Programming Techniques

R.L. Rivest Editor

Hierarchical Binary Search

Arthur Gill
University of California at Berkeley

In hierarchical search the data structure holding the file keys is partitioned into substructures of the same type; these are searched consecutively until the queried key is found or the substructures are exhausted. The interest here is in the conditions under which the performance of a hierarchical organization of static files is superior to that of the nonhierarchical organization and in the construction of the hierarchy when these conditions are met. The performance criterion is the average number of comparisons in a successful search, where averaging extends over all keys and over all permutations of the keys' access probabilities. General properties of hierarchical search are first derived, and attention is then focused on the hierarchical binary organization—the special case where each of the data substructures is a sorted array (or a balanced binary tree) and where the keys are accessed by binary search. It is shown that an advantageous two-stage hierarchy is always implementable when the keys' access density function $\phi(i)$ is "steeper" than Zipf's density function $\zeta(i)$—the steeper it is, the greater the advantage. A simple method for constructing the two-stage hierarchy is formulated, based on finding the intersection of $\phi(i)$ and $\zeta(i)$. For the r-stage hierarchical organization, partitioning procedures are proposed which are based on the iterative application of the two-stage techniques.

Key Words and Phrases: data structures, file organization, hierarchical file organization, searching, binary search

CR Categories: 3.74, 4.34

This research was supported by the National Science Foundation under Grant MCS 76-15036.

Author's address: Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720.

1. Introduction

We are given a set of n keys $X = \{x_1, x_2, \ldots, x_n\}$, organized in a static data structure $\mathscr{D}$ (such as a linked list, a sorted array, a binary tree, etc.). We are also given a search algorithm $\mathscr{A}$ (suitable for accessing any key in $\mathscr{D}$) for which

$s(n)$ = average number of comparisons in a successful search of $\mathscr{D}$,

$u(n)$ = average number of comparisons in an unsuccessful search of $\mathscr{D}$.

We assume that s and u are dependent on the number of keys only and not on the keys' access probabilities. This means that in computing s and u, the averaging extends not only over all n keys, but also over all permutations of the n excess probabilities. We also assume that for all n,

$$u(n) \geq s(n). \tag{1.1}$$

The organization of X in the single structure $\mathscr{D}$ will be referred to as a *simple organization*; the search for a key in $\mathscr{D}$ (using $\mathscr{A}$) will be referred to as a *simple search.*

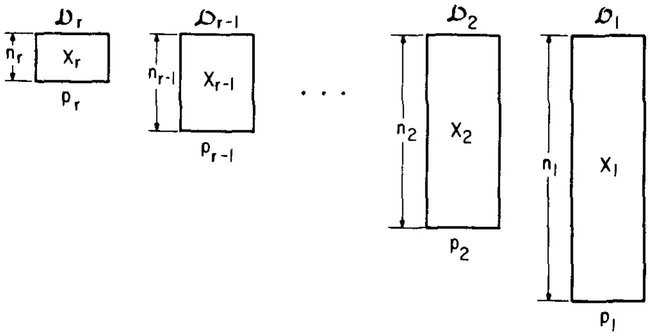
Consider now the reorganization of X in the following manner (see Figure 1): X is partitioned into r nonempty subsets $X_1, X_2, \ldots, X_r$, where for $j = 1, 2, \ldots, r$, X_j contains n_j keys ($\sum_{j=1}^{r} n_j = n$), organized in a data structure $\mathscr{D}_j$ which is identical to $\mathscr{D}$ except for size (e.g., if $\mathscr{D}$ is a sorted array, so is $\mathscr{D}_j$). A search for a key x in this organization is carried out as follows: Using algorithm $\mathscr{A}$, search $\mathscr{D}_r$; if x is found, quit, else search D_{r-1}; if x is found, quit, else search $\mathscr{D}_{r-2}$; $\ldots$; if x is found, quit, else search $\mathscr{D}_1$. The organization of X in this fashion will be referred to as a *hierarchical organization of order r* and the corresponding search scheme as a *hierarchical search of order r.*

The average number of comparisons in a successful hierarchical search of order r is denoted by s_r. Our objective is to construct the partition $\{X_1, X_2, \ldots, X_r\}$ which minimizes s_r (this partition will be referred to as the *optimal* one). We also want to compare the minimal s_r with $s(n)$ and thus determine to what extent and under what conditions the hierarchical search has an advantage over the simple search, insofar as the average successful search time is concerned. This advantage is measured by the *advantage index* $A = s(n)/s_r$.

In the next section we derive some general properties of the hierarchical organization described above. In the remaining sections we focus our attention on *hierarchical binary search*— hierarchical search where $\mathscr{D}$ is a sorted array (or a balanced binary tree) and where $\mathscr{A}$ is the binary search algorithm. We start with the special case where the partition of X is a dichotomy (i.e., $r = 2$).[1] Subsequently we extend our results to hierarchical binary search of any order.

[1] The hierarchical binary search of order 2 was suggested by Reingold, Nievergelt, and Deo (see [3, pp. 274–275]).

Fig. 1. Hierarchical organization of order r.



Throughout the discussion we assume that the set of keys X is static, i.e., not subjected to any addition or deletion operations. We also assume that the access probabilities of the keys are known in advance. The latter assumption, however, is not as rigorous as it sounds; as we shall see, what we require is a knowledge of the "rate" at which this probability drops as we proceed from the more active to the less active keys rather than the precise knowledge of each key's access probability.

2. Some Properties of Hierarchical Organization

Let us consider the hierarchical organization of order r shown in Figure 1, where $X = X_1 \cup X_2 \cup \cdots \cup X_r = \{x_1, x_2, \ldots, x_n\}$. Let us define the density function

$$\phi(i) = \text{Prob[accessing } x_i] \quad (i = 1, 2, \ldots, n).$$

(We assume that $\phi(i) > 0$ for all i (otherwise x_i can be discarded).) The access probability of the subset X_j is

$$p_j = \sum_{\substack{i \text{ such that} \\ x_i \in X_j}} \phi(i).$$

(Since the X_j are nonempty, $p_j > 0$ for all j.)

By s_k $(1 \le k \le r)$ we denote the average number of comparisons in a successful hierarchical search of order k conducted on $\mathcal{D}_k, \mathcal{D}_{k-1}, \ldots, \mathcal{D}_1$ (in that order). Thus

$$s_1 = s(n_1), \tag{2.1}$$
$$s_k = \frac{p_k}{q_k} s(n_k) + \frac{q_{k-1}}{q_k} [u(n_k) + s_{k-1}] \quad (2 \le k \le r),$$

where

$$q_\nu = p_1 + p_2 + \cdots + p_\nu \quad (1 \le \nu \le r).$$

From (2.1) s_r can be computed recursively.

PROPOSITION 1. *For all k and j such that $1 \le j < k \le r$ we can write*

$$s_k = \alpha + \beta s_j,$$

where α and β are positive and depend only on $n_{j+1}, n_{j+2}, \ldots, n_k, p_{j+1}, p_{j+2}, \ldots, p_k$, and $p_1 + p_2 + \cdots + p_j$.

PROOF. We show by induction on d that for all k and d such that $2 \le k \le r$ and $1 \le d \le k - 1$,

$$s_k = \alpha + \beta s_{k-d}, \tag{2.2}$$

where α and β are positive and depend only on $n_{k-d+1}, n_{k-d+2}, \ldots, n_k, p_{k-d+1}, p_{k-d+2}, \ldots, p_k$, and $p_1 + p_2 + \cdots + p_{k-d}$.

Basis. By (2.1), for all k such that $2 \le k \le r$ we can write $s_k = \alpha' + \beta' s_{k-1}$, where α' and β' are positive and depend only on n_k, p_k, and $p_1 + p_2 + \cdots + p_{k-1}$. Hence (2.2) holds for $d = 1$.

Induction step. Hypothesize that (2.2) is true for d. Using (2.1) we can write

$$s_k = \alpha + \beta s_{k-d} = \alpha + \beta(\alpha' + \beta' s_{k-d-1})$$
$$= (\alpha + \beta\alpha') + (\beta\beta')s_{k-d-1} = \alpha'' + \beta'' s_{k-d-1},$$

where α and β (by hypothesis) are positive and depend only on $n_{k-d+1}, n_{k-d+2}, \ldots, n_k, p_{k-d+1}, p_{k-d+2}, \ldots, p_k$, and $p_1 + p_2 + \cdots + p_{k-d}$, and where α' and β' (by (2.1)) are positive and depend only on n_{k-d}, p_{k-d}, and $p_1 + p_2 + \cdots + p_{k-d-1}$. Hence α'' and β'' are positive and depend only on $n_{k-d}, n_{k-d+1}, \ldots, n_k, p_{k-d}, p_{k-d+1}, \ldots, p_k$, and $p_1 + p_2 + \cdots + p_{k-d-1}$. Thus (2.2) holds for $d + 1$. □

PROPOSITION 2. *If s_r is minimal, so is s_k for $k = 2, 3, \ldots, r$.*

PROOF. Suppose s_r is minimal but that some s_k $(2 \le k < r)$ is not minimal. Hence it is possible to reorganize $X_1 \cup X_2 \cup \cdots \cup X_k$ so as to yield an average number of comparisons $s'_k < s_k$ (which replaces s_k) and s'_r (which replaces s_r). By Proposition 1 we can write

$$s_r = \alpha + \beta s_k,$$

where α and β are positive and depend only on $n_{k+1}, n_{k+2}, \ldots, n_r, p_{k+1}, p_{k+2}, \ldots, p_r$, and $p_1 + p_2 + \cdots + p_k$. Since the reorganization of $X_1 \cup X_2 \cup \cdots \cup X_k$ leaves all these quantities unchanged, we can write

$$s'_r = \alpha + \beta s'_k.$$

Hence

$$s'_r - s_r = \beta(s'_k - s_k).$$

Since $\beta > 0$ and $s'_k < s_k$, we have $s'_r < s_r$, which contradicts the assumption that s_r is minimal. Hence s_k must also be minimal. □

PROPOSITION 3. *Let $x_{\bar{l}}$ and $x_{\bar{h}}$ be keys in X_l and X_h, respectively, where $l > h$ and $\phi(\bar{l}) < \phi(\bar{h})$. Then for some j $(1 \le j \le r - 1)$ there exist keys $x_{\bar{j}}$ and $x_{\overline{j+1}}$ in X_j and X_{j+1}, respectively, such that $\phi(\overline{j+1}) < \phi(\bar{j})$.*

PROOF. Let j be the largest integer such that $h \le j < l$ and X_j contains a key $x_{\bar{j}}$ which satisfies $\phi(\bar{l}) < \phi(\bar{j})$. If $j = l - 1$, the proof is complete. Otherwise, pick up any key $x_{\overline{j+1}}$ in X_{j+1}. From the choice of j and $x_{\bar{j}}$ it follows that $\phi(\bar{l}) \ge \phi(\overline{j+1})$. Hence $\phi(\overline{j+1}) < \phi(\bar{j})$. □

PROPOSITION 4. *Suppose s_r is minimal. Let $x_{\bar{l}}$ and $x_{\bar{h}}$ be any keys in X_l and X_h, respectively, where $l > h$. Then $\phi(\bar{l}) \ge \phi(\bar{h})$.*

PROOF. Suppose $\phi(\bar{l}) < \phi(\bar{h})$. By Proposition 3 there exists k $(2 \le k \le r)$ such that there are keys $x_{\bar{k}}$ and $x_{\overline{k-1}}$ in X_k and X_{k-1}, respectively, where $\phi(\bar{k}) < \phi(\overline{k-1})$. From (2.1) we have (defining $q_0 = s_0 = 0$)

$$s_k = \frac{1}{q_k}\{p_k s(n_k) + p_{k-1}[u(n_k) + s(n_{k-1})] + q_{k-2}[u(n_k) + u(n_{k-1}) + s_{k-2}]\}.$$

Let us now reorganize X by interchanging $x_{\bar{k}}$ and $x_{\overline{k-1}}$. As a result, p_k is replaced by $p'_k = p_k + f$ and p_{k-1} by $p'_{k-1} = p_{k-1} - f$, where $f = \phi(\overline{k-1}) - \phi(\bar{k}) > 0$; and q_k, q_{k-2}, and s_{k-2} remain unchanged. Hence s_k changes to

$$s'_k = \frac{1}{q_k}\{p'_k s(n_k) + p'_{k-1}[u(n_k) + s(n_{k-1})] + q_{k-2}[u(n_k) + u(n_{k-1}) + s_{k-2}]\}.$$

Thus

$$\begin{aligned} s_k - s'_k &= \frac{1}{q_k}\{(p_k - p'_k)s(n_k) + (p_{k-1} - p'_{k-1})[u(n_k) + s(n_{k-1})]\} \\ &= \frac{1}{q_k}\{f[u(n_k) - s(n_k)] + fs(n_{k-1})\}. \end{aligned}$$

By (1.1), $u(n) \geq s(n)$ for all n, and hence $u(n_k) - s(n_k) \geq 0$. Since $f > 0$, we have $s_k > s'_k$, which implies that s_k is not minimal. But this, by Proposition 2, implies that s_r is not minimal—a contradiction. Hence we must have $\phi(\bar{l}) \geq \phi(\bar{h})$. □

Proposition 4 is intuitively plausible: It is always advantageous to place the more active keys in those subsets which are searched earlier. What is not obvious, however, is the dependence of this result on condition (1.1), i.e., on the assumption that $u(n) \geq s(n)$.

Incidentally, under the assumption that for each subset X_j the $|X_j|$ keys are equally likely and so are the $|X_j| + 1$ "unsuccessful" intervals (see [2, p. 410]), it is always true that $u(n) \geq s(n)$: It is known that for every search algorithm describable by a binary search tree, we have, under the equal probability assumption,

$$s(n) = \left(1 + \frac{1}{n}\right) u(n) - 1$$

(see [2, p. 427]); thus $u(n) < s(n)$ if and only if $s(n) > n$, which is an impossible condition in a binary search tree.

3. Hierarchical Binary Search of Order 2

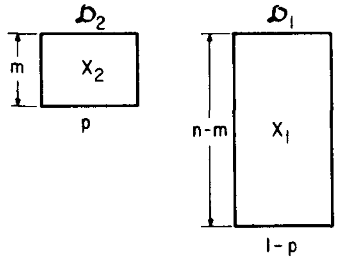
We now turn our attention to the *hierarchical binary search*—the special case where the data structure $\mathscr{D}$ (as well as $\mathscr{D}_1, \mathscr{D}_2, \ldots, \mathscr{D}_r$) is an array sorted by key (or a balanced binary search tree), and where algorithm $\mathscr{A}$ is the binary search algorithm. In this case, under the assumption that all keys are equally likely we have for large n,

$$\begin{aligned} s(n) &\approx \log n - 1 \qquad \text{(see footnote 2)} \\ u(n) &\approx \log n \end{aligned} \tag{3.1}$$

(see [2, p. 411]). When the keys are not equally likely,

[2] log k stands for $\log_2 k$ throughout.

Fig. 2. Hierarchical organization of order 2.



(3.1) is valid if in computing s and n the averaging is extended over all permutations of the n access probabilities, as well as over the n keys. The effect is the same as that of assuming that there is no correlation between the key ordering and the frequency ordering.

Note that $s(n)$ and $u(n)$ of (3.1) satisfy condition (1.1). Thus, in conformance with Proposition 4 an optimal hierarchical binary organization involves two sorting operations: First, the n keys are sorted in ascending order by frequency; the first n_1 keys are then assigned to X_1, the next n_2 keys are assigned to X_2, etc. (assuming that $n_1, n_2, \ldots, n_r$ have been determined); finally each one of the subsets $X_1, X_2, \ldots, X_r$ is sorted by key. (This double sorting, which is done only once for the static file, requires $O(n \log n)$ comparisons.) Since our objective is to construct an optimal organization, we shall henceforth assume that for all i and j, $i \geq j$ implies $\phi(i) \leq \phi(j)$ (i.e., that $\phi(i)$ is a monotonically nonincreasing density function).

Using (3.1) in (2.1) we get

$$\begin{aligned} s_1 &\approx \log n_1 - 1 \\ s_k &\approx \log n_k - \frac{p_k}{q_k} + \frac{q_{k-1}}{q_k} s_{k-1} \end{aligned} \tag{3.2}$$

In this section we focus on the hierarchical binary search of order 2. Denoting $m = n_2$ and $p = p_2$ (see Figure 2), we have from (3.2),

$$s_2 \approx \log m + (1 - p)\log(n - m) - 1. \tag{3.3}$$

In this case hierarchical search is superior to simple search if

$$\begin{aligned} A(n, m) &= \frac{s(n)}{s_2} \\ &\approx \frac{\log n - 1}{\log m + (1 - p)\log(n - m) - 1} > 1. \end{aligned} \tag{3.4}$$

From (3.4) we get the condition on p for which $A > 1$,[3]

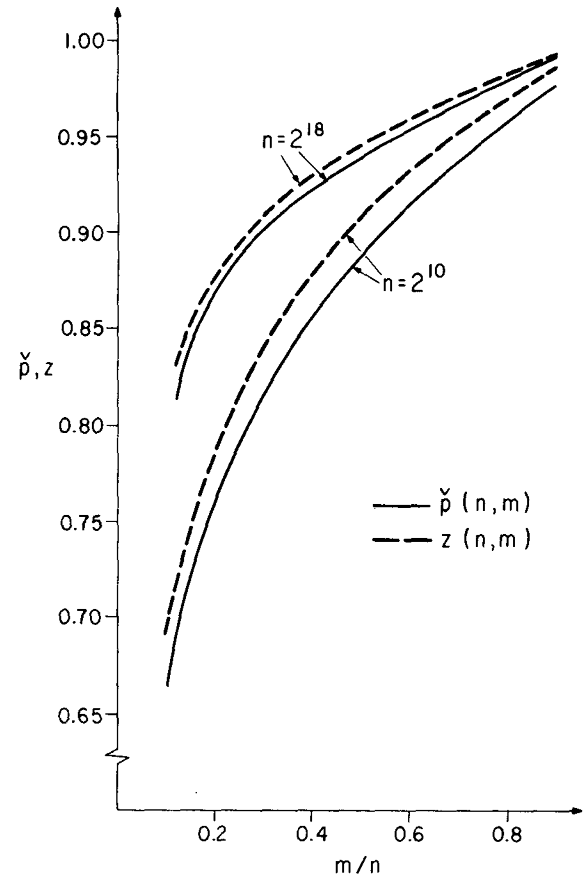
$$p > 1 - \frac{\log(n/m)}{\log n + \log(1 - (m/n))}.$$

The corresponding lower bound on p is denoted by

$$\check{p}(n, m) = 1 - \frac{\log(n/m)}{\log n + \log(1 - (m/n))}. \tag{3.5}$$

[3] This result is essentially the same as in the Solutions Manual to [3].

Fig. 3. $\check{p}(n, m)$ and $z(n, m)$.



In what follows a key role is played by the density function known as *Zipf's density function*,[4] defined by

$$\zeta(i) = \frac{c}{i} \quad (i = 1, 2, \ldots, n), \tag{3.6}$$

where

$$c = \frac{1}{H_n}, \quad H_n = \sum_{\nu=1}^{n} \frac{1}{\nu}.$$

For large n, H_n can be approximated (see [1, p. 74]) by

$$H_n \approx \log_e n + 0.577 = (\log_e 2) \log n + 0.577. \tag{3.7}$$

The corresponding *Zipf's distribution* is

$$z(n, m) = \sum_{i=1}^{m} \zeta(i) = \frac{H_m}{H_n},$$

or for large n,

$$z(n, m) \approx \frac{(\log_e 2)\log m + 0.577}{(\log_e 2)\log n + 0.577},$$

which can be written as

$$z(n, m) \approx 1 - \frac{\log(n/m)}{\log n + 0.833}. \tag{3.8}$$

[4] This density was observed by Zipf [4] to approximate the relative frequency of words in natural-language texts.

Comparing (3.5) and (3.8), we see that $z(n, m)$ and $\check{p}(n, m)$ approximate each other. In fact, computation shows that for large n the two functions differ by less than 5 percent (see Figure 3). Thus Zipf's distribution can be taken as the limiting distribution for judging the profitability of a hierarchical binary search of order 2. If the given distribution is "higher" than Zipf's, then one can implement hierarchical binary search of order 2 which is, on the average, faster than a simple binary search.

It is clear from (3.4) (as well as intuitively) that A becomes larger as p becomes larger. The upper bound on A is given by

$$A \leq \frac{\log n - 1}{\log m - 1},$$

which for given n and m is approached asymptotically as p approaches 1.

In the remainder of this section we deal with hierarchical binary organizations of order 2 where A is "large," i.e., where p is close to 1 and where n/m is large (and hence $n - m$ is close to n).

Let $\tilde{m}$ denote the value of m which maximizes A for specified n and $p(n, m)$. To find $\tilde{m}$ we can maximize $A(n, m)$ of (3.4) or, equivalently, minimize s_2 of (3.3):

$$\frac{ds_2}{dm} \approx \frac{1}{\log_e 2}\left[\frac{1}{m} - \frac{dp}{dm}\log_e(n - m) - \frac{1 - p}{n - m}\right] = 0. \tag{3.9}$$

For large values of n, (3.9) can be approximated by

$$\frac{1}{m} - \frac{dp}{dm}\log_e n \approx 0$$

or

$$\frac{dp}{dm} \approx \frac{1}{m \log_e n}. \tag{3.10}$$

If the density function for the distribution $p(n, m)$ is $\phi(i)$, then (3.10) implies that

$$\phi(\tilde{m}) \approx \frac{1}{\tilde{m} \log_e n}. \tag{3.11}$$

Thus $\tilde{m}$ can be found graphically by locating the intersection of $\phi(i)$ and $1/i \log_e n$ (see Figure 4).

PROPOSITION 5. *Let $\phi(i)$ be a density function used in a hierarchical binary organization of order 2. Then $\tilde{m}$ is approximately the value of i for which*

$$\phi(i) = \zeta(i)$$

(*where $\zeta(i)$ is Zipf's density function*).

PROOF. From (3.6) and (3.7) we can write

$$\zeta(i) = \frac{1}{iH_n} \approx \frac{1}{i(\log_e n + 0.577)},$$

which for large n becomes

$$\zeta(i) \approx \frac{1}{i \log_e n}.$$

The proposition then follows from (3.11). □

We thus see that while Zipf's distribution serves as a yardstick for deciding whether or not a given distribution can yield an advantageous hierarchical binary organization of order 2, Zipf's density helps in the actual design of such an organization (by determining $\tilde{m}$ and hence the optimal partition $\{X_1, X_2\}$).

Examining Figure 4, we can also see that the "steeper" $\phi(i)$ is, the smaller is the value of $\tilde{m}$ and hence the larger is the advantage index A. For example, consider the family of density functions

$$\phi_\delta(i) = \frac{c'}{i^{1+\delta}} \quad (\delta > 0)$$

(δ being a measure of the "steepness" of the function ϕ_δ), where

$$\frac{1}{c'} = \sum_{\nu=1}^{n} \frac{1}{\nu^{1+\delta}}.$$

When n is large, we have

$$\frac{1}{c'} \approx 0.5 + \int_1^\infty \frac{d\nu}{\nu^{1+\delta}} = 0.5 + \frac{1}{\delta}.$$

By Proposition 5, $\tilde{m}$ is the solution of

$$\frac{c'}{i^{1+\delta}} \approx \frac{1}{i \log_e n}$$

or

$$\tilde{m} \approx \left[\left(\frac{\delta}{1 + 0.5\delta}\right)\log_e n\right]^{1/\delta}.$$

When n is large, we have

$$s_2 \approx \log \tilde{m} \approx \frac{1}{\delta} \log \log n$$

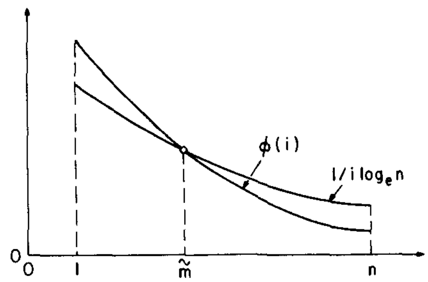
and hence

$$A \approx \frac{\delta \log n}{\log \log n}. \tag{3.12}$$

Thus A is proportional to δ, i.e., to the "steepness index" of the density function.

It should be noted that Proposition 5 entails a number of approximations which cumulatively may cause $\tilde{m}$ to be incorrect unless certain assumptions are valid. First, it should be recalled that (3.1), and hence (3.3) and (3.4), are good approximations only when n and m are not too small (say, greater than 20) and when the key and frequency orderings are not correlated. In deriving (3.11) we further assumed that n is much larger than m (say, $n > 10m$) and that p and ϕ can be regarded as continuous functions (a reasonable assumption when n is large). We also assumed tacitly that ϕ is consistently "steeper" than ζ and hence that ϕ intersects ζ only once, thus yielding a unique value for $\tilde{m}$.

Fig. 4. Finding $\tilde{m}$.



In practical cases the value of $\tilde{m}$ obtained via Proposition 5 may be correct only within an order of magnitude. In these cases it can serve as a good starting point for the trial-and-error evaluation of the correct optimal value.

4. Hierarchical Binary Search of Order r

The recursion (3.2) can be written for large $n_1, n_2, \ldots, n_r$ as

$$s_1 \approx \log n_1 \tag{4.1}$$
$$s_k \approx \log n_k + \frac{q_{k-1}}{q_k} s_{k-1} \quad (2 \le k \le r)$$

The solution of (4.1) for $k = r$ is given by

$$s_r \approx \sum_{j=1}^{r} q_j \log n_j \tag{4.2}$$

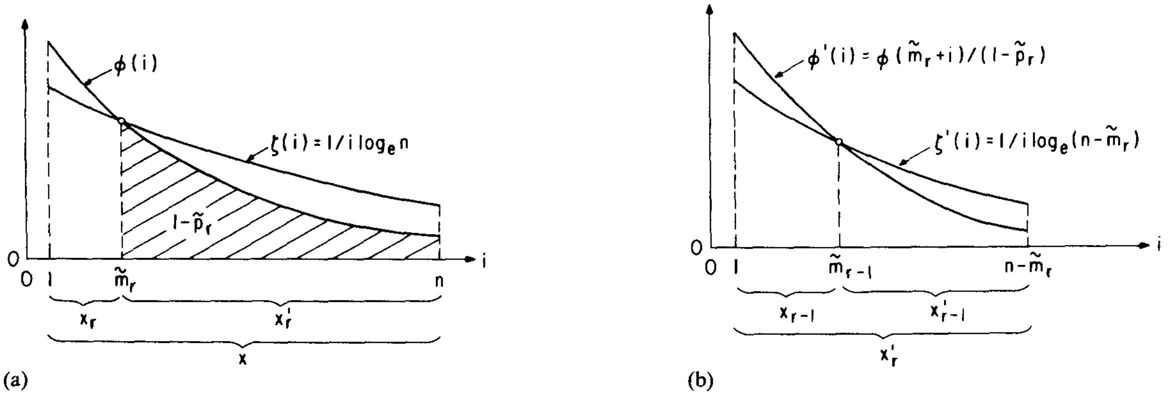
(where $q_r = 1$). We have not discovered a simple way for minimizing s_r in this general case (i.e., for finding the values of r, q_j, and n_j for specified n and $\phi(i)$ which minimize s_r). What we propose, instead, is to apply the technique of Section 3 (for $r = 2$) iteratively to obtain an improvement in the advantage index A in a relatively simple fashion. It should be noted that in those cases where $A > 1$ corresponds to $p_r \gg p_{r-1}$, the improvement achieved either by minimizing (4.2) or by the methods proposed below is only marginal compared to the value of A obtained with $r = 2$.

Using (4.1) we can write

$$A = \frac{s(n)}{s_r} \approx \frac{s(n)}{\log n_r + (1 - p_r)s_{r-1}}.$$

Let us determine n_r as if we designed a hierarchical binary organization with $r = 2$. Using the technique of Section 3, we can find the optimal $n_r = \tilde{m}_r$ and the corresponding value $\tilde{p}_r$ of p_r by intersecting $\phi(i)$ with $\zeta(i) = 1/i \log_e n$ (see Fig. 5(a)). As a result we obtain

Fig. 5. Constructing Binary Hierarchical Organization of Order r.



$$A \approx \frac{s(n)}{\log \tilde{m}_r + (1-\tilde{p}_r)s_{r-1}}.$$

The advantage index A can be improved by reducing s_{r-1}; we do this by partitioning $X_r' = X - X_r$ in the same manner that X has just been partitioned, i.e., by again using the technique of Section 3. In proceeding with the partitioning of X_r' (whose access probability is $1 - \tilde{p}_r$), let us rename the keys $x_{\tilde{m}_r+1}, x_{\tilde{m}_r+2}, \ldots, x_n$ as $x_1', x_2', \ldots, x_{n-\tilde{m}_r}'$, respectively. Correspondingly, let us define the new density functions

$$\phi'(i) = \frac{\phi(\tilde{m}_r + i)}{1 - \tilde{p}_r},$$

$$\zeta'(i) = \frac{1}{i \log_e(n - \tilde{m}_r)},$$

and use these to find the optimal $n_{r-1} = \tilde{m}_{r-1}$ at which $\phi'(i)$ and $\zeta'(i)$ intersect (see Fig. 5(b)). The same partitioning procedure can be repeated now with $X_{r-1}' = X_r' - X_{r-1}$, and so forth. The process can be continued until the access probability of the "remainder" keys becomes sufficiently small to make continuation impracticable.

A simpler method, but equally effective in practice, is to find $\tilde{m}$ for the hierarchical binary organization of order 2 and then let $|X_j| = \tilde{m}$ for $j = 2, 3, \ldots, r$ and $|X_1| = n - \tilde{m}(r-1)$, where r is any integer such that $\tilde{m}(r-1) \ll n$ (see Figure 6). We call this organization a *uniform* hierarchical organization of order r. Using (4.2), we obtain in this case

$$s_r \approx q_1 \log[n - \tilde{m}(r-1)] + \sum_{j=2}^{r} q_j \log \tilde{m},$$

or, approximating further,

$$s_r \approx \log \tilde{m} + p_1 \log n.$$

Thus, in a uniform hierarchical organization of order r we have

$$A \approx \frac{\log n}{\log \tilde{m} + (1-\tilde{p})\log n}.$$

5. An Illustrative Example

To illustrate the preceding discussion, let us consider a set of keys $x_1, x_2, \ldots, x_n$, where $n = 2^{16}$ and where the access probability of x_i is inversely proportional to i^2. In this case (see [1, pp. 74–75]),

$$\phi(i) \approx \frac{6}{\pi^2 i^2} = \frac{0.608}{i^2}.$$

For the binary hierarchical search of order 2, we have

$$A = \frac{s(n)}{s_2} \approx \frac{15}{ps(m) + (1-p)[u(m) + 15]}.$$

Trial-and-error calculations show that A is maximized when $m = 7$ and correspondingly when

$$p \approx 0.919, \quad s(m) \approx 2.43, \quad u(m) \approx 3.00,$$

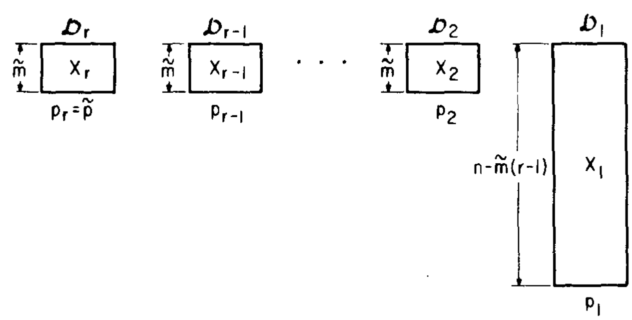
in which case

$$A \approx \frac{15}{3.69} = 4.07.$$

(Note that the approximate formula (3.12) gives in this case $A = 4.00$. However, the proximity of the two values in this case may not be significant, since $m = 7$ is not sufficiently large for (3.12) to be reliable.)

Employing the method of Section 3, the (approximate) value of the maximizing m is the solution of

Fig. 6. Uniform hierarchical organization of order r.



$$\frac{0.608}{i^2} \approx \frac{1}{i \log_e 2^{16}}$$

or

$$i \approx 0.608 \times 16 \times \log_e 2 = 6.74,$$

which agrees with the exact result.

Using the value $\tilde{m} = 7$, let us now design a uniform hierarchical organization with $r = 6$. Thus we have $|X_j| = 7$ for $j = 2, 3, \ldots, 6$ and $|X_1| = 2^{16} - 35$. In this case (see (2.1)),

$$s_1 \approx 15,$$

$$s_k = \frac{p_k}{q_k} s(\tilde{m}) + \frac{q_{k-1}}{q_k} [u(\tilde{m}) + s_{k-1}] \quad (2 \leq k \leq 6),$$

where

$$\begin{aligned} &s(\tilde{m}) \approx 2.43, \quad u(\tilde{m}) \approx 3.00 \\ &\quad p_1 \approx 0.017, \quad p_2 \approx 0.004, \quad p_3 \approx 0.007, \\ &\quad p_4 \approx 0.014, \quad p_5 \approx 0.039, \quad p_6 \approx 0.919 \\ &\quad q_1 \approx 0.017, \quad q_2 \approx 0.021, \quad q_3 \approx 0.028, \\ &\quad q_4 \approx 0.042, \quad q_5 \approx 0.081, \quad q_6 \approx 1.000 \end{aligned}$$

Thus we obtain

$$s_r \approx 3.21 \quad \text{and} \quad A \approx \frac{15}{3.21} = 4.67.$$

Summarizing this example: A hierarchical binary organization of order 2 speeds up the search time by a factor of approximately 4.1, while a uniform hierarchical organization of order 6 speeds up the search by a factor of approximately 4.7.

6. Conclusions

We have explored the conditions under which hierarchical binary search is faster, on the average, than nonhierarchical binary search. We have shown that an advantageous two-stage hierarchy can always be constructed when the keys' access density function $\phi(i)$ is "steeper" (in the graphical sense) than Zipf's density function $\zeta(i)$. The steeper it is, the greater the advantage. A simple method has been formulated in this case for approximating the optimal partition of the keys by finding the intersection of $\phi(i)$ and $\zeta(i)$.

For the r-stage hierarchical search we have not found a simple procedure for constructing the optimal partition. However, we have proposed procedures which are close to optimal in practical cases. The first procedure consists of a repetitive application of the technique developed for the two-stage case. The second—which is the simpler of the two—produces a partition in which the $r - 1$ most active subsets have the same cardinality, which equals that derived for the two-stage case.

The practicality of the results and techniques described in this paper is confined to static data files of moderate to large size (say, $n > 2^{10}$), where the access density function $\phi(i)$ of the keys, or at least the "steepness" of $\phi(i)$ relative to $\zeta(i)$, is known in advance. In such files, with appropriate $\phi(i)$, the speedup in search time achievable with hierarchical organization can be of order $\log n/\log m$, where m is the cardinality of the most active subset in the hierarchy.

It is known that when the n keys are equally likely, the average number of comparisons required by any search algorithm based on key comparisons is at least $\log n$ for large n. The advantage of a binary search is that it achieves this lower bound without requiring additional memory. In this paper we showed that when the keys are not equally likely, the $\log n$ bound may be further lowered (sometimes considerably—by a factor of $\log m$) by means of hierarchical organization. Assuming key access distributions for which such an organization is practicable, the search method offered by the hierarchical binary scheme constitutes an improvement over known conventional search schemes based on key comparison.

Received 1/79; accepted 7/79; revised 1/80

References

1. Knuth, D.E. *The Art of Computer Programming, Vol. 1.* Addison-Wesley, Reading, Mass., 1973.
2. Knuth, D.E. *The Art of Computer Programming, Vol. 3.* Addison-Wesley, Reading, Mass., 1973.
3. Reingold, E.M., Nievergelt, J., and Deo, N. *Combinatorial Algorithms.* Prentice-Hall, Englewood Cliffs, N.J., 1977.
4. Zipf, G.K. *Human Behavior and the Principle of Least Effort.* Addison-Wesley, Reading, Mass., 1949.