IPS4

ITA5002

Problem solving with Data structures and Algorithms

Submitted by: -

Moeenul Islam

Submitted to: -

Dr Vijayan E

School of Computer Science and Engineering

Vellore Institute of Technology, Vellore

22 December, 2021

Sorting

1.Insertion Sort

```cpp
//Moeenul Islam
//Insertion Sort
#include<iostream>
using namespace std;

// insertionSort function
void insertionSort(int[], int);
// printArray function
void printArray(int[], int len);
main(){
    int arr[] = {55, 89, 112, 23, 11, 62};
    int len = sizeof(arr)/sizeof(arr[0]);

    insertionSort(arr, len);
    cout<<"Sorted array is: \n";
    printArray(arr, len); //prints the sorted array
}


void insertionSort(int arr[], int len){
    for (int i = 0; i < len - 1; i++){
        for (int j = i + 1; j >0; j--){
            if (arr[j] < arr[j - 1])
                swap(arr[j], arr[j-1]);
            else
                break;
        }
    }
}
void printArray(int arr[], int len){
    for (int i = 0; i < len; i++){
```

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

}
```

```
moeen@Knightmare MINGW64 ~/Desktop/DSA/CAT-2
$ g++ insertionSort.cpp

moeen@Knightmare MINGW64 ~/Desktop/DSA/CAT-2
$ ./a
Sorted array is:
11 23 55 62 89 112
moeen@Knightmare MINGW64 ~/Desktop/DSA/CAT-2
$
```

2. Selection Sort

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

void selectionSort(int[], int);
void printArray(int[], int);
main()
{
    int arr[] = {500, 423, 663, 123, 229, 543};
    int len = sizeof(arr) / sizeof(arr[0]);
    cout << "Sorted array: after selection sort:\n";
    selectionSort(arr, len);
    printArray(arr, len);
}
```

```
// selection sort
void selectionSort(int arr[], int len)
{
    int minIndex;
    for (int i = 0; i < len - 1; i++)
    {
        minIndex = i;
        for (int j = i+1; j < len; j++){
            if(arr[j] < arr[minIndex])
                minIndex = j;

            swap(arr[minIndex], arr[i]);
        }
    }
}

// Printing the array
void printArray(int arr[], int len)
{
    for (int i = 0; i < len; i++)
    {
        cout << arr[i] << " ";
    }
}
```

```
moeen@Knightmare MINGW64 ~/Desktop/DSA/CAT-2
$ g++ selectionSort.cpp

moeen@Knightmare MINGW64 ~/Desktop/DSA/CAT-2
$ ./a
Sorted array: after selection sort:
123 423 500 543 229 663
moeen@Knightmare MINGW64 ~/Desktop/DSA/CAT-2
$
```

3.Quick Sort

```cpp
//Moeenul Islam
#include <bits/stdc++.h>
using namespace std;

void swap(int *a, int *b);
int partition(int arr[], int low, int high);
void quickSort(int arr[], int low, int high);
void printArray(int arr[], int size);
// Driver Code
int main()
{
    int arr[] = {110, 89, 54, 223, 543};
    int n = sizeof(arr) / sizeof(arr[0]);
    quickSort(arr, 0, n - 1);
    cout << "Sorted array after quick sort: \n";
    printArray(arr, n);
    return 0;
}
```

```c
// A  function to swap two elements
void swap(int *a, int *b)
{
    int t = *a;
    *a = *b;
    *b = t;
}


int partition(int arr[], int low, int high)
{
    int pivot = arr[high]; // pivot
    int i = (low - 1);      // Index of smaller element
and indicates the right position of pivot found so far

    for (int j = low; j <= high - 1; j++)
    {
        // If current element is smaller than the pivot
        if (arr[j] < pivot)
        {
            i++; // increment index of smaller element
            swap(&arr[i], &arr[j]);
        }
    }
    swap(&arr[i + 1], &arr[high]);
    return (i + 1);
}

/* The main function that implements QuickSort
arr[] --> Array to be sorted,
low --> Starting index,
high --> Ending index */
```

```
void quickSort(int arr[], int low, int high)
{
    if (low < high)
    {
        /* pi is partitioning index, arr[p] is now
        at right place */
        int pi = partition(arr, low, high); //partition
index

        // Separately sort elements before
        // partition and after partition
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}

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

```
moeen@Knightmare MINGW64 ~/Desktop/DSA/CAT-2
$ g++ quickSort.cpp

moeen@Knightmare MINGW64 ~/Desktop/DSA/CAT-2
$ ./a
Sorted array after quick sort:
54 89 110 223 543

moeen@Knightmare MINGW64 ~/Desktop/DSA/CAT-2
$
```

4. Bubble sort

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

void swap(int *xp, int *yp)
{
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}

// A function to implement bubble sort
void bubbleSort(int arr[], int n)
{
    int i, j;
    for (i = 0; i < n-1; i++)

    // Last i elements are already in place
    for (j = 0; j < n-i-1; j++)
```

```
        if (arr[j] > arr[j+1])
            swap(&arr[j], &arr[j+1]);
}

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

// Driver code
int main()
{
    int arr[] = {64, 34, 25, 12, 22, 11, 90};
    int n = sizeof(arr)/sizeof(arr[0]);
    bubbleSort(arr, n);
    cout<<"Sorted array after bubble sort \n";
    printArray(arr, n);
    return 0;
}
```

```
moeen@Knightmare MINGW64 ~/Desktop/DSA/CAT-2
$ g++ bubbleSort.cpp

moeen@Knightmare MINGW64 ~/Desktop/DSA/CAT-2
$ ./a
Sorted array after bubble sort
11 12 22 25 34 64 90
```

5.Shell Sort

```cpp
// C++ implementation of Shell Sort
#include <iostream>
using namespace std;

/* function to sort arr using shellSort */
int shellSort(int arr[], int n)
{
    // Start with a big gap, then reduce the gap
    for (int gap = n/2; gap > 0; gap /= 2)
    {

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

            int temp = arr[i];

            int j;
            for (j = i; j >= gap && arr[j - gap] > temp;
j -= gap)
                arr[j] = arr[j - gap];
```

```
            // put temp (the original a[i]) in its
correct location
            arr[j] = temp;
        }
    }
    return 0;
}

void printArray(int arr[], int n)
{
    for (int i=0; i<n; i++)
        cout << arr[i] << " ";
}

int main()
{
    int arr[] = {12, 34, 54, 2, 3}, i;
    int n = sizeof(arr)/sizeof(arr[0]);

    cout << "Array before sorting: \n";
    printArray(arr, n);

    shellSort(arr, n);

    cout << "\nArray after sorting: \n";
    printArray(arr, n);

    return 0;
}
```

```
$ g++ shellSort.cpp

moeen@Knightmare MINGW64 ~/Desktop/DSA/CAT-2
$ ./a
Array before sorting:
12 34 54 2 3
Array after sorting:
2 3 12 34 54
moeen@Knightmare MINGW64 ~/Desktop/DSA/CAT-2
$
```

6. Merge Sort

```
//Moeenul Islam
#include<iostream>
using namespace std;
void swapping(int &a, int &b) {      //swap the content of
a and b
   int temp;
   temp = a;
   a = b;
   b = temp;
}
void display(int *array, int size) {
   for(int i = 0; i<size; i++)
      cout << array[i] << " ";
   cout << endl;
}
void merge(int *array, int l, int m, int r) {
   int i, j, k, nl, nr;
   //size of left and right sub-arrays
   nl = m-l+1; nr = r-m;
```

```
   int larr[nl], rarr[nr];
   //fill left and right sub-arrays
   for(i = 0; i<nl; i++)
      larr[i] = array[l+i];
   for(j = 0; j<nr; j++)
      rarr[j] = array[m+1+j];
   i = 0; j = 0; k = l;
   //marge temp arrays to real array
   while(i < nl && j<nr) {
      if(larr[i] <= rarr[j]) {
         array[k] = larr[i];
         i++;
      }else{
         array[k] = rarr[j];
         j++;
      }
      k++;
   }
   while(i<nl) {        //extra element in left array
      array[k] = larr[i];
      i++; k++;
   }
   while(j<nr) {      //extra element in right array
      array[k] = rarr[j];
      j++; k++;
   }
}
void mergeSort(int *array, int l, int r) {
   int m;
   if(l < r) {
      int m = l+(r-l)/2;
      // Sort first and second arrays
```

```
        mergeSort(array, l, m);
        mergeSort(array, m+1, r);
        merge(array, l, m, r);
    }
}
int main() {
    int n;
    cout << "Enter the number of elements: ";
    cin >> n;
    int arr[n];      //create an array with given number of
elements
    cout << "Enter elements:" << endl;
    for(int i = 0; i<n; i++) {
        cin >> arr[i];
    }
    cout << "Array before Sorting: ";
    display(arr, n);
    mergeSort(arr, 0, n-1);      //(n-1) for last index
    cout << "Array after Sorting: ";
    display(arr, n);
}
```

```
moeen@Knightmare MINGW64 ~/Desktop/DSA/CAT-2
$ ./a
Enter the number of elements: 6
Enter elements:
55 66 88 99 12 36 19 0 2 6
Array before Sorting: 55 66 88 99 12 36
Array after Sorting: 12 36 55 66 88 99
```

7. Bucket Sort

```cpp
// Moeenul Islam
// array using bucket sort
#include <algorithm>
#include <iostream>
#include <vector>
using namespace std;

// Function to sort arr[] of
// size n using bucket sort
void bucketSort(float arr[], int n)
{

    // 1) Create n empty buckets
    vector<float> b[n];

    // 2) Put array elements
    // in different buckets
    for (int i = 0; i < n; i++) {
        int bi = n * arr[i]; // Index in bucket
        b[bi].push_back(arr[i]);
    }

    // 3) Sort individual buckets
    for (int i = 0; i < n; i++)
        sort(b[i].begin(), b[i].end());

    // 4) Concatenate all buckets into arr[]
    int index = 0;
    for (int i = 0; i < n; i++)
        for (int j = 0; j < b[i].size(); j++)
            arr[index++] = b[i][j];
```

```
}

/* Driver program to test above function */
int main()
{
    float arr[]
        = { 0.897, 0.565, 0.656, 0.1234, 0.665, 0.3434 };
    int n = sizeof(arr) / sizeof(arr[0]);
    bucketSort(arr, n);

    cout << "Sorted array is \n";
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";
    return 0;
}
```

```
moeen@Knightmare MINGW64 ~/Desktop/DSA/CAT-2
$ g++ bucketSort.cpp

moeen@Knightmare MINGW64 ~/Desktop/DSA/CAT-2
$ ./a
Sorted array is
0.1234 0.3434 0.565 0.656 0.665 0.897
moeen@Knightmare MINGW64 ~/Desktop/DSA/CAT-2
$
```

Searching

1.Linear search

```
//Moeenul Islam

#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 code
int main(void)
{
    int arr[] = { 2, 3, 4, 10, 40 };
    int x;
    cout<<"Enter number to search:";
    cin>>x;
    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;
}
```

```
moeen@Knightmare MINGW64 ~/Desktop/DSA/CAT-2
$ g++ linearSearch.cpp

moeen@Knightmare MINGW64 ~/Desktop/DSA/CAT-2
$ ./a
Enter number to search:10
Element is present at index 3
```

2. BinarySearch

```cpp
// Moeenul Islam
#include <bits/stdc++.h>
using namespace std;


int binarySearch(int arr[], int l, int r, int x)
{
    while (l <= r) {
        int m = l + (r - l) / 2;

        // Check if x is present at mid
        if (arr[m] == x)
            return m;

        // If x greater, ignore left half
        if (arr[m] < x)
            l = m + 1;

        // If x is smaller, ignore right half
        else
```

```
            r = m - 1;
    }

    // if we reach here, then element was
    // not present
    return -1;
}

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

```
moeen@Knightmare MINGW64 ~/Desktop/DSA/CAT-2
$ g++ BinarySearch.cpp

moeen@Knightmare MINGW64 ~/Desktop/DSA/CAT-2
$ ./a
Enter number to search:40
Element is present at index 4
```