Length Based Non-Recursive Integer Search Algorithm

Anusha Jajur J
DOS in Computer Science,
Davangere University
Davanagere 577005, India.
anusha.anusha325@gmail.com

Prashanth.G.R
DOS in Computer Science,
Davangere University
Davanagere 577005, India.
prashanthgr2009@gmail.com

Vinay.S
DOS in Computer Science,
Davanagere University,
Davanagere 577005, India.
vinaygnanesh@yahoo.com

Abstract: **This paper present a new searching technique based on divide and conquer and brute force principles. In binary search concept input array is divided into left and right sub array and based key element searching is made on left or right sub arrays during searching array is recursively sub divided and key element will be compared with each element. In our new method, array will be divided into sub arrays based on size of the elements in input array. Initially key element size is calculated and based on the size pointer will moved to specific sub array for searching will be made by comparing each element with key element in sequential order.**

Keywords—Binary serach , Linear search, Brute Force method, Divide and Conqure.

I. Introduction

There are lots of searching algorithm are available and are working on different principles example hash search, interpolation search, linear search, binary search. Brute force is a straightforward approach to solving a problem, usually directly based on the problem statement and definitions of the concepts involved . The brute-force approach should not be overlooked as an important algorithm design strategy. First, unlike some strategies, brute force is applicable to a very wide variety of problems and it is difficult to point out problems. Second, for some important problems e.g., searching, string matching the brute-force approach yields reasonable algorithms of at least some practical value with no limitation on instance size. Third, the expense of designing a more efficient algorithm may be unjustifiable. Fourth, a brute-force algorithm can still be useful for solving small-size instances of a problem. Finally, a brute-force algorithm can serve more efficient alternatives for solving a problem. The two applications for the strategy of problem searching are first application deals with the canonical problem of searching for an item of a given value in a given list and second application deals with the string-matching problem.

Divide-and-conquer method is probably the best-known general algorithm design technique. Through this method, we are able to introduce very efficient algorithms for specific implementations of this general strategy. Divide-and-conquer algorithms work according to the following general plan:

1. A problem is divided into several subproblems of the same type, ideally of about equal size.

2. The subproblems are solved (typically recursively, though sometimes a different algorithm is employed, especially when subproblems become small enough).

3. If necessary, the solutions to the subproblems are combined to get a solution to the original problem.

Searching for data is one of the fundamental fields of computing. Often, the difference between a fast program and a slow one is the use of a good algorithm for the data set. In searching, for finding the time efficiency there are three cases for every searching methods they are worst case, average case and best case. There are three types of searching technique they are: Linear Searching, Binary Searching and Interpolation searching. In these searching techniques, we will search the key element in input array elements.

In linear search or sequential search is a Brute force method for finding a particular value in a list, which consists of checking every one of its elements, one at a time and in sequence, until the desired one is found. In linear search, we are searching the key element by incrementing the pointer array in the input list. This method is used for unsorted array. The time efficiency for best case is O(1), for average case is O(n) and for worst case is O(n).

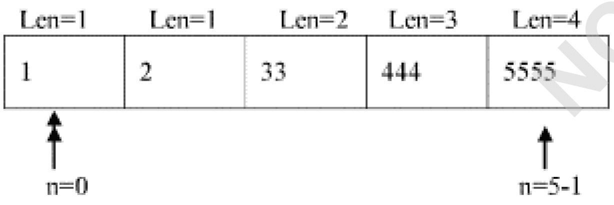
Binary search is a Divide and conquer method, it will locate an item in a sorted array by repeatedly dividing the search interval in half. The initial interval includes the entire array. If the value of the search key is less than the item in the middle of the interval, then the next interval will be the lower half of the current interval. If the value of the search key is greater than the middle item, then the next interval will be the upper half. The search process repeats until the item is found or the search interval is empty. The search start from the midpoint of the file, and goes to the midpoint of the associated remaining half if a match fails. The comparison of their values will decide which half of the file should be tried next time. This process will be repeated until a match is found. This method is used for sorted array that is in ascending order or descending order. The time efficiency for best case is O(1), for average case $O(\log_2 n)$ and for worst case is $O(\log_2 n)$.

In this paper, we are introducing the new concept of searching integer values using divided and conquer principle and Brute force principle. In the first step, elements of input array will be divided into sub group based on the size of the each element of an array and these sub-groups is stored in a one dimensional array which is a non- primitive linear data type. In the second step, we are determining the size of the key element. In the third step, if the size of an sub-array is equal to the size of the key element then the pointer will be moved to specific sub array to search an key element. This method is based on divide and conquer method. In the fourth step, we are searching the key element in the specifed sub-array using brute force method based on linear search algorithm. In the fifth element, if the key element found then we can get the message key element found, otherwise key element not found. This searching technique consists unordered elements. Using this method, it will take less time to search an key element. If the array is of large size, then also searching will be efficient.

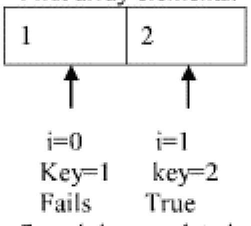
II. Working

In this searching technique, we are searching the key element of an unordered array. User will input n elements of an unordered array. Depending on the size of the array, we are storing the array elements in each one dimensional array. For example, if the array size is one then we are storing in first sub-array and if the array size is two then we are storing in second sub-array and so on. If the size of the key element is equal to the first sub-array then it will search the key element in the first sub-array list, otherwise if the size of the key element is equal to the second sub-array then it will search the key element in the second sub-array list and so on.

For Example: Array consists of 5 elements they are 1, 2, 33, 444, 5555 and the key element is 2.
Step 1: It will calculate the array elements length. Array elements 1 and 2 are of length 1, 33 is of length 2, 444 is of length 3 and 5555 is of length 4.
Step 2: Key element 2 is of length 1.
Step 3: Key length is equal to 1, so the key element will search in the first array.
Step 4: First array consists of array elements 1 and 2. In first step execution of for loop the key element not found, next step execution of for loop the key element will found and it will terminates the execution.



Array size = 5.
Key element to Search =2.
Key length=1.
So it will search in the array:
First array elements:



Search is completed.

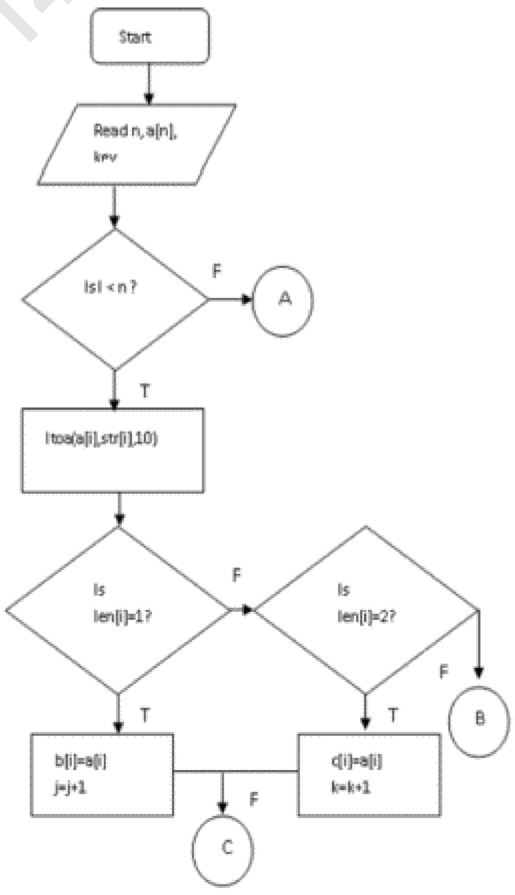
III. Implementation

Step 1:
Reading size of all the elements of input Array and grouping them based on size and store those groups of elements in different arrays.

Algorithm:

```
Step 1.1: read n, a[n], key
Step 1.2: [Finding length of each elements of input array and
grouping them based on its size]
        For i=0 to n in step of 1
        Begin
                ltoa(a[i],str[i],10)
                len[i]=strlen(str[i])
        if len[i]= =1 then
                b[i]=a[i];
                j=j+1;
        end if
        else if  len[i]= =2 then
                c[i]=a[i];
                k=k+1;
        end if
        else if len[i]==3 then
                d[i]=a[i];
                l=l+1;
        end if
        else
                e[i]=a[i];
                m=m+1;
        end if
end;
```

Flow Chart:



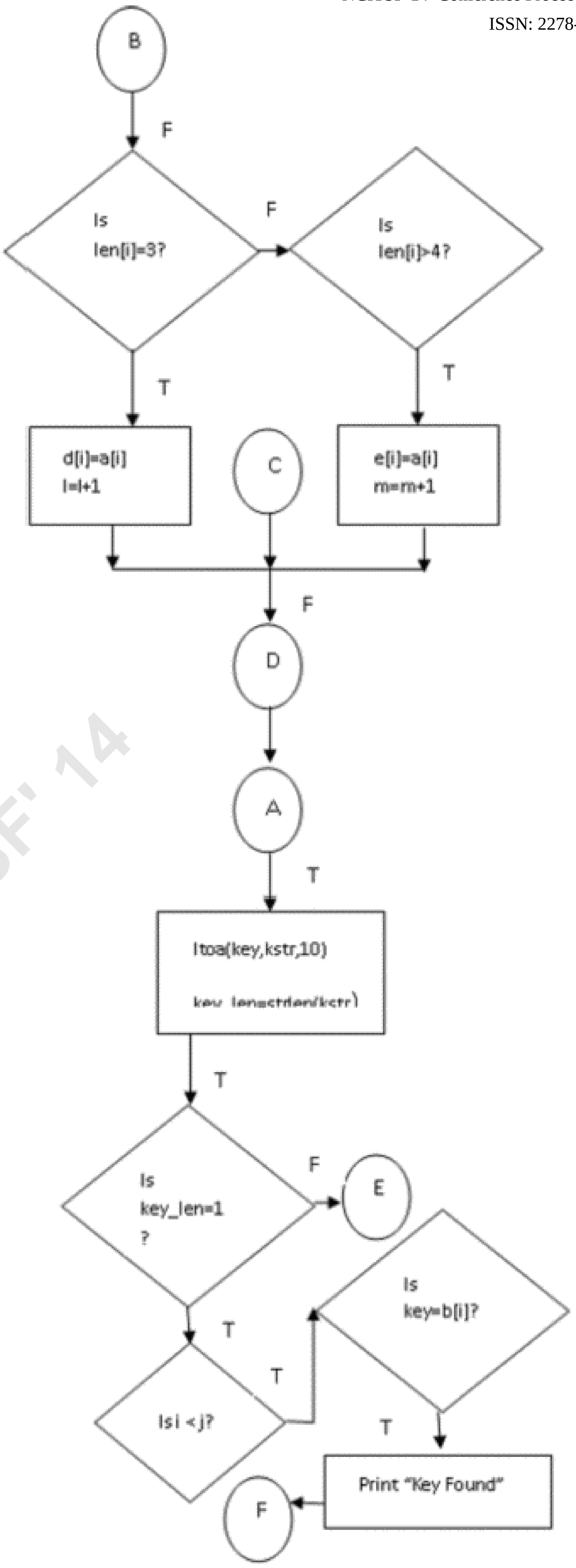
Step 2: Finding Length of Key element and based on size searching made on specific

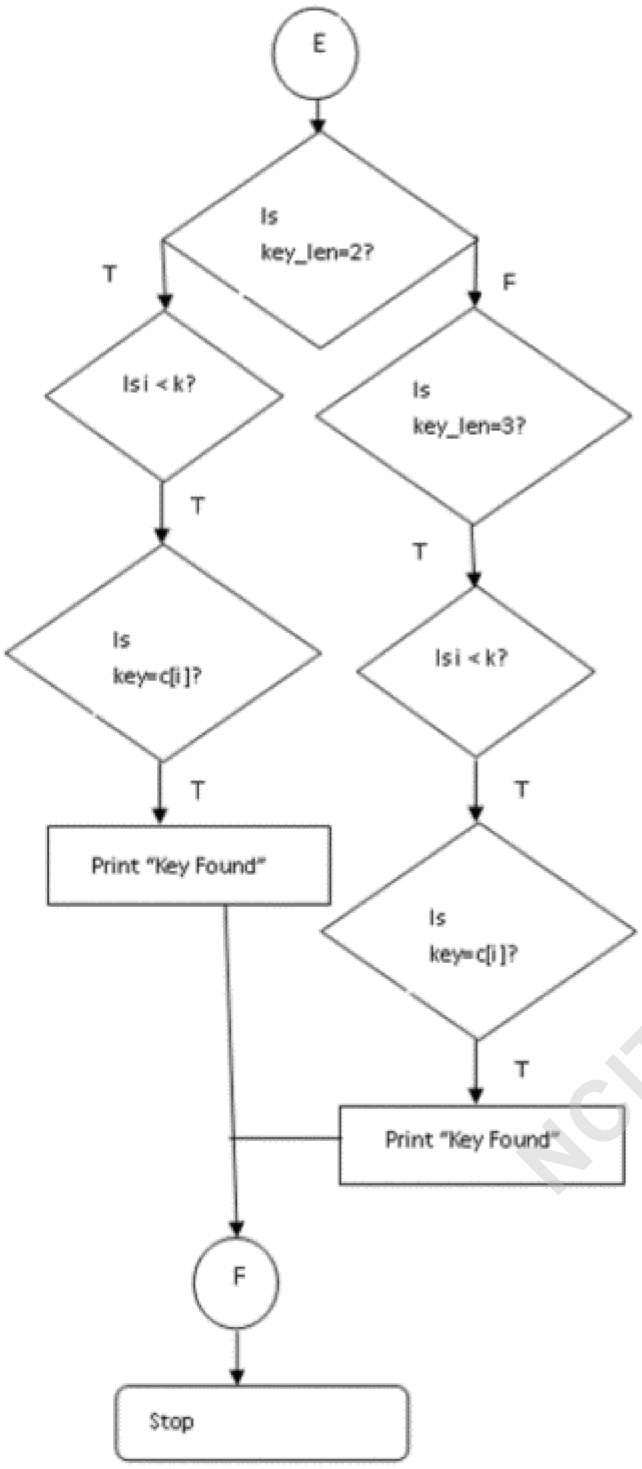
step 2.1: [Convert key element to String and find its string length]

```
        Itoa(key,kstr,10)
        key_len=strlen(kstr)
```

step 2.2: [key element length will be compared and execute only that block]

```
        if key_len= =1 then
                for i=0 to j in step of 1
                begin
                if key= =b[i] then
                printf ' Key element found'
                end if
                end for
        else if key_len= =2 then
                for i=0 to k in step of 1
                begin
                if key==c[i] then
                print 'key element found '
                end if
                end for
        else if key_len==3 then
                for i=0 to l in step of 1
                begin
                if key==d[i] then
                print 'Key element found'
                end if
                end for
        else
                for i=0 to m in step of 1
                begin
                if key==e[i] then
                print 'Key element found'
                end if
                end for
        end if
```





IV. CONCLUSION

In comparison with other searching algorithms, our new algorithm works better with respect to efficiency in the form of time and space. Here comparison is the basic operation which is executed repeatedly. Comparing with linear search which is brute force method, it takes less number of executions of basic operation. In the worst case efficiency of linear search, each and every element of input array is compared with the key element where as in our new algorithm comparison is made only with the group of elements which are of same size as of the size of the key element so at worst case element is matched at the end of that specific array only.
In comparison with binary search which is a divide and conquer method, elements of an input array is divided into sub groups recursively where as in our algorithm, elements of an input array is divided into sub group only once and searching will be made in specific group. In comparison with other searching methods, our algorithm works more efficient. In our algorithm, we are merging both brute force and divide and conquer method.

REFERENCES

[1] "The Design and analysis of algorithm" by Anany Levitin, 2nd Edition Pearson Education publication.
[2] "Design and analsis of computer algorithms" by AHO, Hopcroft, Ullman. Addision Wesley.
[3] "Data Structure" by Lipschutz, Tata Mcgraw hill Eductaion
[4] 4.https://www.princeton.edu/~achaney/tmve/wiki100k/docs/Binary_search_algorithm.html