University of Information Technology and Sciences (UITS)

Department of Information Technology

LAB-2:

IT-214 : Algorithm Lab

Sorting Algorithm

Submitted To:

Sumaiya Akhtar Mitu
Lecturer,
Department of IT, UITS

Submitted By:

Name:Nazmul Zaman
Student ID:2014755055
Department of IT, UITS

April 18, 2022

Contents

1 Abstraction

In this lab report we learn how to worked sorting algorithm and use of sorting algorithm and why we use sorting algorithm and best way for use . A sorting algorithm is a method for reorganizing a large number of items into a specific order, such as alphabetical, highest-to-lowest value or shortest-to-longest distance. Sorting algorithms take lists of items as input data, perform specific operations on those lists and deliver ordered arrays as output. 1.Bubble Sort 2.Selection Sort 3.Selection Sort

2 Introduction

1.Bubble Sort:Bubble sort is a basic algorithm for arranging a string of numbers or other elements in the correct order. The method works by examining each set of adjacent elements in the string, from left to right, switching their positions if they are out of order.Bubble Sort is an easy-to-implement, stable sorting algorithm with a time complexity of $O(n^2)$ in the average and worst cases – and O(n) in the best case

2.Insertion Sort: Insertion sort is a simple sorting algorithm that works similar to the way you sort playing cards in your hands. The array is virtually split into a sorted and an unsorted part. Values from the unsorted part are picked and placed at the correct position in the sorted part.nsertion Sort is an easy-to-implement, stable sorting algorithm with time complexity of $O(n^2)$ in the average and worst case, and O(n) in the best case

3.Selection Sort: The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from unsorted part and putting it at the beginning. The algorithm maintains two subarrays in a given array.In computer science, selection sort is an in-place comparison sorting algorithm. It has an O(n2) time complexity,

1) The subarray which is already sorted. 2) Remaining subarray which is unsorted.

3 Source Code

4 BUBBLE SORT

```
1 //Q:  Bubble  Sort
2
3 //NAZMUL ZAMAN-Bsc in (IT)
4
5 #include<bits/stdc++.h>
6 using namespace std;
```

```
7
8  void swap(int *x,int *y) // this funtion for swap minimum value and
       used pointer
9  {
10     int temp =  *x;
11    *x =  *y;
12     *y = temp;
13     int  flag=1; // use (flag) for less time complexity
14 }
15
16
17 void Bubble_Sort(int arr[],int n)  // base function for iteration
     and compair
18 {int i,j;
19     for(i=0;i<n-1;i++)
20     {
21         for(j=0;j<n-i-1;j++)
22         {
23             if(arr[j]>arr[j+1])
24             swap(&arr[j],&arr[j+1]);
25         }
26     }
27 }
28
29
30 void printArray(int arr[],int n)  // printing funtion
31 {
32   int flag;
33     for(int i=0;i<n;i++)
34     {
35         cout<<arr[i]<<" ";
36
37     if(!flag)
38 {
39 break;
40 }
41
42     }
43
44 }
45
46 int main()
47 {
48      cout<<"Enter the size of an array : ";
49     int n;
50     cin>>n;
51     int arr[n];
52
53
54     for(int i=0;i<n;i++)
55     {
```

```
56          cout<<"Enter the value of index "<<i<<" is : ";
57          cin>>arr[i];
58
59      }
60       cout<<"Before sorted array is : \n";
61         for(int i=0;i<n;i++)
62         {
63          cout<<arr[i]<<" ";
64         }
65
66      Bubble_Sort(arr,n);
67      cout<<"\nAfter Sorted array : "<<endl;     cout<<"\nAfter Sorted
        array : "<<endl;
68      printArray(arr,n);
69
70      return 0;
71  }
```

[?]

4.1 Output

```
Enter the size of an array : 3
Enter the value of index 0 is : 22
Enter the value of index 1 is : 4
Enter the value of index 2 is : 4444
Before sorted array is :
22 4 4444
After Sorted array :

After Sorted array :
4 22 4444
```

Figure 1: BUBBLE SORT

5 INSERTION SORT

```
 //Q:  Bubble  Sort
//NAZMUL ZAMAN-Bsc in (IT)

 #include<bits./stdc++.h>
 using namespace std;
void Insertion_Sort(int arr[],int n)
{
    int i,j,temp;
    for(i=1;i<n;i++)
    {
        temp=arr[i];
        j=i-1;

        while(j>=0 && arr[j]>temp)
        {
            arr[j+1]=arr[j];
            j--;
        }
        arr[j+1]=temp;
    }
}
void Print_Array(int arr[], int n)
 {
     for(int i = 0;i<n; i++)
     cout << arr[i] << " "; // printing an array with sorting

 }


 int main()
 {
     cout<<"Enter the size of an array : "<<endl;
     int n,temp;
     cin >> n;
     int arr[n];
     for(int i =0; i<n; i++)
     {
         cout << "Enter the value of index " << i <<" is : "; //
   this line for taking array value
         cin >> arr[i];
     }
       cout<<"Before sorted array is : \n";
       for(int i=0 ;i<n; i++)
       {
           cout << arr[i] <<" "; // this line for printing user
   value  before sort
       }
      cout<<"\nAfter Sorted array : " <<endl;

```

```
48      Insertion_Sort(arr, n); // call function
49      Print_Array(arr, n);  // call function
50      return 0;
51
52  }
```

[?]

5.1 Output

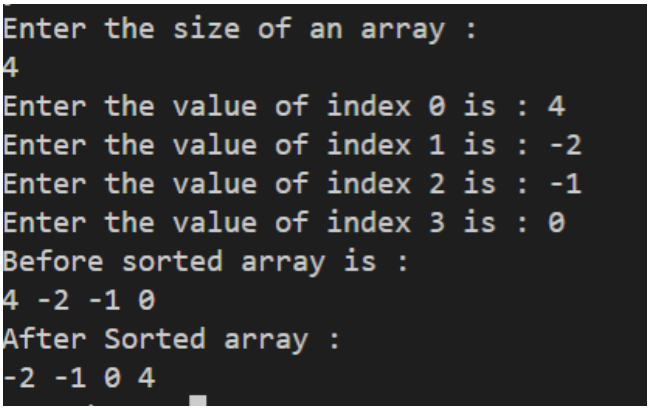

Figure 2: INSERTION SORT

6 SELECTION SORT

```
1  //Q:  Selection Sort
2
3  //NAZMUL ZAMAN-Bsc in (IT)
4
5  #include<bits/stdc++.h>
6  using namespace std;
7
8  void swap(int *x,int *y) // this funtion for swap minimum value and
       used pointer
9  {
10     int temp =  *x;
11    *x =  *y;
12     *y = temp;
13 }
14
15
16 void selectionSort(int arr[],int n)  // base function for iteration
       and compire
17 {int i,j,min_indx;
18     for( i=0 ;i<n-1; i++) //outer loop for iteration
19     {
20         min_indx = i; //i==0; set minimum as first index
21
22         for( j = i+1 ;j<n; j++) // inner loop for compair
23
24             if(arr[j] < arr[min_indx])  // if ture then call swap
       function
25
26                 min_indx = j; // change i into j for stored minimum
       value
27
28      swap(&arr[min_indx], &arr[i]);     // this is swap function
       for swap minmum value
29     }
30 }
31
32
33
34 void printArray(int arr[],int n)  // printing funtion
35 {
36
37     for(int i=0;i<n;i++)
38     {
39         cout<<arr[i]<<" ";
40
41     }
42 }
43
44 int main()
```

```
{
      cout<<"Enter the size of an array : ";
    int n;
    cin>>n;
    int arr[n];


    for(int i=0;i<n;i++)
    {
        cout<<"Enter the value of index "<<i<<" is : ";
        cin>>arr[i];

    }
     cout<<"Before sorted array is : \n";
       for(int i=0;i<n;i++)
       {
        cout<<arr[i]<<" ";
       }

    selectionSort(arr,n);
    cout<<"\nAfter Sorted array : "<<endl;
    printArray(arr,n);

    return 0;
}
```

[?]

6.1 Output

```
Enter the size of an array : 4
Enter the value of index 0 is : 2
Enter the value of index 1 is : -1
Enter the value of index 2 is : 44
Enter the value of index 3 is : 4
Before sorted array is :
2 -1 44 4
After Sorted array :
-1 2 4 44
```

Figure 3: SELECTION SORT

7 Reorder elements of the first array by the order of elements defined by the second array.(Source Code)

```
1
2  // Nazmul_Zaman Bsc in IT
3
4  // A C++ program to sort an array according to the order defined
5  // by another array
6  #include <bits/stdc++.h>
7  using namespace std;
8
9  // A Binary Search based function to find index of FIRST occurrence
10 // of x in arr[]. If x is not present, then it returns -1
11
12 // The same can be done using the lower_bound
13 // function in C++ STL
14 int first(int arr[], int low, int high, int x, int n)
15 {
16
17   // Checking condition
18   if (high >= low) {
19
20     // FInd the mid element
21     int mid = low + (high - low) / 2;
22
23     // Check if the element is the extreme left
24     // in the left half of the array
25     if ((mid == 0 || x > arr[mid - 1]) && arr[mid] == x)
26       return mid;
27
28     // If the element lies on the right half
29     if (x > arr[mid])
30       return first(arr, (mid + 1), high, x, n);
31
32     // Check for element in the left half
33     return first(arr, low, (mid - 1), x, n);
34   }
35
36   // ELement not found
37   return -1;
38 }
39
40 // Sort A1[0..m-1] according to the order defined by A2[0..n-1].
41 void sortAccording(int A1[], int A2[], int m, int n)
42 {
43   // The temp array is used to store a copy of A1[] and visited[]
44   // is used mark the visited elements in temp[].
45   int temp[m], visited[m];
```

```
46    for (int i = 0; i < m; i++) {
47      temp[i] = A1[i];
48      visited[i] = 0;
49    }
50
51    // Sort elements in temp
52    sort(temp, temp + m);
53
54    // for index of output which is sorted A1[]
55    int ind = 0;
56
57    // Consider all elements of A2[], find them in temp[]
58    // and copy to A1[] in order.
59    for (int i = 0; i < n; i++) {
60      // Find index of the first occurrence of A2[i] in temp
61      int f = first(temp, 0, m - 1, A2[i], m);
62
63      // If not present, no need to proceed
64      if (f == -1)
65        continue;
66
67      // Copy all occurrences of A2[i] to A1[]
68      for (int j = f; (j < m && temp[j] == A2[i]); j++) {
69        A1[ind++] = temp[j];
70        visited[j] = 1;
71      }
72    }
73
74    // Now copy all items of temp[]
75    // which are not present in A2[]
76    for (int i = 0; i < m; i++)
77      if (visited[i] == 0)
78        A1[ind++] = temp[i];
79  }
80
81  // Utility function to print an array
82  void printArray(int arr[], int n)
83  {
84
85    // Iterate in the array
86    for (int i = 0; i < n; i++)
87      cout << arr[i] << " ";
88    cout << endl;
89  }
90
91  // Driver Code
92  int main()
93  {
94    int A1[] = { 5, 8, 9, 3, 5, 7, 1, 3, 4, 9, 3, 5, 1, 8, 4 };
95    int A2[] = {3, 5, 7, 2 };
96    int m = sizeof(A1) / sizeof(A1[0]);
```

```
97    int n = sizeof(A2) / sizeof(A2[0]);
98
99    // Prints the sorted array
100   cout << "Sorted array is \n";
101   sortAccording(A1, A2, m, n);
102   printArray(A1, m);
103   return 0;
104 }
```

[?]

7.1 Output

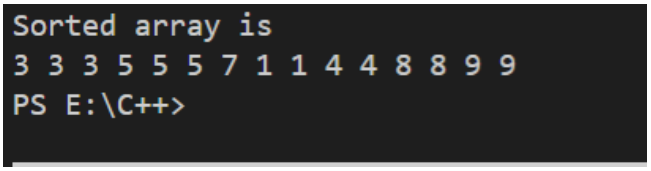

Figure 4: PROBLEM 1

8 Conclusion

In this lab report we learn about different type of sorting algorithm and use of sorting algorithm . A sorting algorithm will put items in a list into an order, such as alphabetical or numerical order. By using these algorithm we can solve different type of sorting algorithm problem and it's also helpful for competitive programming.

References

9 References

1.https://github.com/ZamanNazmul/Data$_S tructure$