

Experiment - 2

Aim:- To implement Linear Search and Binary Search and analyse its time complexity.

Software used:- Turbo C++.

Theory:-

Linear Search:- It is a method for finding a target value within a list. It sequentially checks each element of the list for the target value until a match is found or until all the elements have been searched.

Binary Search:- Also known as half-interval search, logarithmic search, or binary chop, is a search algo. that finds the position of a target value within a sorted array. It compares the target value to the middle elements of the array.

Algo:-

Linear Search	Inst	rough time taken
for (i=0 ; i < n ; i++)	C ₁	
{		
if (array[i] == n)	C ₂	
return i;	C ₃	
}		

Best case:- When the number to be searched found at 1st location $O(1)$

Worst case:- When the number to be searched is not found, then we check the whole array of size n .
 $O(n)$

Average case:- The no. It is the avg. of all the cases, where no. can be found at 1st, 2nd - - - n th position.

$$\frac{1 + 2 + 3 + \dots + n}{n} = \frac{n(n+1)}{2n} \approx \frac{n}{2}$$

Binary Search:-

```

while (beg <= last)
{
    mid = (beg + last) / 2
    if (array[mid] == m)
        { return i; }
    else if (m < array[mid])
        last = mid - 1
    else if (m > array[mid])
        beg = mid + 1
}

```

Best case:- no. is found, at the middle of array $O(1)$

Worst case:- no. is not in the array. $\log_2(n)$

$$T(n) = T(n/2)$$

Time Complexity

1st Comparison	Approx No. of Items left
1st	$n/2$
2nd	$n/4$
3rd	$n/8$
!	
i	$n/2^i$

The list will end up when it has only one item left

$$\therefore \frac{n}{2^i} = 1$$

$$n = 2^i$$

$$\log_2 n = i$$

\therefore Time complexity is $O(\log n)$

Result:- Successfully understand the linear and binary search and analysed their time complexity.