# Randomized Binary Search Technique

S. R. Arora and W. T. Dent
*University of Minnesota,* Minneapolis, Minn.*

**A mathematical model is developed for the mean and variance of the number of trials to recover a given document in a randomly received list of files. The search method described is binary in nature and offers new potential for information retrieval systems.**

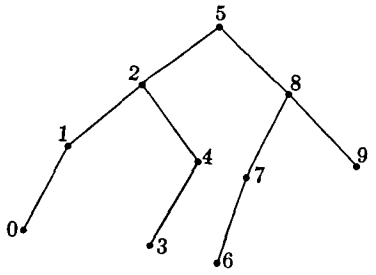
KEY WORDS AND PHRASES: binary pattern, file examination, graph theory, information retrieval, mathematical model, partitioning, probabilistic method, random sequencing, search techniques, tree structures
CR CATEGORIES: 3.74, 5.32

## Introduction

In the last decade the information explosion problem has become increasingly important. Libraries, legal advisory services, medical centers, and professional scientific organizations must cope with a huge volume of new literature in pertinent fields. The problem of storage and retrieval of relevant articles and documents is as complex as that of "keeping up with the literature." The randomized binary search technique offered below has the potential of being quite useful in information retrieval.

This technique has been introduced by Sussenguth [1] and comparisons of search costs with alternate methods looked at by Clampett [2].

To illustrate the technique let us consider the following example of a list of ten files numbered 0 to 9. Assume that these files arrived into a given system in a random order and were stored in order of arrival. Suppose a particular arrival pattern is 5, 8, 2, 9, 7, 4, 1, 3, 0, 6 and a call is made for file 3. Under the binary storage method the documents are stored in the following tree structure.




From every node of the tree two branches originate. The right branch leads to higher numbers, the left branch to lower numbers. To find 3, since 3 is less than 5, the left branch out of 5 is selected. Since 3 is greater than 2, the process selects the right-hand branch from 2. This leads to 4, and from there the left-hand branch leads to 3. It can be seen that at each stage the range of search is narrowed as the process eliminates intervals wherein the number sought cannot lie. In the selection of the left-hand branch at the first stage, the range of search is immediately limited to [0, 4]. The right-hand branch from the node 2 eliminates the interval [0, 2] so that [3, 4] only remains to be searched, and so on until the range of search is simply one number, the one sought.

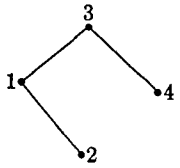
* School of Mechanical and Aerospace Engineering, Department of Mechanical Engineering.

TABLE I

| | |
|---|---|
| *Order of numbers* | 4 4 4 4 4 4 3 3 3 3 3 3 2 2 2 2 2 2 1 1 1 1 1 1<br>3 3 2 2 1 1 4 4 2 2 1 1 4 4 3 3 1 1 4 4 3 3 2 2<br>2 1 3 1 3 2 2 1 4 1 4 2 3 1 4 1 4 3 3 2 4 2 4 3<br>1 2 1 3 2 3 1 2 1 4 2 4 1 3 1 4 3 4 2 3 2 4 3 4 |
| *Number of trials to find 2 (by binary search)* | 3 4 2 2 4 3 2 3 2 2 3 3 1 1 1 1 1 1 4 3 3 3 2 2 |

To set up the mathematical model to find the expected value and variance of the number of trials required to recover a particular file, a further simplified example is used. Suppose the documents are coded as 1, 2, 3, 4, and that 2 is the catalog of the desired article. The arrival of the documents is random and thus 24 equally likely arrangements exist. Table I gives a full enumeration of these permutations.

The permutation 3, 4, 1, 2 may be arranged as




and requires 3 trials to find 2. The average number of trials to recover 2 is 56/24. The variance of the number of trials is 35/36.

## Mathematical Model

Consider the given list of numbers arranged in monotonic order such as 1, 2, 3, 4. Let $x(i, j)$ denote the number of trials required to recover a particular number $(i, j)$ which has in this ordered list $i$ numbers to its left and $j$ to its right. In the above example $x(1, 2)$ is the number of trials to recover $2 = (1, 2)$. In the search to recover $(1, 2) = 2$ we examine the first element in the file. If it is $(1, 2)$, we are finished, and since $(1, 2)$ is the first number in 1/4 of the permutations, $x(1, 2) = 1$ with probability 1/4. If the first search yields $1 = (0, 3)$, the problem is reduced to finding the number 2 in the group 2, 3, 4, since the number 1 is eliminated. Thus $x(1, 2) = 1 + x(0, 2)$ with probability 1/4. If the first search yields the number $3 = (2, 1)$, then because of the tree structure the next searches are narrowed to the set 1, 2 so that $x(1, 2) = 1 + x(1, 0)$ with probability 1/4. If the first search element is 4, then $x(1, 2) = 1 + x(1, 1)$ with probability 1/4.

In the general case the search is for the number $(i, j)$, and if $(i, j)$ is the first number examined, then $x(i, j) = 1$

with probability $1/(i+j+1)$ since there are $i+j+1$ numbers in the total, and they each appear first in the same number of permutations. If the first number examined is $(k_1, i+j-k_1)$ where $k_1 \leq i$, then the search is restricted to the numbers to the right, so that $x(i,j) = 1 + x(i-k_1, j)$ with probability $1/(i+j+1)$. If the first number examined is $(i+j-k_2, k_2)$ where $k_2 \geq j$, a similar argument shows that $x(i,j) = 1 + x(i, j-k_2)$. Then obviously

$$x(i,j) = \begin{Bmatrix} 1 \\ 1 + x(i-k_1, j) \\ 1 + x(i, j-k_2) \end{Bmatrix}$$

$$\text{with probability} \begin{cases} \dfrac{1}{1+i+j}; & \\ \dfrac{1}{1+i+j}; & k_1 = 1, \cdots, i; \\ \dfrac{1}{1+i+j}; & k_2 = 1, \cdots, j. \end{cases} \tag{1}$$

Let $n(i,j) = E[x(i,j)]$. Then taking expectations in (1), for $i \neq 0$, $j \neq 0$,

$$n(i,j) = 1 + \frac{1}{1+i+j}\left[\sum_{k_1=1}^{i} n(i-k_1, j) + \sum_{k_2=1}^{j} n(i, j-k_2)\right]. \tag{2}$$

The solution to eq. (2) under the boundary condition $n(0,0) = 1$ can be found using the following identity when neither $i$ nor $j$ is zero:

$$n(i,j) + n(i-1, j-1) - n(i, j-1) - n(i-1, j) \equiv 0. \tag{3}$$

Using eq. (3) recursively yields

$$n(i,j) = n(0,j) + n(i,0) - n(0,0). \tag{4}$$

For $j = 0$ the appropriate version of eq. (2) is

$$n(i,0) = 1 + \frac{1}{1+i}\sum_{k_1=1}^{i} n(i-k_1, 0). \tag{5}$$

Similarly

$$n(0,j) = 1 + \frac{1}{1+j}\sum_{k_2=1}^{j} n(0, j-k_2). \tag{6}$$

From eq. (5)

$$n(i-1,0) = 1 + \frac{1}{i}\sum_{k_1=1}^{i-1} n(i-k_1-1, 0). \tag{6a}$$

But

$$\begin{aligned} \sum_{k_1=1}^{i-1} n(i-k_1-1, 0) &= \sum_{k_1=1}^{i} n(i-k_1) - n(i-1,0) \\ &= (1+i)[n(i,0) - 1] - n(i-1,0), \end{aligned} \tag{6b}$$

from eq. (5). Substituting this expression in (6a)

$$n(i,0) - n(i-1,0) = \frac{1}{1+i}. \tag{7}$$

Using this recursively and $n(0,0) = 1$,

$$n(i,0) = \sum_{k_1=1}^{i+1} \frac{1}{k_1}, \qquad i \geq 1. \tag{8}$$

Similarly,

$$n(0,j) = \sum_{k_2=1}^{j+1} \frac{1}{k_2}, \qquad j \geq 1. \tag{9}$$

Substituting eqs. (8) and (9) in eq. (4), if neither $i$ nor $j$ is zero,

$$E[x(i,j)] = \sum_{k_1=1}^{i+1} \frac{1}{k_1} + \sum_{k_2=1}^{j+1} \frac{1}{k_2} - 1. \tag{10}$$

The variance of $x(i,j)$ is found from eq. (1) by squaring both sides and taking expectations. If $s(i,j) = E[x(i,j)]^2$,

$$s(i,j) + 1 - 2n(i,j) = \frac{1}{1+i+j}\left[\sum_{k_1=1}^{i} s(i-k_1, j) + \sum_{k_2=1}^{j} s(i, j-k_2)\right]. \tag{11}$$

For $i \neq 0, j = 0$, the appropriate version of eq. (11) is

$$s(i,0) + 1 - 2n(i,0) = \frac{1}{1+i}\sum_{k_1=1}^{i} s(i-k_1, 0). \tag{12}$$

so that, using methods similar to those employed in the derivation of eq. (7),

$$(1,i)[s(i,0) - s(i-1,0)] - 2n(i-1,0) - 1 = 0. \tag{13}$$

Thus, using $s(0,0) = 1$, recursion on this equation yields

$$s(i,0) = \sum_{k_1=1}^{i} \frac{1}{k_1+1}[2n(k_1-1, 0) + 1] + 1, \qquad i \geq 1. \tag{14}$$

Similarly,

$$s(i,0) = \sum_{k_2=1}^{j} \frac{1}{k_2+1}[2n(0, k_2-1) + 1] + 1, \qquad j \geq 1. \tag{15}$$

For $i \neq 0, j \neq 0$, using earlier results for $n(i,j)$ developed in eqs. (4) and (10), the following relation can be deduced from (11):

$$\begin{aligned} s(i,j) + s(i-1, j-1) - s(i, j-1) - s(i-j) &= \frac{2(i+j)}{(i+j+1)}[n(i,j) + n(i-1, j-1) \\ &\quad - n(i, j-1) - n(i-1, j)] \\ &\quad + \frac{2}{(i+j+1)}[n(i,j) - n(n-1, j-1)] \\ &= \frac{2}{(i, j+1)}\left[\frac{1}{i+1} + \frac{1}{j+1}\right]. \end{aligned} \tag{16}$$

TABLE II. Mean (Upper Figure) and Variance of Trials to Retrieve $(i, j)$

| i/j | 0 | 1 | 2 | 3 | 4 | 5 | 10 | 15 | 20 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1.0000 | 1.5000 | 1.8333 | 2.0833 | 2.2833 | 2.4500 | 3.0199 | 3.3807 | 3.6454 |
| | 0.0000 | 0.2500 | 0.4722 | 0.6597 | 0.8197 | 0.9586 | 1.4618 | 1.7964 | 2.0469 |
| 1 | 1.5000 | 2.0000 | 2.3333 | 2.5833 | 2.7833 | 2.9500 | 3.5199 | 3.8807 | 4.1454 |
| | 0.2500 | 0.6667 | 0.9722 | 1.2097 | 1.4031 | 1.5658 | 2.1285 | 2.4876 | 2.7515 |
| 2 | 1.8333 | 2.3333 | 2.6667 | 2.9167 | 3.1167 | 3.2833 | 3.8532 | 4.2141 | 4.4787 |
| | 0.4722 | 0.9722 | 1.3222 | 1.5875 | 1.7999 | 1.9765 | 2.5751 | 2.9494 | 3.2218 |
| 3 | 2.0833 | 2.5833 | 2.9167 | 3.1667 | 3.3667 | 3.5333 | 4.1032 | 4.4641 | 4.7287 |
| | 0.6597 | 1.2097 | 1.5875 | 1.8706 | 2.0955 | 2.2814 | 2.9045 | 3.2897 | 3.5681 |
| 4 | 2.2833 | 2.7833 | 3.1167 | 3.3667 | 3.5667 | 3.7333 | 4.3032 | 4.6641 | 4.9287 |
| | 0.8197 | 1.4031 | 1.7999 | 2.0955 | 2.3293 | 2.5218 | 3.1630 | 3.5563 | 3.8394 |
| 5 | 2.4500 | 2.9500 | 3.2833 | 3.5333 | 3.7333 | 3.9000 | 4.4699 | 4.8307 | 5.0954 |
| | 0.9586 | 1.5658 | 1.9765 | 2.2814 | 2.5218 | 2.7194 | 3.3745 | 3.7742 | 4.0610 |
| 10 | 3.0199 | 3.5199 | 3.8532 | 4.1032 | 4.3032 | 4.4699 | 5.0398 | 5.4006 | 5.6652 |
| | 1.4618 | 2.1285 | 2.5751 | 2.9045 | 3.1630 | 3.3745 | 4.0695 | 4.4884 | 4.7866 |
| 15 | 3.3807 | 3.8807 | 4.2141 | 4.4641 | 4.6641 | 4.8307 | 5.4006 | 5.7615 | 6.0261 |
| | 1.8517 | 2.4876 | 2.9494 | 3.2897 | 3.5563 | 3.7742 | 4.4884 | 4.9170 | 5.2212 |
| 20 | 3.6454 | 4.1454 | 4.4787 | 4.7287 | 4.9287 | 5.0954 | 5.6652 | 6.0261 | 6.2907 |
| | 2.0469 | 2.7515 | 3.2218 | 3.5681 | 3.8394 | 4.0610 | 4.7866 | 5.2212 | 5.5293 |

On intuitive investigation and by employing the results found above, the general solution for eq. (16), when $i \neq 0$, $j \neq 0$, is found to be

$$s(i,j) = 2\sum_{k_1=0}^{i-1}\sum_{k_2=0}^{j-1}\frac{1}{1+i+j-k_1-k_2}\cdot\left(\frac{1}{i+1-k_1}+\frac{1}{j+1-k_2}\right) + s(i,0) + s(0,j) - s(0,0). \quad (17)$$

From $s(i, j)$ the variance of $x(i, j)$ is easily found.

If every file in a list were to be called with equal frequency, the long-run average number of trials to recover a document would be

$$E[N] = \frac{\sum_{k_3=0}^{i+j+1} n(k_3, i+j+1-k_3)}{(i+j+1)}. \quad (18)$$

The complete distribution of $x(i, j)$ can be approached by setting up difference equations of the probability generating function of $x(i, j)$ but these equations are too complex to have a solution in closed analytic form. Some numerical values of the mean and variance of the number of trials to find document $(i, j)$ are presented in Table II.

A possible application of the binary search technique is in situations where the order of randomly arriving files must be presented. For example Western Union is required to store messages on drum for a certain period after their arrival before transferring them to slower access equipment. If messages can be uniquely coded (according to place of origin and name of receiver, say) the 3-cell technique demonstrated by Clampett may be modified to

TABLE III. Drum Storage Patterns

| Cell number | Message code and time of arrival | Address of cell with next lower code in tree and time of entry | Address of cell with next higher code in tree and time of entry | Location information (cell number) |
|---|---|---|---|---|
| | A. Storage Pattern 1 | | | |
| 1 | $73_{t_1}$ | $5_{t_2}$ | $9_{t_3}$ | 4 |
| 5 | $62_{t_2}$ | $13_{t_4}$ | $21_{t_6}$ | 8 |
| 9 | $84_{t_3}$ | $25_{t_7}$ | $29_{t_8}$ | 12 |
| 13 | $10_{t_4}$ | | $17_{t_5}$ | 16 |
| 17 | $27_{t_5}$ | | | 20 |
| 21 | $68_{t_6}$ | $33_{t_9}$ | | 24 |
| 25 | $75_{t_7}$ | | | 28 |
| 29 | $91_{t_8}$ | | | 32 |
| 33 | $66_{t_9}$ | | | 36 |
| | B. Drum Storage Pattern 2 | | | |
| 1 | $12_{t_{10}}$ | | | 4 |
| 5 | $62_{t_2}$ | $13_{t_4}$ | $21_{t_6}$ | 8 |
| 9 | $84_{t_3}$ | $25_{t_7}$ | $29_{t_8}$ | 12 |
| 13 | $10_{t_4}$ | | $17_{t_5}$ | 16 |
| 17 | $27_{t_5}$ | $1_{t_{10}}$ | | 20 |
| 21 | $68_{t_6}$ | $33_{t_9}$ | $9_{t_1+T^*}$ | 24 |
| 25 | $75_{t_7}$ | | | 28 |
| 29 | $91_{t_8}$ | | | 32 |
| 33 | $66_{t_9}$ | | | 36 |
| | C. Drum Storage Pattern 3 | | | |
| 1 | $12_{t_{10}}$ | | | 4 |
| 5 | $74_{t_{11}}$ | | | 8 |
| 9 | $84_{t_3}$ | $25_{t_7}$ | $29_{t_8}$ | 12 |
| 13 | $10_{t_4}$ | | $17_{t_5}$ | 16 |
| 17 | $27_{t_5}$ | $1_{t_{10}}$ | | 20 |
| 21 | $68_{t_6}$ | $33_{t_9}$ | $9_{t_1+T^*}$ | 24 |
| 25 | $75_{t_7}$ | $5_{t_{11}}$ | | 28 |
| 29 | $91_{t_8}$ | | | 32 |
| 33 | $66_{t_9}$ | $13_{t_2+T^*}$ | | 36 |

a 4-cell technique with one cell to hold the message code, two cells to indicate drum addresses of higher/lower code numbers, and a last cell giving data on actual physical message location. A call for verification of a particular message is readily traceable using the binary search technique, and the message may be removed after the specified period with a new ordered tree of messages remaining. To illustrate how this could be effected, let us consider the following nonrigorous example.

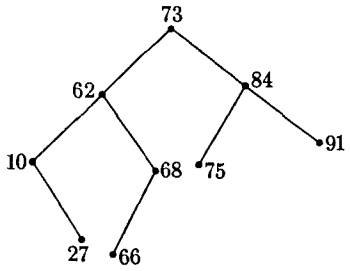
Let $T^*$ be the required time messages are held on drum, and suppose there are 4 $m$ available cells on the drum. Assume no more than $m$ messages arrive per period $T^*$. Denote the messages $m_1, m_2, \cdots$. Upon arrival at time $t_i$ message $m_i$ will enter cell

$$4\{i - [i - 1/m]\cdot m\} - 3,$$

where $[x]$ denotes the integral part of the number $x$. Messages will be coded and the code representations connected by a binary tree structure. At any time $t$ there will exist a tree structure of no more than $m$ codes.

Consider a specific case where $m = 9$, with the first 11 messages having codes 73, 62, 84, 10, 27, 68, 75, 91, 66, 12, 74. These messages arrive at times $t_1, \cdots, t_{11}$ with

$t_{10} \geq t_1 + T^*$. The first complete tree is:




The corresponding drum storage pattern just before time $t_1 + T^*$ is shown in Table III, where cells are counted down successive columns.

Consider the arrival of the message with code 27. From the tree structure we see that 27 lies on the right-hand branch from 10. Hence at time $t_5$ (when the message with code 27 arrives) the code 27 will be entered in cell 17, and in cell 15 (corresponding to the right-hand branch from 10) the address cell 17 will be entered. A similar procedure is used for codes on left-hand branches in the tree.

Suppose now that a call is made for the message with code 66. Entry in this case is made to cell 1. Since 66 is less than 73, the left-hand branch from 73 is required. This leads to cell 5. In cell 5 the code 62 is found. Since 66 is greater than 62 the right-hand branch from 62 is needed, leading to cell 21. Here code 68 is located, and the left-hand branch from this point to cell 33 uncovers code 66. In cell 36 is information concerning the actual message.

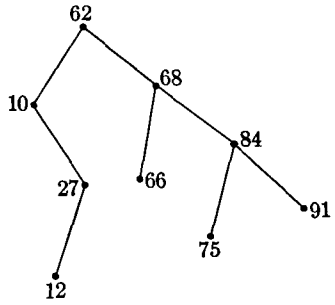
The greater advantage of the binary search technique however lies in its ability to change readily as new messages enter and leave the drum. At time $t_1 + T^*$ cells 1 to 4 are to be erased as the code information is transferred elsewhere. The original tree structure is to be destroyed and a new structure formed with leading element the code of message $m_2$, or 62. The following general scheme applies to form this new construction when the code of $m_i$ is less than the code of $m_{i-1}$. (A simple analogy holds in the reverse case.)

STEP 1. Let the present tree structure be made up of message codes $a_1, \cdots, a_m$ (where $a_1$ is the code of message $m_i$ for some $i$). $a_1$ is the leading code which has to be deleted and replaced by, say, $a_j$. $a_j$ is less than $a_1$. The left-hand side of the tree (from $a_1$) is searched for the most-right-hand element, say $a_j{}^*$. This is easily found by tracing the left-hand branch from $a_1$ and then successive right-hand branches from that point until no more exist at code $a_j{}^*$.

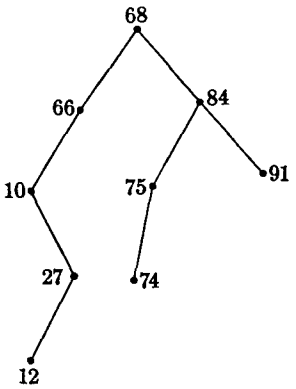
STEP 2. In the drum storage pattern if $a_1$ is in cell $p^* = 4p - 3$ $(1 \leq p \leq 9)$, and $a_j{}^*$ in cell $q^* = 4q - 3$ $(1 \leq q \leq 9)$ the number in cell $p^* + 2$ is transferred to cell $q^* + 2$ (which is previously empty), and cells $p^*$ to $p^* + 3$ are erased as message $a_1$ is transferred. At this point the code on the left-hand branch from $a_1$ in the original tree, say $a_1{}^*$, heads a new tree structure. Note that the code of any new message arriving may be appended now in its appropriate position in the new tree structure, using the just vacated drum cells. If $a_1{}^* = a_j$ the process is finished. If $a_1{}^* \neq a_j$ the following step is required.

STEP 3. The same procedure as in step 2 is followed to replace $a_1{}^*$ by $a_j{}^*$. That is, the most right-hand element on the left side of the tree headed by $a_1{}^*$ (assuming $a_j < a_1{}^*$) is found, and the transfer of cell addresses effected. However, now the information pertaining to $a_1{}^*$ is not erased. Via step 1 the process is repeated until at some stage $a_1{}^* = a_j$ in step 2.

To demonstrate, consider the previous example where code 73 is to be erased, and code 62 to head a new tree structure. $a_1 = 73$, $a_j = 62$, $p^* = 1$, $q^* = 21$. In step 1, $a_j{}^*$ is located as $a_j{}^* = 68$. In step 2, the number 9 in cell $p^* + 2 = 3$ is transferred to cell $q^* + 2 = 23$. Cells 1 to 4 are erased. $a_1{}^* = 62 = a_j$ so the process terminates. If 12 is the code of message $m_{10}$, the following tree structure and drum storage pattern are obtained at time $t_{10}$.




When code 62 is to be transferred (see Table III. B) at time $t_2 + T^*$ and replaced by code 84 leading a new structure, we have $a_1 = 62$, $a_j = 84$. By step 1 $a_j{}^* = 66$. By step 2 $a_1{}^* = 68$ and $p^* = 5$, $q^* = 33$. Hence the element in cell 6 is transferred to cell 34, and cells 5 to 8 are erased. With the code of message $m_{12}$ appended, the tree structure and drum storage pattern after Step 2 are




In one step the final pattern with 84 leading the tree structure will be obtained (see Table III. C). Although there is some delay in constructing new tree structures at each instant of time, a practical scheme would no doubt employ sophisticated versions of the above steps to minimize this aspect. The advantages of the binary search technique are now obvious—it has the ability to deal with random arrivals and yet it can, at the same time, provide an extremely efficient method of retrieval.

RECEIVED JUNE, 1967; REVISED SEPTEMBER, 1968

## REFERENCES

1. SUSSENGUTH, E. H., JR. Use of tree structures for processing files. *Comm. ACM 6*, 5 (May 1963), 272-279.
2. CLAMPETT, H. A., JR. Randomized binary searching with tree structures. *Comm. ACM 7*, 3 (Mar. 1964), 163-165.