Name : Jay Lapani

Roll No.: 21BCP150 Div : 3 G5

Insertion Sort

Aim : Sorting an array using Insertion sort

Theory : Insertion sort is a sorting algorithm that places an unsorted element at its suitable place in each iteration. Insertion sort works similarly as we sort cards in our hand in a card game. We assume that the first card is already sorted then, we select an unsorted card. If the unsorted card is greater than the card in hand, it is placed on the right otherwise, to the left. In the same way, other unsorted cards are taken and put in their right place. A similar approach is used by insertion sort.

Algorithm :

The simple steps of achieving the insertion sort are listed as follows -

Step 1 - If the element is the first element, assume that it is already sorted. Return 1.

Step2 - Pick the next element, and store it separately in a **key.**

Step3 - Now, compare the **key** with all elements in the sorted array.

Step 4 - If the element in the sorted array is smaller than the current element, then move to the next element. Else, shift greater elements in the array towards the right.

Step 5 - Insert the value.

Step 6 - Repeat until the array is sorted.

Problem:

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

int insertionsort(int a[], int b)
{
   int z=0;
   int t=0;
   int j;
   for(int i=1;i<b;i++)
   {
      t = a[i];
      for(j = i-1;j>=0;j--)
      {
         if(t<a[j])
         {
            a[j+1] = a[j];
            z++;
         }
         else
            break;
      }
      a[j+1] = t;
   }
   return z;
}
```

```
int main()
{
   int a[] = {6,5,4,3,2,1};
   int n = sizeof(a) / sizeof(int);

   int s = insertionsort(a, n);
   for (int i = 0; i < n; i++)
   {
      cout<<a[i]<<" ";
   }
   cout<<endl<<s;
}
```

/*

Output :

best case{1,2,3,4,5,6} --> comp = 0 times

average case{5,4,6,2,1,3} --> comp = 11 times

worst case{6,5,4,3,2,1} --> comp = 15 times

*/

Selection Sort

Aim : Sorting an Array using selection sort

Theory : In the selection sort technique, the list is divided into two parts. In one part all elements are sorted and in another part the items are unsorted. At first, we take the maximum or minimum data from the array. After getting the data (say minimum) we place it at the beginning of the list by replacing the data of first place with the minimum data. After performing the array is getting smaller. Thus this sorting technique is done.

Algorithm:

Algorithm of the Selection Sort Algorithm

The selection sort algorithm is as follows:

Step 1: Set Min to location 0 in Step 1.

Step 2: Look for the smallest element on the list.

Step 3: Replace the value at location Min with a different value.

Step 4: Increase Min to point to the next element

Step 5: Continue until the list is sorted.

Program:

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

```
int selectionsort(int a[], int b)
{
   int z;
   int m=INT_MAX;
   int s=0;
   for (int i = 0; i < b; i++)
   {
      for (int j = i; j < b; j++)
      {
         if (a[j] < m)
         {
            s++;
            m = a[j];
            z = j;


         }
      }
      m = INT_MAX;
      int t = a[i];
      a[i] = a[z];
      a[z] = t;
   }
   return s;
}
int main()
{
```

```
    int a[] = {3,4,5,2,6,1};
    int n = sizeof(a) / sizeof(int);

    int s = selectionsort(a, n);
    for (int i = 0; i < n; i++)
    {
        cout<<a[i]<<" ";
    }
    cout<<endl<<s;
}
```

/*

Output :

best case{1,2,3,4,5,6} --> comp = 6 times

average case{3,4,5,2,6,1} --> comp = 12 times

worst case{6,5,4,3,2,1} --> comp = 15 times

*/