Assignment 1

Design and Analysis of Algorithm Lab

By Smit Sutariya
Roll no. 21BCP142

Submitted To

COMPUTER ENGINEERING

School of Technology,
Pandit Deendayal Energy University
January – 2023

Title: Insertion and Selection Sort

Aim: To analyse the Insertion and Selection sort algorithms

Theory:

Insertion sort: It is a simple sorting algorithm that works similarly 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 in the correct position in the sorted part. 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. The time complexity of the Insertion sort is O(n^2).

Selection Sort: The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from the unsorted part and putting it at the beginning. The algorithm maintains two subarrays in a given array. The subarray which already sorted and the remaining subarray was unsorted. In every iteration of the selection sort, the minimum element (considering ascending order) from the unsorted subarray is picked and moved to the beginning of the unsorted subarray. After every iteration sorted subarray size increase by one and the unsorted subarray size decrease by one. The complexity of the selection sort is also O(n^2).

Algorithm:

For Insertion Sort

```
insertionSort(array)
  mark first element as sorted
  for each unsorted element X
  'extract' the element X
    for j <- lastSortedIndex down to 0
      if current element j > X
        move sorted element to the right by 1
    break loop and insert X here
end insertionSort
```

For Selection Sort

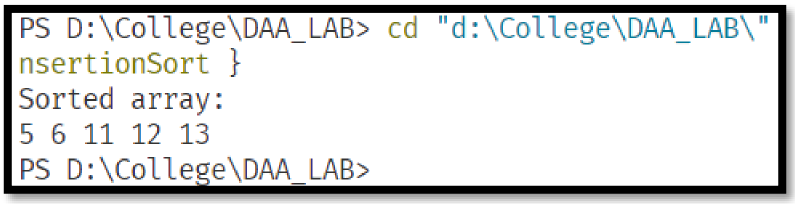
```
selectionSort(array, size)
  repeat (size - 1) times
  set the first unsorted element as the minimum
  for each of the unsorted elements
    if element < currentMinimum
      set element as new minimum
  swap minimum with first unsorted position
end selectionSort
```

Code:

Insertion Sort

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

void insertionSort(int arr[], int n){
    int i, key, j;
    for (i = 1; i < n; i++){
        key = arr[i];
        j = i - 1;
        while (j >= 0 && arr[j] > key){
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}

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

int main(){
    int arr[] = {12, 11, 13, 5, 6};
    int n = sizeof(arr)/sizeof(arr[0]);
    insertionSort(arr, n);
    cout << "Sorted array: \n";
    printArray(arr, n);
    return 0;
}
```

Output:

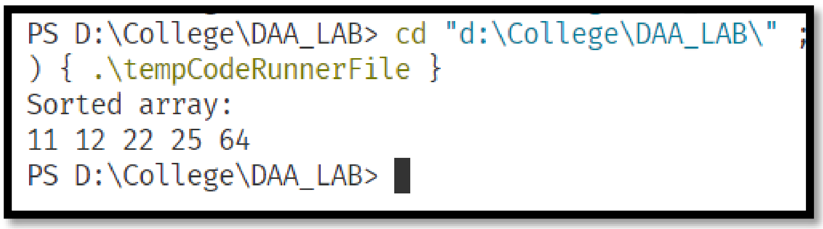


```
PS D:\College\DAA_LAB> cd "d:\College\DAA_LAB\"
nsertionSort }
Sorted array:
5 6 11 12 13
PS D:\College\DAA_LAB>
```

Selection Sort

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

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

void selectionSort(int arr[], int n){
     int i, j, min_idx;
     for (i = 0; i < n-1; i++){
          min_idx = i;
          for  (j  =  i+1;  j  <  n;  j++)
            if (arr[j] < arr[min_idx])
              min_idx = j;
          swap(arr[min_idx], arr[i]);
     }
}

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

int main() {
     int arr[] = {64, 25, 12, 22, 11};
     int n = sizeof(arr)/sizeof(arr[0]);
     selectionSort(arr, n);
     cout << "Sorted array: \n";
     printArray(arr, n);
     return 0;
}
```

Output:



```
PS D:\College\DAA_LAB> cd "d:\College\DAA_LAB\" ;
) { .\tempCodeRunnerFile }
Sorted array:
11 12 22 25 64
PS D:\College\DAA_LAB> 
```