
Daffodil International University

LAB TASK

COUSE CODE : CSE214

COURSE TITLE : ALGORITHM LAB

TOPIC :

LINEAR SEARCH IMPLEMENTATION , ANALYSIS , WORST CASE , BEST CASE, AVERAGECASE &

BUBBLE SORT VISUALIZATION .

SUBMITTED TO :

MR.SUBROTO NAG PINKU

DEPARTMENT OF CSE,

DAFFODIL INTERNATIONAL UNIVERSITY .

SUBMITTED BY :

SHAZID NAWAS SHOVON

ID : 191-15-12929

SECTION : D-14 (S)

DEPARTMENT OF CSE ,

DAFFODIL INTERNATIONAL UNIVERSITY .

# Linear Search

## Implementation:

```
intlinearsearch(intarr[],intn,int x)
{
inti,index=-1;
for(i=0; i<n; i++)
   {
      if(arr[i]==x)
      {
         index = i;
         break;
      }
   }
return index;
}
```

## Analysis:

| 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|

let consider this array having 5 elements that means n=5. We want to search the value x = 1

At the begging of the iteration when i=0 which is less then n, so it will enter in the loop. After checking the condition if the value is found then it will break and return the index. But in this case the value we want to search that is in the last positon or last index of array which is 4.

For that reason, this loop will be executed for 5 times as n=5.

## Worst Case:

If there are n elements and the value either exist in the last positon n-1 or not exist, the loop will run for n times.

Therefore, the complexity would be O(n).

## Best Case:

| 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|

If x = 5 which is in the begging of the array, the loop will run for only 1 time.

Therefore, for best case, the complexity would be O(1).

**Average Case:**

We know average case = All possible case time / No. of case (Till n)

$$= \frac{1+2+3\ldots\ldots+n}{n}$$

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

$$= \frac{n+1}{2}$$

Ignoring the constant co-efficient, we can say that the complexity in average case of linear search is O(n).

**<u>Bubble Sort Visualization</u> :**

| 3 | 1 | 2 | 4 | 5 |
|---|---|---|---|---|

If we bubble sort this array the visualization will be

the loop will check 1st array index & 2nd array. 1st >2nd index will swap. So we will get 3 > 1.

| 1 | 3 | 2 | 4 | 5 |
|---|---|---|---|---|

Again it will swap 3rd > 2nd array index

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|

Then the loop will check the 3rd and 4th index of array like the above . If the index in greater then it will swap otherwise it will stop .

Therefore the final visualization of sort will be

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|