Average case lower bounds on the construction and searching of partial orders

H.G. Mairson

INRIA
CENTRE DE ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél. (3) 954 90 20

Rapports de Recherche

N° 415

AVERAGE CASE LOWER BOUNDS ON THE CONSTRUCTION AND SEARCHING OF PARTIAL ORDERS

Harry G. MAIRSON

Juin 1985

AVERAGE CASE LOWER BOUNDS ON THE CONSTRUCTION AND SEARCHING OF PARTIAL ORDERS

Harry G. Mairson
Stanford University *

INRIA
Domaine de Voluceau - Rocquencourt
78153 Le Chesnay
FRANCE

Résumé :

Nous analysons le compromis entre le tri et la recherche, en utilisant le modèle des arbres de comparaison. Soit P un algorithme qui produit un ensemble d'ordres partiels, étant donné un ensemble U de n éléments. Soit S un algorithme de recherche dans ces ordres partiels. Si les $n!$ permutations des éléments sont équiprobables, et si on recherche chaque $y \in U$ avec la même probabilité, on peut calculer les coûts moyens $\bar{P}(n)$ et $\bar{S}(n)$ (i.e., le nombre de comparaisons) pour les algorithmes P et S. Nous démontrons un compromis de la forme

$$\bar{P}(n) + n\log\bar{S}(n) = \Omega(n\log n),$$

dans les cas de recherches avec succès et de recherches avec échec. Cette borne peut être réalisée par une borne supérieure.

En démontrant ce compromis, nous montrons dans le cas moyen une bonne inférieure pour la recherche dans un ordre partiel. Soit A un ordre partiel sur n éléments qui est compatible avec Π extensions linéaires. Nous démontrons que $\bar{S}(n) = \Omega(\Pi^{3/n}/n^2)$ pour des recherches avec succès, et $\bar{S}'(n) = \Omega(\Pi^{2/n}/n)$ pour des recherches avec échec. Ces bornes inférieures montrent dans le cas moyen, par example, que $\Omega(n)$ comparaisons sont nécessaires pour la recherche dans un tas.

* Chercheur invité par le projet VLSI du 1/1/83 au 30/6/85

Average Case Lower Bounds on the Construction and Searching of Partial Orders

Harry G. Mairson
INRIA
Domaine de Voluceau - Rocquencourt
78153 Le Chesnay
FRANCE

Abstract :

It is very well known in computer science that partially ordered files are easier to search. In the worst case, for example, a totally unordered file requires no preprocessing, but $\Omega(n)$ time to search, while a totally ordered file requires $\Omega(n \log n)$ preprocessing time to sort, but can be searched in $O(\log n)$ time. Behind the casual observation, then, lurks the notion of a computational tradeoff between sorting and searching.

We analyze this tradeoff in the average case, using the decision tree model. Let P be a *preprocessing algorithm* that produces partial orders given a set U of n elements, and let S be a *searching algorithm* for these partial orders. Assuming any of the $n!$ permutations of the elements of U are equally likely, and that we search for any $y \in U$ with equal probability (in unsuccessful search, all "gaps" are considered equally likely), the average costs $\overline{P}(n)$ of preprocessing and $\overline{S}(n)$ of searching may be computed. We demonstrate a tradeoff of the form

$$\overline{P}(n) + n \log \overline{S}(n) = \Omega(n \log n),$$

for both successful and unsuccessful search. The bound is tight up to a constant factor.

In proving this tradeoff, we show a lower bound on the average case of searching a partial order. Let A be a partial order on n elements consistent with Π permutations. We show $\overline{S}(n) = \Omega(\Pi^{3/n}/n^2)$ for successful search of A, and $\overline{S}'(n) = \Omega(\Pi^{2/n}/n)$ for unsuccessful search. These lower bounds show, for example, that heaps require linear time to search on the average.

1. Introduction

Sorting and searching problems are obviously ubiquitous in computer science, both for reasons of clear practicality in applications, as well as for the wealth of combinatorial problems suggested in theory. Some recent research has focused on the optimality of sorting and searching in the presence of partial information, where certain sorting information is known in advance.

There have been a series of recent results establishing that the so-called "information theoretic" bounds on sorting, searching, merging, etc. are good in the worst case [Linial and Saks] [Kahn and Saks] [Linial] when *partial order* information is known. These results obviously suggest as a topic for further research what can be said about this genre of problems in the average case. We investigate here some of these questions, although using quite different analytical methods.

Another, related subject of research is that of "implicit" data structures, where information is stored by the order in which the keys themselves are stored in memory, rather than by explicit pointers (see [Munro and Suwanda] [Alt et al.]). Some recent results by Alt and Mehlhorn show $\Omega(\Pi^{1/n})$ lower bounds on searching a *semisorted* table, where the data is known to be stored in one of Π different permutations [Alt and Mehlhorn]. Note, however, that these permutations need not be characterized by a single partial order. This peculiar bound is established in the decision tree model under worst case, average case, and particular nondeterministic measures. Their model requires that y, the key being searched for, appears in all comparisons, and they hence pose the open question of whether the lower bounds hold when arbitrary comparisons are allowed.

We analyze here as well the tradeoff between the *construction* of partial information and the complexity of subsequent searching using this information. The paper [Borodin et al.] showed that if $P(n)$ is the *worst case* complexity of an algorithm P that computes partial orders, and $S(n)$ the worst case of searching one of these partial orders, then in the decision tree model $P(n) + n \log_2 S(n) \geq n \log_2 n + O(n)$. For example, making n/k lists of length k costs $n \log_2 k$ comparisons, and this partial order can be searched in at most $\frac{n}{k} \log_2 k$ comparisons, thus

$$P(n) + n \log_2 S(n) \leq n \log_2 k + n \log_2 \left(\frac{n}{k} \log_2 k \right) \leq n \log_2 n + n \log_2 \log_2 k;$$

the lower bound is then tight in the worst case. We show that a similarly tight tradeoff exists, up to a constant factor, in the average case.

2. The model

Let $U = \{x_1, x_2, ..., x_n\}$ denote a particular set of n distinct elements to be *preprocessed* so that subsequent queries of the form "Is $x \in U$?" can be answered quickly. During the preprocessing stage, we use a *decision tree algorithm* P (see [Knuth] or [Aho et al.]) to produce a class of *partial orders*. Such an algorithm can be represented by a binary tree, where each internal node makes a comparison $x_i : x_j$, branching left if $x_i < x_j$, and right if $x_i > x_j$. At each leaf v of the tree, a particular partial order A_v is implied, consistent with the comparisons made on the internal path to v.

Each leaf v of P has as well an associated *search algorithm* S_v, where comparisons are of the form $x_i : x_j$ or $y : x_i$. Note that the latter type of comparison has three possible outcomes, so that S_v is not strictly binary. The leaves of S_v contain either "$y = x_i$" for some particular x_i, denoting a *successful search*, or "$y \notin U$," denoting *unsuccessful search*.

Associated with the set U is a particular permutation π of $\{1, 2, ..., n\}$, where $x_{\pi(1)} < x_{\pi(2)} < ... < x_{\pi(n)}$. We will speak then of a *permutation consistent with* U. Furthermore, without loss of generality, we may assume that the set U is itself $\{1, 2, ..., n\}$ since a comparison-based model precludes any radix-method sorting or searching. Each π causes a different leaf of P to be reached. We then assume each of the $n!$ permutations π to be equally likely, and associate with each leaf a probability p_v. Using these probabilities, we define the *average cost of preprocessing* $\overline{P}(n)$ as the weighted path length (see [Knuth]) of the tree P.

For each search algorithm S_v, we assume that any π consistent with A_v (also called a *linear extension* of A_v) is equally likely. For successful search we assume $y = x_i$ with equal probability, and for unsuccessful search we assume $x_{\pi(i)} < y < x_{\pi(i+1)}$ with equal probability. The average cost of searching A_v, denoted $\overline{S}_v(n)$ for successful search and $\overline{S}_v'(n)$ for unsuccessful search, can then be defined similarly in terms of average path lengths. The total average cost of searching is then defined as $\overline{S}(n) = \sum_v p_v \overline{S}_v(n)$ for successful search, and $\overline{S}'(n) = \sum_v p_v \overline{S}_v'(n)$ for unsuccessful search.

3. Lower bounds on searching

Before proceeding with the details of the lower bound proofs, we present an informal account of their basic structure and motivation. These intuitions, it is hoped, will motivate the reader in comprehending the varied combinatorial twists and turns to follow.

The fundamental idea that must be exploited in both worst-case and average-case lower bonds on searching a partial order A is the idea of an *independent set* (also sometimes called an *antichain*) of incomparable elements. We call a set $W \subset U$ *independent* if any permutation of the elements of W is consistent with some linear extension of A.

In section 3.1, we briefly present a worst-case lower bound on searching and a tradeoff with preprocessing. While based on the analysis of [Borodin et al.], we present a different proof of their result which does not use Dilworth's Theorem [Dilworth], a fundamental theorem relating minimal chain decompositions and maximum antichains in partial orders. We introduce instead the idea of a *permutation tree*, a construction which can be generalized in the more complicated average-case analysis to follow.

Let $W \subset U$ be a maximal independent set, where $w = |W|$ (also called the *width* of U), so that any permutation of the elements of W is consistent with the partial ordering A of U. The set U can then be decomposed as $U = U_< \cup W \cup U_>$ (see Figure 1), where $U_> = \{x \in U | \exists z \in W, x >_A z\}$ and $U_< = \{x \in U | \exists z \in W, x <_A z\}$. This decomposition is well defined since A can be partitioned into distinct chains, where no more than one element of W appears on each chain.

In the *worst case* it is clear by an adversary argument that w comparisons are necessary to answer a query "Is $y \in U$?" The adversary assumes a permutation π of U consistent with A such that, for any $x \in U_>$, $x' \in W$, and $x'' \in U_<$, the inequality $x > x' > x''$ holds in π. In response to comparisons $u : v$ demanded by the search algorithm, the adversary answers truthfully, *except* for comparisons of the form $y : x_i$. If a particular $x \in W$ has not yet been mentioned in a previous comparison $y : v$, the adversary can pretend $y = x$ and still be consistent in its responses. This adversary strategy can always force the search algorithm to make w comparisons, from which a $\Omega(w)$ lower bound results.

In the *average case*, however, the analysis is much more difficult, because a worst case permutation of U as suggested in Figure 1 cannot be assumed. The worst case lower bound depended on the fact that, after having made t comparisons, the search algorithm could

have eliminated at most t candidates $x_i \in W$ as being equal to y. Such a condition does not hold in the average case, as the example of Figure 2 shows.

To search the partial order of Figure 2, we use an algorithm that first compares y to u, and if necessary linearly searches (i.e., performs comparisons $y : x_i$) to all candidates x_i after the comparison $y : u$, in some arbitrary order. It is entirely possible that $y > u$, in which case one comparison has certified that $y \notin \{x_1, x_2, ..., x_t\}$, and thus half of the set $W = \{x_1, x_2, ..., x_{2t}\}$ has been eliminated. A lower bound proof in the average case, then, must demonstrate that comparisons cannot have such dire effects on a regular basis!

At any particular point in the calculations of the search algorithm, certain x_i have been *cancelled*, i.e., it is known with certainty that $y \neq x_i$. In the average case, it was initially assumed that $y = x_i$ with probability $1/n$ (here we speak of successful search). A lower bound proof must also show that if x_i has not been cancelled yet by the search algorithm, the probability that $y = x_i$ has not markedly decreased, given the comparisons previously made. For example, if after a series of comparisons a certain $x \in U$ has not been cancelled, but $Pr\{y = x\} = e^{-n}$ given comparisons already made, from the perspective of an average case analysis it has effectively been cancelled, since the search algorithm can be very stupid in the case that $y = x$ and still be efficient on the average, reducing the lower bound analyst to bitter tears and frustration.

Every lower bound depends critically on what information can be "given away for free." In the worst case analysis, we give away the fact that $y \in W$ (or $y \pm \epsilon \in W$ for unsuccessful search), and then prove that W is large if the preprocessing done to create the partial order A was small. In the restricted average case analysis of section 3.2, we assume in addition that the elements of W are all maximum elements in A, or all minima, give away for free all $y : x$ comparisons where $x \in U - W$, and show that enough elements of W still remain uncancelled to provide a strong lower bound. Finally, in section 3.7 we demonstrate that every partial order A can be "cut" into two parts A^+ and A^- by comparing all $x \in U$ to some value α depending on A, so that on the average, A^+ has a large independent set of minimal elements, or A^- has a large set of maximal elements.

For each of the stages mentioned above, we will construct a table T_i, where $T_i(\pi, x)$ records which elements of W remain uncancelled after stage i, for each permutation π consistent with the partial order A and possibilities $y = x$ where $x \in W$ (see Figure 3). Note we derive a lower bound by only considering the cases $y \in W$, and for the moment we concern ourselves only with successful search. For notational convenience we write $W \subset U$ as $\{x_1, ..., x_w\}$.

3.1 Worst case analysis: a brief review

Let A be a partial order of the elements of U consistent with Π permutations, i.e., there are Π linear extensions of A to a total order. We define a *permutation tree* of A to be a tree of height n which lexicographically orders the permutations consistent with A. Figure 4 depicts a partial order and its associated permutation tree.

The root node has as children the maximal elements of A. Each such child x_i is in turn the root of a subtree of height $n-1$, representing the linear extensions of A where $x_i = n$. Every path in the tree is of length n since a path corresponds to a unique permutation consistent with A. Furthermore, for every node v in the permutation tree, the children of v are independent in A, that is, every permutation of them is consistent with some linear extension of A.

Lemma 3.1.1. *Every partial order A on n elements and consistent with Π permutations has an independent set of size at least $\Pi^{1/n}$.*

Proof. The permutation tree for A has depth n and Π leaves, thus there exists one or more internal nodes with at least $\Pi^{1/n}$ children; otherwise there is not enough branching in the tree to reach Π leaves in depth n. $\diamond$

Lemma 3.1.2. *The worst case of searching A requires $S(n) \geq \Pi^{1/n}$.*

Proof. By the previous lemma we know there exists a set of independent elements of U of size $\Pi^{1/n}$; extend this set to a maximal set W. Decompose U as $U = U_> \cup W \cup U_<$, and use the adversary argument sketched earlier to show $|W|$ comparisons are necessary. $\diamond$

Let $P(n)$ denote the height of the preprocessing tree described in section 2, and $S(n)$ the largest height of any search algorithm S_v. These two quantities correspond to the worst cases of preprocessing (sorting) and searching. We can then prove the following worst-case tradeoff.

Theorem 3.1.3. $P(n) + n \log_2 S(n) \geq \log_2 n!$.

Proof. Since the height of the preprocessing tree P is $P(n)$, then there exists one leaf (partial order) A of P consistent with $\lceil n!/2^{P(n)} \rceil$ permutations. By the previous lemma, we have

$$P(n) + n \log_2 S(n) \geq P(n) + n \log_2 \left(\frac{n!}{2^{P(n)}} \right)^{1/n} \geq \log_2 n! = \Theta(n \log n)$$

$\diamond$

3.2 Average case analysis of searching a restricted class of partial orders

With the intuitions of the worst case well understood, we now begin consideration of the more difficult average case. For the moment, however, we only prove a lower bound on searching based on the number of minimal elements of A. Note that since a minimum of n elements can be produced by the preprocessing algorithm in $O(n)$ time, thus giving a whopping $\Omega(1)$ lower bound for searching, this approach is *by itself* quite uninteresting! However, in section 3.7 we will show that if A is consistent with Π permutations, then A can be "cut" or "factored" into two parts A^+ and A^-, where the average number of minimal elements of A^+ is $\Omega(\Pi^{1/n})$. So the reader should rest assured that the techniques presented in this section will be put to good use.

On the other hand, a variety of classical preprocessing (i.e., sorting) methods maintain a maximal independent set as extrema, including merge sort, heap sort, bubble sort, and (subject to a little hand-waving argument) quicksort; the lower bound technique is not in this light so absurd.

We give away in the searching algorithm two stages of free information. In stage 1, we sort the elements in $U_> = U - W$, producing a partial order of the form suggested in Figure 5. It should be clear that regardless of the comparisons C involved in sorting $U_>$, which we call the *upper chain*, $\Pr\{y = x | x \in W \text{ and } C\} = 1/n$. The result of stage 1 is a set of partial orders of the form suggested in Figure 5. Because these partial orders all have the same general structure, we refer to them generically as A, one hopes without confusion.

In stage 2, we compare y, the subject of the query "Is $y \in U$?", to every element of the upper chain, using linear search, binary search, or the reader's favorite method (take your time– it's free). The decision tree equivalent of these stages is a decision tree of the comparisons comprising the said free information, with copies of the same search algorithm S spliced in at the leaves. We then bound the average number of comparisons done by S alone.

Let $T_2(\pi, x)$ denote the elements of W remaining after stage 2 when $y = x (x \in W)$, and where π is a permutation of U consistent with the partial order A.

Lemma 3.2.1. *For any permutation π consistent with A,*

$$\sum_{x \in W} | T_2(\pi, x)| \geq \frac{w^2}{2}.$$

Proof. When $x \in W_i$ (see Figure 6), $x < u_i$ in all permutations π consistent with A, $0 \leq i < k$. Therefore the largest set of elements of W that can be cancelled by x is $W_0 \cup W_1 \cup ... \cup W_{i-1}$, this occuring when x is as large as possible. We then derive

$$\begin{aligned}\sum_{x \in W} |T_2(\pi, x)| &\geq w_0(w) + w_1(w - w_0) + w_2(w - w_0 - w_1) + \cdots \\ &\quad + w_k(w - w_0 - \cdots - w_{k-1}) \\ &\geq w^2 - \sum_{0 \leq i < j \leq k} w_i w_j \\ &\geq \frac{w^2}{2} + \frac{1}{2} \sum_{0 \leq i \leq k} w_i^2 \geq \frac{w^2}{2}.\end{aligned}$$

$\diamond$

Corollary 3.2.2. *Let π be a permutation of U consistent with A and $x \in W$, both chosen uniformly at random. Then the expected number of uncancelled elements of W after stage 2 is $E[|T_2(\pi, x)|] \geq w/2$.*

The corollary has an intuitive explanation in terms of the diagram of A in Figure 6, where we observe that for x, $x' \in W$, x can only cancel x' if x' is "to the left of" x, and then not always. Since the average x is in the middle, it only has $w/2$ elements of W to its left, so the $w/2$ elements to its right are left uncancelled.

Let C_y denote the comparisons made with y during the last stage; since for all $x \in W$ it was known before that $\Pr\{y = x\} = 1/n$, we would like to ensure that the conditional probability $\Pr\{y = x | C_y\}$ for uncancelled x is not much less, so that x is preserved as a "viable candidate" for y in the ensuing search. We now show that this conditional probability is at least $\frac{1}{n+1}$.

3.3 Some lower bounds on conditional probabilities

Let A denote a partial order on n elements, and B denote a partial order on $n+1$ elements derived by adding an isolated element h to A. We use $\Pr_A$ and $\Pr_B$ to express probabilities in these respective partial orders, and denote by C_y a set of inequalities of the form $u_i < y < v_i$, where u_i and v_i are both in the upper chain, or both in the lower chain, as well as inequalities of the form $u_j < v_j < y$ and $y < u_j < v_j$. Assume $x \in W$ does not appear in any inequality in C_y.

Lemma 3.3.1. $\Pr_A\{y = x|C_y\} \geq \Pr_B\{y = x|C_y\}$.

Proof. We write $\Pr_A\{y = x|C_y\}$ as N_A/D_A, where

$$N_A = |\{\pi \mid \pi \textit{ consistent with } C_x \textit{ in } A\}|,$$

$$D_A = |\{(\pi, x) \mid \pi \textit{ consistent with } C_x \textit{ in } A\}|,$$

and write $\Pr_B\{y = x|C_y\}$ as N_B/D_B similarly. Clearly $N_B = (n+1)N_A$, since for every permutation of $\{1, 2, ..., n\}$ consistent with C_x, h can be inserted into the permutation in one of $n+1$ positions. We note that

$$D_A = \sum_{x \in A} |\{(\pi, x)|\ C_x\}|,$$

and

$$\begin{aligned} D_B &= (n+1)D_A + |\{(\pi, h)|\ \pi \textit{ a permutation of } \{1, 2, ..., n+1\} \textit{ and } C_h\}| \\ &\geq (n+1)D_A. \end{aligned}$$

Since $(n+1)N_A = N_B$ and $D_B \geq (n+1)D_A$ we have $N_A D_B \geq N_B D_A$, and the result follows. $\Diamond$

Lemma 3.3.2. $\Pr_B\{y = x|\ C_y\} \geq \Pr_B\{y = h|\ C_y\}$.

Proof. Let π be a permutation of $\{1, 2, ..., n+1\}$ consistent with B and C_x. The position of x in the permutation is constrained by the partial order information of B, and *further* constrained by C_x. However, h can appear in any of the $n+1$ positions of the permutation, since h is an isolated element in B, and h is not mentioned in the inequalities C_x.

Now consider a permutation π' consistent with B and C_h. We can *always* exchange the positions of x and h in π', producing a permutation π satisfying both B and C_x. Since this exchange describes an injection from $\{\pi'|C_h\}$ into $\{\pi|C_x\}$, the lemma is proven. $\Diamond$

Lemma 3.3.3. $\Pr_B\{y = h|\ C_y\} \geq \frac{1}{n+1}$.

Proof. Let π be a permutation consistent with A and possibly C_y; we consider the number of ways h can be inserted into π preserving the same conditions in B. Clearly, if π is consistent with A, then h can be inserted in $n+1$ different places preserving the conditions of B. Furthermore, there are $t_\pi \geq 1$ ways to insert h preserving C_h. Then

$$\Pr_B\{y = h|\ C_y\} = \frac{\sum_\pi t_\pi}{\sum_\pi (n+1)} \geq \frac{1}{n+1}.$$

$\Diamond$

Combining the previous three lemmas, we derive the following with respect to partial order A:

Theorem 3.3.4. $\Pr\{y = x |\ C_y\} \geq \frac{1}{n+1}$.

Theorem 3.3.4 will be important in the analysis that follows, because it will ensure that if $x \in W$ has not been cancelled the disclosure of free information involving y, then x remains with reasonably high probability a candidate for y.

What does the above theorem mean *combinatorially*? Arrange the values of $T_2(\pi, x)$ as a $\Pi \times n$ table; the entry $T_2(\pi, x)$ notes the subset $Q_j \subset W$ remaining uncancelled after y has been compared to every element in the upper chain, where $Q_j = W_j \cup W_{j+1} \cup \cdots \cup W_k$, $0 \leq j \leq k$ (see Figure 6.). If there are $N(Q_j)$ occurances of Q_j in table T_2, Theorem 3.3.4 says that at least $N(Q_j)/(n+1)$ occurances of Q_j are in column x, for every $x \in Q_j$.

3.4 The graph model

After the above disclosure of free information, we are left with a refined partial order on the set $U = L \cup M \cup Q$, where L are elements known to be less than y, M are elements known to be greater than y, and $Q \subset W$ are mutually incomparable elements, any one of which could be equal to y. Now we begin to charge the search algorithm S for comparisons made.

We use a graph $G(V, E)$ as a device to model the state of knowledge of the search algorithm, where $V = \overline{y} \cup Q$ and $E = \emptyset$ initially at the end of stage 2. As comparisons are made during the continuing search, we modify the graph as follows: Define a function $f : U \rightarrow V$ where $f(x) = x$ if $x \in Q$, otherwise $f(x) = \overline{y}$. When comparison $u : v$ is made in the search algorithm, we add the edge $(f(u), f(v))$ to E. We then observe the following two facts:

Lemma 3.4.1. *If a leaf is reached in the search algorithm labelled "$y = x_i$," then $\overline{y}$ and x_i must be connected by an edge in G.*

Proof. If not, then y and x_i were never compared, in which case the algorithm would not be correct when $y = x_i + \epsilon$: such a value for y would satisfy all comparisons made, i.e., the algorithm would behave exactly as if $y = x_i$, but would give the wrong answer. $\Diamond$

Lemma 3.4.2. *If $y \notin U$, then when a leaf of the search algorithm is reached, G must be a single connected component.*

Proof. Let x be an element of a component not connected to $\overline{y}$. By a similar argument, there exists a permutation π where all the same comparisons would be made if $y = x$. $\Diamond$

3.5 Width-related lower bounds on searching: unsuccessful case

By Lemma 3.4.2, if Q denotes set of the elements remaining after stage 2, and $y \notin U$, then at least $|Q|$ subsequent comparisons must be made, since when the search algorithm terminates, G must have at least $|Q|$ edges. Note that the result of Corollary 3.2.2 still holds when we analyze unsuccessful search, i.e., we replace "$y = x$" by "$y = x \pm \epsilon$" for some small $\epsilon > 0$. We then have directly the following theorem.

Theorem 3.5.1. *Let A be a partial order on n distinct elements with width w. If any linear extension π consistent with A is equally likely, and for any x_i we assume $y \in [x_i - \epsilon_i, x_i) \cup (x_i, x_i + \epsilon_{i+1})$ with equal probability, where $x_{\pi(1)} < x_{\pi(2)} < \ldots < x_{\pi(n)}$, and $\epsilon_i = (x_{\pi(i)} - x_{\pi(i-1)})/2$, then the expected number of comparisons made by the search algorithm is bounded below by $\frac{w^2}{2n}$.*

Proof. By the above corollary, when $x_i - \epsilon_i < y < x_i + \epsilon_i$ for some $x_i \in W$, then the average value of $|Q|$ is $w/2$, and this event happens with probability w/n. $\diamond$

The reader may think that a $\Omega(w^2/n)$ lower bound is weak, but the following example informally suggests that a lower bound is going to have the form $\Omega(w^{k+1}/n^k)$ for some integer $k \geq 1$. Consider the problem of searching the partial order depicted in Figure 7: suppose we first compare y and u, subsequently binary searching $U_>$ if $y > u$, and linear searching W if $y < u$. Since the former happens with probability $\frac{n-w-1}{n}$ and the latter with probability w/n, a quick-and-dirty calculation shows that the average number of comparisons for successful search is no more than

$$1 + \frac{n-w-1}{n} \log_2(n-w-1) + \frac{w}{n}\left(\frac{w}{2}\right) = O\left(\log_2 n + \frac{w^2}{n}\right).$$

A $\Omega(\log_2 n)$ bound on searching seems pretty obvious, and if one believes that this algorithm is good, the example shows that $\Omega(\log_2 n + w^2/n)$ is the strongest lower bound that can be proven.

3.6 Width-related lower bounds on searching: successful case

In the case of successful search, we have to work harder than in the unsuccessful case, and derive a slightly weaker result. At the end of stage 2, only the elements of some

$Q \subset W$ remain as possible candidates for y: every other element of U is known to be either less than y or greater than y. Furthermore, any permutation of Q is consistent with the comparisons made in earlier stages, though not necessarily equiprobable.

Two additional tools are needed to complete the analysis. First, we transform the search algorithm S into another search algorithm S^* which preserves the following property: if a set $Q \subset W$ are all incomparable with y after a certain series of comparisons, then the elements of Q are also mutually incomparable. (A long winded way of saying no order information is known about Q.) This property will ensure that we can continue to use Theorem 3.3.4, which says that the conditional property $\Pr\{y = x | x \in Q\} \geq \frac{1}{n+1}$. The average cost $\overline{S}^*$ of S^* will be bounded as $\overline{S}^*(n) \leq 3\overline{S}(n)$, thus providing a lower bound on the average cost $\overline{S}(n)$ of algorithm S up to a constant factor. Second, we introduce an *accounting game* to model successful search using S^*.

The transformation of S into S^* is easy and follows a simple rule: if $u : v$ is a comparison in S where $y \notin \{u, v\}$ and either u or v are not connected in G (and are therefore incomparable), compare $y : u$ and $y : v$ first, copying subtrees of S as necessary. Figure 8 gives an example of this transformation.

Lemma 3.6.1. *In the decision tree S^*, if $y = x$ is consistent with comparisons made thus far (i.e., there exists a linear extension of the current partial order with $y = x$), then the conditional probability $\Pr\{y = x\} \geq \frac{1}{n+1}$.*

Proof. Note that the comparisons made satisfy the conditions of Theorem 3.3.4. $\diamond$

We now describe an *accounting game* that measures the average cost of S^*. Assume that after stage 2, $Q \subset W$ elements of U remain uncancelled. Define a table $T_3(\pi, x)$ where $x \in W$, and π is a permutation consistent with the initial partial order A. Let $Q \subset W$ be fixed: if $T_2(\pi, x) = Q$ then we call position (π, x) of T_3 *active*, otherwise *cancelled*. Initially all active positions of T_3 are set to zero.

An *accounting game for Q* is made up of one or more subgames, each subgame corresponding to particular disjoint subtrees of S^*. The game is played in successive *levels* $\ell = 1, 2, 3, \ldots$ where at level ℓ (corresponding to a comparison at depth ℓ in S^*), we add 1 to each active position, and then for each subgame, choose one of the following two types of moves:

1. *(Cancel)* Cancel all active elements of column x of the subgame (corresponding to a comparison $y : x$ in the tree).

2. *(Split)* Divide the permutations in the subgame into two groups, and split the subgame into two for the next level (corresponding to a comparison $u : v$ in the tree, where the result of $y : u$ and $y : v$ is already known).

The game is played by running the search algorithm S^* until all active positions have been cancelled. The *final score* of the game, equalling the sum of all cancelled positions $T_3(\pi, x)$ such that $T_2(\pi, x) = Q$, is precisely the sum of the number of comparisons made by S^* for every (π, x) consistent with the result Q of stage 2.

Lemma 3.6.2. *The final score of an accounting game for Q is at least*

$$\frac{N(Q)|Q|^2}{2(n+1)},$$

where $N(Q) = |\{(\pi, x) | T_2(\pi, x) = Q\}|$.

Proof. In any subgame, at most j columns of active positions have been cancelled after level j. By Theorem 3.3.4, if there were t active positions in the subgame, at least $t/(n+1)$ occur in any column containing *some* active positions. To get a lower bound on the final score, we assume active positions are removed as early as possible (i.e., when $T_3(\pi, x)$ is still small). Then if there were t active positions in a subgame, the contribution of the subgame to the final score is at least $\frac{t}{n+1}(1 + 2 + \cdots + |Q|)$. Summing over all subgames gives the bound. $\diamond$

Finally, we can derive a lower bound on successful search:

Theorem 3.6.3.

$$\overline{S}(n) \geq \frac{w^3}{24n(n+1)}.$$

Proof. We know that for algorithm S^*,

$$\overline{S}^*(n) \geq \frac{1}{n\Pi} \sum_{Q \subset W} \sum_{\substack{\pi, x \\ T_2(\pi, x) = Q}} T_3(\pi, x),$$

where Π is the number of permutations consistent with the initial partial order. By the previous lemma, then, we have

$$\begin{aligned} \overline{S}^*(n) &\geq \frac{1}{n\Pi} \sum_{Q \subset W} \frac{N(Q)|Q|^2}{2(n+1)} \\ &\geq \frac{w}{2n(n+1)} \sum_{Q \subset W} \frac{N(Q)}{w\Pi} |Q|^2. \end{aligned}$$

The last sum can be represented as the expected value $E[X^2]$, where X is a random variable defined as the value of $|T_2(\pi, x)|$ over all π and $x \in W$, these chosen uniformly at random. Since $E[X^2] \geq E[X]^2$, and $E[X] \geq w/2$ by Corollary 3.2.2, we derive

$$\overline{S}^*(n) \geq \frac{w}{2n(n+1)}\left(\frac{w}{2}\right)^2$$
$$\geq \frac{w^3}{8n(n+1)}.$$

As $\overline{S}^*(n) \leq 3\overline{S}(n)$, we finally derive that the average number of comparisons in the successful search of a partial order A on n elements, where A has w mutually incomparable elements, is bounded below as

$$\overline{S}(n) \geq \frac{w^3}{24n(n+1)}.$$

◇

Corollary 3.6.4. *In the average case, $\Omega(n)$ comparisons are required to search a heap.*

Proof. Note that in a heap on n elements, there are $\Omega(n)$ leaves, and these constitute the width of the partial order. We remark that this result is very obvious in the worst case, but not at all clear in the average case. ◇

3.7 Average case lower bounds on searching

To conclude the analysis of searching in the average case, we wish to establish lower bounds not based on the number of minimal elements of a partial order A, but rather on the number of its linear extensions. We establish such bounds by "cutting" A into two parts A^+ and A^-, showing that if A has many linear extensions, then A^+ must have, on the average, many minimal elements.

More precisely, we define a *cut* $\langle A, \alpha \rangle$ as a partitioning $A = A^+ \cup A^-$ where the elements in A^+ are $\{x \in U | x > \alpha\}$ and the elements of A^- are $\{x \in U | x < \alpha\}$. We imagine α to be some value not equal to any $x \in U$. The partial order information in A^+ and A^- is just the order information known for the two respective sets.

We now reconsider the *permutation tree* described in section 3.1. In terms of this tree, a cut $\langle A, \alpha \rangle$ now corresponds to a horizontal slice through the tree, in between two levels of internal nodes (see Figure 9). Each of the nodes u_i of the tree has a *weight*, given by the proportion of permutations represented in the tree by a path through u_i. The *average size of a cut* $\langle A, \alpha \rangle$ is then the weighted average number of children of the u_i.

Explained in terms of the decision-tree model, we can imagine inserting a new element α in the partial order and beginning the search algorithm with comparisons $x_1 : \alpha, x_2 :$

$\alpha, ..., x_n : \alpha$. The average size of the cut then expresses, averaging over all permutations consistent with A, the number of minimal elements in the partial order formed by the elements of A^+. Note that by a symmetrical argument based on permutation trees of the *reversals* of permutations, we can compute the average number of maximal elements in A^-.

Define $\overline{C}(\Pi, n)$ as the number of children of an "average" internal node in a permutation tree T, where the nodes are weighted by the proportion of permutations represented in the tree by a path through the node, Π is the number of permutations represented in the tree, and n is the number of elements in the permutations. We express this value more formally as

$$\overline{C}(\Pi, n) = \frac{1}{\Pi} \sum_{\pi} \frac{1}{n} \sum_{v \in Path_T(\pi)} d(v),$$

where $Path_T(\pi)$ is defined as the nodes on the path in the permutation tree T corresponding to a permutation π, and $d(v)$ is the degree (i.e., number of children) of a node v in the tree.

Theorem 3.7.1. $\overline{C}(\Pi, n) \geq \Pi^{1/n}$.

Proof. Following a similar derivation in [Alt and Mehlhorn] we proceed by induction on n. For $n = 1$ we have $\Pi = 1$ and the theorem is trivially true.

For $n > 1$ we imagine the root r of the permutation tree has k children (see Figure 10). We say $\pi \in T_i$ when π is one of the Π_i permutations represented by a path through subtree T_i, and rewrite $\overline{C}(\Pi, n)$ as

$$\begin{aligned}
\overline{C}(\Pi, n) &= \frac{1}{n\Pi} \sum_{1 \leq i \leq k} \sum_{\pi \in T_i} \left(d(r) + \sum_{v \in Path_{T_i}(\pi)} d(v) \right) \\
&= \frac{1}{n\Pi} \sum_{1 \leq i \leq k} \Pi_i k + \Pi_i (n-1) \overline{C}(\Pi_i, n-1) \quad , \\
&\geq \frac{1}{n\Pi} \left(\Pi k + (n-1) \sum_{1 \leq i \leq k} \Pi_i^{1 + \frac{1}{n-1}} \right)
\end{aligned}$$

the latter inequality resulting from application of the inductive hypothesis. The sum $\sum_{1 \leq i \leq k} \Pi_i^{1+\frac{1}{n-1}}$, subject to the constraint $\Pi = \sum_{1 \leq i \leq k} \Pi_i$, takes its minimum value when all $\Pi_i = \Pi/k$, thus

$$\begin{aligned}
\overline{C}(\Pi, n) &\geq \frac{1}{n\Pi} \left(\Pi k + (n-1) k \left(\frac{\Pi}{k} \right)^{1 + \frac{1}{n+1}} \right) \\
&\geq \frac{1}{n} \left(k + (n-1) \left(\frac{\Pi}{k} \right)^{\frac{1}{n-1}} \right).
\end{aligned}$$

Rewriting the last right-hand expression as $\frac{1}{n}f(k)$, we note $f'(k) = 1 - \Pi^{\frac{1}{n-1}}k^{-\frac{n}{n-1}}$, which takes value 0 at $k = \Pi^{1/n}$, and $f''(k) > 0$ for positive k, indicating a minimum. Then

$$\overline{C}(\Pi, n) \geq \frac{1}{n}\left(\Pi^{1/n} + (n-1)\left(\Pi^{1-\frac{1}{n}}\right)^{\frac{1}{n-1}}\right) \geq \Pi^{1/n}.$$

$\diamond$

Each internal node v in the permutation tree makes a certain contribution to the average $\overline{C}(\Pi, n)$. The nodes appear at levels $0, 1, ..., n-1$, where level 0 is the root. By the pigeonhole principle, the nodes L_ℓ at some level $0 \leq \ell \leq n-1$ of the tree have $\overline{C}(\Pi, n)$ children on the average, that is,

$$\frac{1}{\Pi}\sum_{\pi}\sum_{v \in Path_T(\pi) \cap L_\ell} d(v) \geq \Pi^{1/n}.$$

We summarize the above in the following theorem:

Theorem 3.7.2. *Let A be a partial order on n elements consistent with Π permutations. Then there exists an α depending on A such that the cut $\langle A, \alpha\rangle$ factors A into partial orders A^+ and A^-, where the average number of minimal elements in A^+ is at least $\Pi^{1/n}$.*

Proof. Take $\alpha = \ell + 1/2$ in the above analysis (where we assume $U = \{1, 2, ..., n\}$), or equivalently, $\alpha = (x + x')/2$, where x and x' are the elements (i.e., values) of rank ℓ and $\ell + 1$ in U. $\diamond$

We finally are able to characterize the number of comparisons required on the average by a search algorithm S_A for partial order A, which we denote as $\overline{S}_A(n)$ in the successful case and $\overline{S}'_A(n)$ in the unsuccessful case:

Theorem 3.7.3.

$$\overline{S}_A(n) \geq \frac{\Pi^{3/n}}{24n(n+1)}.$$

Proof. Modify search algorithm S_A by giving away comparisons $x_1 : \alpha$, $x_2 : \alpha$, ..., $x_n : \alpha$ for free, where $\langle A, \alpha\rangle$ is a cut satisfying the result of Theorem 3.7.2 (see Figure 11).

By Theorem 3.6.3 we have

$$\overline{S}_A(n) \geq \sum_w \Pr\{X = w\}\frac{w^3}{24n(n+1)} = \frac{1}{24n(n+1)}E[X^3],$$

where X is a random variable defined as the number of minimal elements of A^+ given by the cut $\langle A, \alpha\rangle$. Since $X \geq 0$ always we know $E[(X + 2\mu)(X - \mu)^2] \geq 0$ where $\mu = E[X]$, or equivalently $E[X^3] \geq E[X]^3 = \Pi^{3/n}$. $\diamond$

Theorem 3.7.4.

$$\overline{S}'_A(n) = \frac{\Pi^{2/n}}{2n}.$$

Proof. Modify search algorithm S_A as in the previous theorem and use the result of unsuccessful search in Theorem 3.5.1. $\diamond$

4. Lower bounds on sorting–searching tradeoffs

4.1 A bound on $\overline{P}(n)$

Associated with the jth leaf ($1 \le j \le L$) of the preprocessing algorithm is a particular partial order A_j of width w_j, as well as a search algorithm S_j. This leaf is reached by $p_j n!$ of the possible input permutations π of U, so that p_j represents the probability of reaching the jth leaf when any π is equally likely. The average cost $\overline{P}(n)$ of preprocessing is then the path length of the decision tree weighted by the p_j. The problem is therefore similar to that of generating a set of prefix codes for a source of L symbols in a noiseless channel, which is treated by Shannon's first theorem of information theory [Shannon], and bounded by the entropy of the source.

Theorem 4.1.1. $\overline{P}(n) \ge \sum_{1 \le j \le L} p_j \log_2 \frac{1}{p_j}$.

Proof. If ℓ_j is the length of the path (equivalently, the number of comparisons made on the path) to the jth leaf, we notice that since P is a complete binary tree, $\sum_{1 \le j \le L} 2^{-\ell_j} = 1$; we then define $q_j = 2^{-\ell_j}$ as the "pseudoprobability" of reaching leaf j. As $\ln x \le x - 1$ for all real x, we derive the well known inequality

$$\sum_{1 \le j \le L} p_j \ln\left(\frac{q_j}{p_j}\right) \le \sum_{1 \le j \le L} p_j \left(\frac{q_j}{p_j} - 1\right) = 0,$$

or

$$\sum_{1 \le j \le L} p_j \log_2 \frac{1}{p_j} \le \sum_{1 \le j \le L} p_j \log_2 \frac{1}{q_j} = \sum_{1 \le j \le L} p_j \ell_j = \overline{P}(n).$$

$\diamond$

4.2 A bound on $\overline{S}(n)$

The jth leaf of the preprocessing tree P is associated with a partial order A_j consistent with $\Pi_j = p_j n!$ permutations. Since the associated search algorithm S_j is used with probability p_j, we derive an overall bound on searching by weighting the algorithms S_j with weights p_j. For successful search, we have

$$\begin{aligned}\overline{S}(n) &= \sum_{1\le j\le L} p_j \overline{S}_j(n) \\ &\ge \sum_{1\le j\le L} \frac{p_j}{24n(n+1)} (p_j n!)^{3/n}.\end{aligned}$$

Since $2n \ge n+1$ and $n! \ge (n/c)^n$ for some constant c when $n \ge 1$, we have

$$\overline{S}(n) \ge \frac{n}{48c^3} \sum_{1\le j\le L} p_j^{1+3/n}.$$

Similarly, for unsuccessful search, we have

$$\overline{S}'(n) \ge \frac{n}{2c^2} \sum_{1\le j\le L} p_j^{1+2/n}.$$

4.3 A general tradeoff theorem

After a rather detailed combinatorial journey, we are almost prepared to derive the average case tradeoff between constructing and searching partial orders. The proof of this tradeoff depends on the minimization of some formulas $f(p_1, \ldots, p_L)$ subject to the constraint $\sum p_i = 1$, in which case the following lemma turns out to be useful.

Lemma 4.3.1. *If a is a positive constant and $\sum_{1\le j\le L} p_j = 1$, then*

$$a \sum_{1\le j\le L} p_j \ln \frac{1}{p_j} + n \ln \sum_{1\le j\le L} p_j^{1+a/n} \ge 0.$$

Proof. We rewrite $p_j^{1+a/n}$ as

$$p_j^{1+a/n} = p_j \exp\left(\frac{a \ln p_j}{n}\right) = p_j \left(1 + \frac{a \ln p_j}{n} + \frac{1}{2!}\left(\frac{a \ln p_j}{n}\right)^2 + \cdots\right),$$

and define

$$R = \sum_{1 \le j \le L} p_j^{1+a/n} = 1 + \frac{1}{1!} \sum_{1 \le j \le L} p_j \left(\frac{a \ln p_j}{n} \right) + \frac{1}{2!} \sum_{1 \le j \le L} p_j \left(\frac{a \ln p_j}{n} \right)^2 + \cdots.$$

Now rewrite $a \sum_{1 \le j \le L} p_j \ln \frac{1}{p_j}$ as $n \ln Q$, so that

$$\begin{aligned} Q &= \exp\left(-\frac{a}{n} \sum_{1 \le j \le L} p_j \ln p_j \right) \\ &= 1 - \sum_{1 \le j \le L} p_j \left(\frac{a \ln p_j}{n} \right) + \frac{1}{2!} \left(\sum_{1 \le j \le L} p_j \left(\frac{a \ln p_j}{n} \right) \right)^2 - \cdots. \end{aligned}$$

We now claim $QR \ge 1$, from which $n \ln Q + n \ln R$ follows immediately. Let X be a random variable that takes value $\frac{a \ln p_j}{n}$ with probability p_j. Q and R can then be written in the convenient form

$$\begin{aligned} Q &= e^{-E[X]} = 1 - E[X] + \frac{1}{2!} E[X]^2 - \frac{1}{3!} E[X]^3 + \cdots, \\ R &= E[e^X] = 1 + E[X] + \frac{1}{2!} E[X^2] + \frac{1}{3!} E[X^3] + \cdots, \end{aligned}$$

so that

$$QR = 1 + \frac{1}{2!} E[(X - E[X])^2] + \frac{1}{3!} E[(X - E[X])^3] + \cdots.$$

If we let $Y = X - E[X]$, then, we derive

$$\begin{aligned} QR &= E\left[1 + \frac{Y^2}{2!} + \frac{Y^3}{3!} + \cdots \right] \\ &= E\left[e^Y - Y \right]. \end{aligned}$$

Since for any real number x, $e^x - x \ge 1$, we have

$$\begin{aligned} QR &= \sum_x Pr\{Y = x\}(e^x - x) \\ &\ge \sum_x Pr\{Y = x\} \\ &\ge 1, \end{aligned}$$

so that $QR \ge 1$, and $n \ln QR \ge 0$. The minimum value 0 is achieved when one of the p_j are set to 1, and the rest to zero.

We also note that $QR \geq 1$ could be proven by using Jensen's inequality (see, for example, [McEliece]), which states that if f is a real valued function and convex cup, then $E[f(X)] \geq f(E[X])$ for any real random variable X, in which case $R \geq e^{E[X]}$. $\diamond$

The remainder is easy. Using the lower bound on $\overline{P}(n)$ derived in section 4.1, the lower bound on $\overline{S}(n)$ derived in section 4.2, and the above lemma with $a = 3$, we derive for successful search,

Theorem 4.3.2. $3\overline{P}(n) + n \log_2 \overline{S}(n) \geq n \log_2 n + O(n)$.

Proof.

$$\begin{aligned} 3\overline{P}(n) + n \log_2 \overline{S}(n) &\geq 3 \sum_{1 \leq j \leq L} p_j \log_2 \frac{1}{p_j} + n \log_2 \left(\frac{n}{48c^3} \sum_{1 \leq j \leq L} p_j^{1+3/n} \right) \\ &\geq n \log_2 n + O(n) + \frac{1}{\ln 2} \left[3 \sum_{1 \leq j \leq L} p_j \ln \frac{1}{p_j} + n \ln \sum_{1 \leq j \leq L} p_j^{1+3/n} \right] \\ &\geq n \log_2 n + O(n). \end{aligned}$$

$\diamond$

For unsuccessful search we use the lower bound on $\overline{S}'(n)$ and Lemma 4.3.1 with $a = 2$, deriving

Theorem 4.3.3. $2\overline{P}(n) + n \log_2 \overline{S}'(n) \geq n \log_2 n + O(n)$.

Proof.

$$\begin{aligned} 2\overline{P}(n) + n \log_2 \overline{S}'(n) &\geq 2 \sum_{1 \leq j \leq L} p_j \log_2 \frac{1}{p_j} + n \log_2 \left(\frac{n}{2c^2} \sum_{1 \leq j \leq L} p_j^{1+2/n} \right) \\ &\geq n \log_2 n + O(n) + \frac{1}{\ln 2} \left[2 \sum_{1 \leq j \leq L} p_j \ln \frac{1}{p_j} + n \ln \sum_{1 \leq j \leq L} p_j^{1+2/n} \right] \\ &\geq n \log_2 n + O(n). \end{aligned}$$

$\diamond$

5. Remarks

The results of this paper could be extended in several directions. The lower bounds can be made stronger; we conjecture that searching requires an average of $\Omega(\Pi^{1/n})$ comparisons in successful and unsuccessful case, which will improve the constant in the tradeoff

theorem. Better upper bounds, not of the "existential" variety (i.e., algorithms, not proofs of existence of decision trees) remain to be found. Finally, the problems should be considered in more generalized models of computation, such as decision trees where linear or quadratic tests are allowed, if not altogether different models.

Acknowledgements. I would like to thank many friends and colleagues at INRIA and elsewhere who patiently listened to many versions of the above proofs, including Gaston Gonnet, Alice Wideman, and especially Thomas Strothotte. I also want to acknowledge the assistance of Andy Yao, to whom I offer my thanks for helping me take my first difficult and often frustrating steps in doing research, and who suggested I work on this problem.

References

[Aho et al.] Aho, Alfred V., John Hopcroft, and Jeffrey D. Ullman. **The Design and Analysis of Computer Algorithms.** Reading: Addison Wesley, 1974.

[Alt et al.] Alt, Helmut, Kurt Mehlhorn, and J. Ian Munro. *Partial Match Retrieval in Implicit Data Structures.* **Proceedings MFCS**, 1981, Springer Lecture Notes in Computer Science, vol. 118, pp. 156-161.

[Alt and Mehlhorn] Alt, Helmut, and Kurt Mehlhorn. *Searching Semisorted Tables.* Computer Science Department, Penn State University, Technical Report CS-82-28, December 1982.

[Borodin et al.] Borodin, Allan, Leo J. Guibas, Nancy A. Lynch, and Andrew C. Yao. *Efficient Searching Using Partial Ordering.* **Inf. Proc. Lett.**, volume 12, number 2 (April 1981), pp. 71-75.

[Dilworth] Dilworth, R.P. *A decomposition theorem for partially ordered sets.* **Ann. Math** 51 (1950), pp. 161-166.

[Kahn and Saks] Kahn, Jeff, and Michael E. Saks. *Every Poset Has a Good Comparison.* **16th STOC**, 1984, pp. 299-301.

[Knuth] Knuth, Donald E. **The Art of Computer Programming**, vol. 3. Reading: Addison Wesley, 1973.

[Linial] Linial, Nathan. *The Information Theoretic Bound is Good for Merging.* **SIAM J. Comp.**, to appear.

[Linial and Saks] Linial, Nathan, and Michael E. Saks. *Information Bounds are Good for Search Problems on Ordered Data Structures.* **24th FOCS**, 1983, pp. 473-475.

[McEliece] McEliece, Robert J. **The Theory of Information and Coding**. Reading: Addison Wesley, 1977.

[Munro and Suwanda] Munro, J. Ian, and Hendra Suwanda. *Implicit Data Structures.* **11th STOC**, 1979, pp. 108-117.

[Shannon] Shannon, Claude E. **Introduction to the Theory of Communication.** Urbana: U. Illinois Press, 1949.

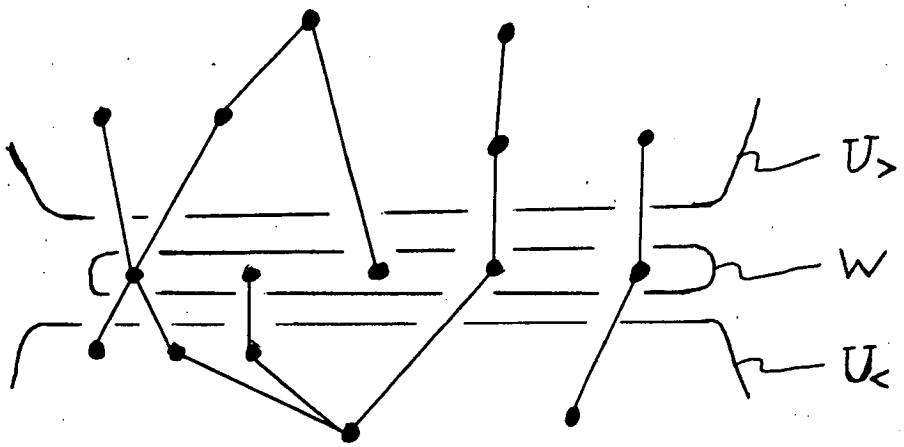


Figure 1

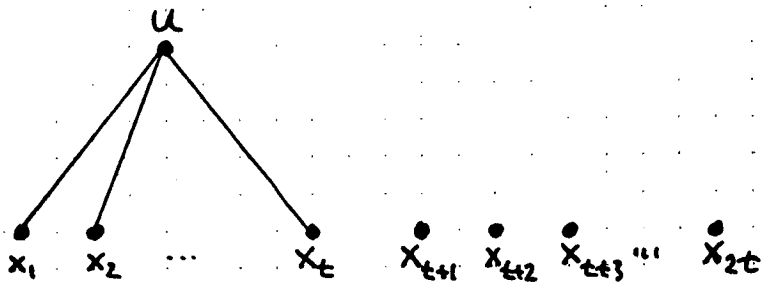


Figure 2

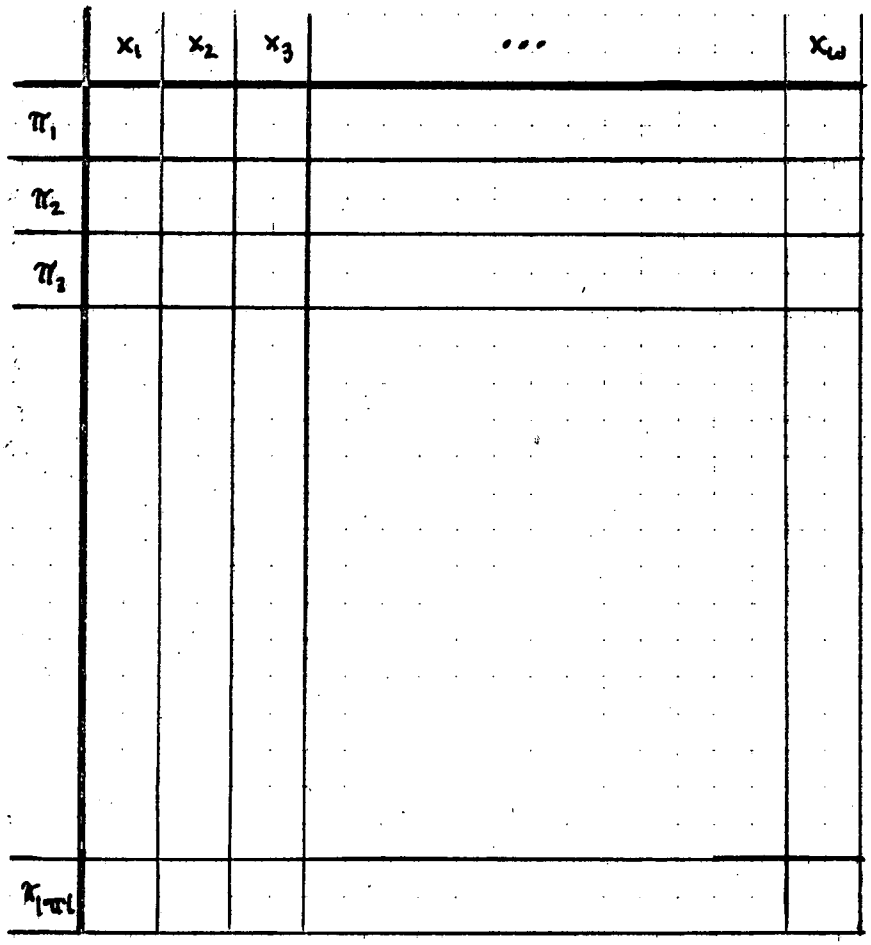


Figure 3

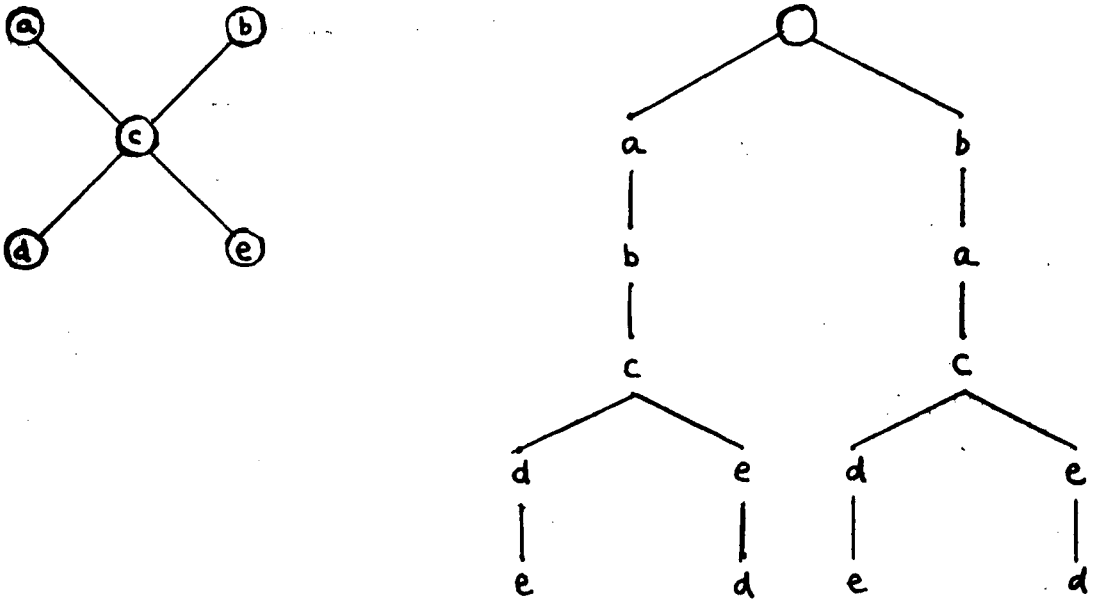
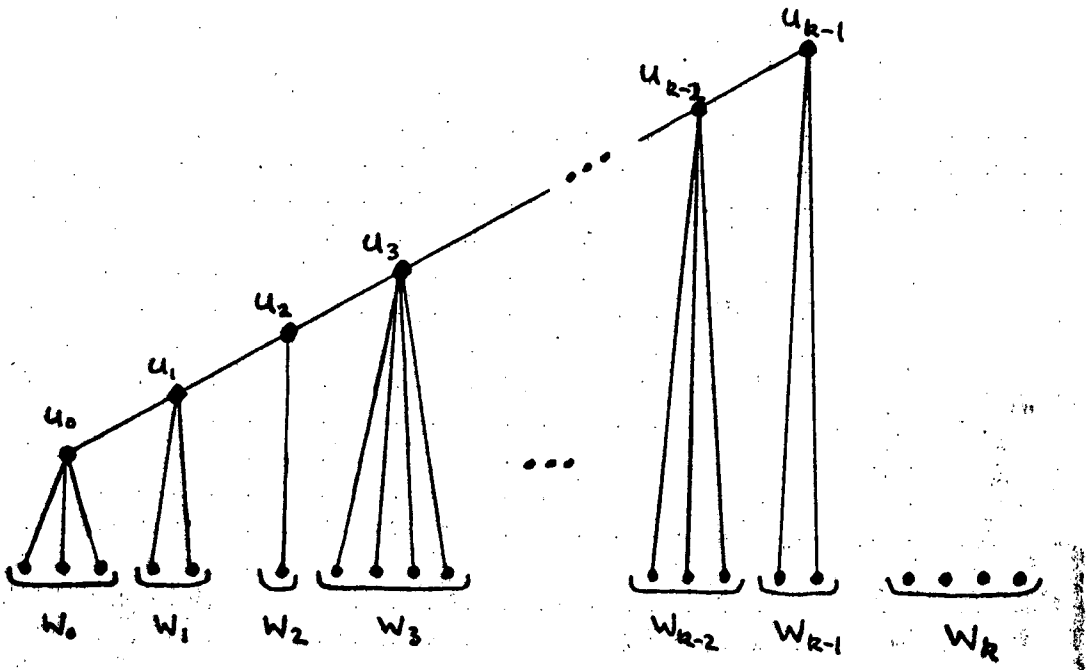


Figure 4



$$W = \bigcup_i W_i$$

$$|U| = n = |W| + k$$

Figure 5.

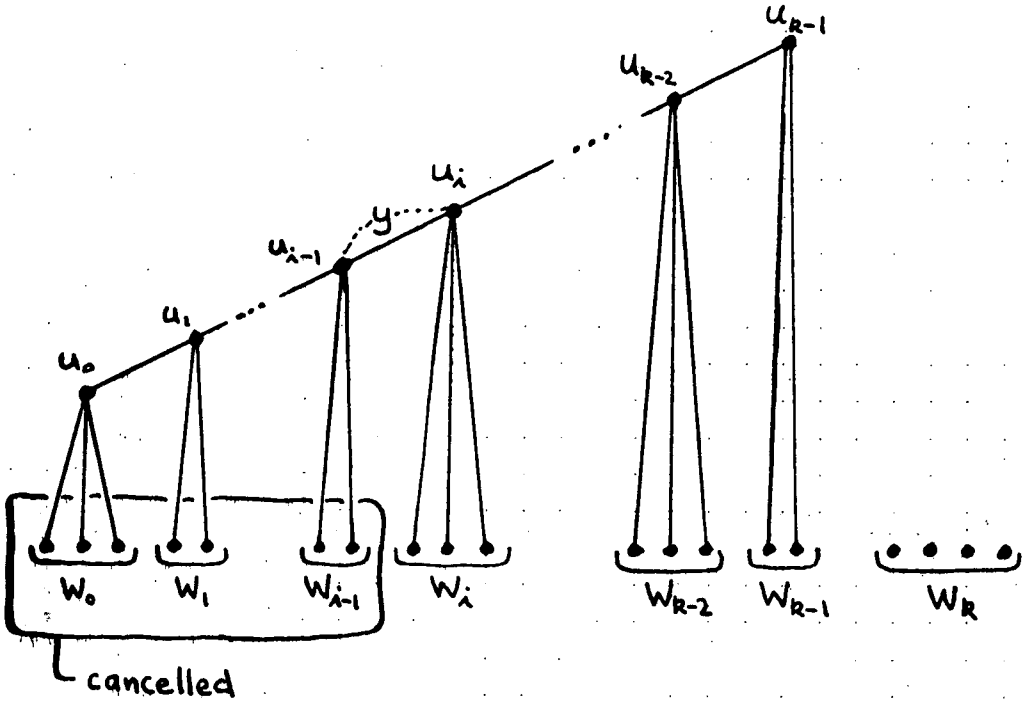


Figure 6

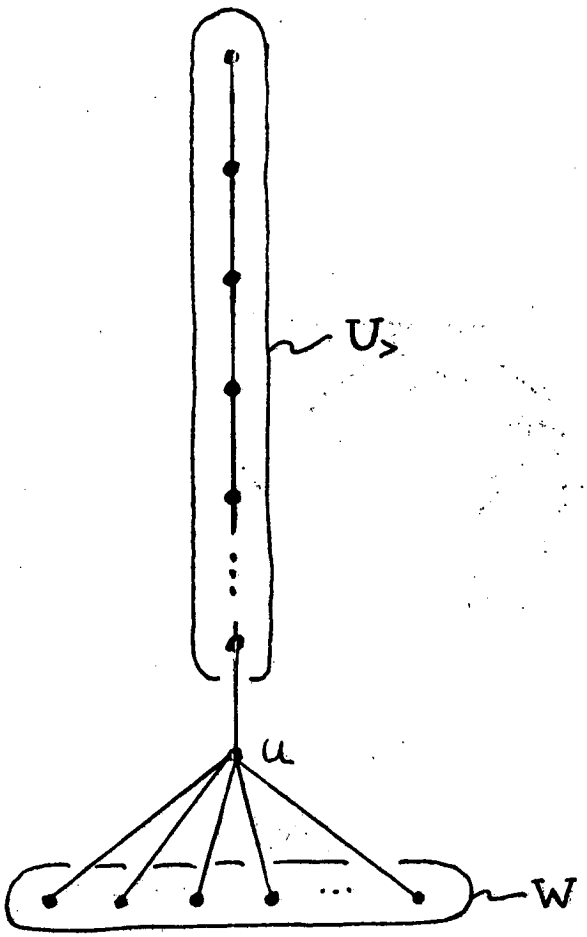


Figure 7

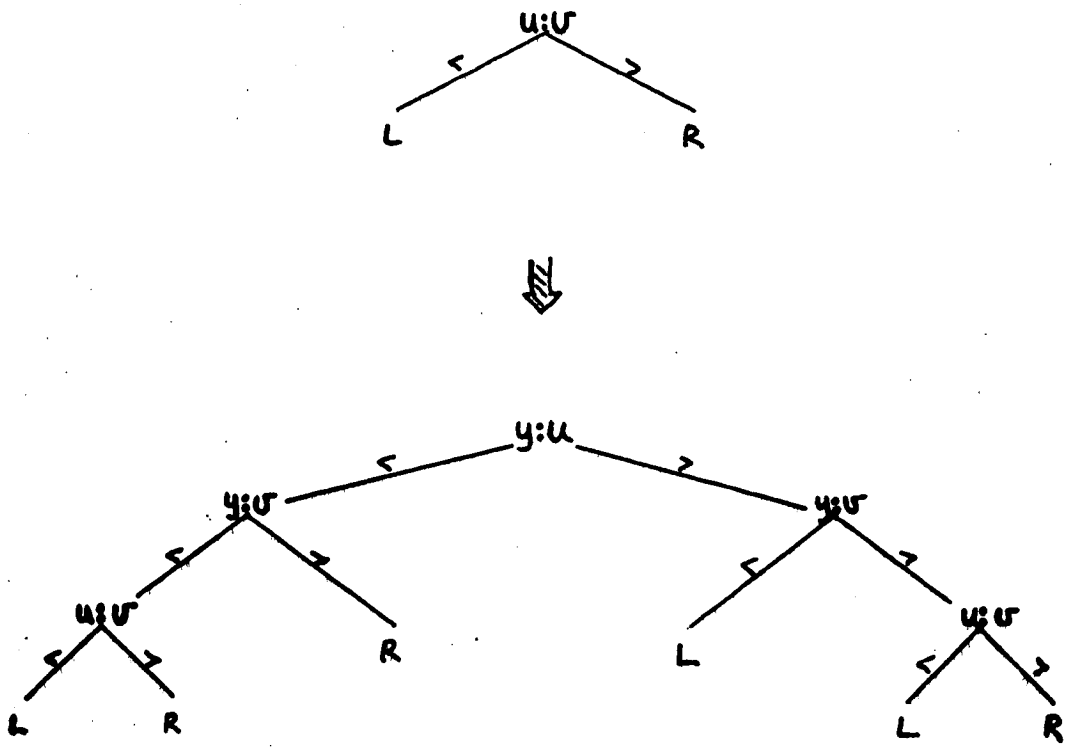


Figure 8

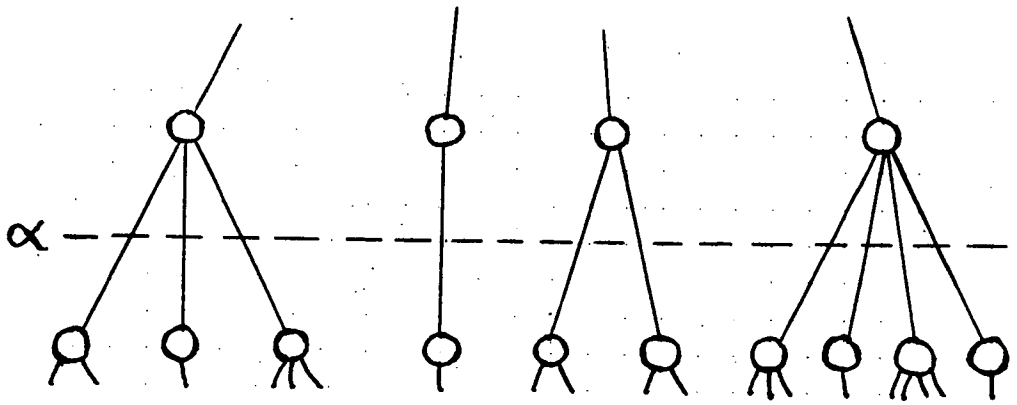


Figure 9

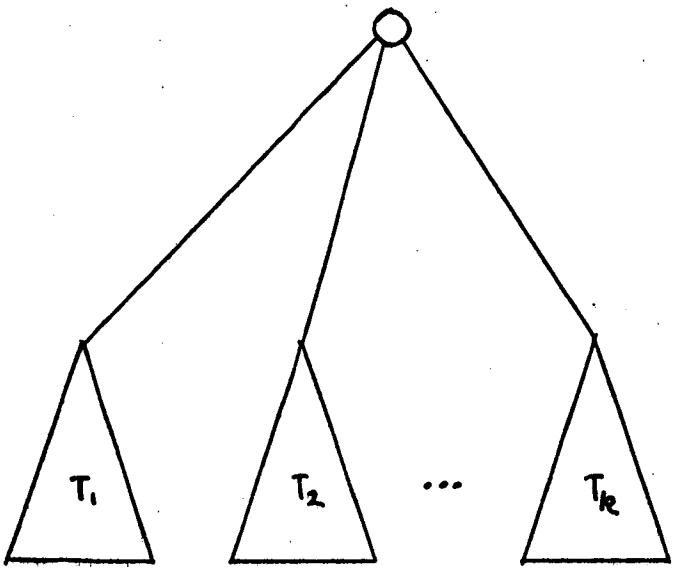


Figure 10

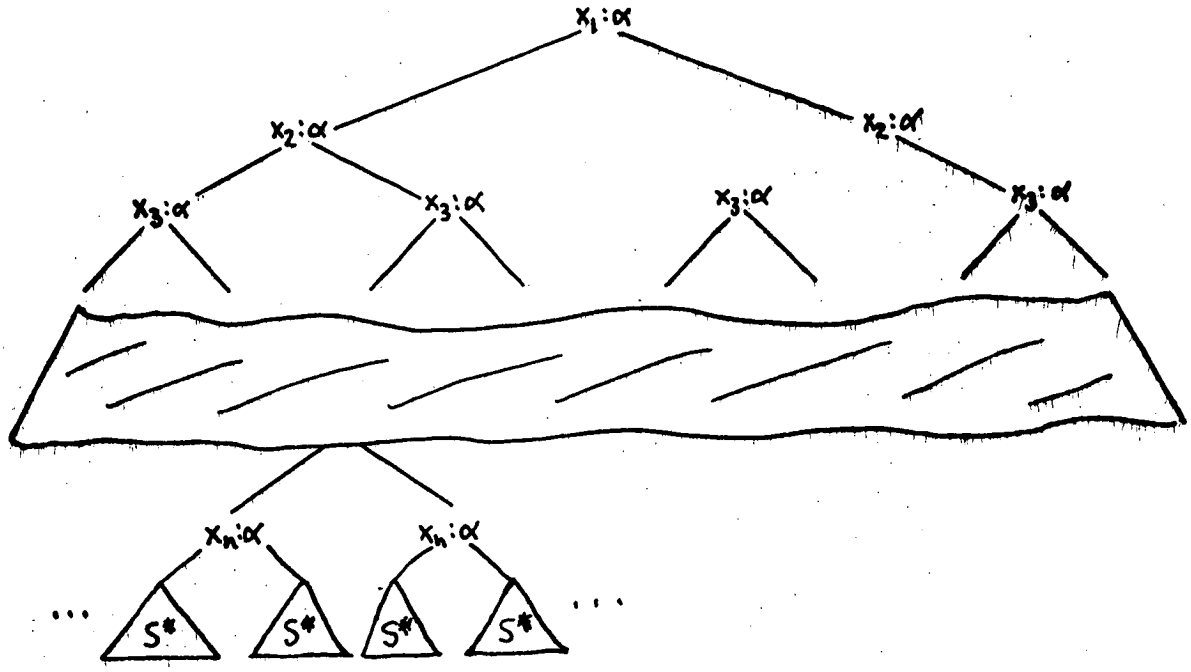


Figure 11

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique