Practical 03

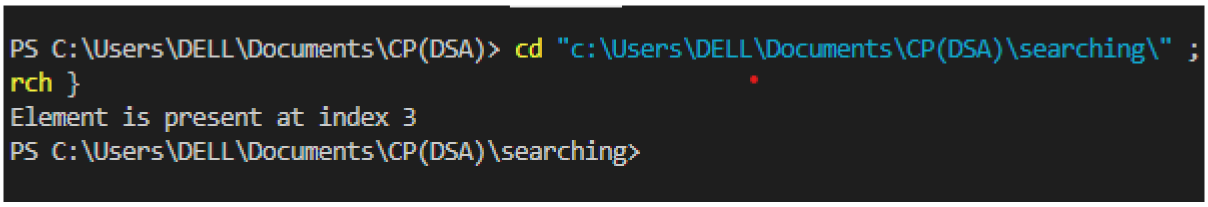
Searching algorhithams:

1)linear search

```
// C++ code to linearly search x in arr[]. If x

// is present then return its location, otherwise

// return -1


#include <iostream>

using namespace std;


int search(int arr[], int N, int x)

{

        int i;

        for (i = 0; i < N; i++)

                if (arr[i] == x)

                        return i;

        return -1;

}


// Driver's code

int main(void)

{

        int arr[] = { 2, 3, 4, 10, 40 };

        int x = 10;
```

```
    int N = sizeof(arr) / sizeof(arr[0]);

    // Function call
    int result = search(arr, N, x);
    (result == -1)
        ? cout << "Element is not present in array"
        : cout << "Element is present at index " << result;
    return 0;
}
```

Output:



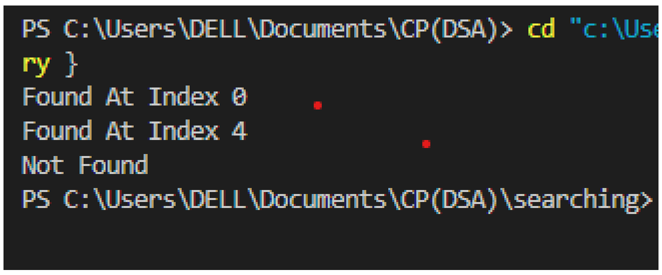
```
PS C:\Users\DELL\Documents\CP(DSA)> cd "c:\Users\DELL\Documents\CP(DSA)\searching\" ;
rch }
Element is present at index 3
PS C:\Users\DELL\Documents\CP(DSA)\searching>
```

2)BINARY SERACH

```
#include <bits/stdc++.h>
#include <iostream>
using namespace std;

int binarySearch(vector<int> v, int To_Find)
{
    int lo = 0, hi = v.size() - 1;
    int mid;
    // This below check covers all cases , so need to check
    // for mid=lo-(hi-lo)/2
```

```
        while (hi - lo > 1) {
                int mid = (hi + lo) / 2;
                if (v[mid] < To_Find) {
                        lo = mid + 1;
                }
                else {
                        hi = mid;
                }
        }
        if (v[lo] == To_Find) {
                cout << "Found"
                        << " At Index " << lo << endl;
        }
        else if (v[hi] == To_Find) {
                cout << "Found"
                        << " At Index " << hi << endl;
        }
        else {
                cout << "Not Found" << endl;
        }
}

int main()
{
        vector<int> v = { 1, 3, 4, 5, 6 };

        int To_Find = 1;

        binarySearch(v, To_Find);

        To_Find = 6;

        binarySearch(v, To_Find);

        To_Find = 10;

        binarySearch(v, To_Find);

        return 0;

}
```

Output:

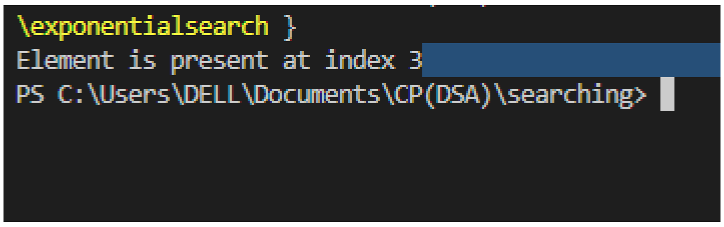


```
PS C:\Users\DELL\Documents\CP(DSA)> cd "c:\Use
ry }
Found At Index 0
Found At Index 4
Not Found
PS C:\Users\DELL\Documents\CP(DSA)\searching>
```

3)Exponential Search

```
// C++ program to find an element x in a
// sorted array using Exponential search.
#include <bits/stdc++.h>
using namespace std;

int binarySearch(int arr[], int, int, int);

// Returns position of first occurrence of
// x in array
int exponentialSearch(int arr[], int n, int x)
{
        // If x is present at first location itself
        if (arr[0] == x)
                return 0;

        // Find range for binary search by
        // repeated doubling
        int i = 1;
        while (i < n && arr[i] <= x)
                i = i*2;

        // Call binary search for the found range.
        return binarySearch(arr, i/2,
                                                min(i, n-1), x);
}

// A recursive binary search function. It returns
// location of x in given array arr[l..r] is
// present, otherwise -1
int binarySearch(int arr[], int l, int r, int x)
{
        if (r >= l)
        {
                int mid = l + (r - l)/2;

                // If the element is present at the middle
                // itself
                if (arr[mid] == x)
                        return mid;

                // If element is smaller than mid, then it
                // can only be present n left subarray
                if (arr[mid] > x)
                        return binarySearch(arr, l, mid-1, x);

                // Else the element can only be present
                // in right subarray
                return binarySearch(arr, mid+1, r, x);
        }

        // We reach here when element is not present
        // in array
        return -1;
}

// Driver code
int main(void)
{
int arr[] = {2, 3, 4, 10, 40};
int n = sizeof(arr)/ sizeof(arr[0]);
int x = 10;
int result = exponentialSearch(arr, n, x);
(result == -1)? cout <<"Element is not present in array"
                                        : cout <<"Element is present at index " << result;
return 0;
}
```

Output:



```
\exponentialsearch }
Element is present at index 3
PS C:\Users\DELL\Documents\CP(DSA)\searching>
```

Sorting algorhithams

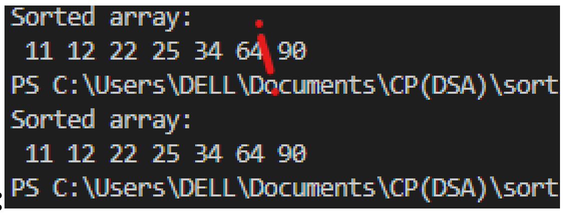
1) Bubble Sort Algorithm

```
// Optimized implementation of Bubble sort

#include <bits/stdc++.h>

using namespace std;



// An optimized version of Bubble Sort

void bubbleSort(int arr[], int n)

{

int i, j;

bool swapped;

for (i = 0; i < n-1; i++)

{

          swapped = false;

          for (j = 0; j < n-i-1; j++)
```

```
        {
                if (arr[j] > arr[j+1])
                {
                swap(arr[j], arr[j+1]);
                swapped = true;
                }
        }

        // IF no two elements were swapped
        // by inner loop, then break
        if (swapped == false)
                break;
}
}

// Function to print an array
void printArray(int arr[], int size)
{
        int i;
        for (i = 0; i < size; i++)
                cout <<" "<< arr[i];
}

// Driver program to test above functions
```

```
int main()

{

        int arr[] = {64, 34, 25, 12, 22, 11, 90};

        int N = sizeof(arr)/sizeof(arr[0]);

        bubbleSort(arr, N);

        cout <<"Sorted array: \n";

        printArray(arr, N);

        return 0;

}
```

Output:



```
Sorted array:
 11 12 22 25 34 64 90
PS C:\Users\DELL\Documents\CP(DSA)\sort
Sorted array:
 11 12 22 25 34 64 90
PS C:\Users\DELL\Documents\CP(DSA)\sort
```

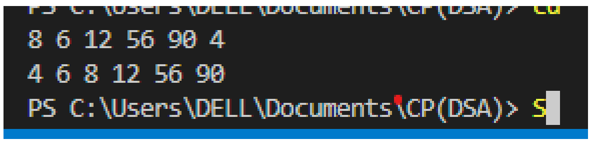
:

2)Selection sort:

```
#include <iostream>

using namespace std;

void print(int arr[],int n){

   for(int i=0; i<n; i++){

      cout<<arr[i]<<" ";

   }

}

void selection(int arr[], int n){

   for(int i=0; i<n-1;i++){

      int indmin=i;
```

```
    for(int j=i+1;j<n;j++)
    {
        if(arr[j]< arr[indmin]){
            indmin=j;
        }
        int temp=arr[i];
        arr[i]=arr[indmin];
        arr[indmin]=temp;
    }
}

int main(){
  int arr[6]={8,6,12,56,90,4};
  int size=6;
  print(arr,size);
  cout<<endl;
  selection(arr,size);
    print(arr,size);
}
```

Output:



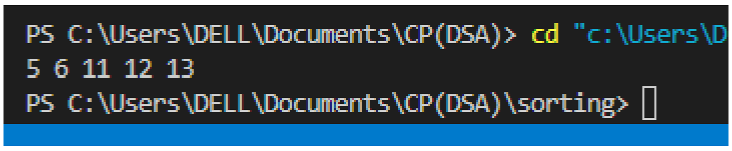
```
PS C:\Users\DELL\Documents\CP(DSA)> cd
8 6 12 56 90 4
4 6 8 12 56 90
PS C:\Users\DELL\Documents\CP(DSA)> S
```

3)Insertion sort

```
// C++ program for insertion sort


#include <bits/stdc++.h>

using namespace std;

// Function to sort an array using

// insertion sort

void insertionSort(int arr[], int n)

{

	int i, key, j;

	for (i = 1; i < n; i++)

	{

		key = arr[i];

		j = i - 1;

		// Move elements of arr[0..i-1],

		// that are greater than key, to one

		// position ahead of their

		// current position

		while (j >= 0 && arr[j] > key)

		{

			arr[j + 1] = arr[j];

			j = j - 1;

		}

		arr[j + 1] = key;
```

```
        }
}

// A utility function to print an array
// of size n
void printArray(int arr[], int n)
{
        int i;
        for (i = 0; i < n; i++)
                cout << arr[i] << " ";
        cout << endl;
}
// Driver code
int main()
{
        int arr[] = { 12, 11, 13, 5, 6 };
        int N = sizeof(arr) / sizeof(arr[0]);
        insertionSort(arr, N);
        printArray(arr, N);
        return 0;
```

output:



```
PS C:\Users\DELL\Documents\CP(DSA)> cd "c:\Users\D
5 6 11 12 13
PS C:\Users\DELL\Documents\CP(DSA)\sorting> 
```