UNIT 1 SEARCHING TECHNIQUES

Structure

1.0 INTRODUCTION

Information retrieval in the required format is the central activity in all computer applications. This involves searching, sorting and merging. This block deals with all three, but in this block we will be concerned with searching techniques.

Searching methods are designed to take advantage of the file organisation and optimize the search for a particular record or to establish its absence. The file organisation and searching method chosen can make a substantial difference to an application's performance.

We will now discuss two searching methods and analyze their performance. These two methods are:

- The sequential search

The binary search

1.1 OBJECTIVES

After going through this unit you will be able to:

list searching methods

discuss algorithms of Binary Search

analyse the performance of searching methods

1.2 SEQUENTIAL SEARCH

This is the most natural searching method. Simply put it means to go through a list or a file till the required record is found. It makes no demands on the ordering of records. The algorithm for a sequential search procedure is now presented.

ALGORITHM: SEQUENTIAL SEARCH

This represents the algorithm to search a list of values of to find the required one.

```
INPUT: List of size N. Target value T

OUTPUT: Position of T in the list - I

BEGIN

Set FOUND to false

Set I to O

While (I < = N) and (FOUND is false)

If List [I]             = T
FOUND                   = true
Else
                          I = I + 1

If FOUND is false.

T is not present in List.

END
```

This algorithm can easily be extended for searching for a record with a matching key value.

Analysis of Sequential Search

Whether the sequential search is carried out on lists implemented as arrays or linked lists or on files, the criterial part in performance is the comparison loop step 2. Obviously the fewer the number of comparisons, the sooner the algorithm will terminate.

The fewest possible comparisons = 1. When the required item is the first item in the list. The maximum comparisons = N when the required item is the last item in the list. Thus if the required item is in position I in the list, I comparisons are required.

Hence the average number of comparisons done by sequential search is

$$\frac{1+2+3.....+I+...+N}{N}$$

$$=\frac{N(N+1)}{2*N}$$

$$=(N+1)/2$$

Sequential search is easy to write and efficient for short lists. It does not require sorted data. However it is disastrous for long lists. There is no way of quickly establishing that the required item is not in the list or of finding all occurrences of a required item at one place.

We can overcome these deficiencies with the next searching method namely the Binary search.

1.3 BINARY SEARCH

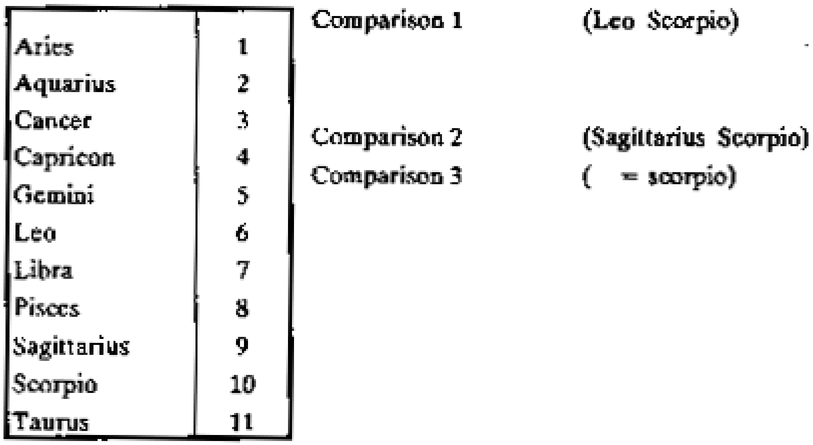
The drawbacks of sequential search can be eliminated if it becomes possible to eliminate large portions of the list from consideration in subsequent iterations. The binary search method just that, it halves the size of the list to search in each iterations.

Binary search can be explained simply by the analogy of searching for a page in a book. Suppose you were searching for page 90 in book of 150 pages. You would first open it at random towards the later half of the book. If the page is less than 90, you would open at a page to the right, it it is greater than 90 you would open at a page to the left, repeating the process till page 90 was found. As you can see, by the first instinctive search, you dramatically reduced the number of pages to search.

Binary search requires sorted data to operate on since the data may not be contiguous like the pages of a book. We cannot guess which quarter of the data the required item may be in. So we divide the list in the centre each time.

We will first illustrate binary search with an example before going on to formulate the algorithm and analysing it.

Example: Use the binary search method to find 'Scorpio' in the following list of 11 zodiac signs.



Aries	1
Aquarius	2
Cancer	3
Capricon	4
Gemini	5
Leo	6
Libra	7
Pisces	8
Sagittarius	9
Scorpio	10
Taurus	11

Comparison 1 (Leo Scorpio)

Comparison 2 (Sagittarius Scorpio)

Comparison 3 (= scorpio)

This is a sorted list of size 11. The first comparison is with the middle element number 6 i.e. Leo. This eliminates the first 5 elements. The second comparison is with the middle element from 7 to 11, i.e. 9 Sagittarius. This eliminates 7 to 9. The third comparison is with the middle element from 9 to 11, i.e. 10 Scorpio. Thus we have found the target in 3 comparisons. Sequential search would be taken 10 comparisons. We will now formulate the algorithm for binary search.

ALGORITHM BINARY SEARCH

This represents the binary search method to find a required item in a list sorted in increasing order.

INPUT: Sorted LIST of size N, Target Value TP

OUTPUT: Position of T in the LIST = I

```
BEGIN
        {
                MAX = N-1;
                MIN=0;
                FOUND=0;
                WHILE (( FOUND == 0) && (MAX > = MIN))
                { MID = (INT) (( MAX + MIN))
        IF       T = =  LIST[MID]
        { I = ++ MID;
                 FOUND = 1;
        }

        ELSE

                 {IF ( T < LIST[MID])
                          MAX = MID - 1;

                 ELSE
                          MIN = MID + 1;

                          }

                 }
        }

END
```

It is recommended that the student apply this algorithm to some examples.

Analysis of Binary search

In general, the binary search method needs no more than [$\log_2 n$] + 1 comparisons This implies that for an array of a million entries, only about twenty comparisons will be needed. Contrast this with the case of sequential search which on the average will need (n +1) / 2comparisons.

The conditions (MAX = MIN) is necessary to ensure that step 2 terminates even in the case that the required element is not present. Consider the example of Zodiac signs. Suppose the 10th item was Solar (an imaginary Zodiac sign). Then at that point we would have

```
MID  = 10
MAX  = 11
MIN  = 9
```

and, from 2.2 get

```
MAX = MID-1 = 9
```

In the next iteration we get

```
(2.1)       MID  = (9 + 9)DIV 2 = 9
(2.2)       MAX  = 9-1 = 8.
```

Since MAX < MIN, the loop terminates. Since FOUND is false, we consider the target was not found.

In the binary search method just described above, it is always the key in the middle of the list currently being examined that is used for comparison. The splitting of the list can be illustrated through a binary decision tree in which the value of a node is the index of the key being tested. Suppose there are 31 records, then the first key compared is at location 16 of the list since (1 + 31)/2 = 16. If the key is less than the, key at location 16 the location 8 is tested since (1 + 15)/2 = 8; or if key is less than the key at location 16, then the location 24 is tested. The binary tree describing this process is shown below (Figure 1)

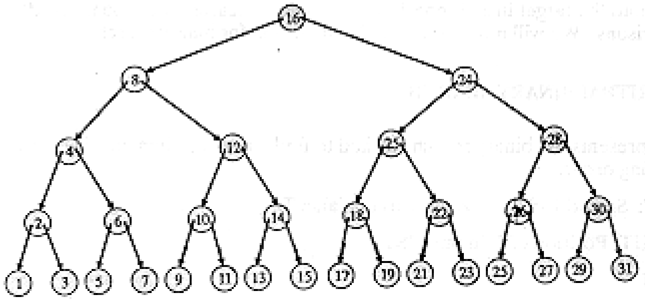


Figure 1. Searching Process In Binary Search

1.4 SUMMARY

This unit concentrates on searching techniques used for information retrieval. The sequential search method was seen to be easy to implement and relatively efficient to use for small lists. But very time consuming for long unsorted lists. The binary search method is an improvement, in that it eliminates half the list from consideration at each iteration; It has checks to incorporated to ensure speedy termination under all possible conditions. It requires only twenty comparisons for a million records and is hence very efficient. The prerequisite for it is that the list should be sorted in increasing order

1.5 EXERCISES

1. Implement the sequential search algorithm to search a linked list (in C language).

2. Modify the above program to find all occurrences of a required item in the list.

3. Implement the binary search algorithm to search a list implemented as an array (in C).

4. Modify the above to find all occurrences of a given item.

5. Modify the binary search algorithm to search a list sorted in descending order.

SUGGESTED READINGS

How to Solve it BY Computer: R.G. Dromey - PHI

Data Structure's & Program Design: Robert. L. Kruse - PHI