→ Palash Mishra.

2018172

Date __/__/____
Page _____

Solution 1

Here time complexity —

for linear search is $O(n)$
for inner loop is $O(\log n)$
for outer loop is $O(n)$

Thus by adding all the time complexity.

∴ Time complexity is $O(n * \log n)$.

2018172

Date __/__/____
Page _____

Solution 2

Recursive function for Binary Search

```
Int BinarySearch (int[] A, int low, int high, int x)
{
  if (low > high){
    return -1; }
  int mid = (low + high)/2;
  if (x == A[mid]){
    return mid; }
  else if (x < A[mid]){
   return BinarySearch (A, low, mid-1, x); }
  else {
    return BinarySearch (A, mid+1, high, x); }
}
```

Suppose we have a total time T(n), In recursive Binary search our trend is going half like $n \rightarrow n/2 \rightarrow n/4 \rightarrow n/8$ ...

$\therefore$ Recurrence relation $\rightarrow$ $\boxed{T(n) = T(n/2) + c}$

By backward Substitution method.

We have recurrence relation

$T(n) = T(n/2) + c$ ——— (1)

$T(n/2) = T(n/4) + c$ → for $n = n/2$ ——— (2)

$T(n/4) = T(n/8) + c$ → for $n = n/4$ ——— (3)

Date ___/___/___
Page _____

Substituting eq ② in eq ①

$T(n) = T(n/4) + c + c \longrightarrow ④$

Substituting eq ③ in eq ④

$T(n) = T(n/8) + c + c + c$

$T(n) = T(n/2^3) + 3c$

Similarly if we go for k no. of steps

$T(n) = T(n/2^k) + kc$

To terminate this function we know that $T(1) = 1$

$\therefore \boxed{\frac{n}{2^k} = 1 \quad \text{i.e} \quad n = 2^k \quad \text{ie} \quad k = \log n.}$ ← after taking log on both sides

$\therefore T(n) = T(n/n) + kc$

$T(n) = T(1) + kc$

$T(n) = 1 + kc$

$T(n) = 1 + \log n\, c$

$T(n) = O(\log_2 n)$

Solution-03

Recursive program for linear Search.

```
#include<iostream>
using namespace Std;
int Search(int arr[], int size, int key)
{
   int Temp;
   Size--;
   If(Size.>=0)
     {
        If(arr[size]==Key)
        Return size;
        else
        Temp = Search(arr,size,key)
     }
    else
    return -1;
   Return Temp;
}
int main(void){
int arr[]={18,34,09,58,6}
int size= sizeof(arr)/ Size of arr[0];
int key=3;
int Index;
   Index= elem Search(arr,Size,key);
if(Index !=-1)
Cout<<"Element"<<Key<<"is present at index"<<index<<endl;
else
cout<<"Element"<<key<<" is not present";<<endl;
Return 0; }
```

Recursive function for linear Search.

Date ___/___/____
Page _____

Now, as we know, recurrence relation for the linear search recursive algorithm is

$$\boxed{T(n) = T(n-1) + C}$$

using backward substitution

$$T(n) = T(n-1) + C \quad \text{— (1)}$$
$$T(n-1) = T(n-2) + C \quad \text{— (2)}$$
$$T(n-2) = T(n-3) + C \quad \text{— (3)}$$
$$\vdots$$
$$T(n-i) = T(n-(i+1)) + C \quad \text{— (4)}$$

Now we back substitute the equation we get

$$T(n) = T(n-2) + 2C$$
$$T(n) = T(n-3) + 3C$$
$$\vdots$$
$$T(n) = T(n-i) + iC \longrightarrow (5)$$

as we know $T(1) = 1$

then $n - i = 1$

$n - 1 = i$

$\boxed{i = n-1}$ putting value in eq.

we get

$$T(n) = 1 + (n-1)C$$
$$T(n) = 1 + (nC - C)$$
$$T(n) = nC - C + 1$$
$$T(n) = O(nC - C + 1) \quad \boxed{T(n) = O(n)}$$ as constant be eleminated.

Date ___/___/____
Page _____

Solution 04

```
#include<iostream>
using namespace std;

int Binary_Search (int A[], int low, int high, int key)
{
    while (low<=high)
    {
      int mid=(low+high)/2;
      if (A[mid]<key)
      {
        low=mid+1;
      }
      else if (A[mid]>key)
      {
        high=mid-1;
      }
      else
      {
        return mid+1;
      }
    }
    return -1;
}
int main()
{
  int low=0, high=10, key;
  int A[]={20, 30, 45, 77, 89, 90, 94, 99, 100, 150};
  cout<<"Enter the key :";
  cin>>key;
```

Date ___/___/____
Page _____

```
cout<< Binary_Search(A, low, high, key);
return 0;
}
```

(i) Enter the key : 10
-1

(ii) Enter the key : 152
-1

(iii) Enter the key : 45
3

(iv) Enter the key : 89
5

Best Case: The best case of binary search occurs when the element or value which we need to find is present at the middle. In this case the best case will be arrise when key=90

Average Case: Average case lies in b/w worst case and best case it is the key element can be found in less the worst iteration but greater than best case iteration
In this case Average case arrise when key = 77 or 99.

Worst case: In this it takes maximum iteration or time, this happens when element is found after all n iterations In this cast the worst case arrise when key = 99.