q(u)) \times \frac{e^{-u}\, du}{((e^z + 1 - u)^2 - 2e^z)^{1/2}},$$

where

$$\mathcal{C}_{AS}(x) = \sum_{k \geq 0} (CA_k + CS_{k+1}) \frac{x^k}{k!} \quad \text{and} \quad \mathcal{C}_Q(u) = \sum_{k \geq 0} CQ_k \frac{x^k}{k!}.$$

6. Application to Computing Integrated Costs of Some Relevant Data Organizations

It now remains to use the individual costs tabulated in Fig. 5, Section 2, in conjunction with the integral formulas given in Section 5. With the exception of binomial queues, all the individual costs are linear combinations of the functions (of k)

$$\frac{1}{k};\ \frac{1}{k+1};\ H_k;\ 1;\ k;\ k^2;\ kH_k,$$

whose ordinary generating functions have the simple forms

$$\ln\frac{1}{1-x};\ \frac{1}{x}\ln\frac{1}{1-x};\ \frac{1}{1-x}\ln\frac{1}{1-x};\ \frac{1}{1-x};$$

$$\frac{1}{(1-x)^2};\ \frac{x}{(1-x)^3};\ \frac{x}{(1-x)^2}\left(1 + \ln\frac{1}{1-x}\right).$$

Vuillemin's binomial queue implementation of priority queues represents a somewhat different problem: the unitary costs here are

$$C_k = CA_k + CS_{k+1} = \frac{1}{k+1} \sum_{i \geq 0} ib_i 2^i,$$

where $\Sigma_{i \geq 0} b_i 2^i$ is the binary representation of $(k + 1)$. The corresponding integrated costs have been evaluated (using Theorem 2,PQ) by Cheno [4],

TABLE 6D
Integrated Costs for Stationary Dictionary Structures

Structure	Integrated cost for $\mathcal{K}_n$
Sorted list	$(n^2 + 9n - 4)/12$
Unsorted list	$(9n^2 + 47n - 62)/120$
Binary search tree	$2n(H_n - 2) + O(\log^2 n)$

who finds

$$\frac{K_{2n}}{1 \cdot 3 \cdot 5 \cdot \cdots \cdot (2n-1)} = n \cdot \log_2 n - n \cdot \alpha(n) + o(n),$$

where $\alpha(n)$ is some periodic function of $\log_2 n$.

Tables 6D, 6PQ, and 6LL give integrated costs for stationary dictionaries, priority queues, and linear lists, respectively, corresponding to data organizations of Fig. 5. The only case in which we were unable to derive a simple expression is the position tournament representation for linear lists.

Such costs are not necessarily decisive in evaluating the practical value of these structures, since we count only the cost of key comparison. There is, however, nothing to stop us from computing the integrated cost for a more realistic measure, say the execution time in MIX units, as Knuth [23] is fond of doing. This is a large but routine computational task!

TABLE 6PQ
Integrated Costs for Stationary Priority Queue Structures[a]

Structure	Integrated cost for $\mathcal{K}_{2n}$
Sorted list	$n(n+5)/6$
Binary search tree	$\frac{2n!}{n?}\left\{\sum_{1\le i<n}\frac{i?}{i!}\left[\frac{i(2^{n-i}-i)}{2i-1}H_{n-i}+\frac{2^{n-i}-1}{n-i}\right]+\frac{2^n-1}{n}\right\}-2n$ $= n \ln n + O(n)$
Binary tournament	$\frac{n!}{n?}\left\{\sum_{1\le i<n}\frac{i?}{i!}\left[\frac{3i(2^{n-i}-i)}{2i-1}H_{n-i}+5\frac{2^{n-i}-1}{n-i}\right]+5\frac{2^n-1}{n}\right\}-\frac{9n}{2}$ $= \frac{3}{2} n \ln n + O(n)$
Pagoda	Identical to binary search trees
Binomial queue	$n \log_2 n - n \cdot \alpha(n) + o(n)$,

[a] Here $n? = 1 \cdot 3 \cdot 5 \cdot \cdots \cdot 2n - 1$ and $\ln x$ is the natural logarithm of x.

TABLE 6LL
Integrated Costs for Stationary Linear List Structures[a]

Structure	Integrated cost for $\mathcal{H}_{2n}$
Sequential list	$(E_{2n+2} + 4n)/(4E_{2n}) - n - 1/4$

[a]Here E_{2n} is the secant number: $\Sigma_{n\geq 0} E_{2n}(z^{2n}/(2n)!) = 1/\cos z)$.

7. Conclusions; Directions for Further Research

We have presented here in detail a method for analyzing sequences of operations in stationary data structures belonging to one of five basic types. This approach can be extended in several different ways:

(a) by varying the set of histories over which the analysis is performed: initial and final conditions can be altered, and the condition that histories go back to the empty file can be relaxed. The generating functions given in Section 4 are general enough to yield convolution integral expressions for generating functions of integrated costs.

(b) by varying the universe of possible keys: the case where keys are drawn from a finite set (a "reference file") can in some instances be dealt with along similar lines (see Flajolet and Françon [9]).

(c) by considering other data types: those where only the four basic operations are allowed are amenable to the continued fraction approach. If further, the possibility functions are linear in the size of the file, the convergent polynomials can be explicitly determined and lead to the Meixner classification; this is the case for dequeues, double-ended priority queues, priority queues with various types of interrogations, ... On the other hand we are lacking a general approach for data types involving union as in mergeable priority queues or dictionaries. Histories for "hash coding" dictionaries should also be of interest.

(d) by establishing more connections with probabilistic approaches: for instance, Jonassen and Dahl [20] have established that, for priority queues, drawing keys from an exponential distribution entails the equiprobability of histories (see also Knuth [25]; Jonassen and Knuth [21]).

Acknowledgments

The authors would like to thank J. Giraud and G. Viennot for several interesting discussions relative to this work.

REFERENCES

1. A. V. AHO, J E. HOPCROFT, AND J. D. ULLMAN, "The Design and Analysis of Computer Algorithms," Addison–Wesley, Reading, Mass., 1974.
2. M. R. BROWN, Implementation and analysis of binomial queue algorithms, *SIAM J. Comput.* 7, No. 3 (August 1978), 298–319.
3. R. BROWN AND R. E. TARJAN, A representation for linear lists with movable fingers, *in* "Proceedings, Tenth Annual ACM Symposium on the Theory of Computing, 1978," pp. 19–29.
4. L. CHÉNO, "Formes asymptotiques des coûts de files de priorité," Mémoire de DEA; Fac. Sci. d'Orsay, 1979.
5. T. S. CHIHARA, "An Introduction to Orthogonal Polynomials," Gordon & Breach, New York, 1978.
6. J. DOYLE AND R. L. RIVEST, Linear expected time of a simple union–find algorithm, *Inform. Proc. Lett.* 5 (1976), 146–148.
7. P. FLAJOLET, "Analyse d'algorithmes de manipulation de fichiers," Iria-Laboria Report No. 321, August 1978.
8. P. FLAJOLET, "Analyse d'algorithmes de manipulation d'arbres et de fichiers," Thèse, Fac. Sci. Orsay, Sept. 1979.
9. P. FLAJOLET AND J. FRANÇON, Sequence of operations analysis of data structures under restricted sets of keys, in preparation.
10. P. FLAJOLET, J. FRANÇON, G. VIENNOT, AND J. VUILLEMIN, Algorithmique et combinatoire des arbres et des permutations, to appear.
11. P. FLAJOLET, J. FRANÇON, AND J. VUILLEMIN, Computing the integrated cost of dictionaries, *in* "Proceedings, 11th ACM Symposium of Theory of Computing, 1979."
12. J. FRANÇON, Arbres binaires de recherche, propriétés combinatoires et applications, *RAIRO Inform. Theor.* **10** (1976), 35–50.
13. J. FRANÇON, Histoires de fichiers, *RAIRO Inform. Theor.* **12** (1978), 49–67.
14. J. FRANÇON, "Combinatoire des structures de données," Thèse, Université Louis Pasteur, Strasbourg, 1979.
15. J. FRANÇON AND G. VIENNOT, Permutations selon les pics, creux, doubles, montées, doubles descentes, nombre d'Euler et nombres de Genocchi, *Discrete Math.*, in press.
16. J. FRANÇON, G. VIENNOT, AND J. VUILLEMIN, "Description et analyse d'une représentation performante des files de priorité," Rapport No. 12 du Laboratoire d'Informatique, 91405 Orsay; *Acta Inform.*, in press.
17. L. J. GUIBAS, E. M. MCCREIGHT, M. F. PLASS, AND J. R. ROBERTS, A new representation for linear lists, *in* "Proceedings, 9th Annual ACM Symposium on the Theory of Computing, Boulder, Colorado, 1977," pp. 49–60.
18. L. J. GUIBAS AND R. SEDGEWICK, A dichromatic framework for balanced trees, *in* "Proceedings, 19th annual IEE Symp. on the Foundations of Computer Science, 1978," pp. 8–21.
19. T. N. HIBBARD, Some combinatorial properties of certain trees with applications to searching and sorting, *J. Assoc. Comput. Mach.* **9** (1962), 13–28.
20. A. JONASSEN AND O.-J. DAHL, "Analysis of an Algorithm for Priority Queue Administration," Math. Inst., University of Oslo, 1975.
21. A. JONASSEN AND D. E. KNUTH, "A Trivial Algorithm Whose Analysis Isn't," Stanford University, Report STAN-CS-77-598, March 1977.
22. G. D. KNOTT, "Deletion in Binary Storage Trees," Ph.D. thesis, Computer Science Department, Stanford University, Report STAN-CS-75-491, May 1975.
23. D. E. KNUTH, "The Art of Computer Programming," Vol. 1, "Fundamental Algorithms," Addison–Wesley, Reading, Mass., 1968.

24. D. E. KNUTH, "The Art of Computer Programming," Vol. 3, "Sorting and Searching," Addison–Wesley, Reading, Mass., 1973.
25. D. E. KNUTH, Deletions that preserve randomness, *IEEE Trans. Software Engrg.* **SE 3** (1977), 351–359.
26. D. E. KNUTH AND A. SCHÖNAGE, "The Expected Linearity of a Simple Equivalence Algorithm," Stanford University, Report STAN-CS-77-599, March 1977.
27. MEIXNER, Orthogonale Polynomsysteme mit einem besonderen Gestalt der erzeugenden Funktion, *J. London Math. Soc.* **9** (1934), 6–13.
28. O. PERRON, "Die Lehre von den Kettenbrüchen," 2 vols., Teubner Verlagsgesellschaft, Stuttgart, 1954.
29. R. E. TARJAN, Efficiency of a good but not linear set union algorithm, *J. Assoc. Comput. Mach.* **22** (1975), 215–225.
30. J. VUILLEMIN, A data structure for manipulating priority queues, *Comm. ACM* **21**, No. 4 (1978), 309–315.
31. J. VUILLEMIN, A representation for linear lists with good average time performance, *in* "Proceedings, ISCAJ Symposium, Tokyo, 1979."
32. H. S. WALL, "Analytic Theory of Continued Fractions," Chelsea, New York, 1967.
33. A. C.-C. YAO, On the average behavior of set merging algorithms (extended abstract), *Proc. ACM Symp. Theor. Comput.* **8** (1976), 192–195.
34. C. A. CRANE, "Linear Lists and Priority Queues as Balanced Binary Trees," Ph.D. thesis, STAN-CS-72-259, Stanford University, 1972.
35. L. SNYDER, On uniquely represented data structures, *Proc. IEEE Symp. Foundations Comput. Sci.* **18** (1977), 142–146.