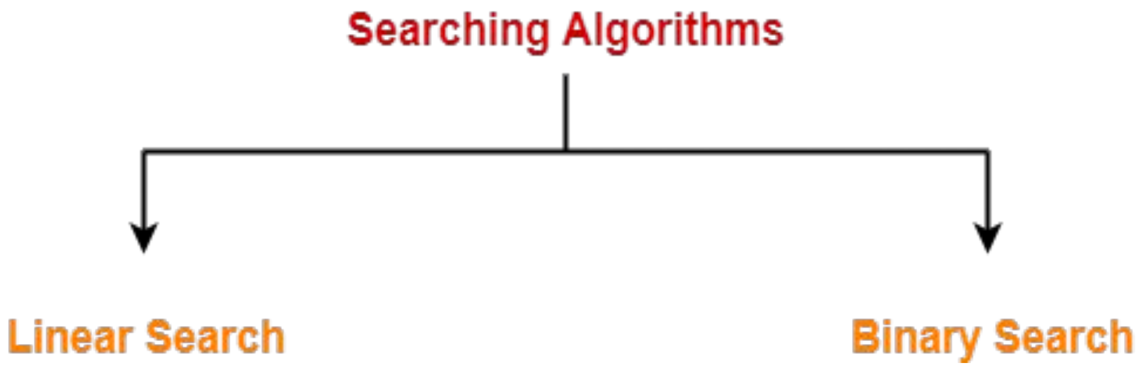
BINARY SEARCH

BinarySearch

Complexity of binary search

BINARY SEARCH



Binary Search

Session Learning Outcome-SLO-Solve problems using divide and conquer approaches

- Motivation of the topic

  Binary Search is one of the fastest searching algorithms.
  - It is used for finding the location of an element in a linear array.
  - It works on the principle of divide and conquer technique.

*Binary Search Algorithm can be applied only on **Sorted arrays**.*

- So, the elements must be arranged in-
  - Either ascending order if the elements are numbers.
  - Or dictionary order if the elements are strings.
- To apply binary search on an unsorted array,
  - First, sort the array using some sorting technique.
  - Then, use binary search algorithm.

- Binary search is an efficient searching method. While searching the elements using this method the most essential thing is that the elements in the array should be a sorted one.
- An element which is to be searched from the list of elements stored in the array A[0---n- 1] is called as Key element.
- Let A[m] be the mid element of array A. Then there are three conditions that need to be tested while searching the array using this method.
- They are given as follows
  - ❖ If key==A[m] then desired element is present in the list.
  - ❖ Otherwise if key <=A[m] then search the left sub list
  - ❖ Otherwise if Key>=A[m] then search the right sub list The following algorithm explains about binary search.

Recursive Binary search algorithm

Algorithm

Input: A array A[0...n-1] sorted in ascending orderand search key K.

Output: An index of array element which is equal to k

Low<- 0;high <-n-1

while low ≤ high do

    mid <- (low+high)/2
    if K=A[mid] return mid
    else if K,A[m] high <- mid-1
    else low<-mid+1

return

EXAMPLE FOR BINARY SEARCH

Step:1 Consider a list of elements sorted in array A as

0	1	2	3	4	5	6	
10	20	30	40	50	60	70	

Low high

The Search key element is key=60

Now to obtain middle element we will apply formula

mid=(low+high)/2

mid=(0+6)/2

mid=3

Then check A[mid]==key, A[3]=40

A[3]!=60 Hence condition failed

Then check key>A[mid],A[3]=40

60>A[3] Hence condition satisfied so search the Right Sublist

Step 2:

- The Right Sublist is

50	60	70

- Now we will again divide this list to check the mid element

4	5	6
50	60	70

Left sublist mid right sublist

- mid=(low+high)/2
- mid=(4+6)/2
- mid=5
- Check if A[mid]==key
- (i.e)A[5]==60.Hence condition is satisfied. The key element is present in position 5.
- **The number is present in the Array A[ ] at index position 5.**

Thus we can search the desired number from the list of the elements.

ANALYSIS

- The basic operation in binary search is comparison of search key with array elements.
- To analyze efficiency of binary search we must count the number of times the search gets compared with the array elements.
- The comparison is also called three way comparisons because the algorithm makes the comparison to determine whether key is smaller, equal to or greater than A[m].
- In this algorithm after one comparison the list of n elements is divided into n/2 sub list.
- The worst case efficiency is that the algorithm compares all the array elements for searching the desired element.
- In this method one comparison is made and based on the comparison array is divided each time in n/2 sub list.

- Hence worst case time complexity is given by

  Cworst(n) = Cworst (n/2) + 1 for n > 1

  Time required to Compare left or Right sub list

  one comparison made with middle element

  Also Cworst(1) =1

- But as we consider the rounded down values when array gets divided the above equation can be written as

  Cworst (n) = Cworst (n / 2) +1 for n>1

  Cworst (1)= 1

- We can analyse the best case , Worst case and Average case . The time complexity of binary search is given as follows

Best case	Average case	Worst Case
$\theta(1)$	$\theta(logn)$	$\theta(logn)$

- In conclusion we are now able to completely describe the computing time of binary search by giving formulas that describe best, average and worst cases
- Successful searches unsuccessful searches

$\theta(1)$ $\theta(logn)$ $\theta(logn)$ $\theta(logn)$

- best average worst best, average, worst

- **Advantages of Binary search:**
  - Binary search is an optimal searching algorithm using which we can search the desired element very efficiently
- **Disadvantages of binary Search :**
  - This Algorithm requires the list to be sorted . Then only this method is applicable
- **Applications of binary search:**
  - The binary search is an efficient searching method and is used to search desired record from database
  - For solving with one un known this method is used

Summary:

- Binary Search time complexity analysis is done below-
  - In each iteration or in each recursive call, the search gets reduced to half of the array.
  - So for n elements in the array, there are $\log_2 n$ iterations or recursive calls.
- **Time Complexity of Binary Search Algorithm is $O(\log_2 n)$.**
  - Here, n is the number of elements in the sorted linear array.

Home assignment:

- Search the Element 15 from the given array using Binary Search Algorithm.

3	10	15	20	35	40	60
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]

Binary Search Example