Name: - BADRI VISHAL SINGHAL

Scholar Number: - 20U03001

Date: - 05-08-2021

Experiment 1

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.
1) The subarray which is already sorted.
2) Remaining subarray which is unsorted.
In every iteration of selection sort, the minimum element (considering ascending order) from the unsorted subarray is picked and moved to the sorted subarray.
Following example explains the above steps:

arr[] = 64 25 12 22 11

// Find the minimum element in arr[0...4]

// and place it at beginning

11 25 12 22 64

// Find the minimum element in arr[1...4]
// and place it at beginning of arr[1...4]
11 **12** 25 22 64

// Find the minimum element in arr[2...4]
// and place it at beginning of arr[2...4]
11 12 **22** 25 64

// Find the minimum element in arr[3...4]
// and place it at beginning of arr[3...4]
11 12 22 **25** 64

Algorithm of Selection Sort: -
1. For i=0 to i<n-1 do the following
2. For j=i+1 to j<n
3. Now find the minimum Element index in the arr[i+1 to n]
4. Now swaps the minimum Element with the arr[i]
5. End for loop 2
6. End for loop 1
7. End

Code: -

```
// C++ program for implementation of selection sort
#include <bits/stdc++.h>
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;

    // One by one move boundary of unsorted subarray
    for (i = 0; i < n-1; i++)
    {
        // Find the minimum element in unsorted array
        min_idx = i;
        for (j = i+1; j < n; j++)
        if (arr[j] < arr[min_idx])
```

```
            min_idx = j;

        // Swap the found minimum element with the first element
        swap(&arr[min_idx], &arr[i]);
    }
}

/* 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 program to test above functions
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: -

```
Sorted array:
11 12 22 25 64
PS C:\Users\hp\Desktop\C++ DSA\Practice2>
```

Time Complexity: -

$O(n^2)$ as there are two nested loops.

Space Complexity: -

O(1) as we are not using any space.

Bubble Sort

Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in wrong order.

Example:

First Pass:

(**5 1** 4 2 8) –> (**1 5** 4 2 8), Here, algorithm compares the first two elements, and swaps since 5 > 1.

(1 **5 4** 2 8) –> (1 **4 5** 2 8), Swap since 5 > 4

(1 4 **5 2** 8) –> (1 4 **2 5** 8), Swap since 5 > 2

(1 4 2 **5 8**) –> (1 4 2 **5 8**), Now, since these elements are already in order (8 > 5), algorithm does not swap them.

Second Pass:

(**1 4** 2 5 8) –> (**1 4** 2 5 8)

(1 **4 2** 5 8) –> (1 **2 4** 5 8), Swap since 4 > 2

(1 2 **4 5** 8) –> (1 2 **4 5** 8)

(1 2 4 **5 8**) –> (1 2 4 **5 8**)

Now, the array is already sorted, but our algorithm does not know if it is completed. The algorithm needs one **whole** pass without **any** swap to know it is sorted.

Third Pass:

(**1 2** 4 5 8) –> (**1 2** 4 5 8)

(1 **2 4** 5 8) –> (1 **2 4** 5 8)

(1 2 **4 5** 8) –> (1 2 **4 5** 8)
(1 2 4 **5 8**) –> (1 2 4 **5 8**)

Algorithm of Bubble Sort: -

1. For i=0 to i<n-1 do the following
2. For j=0 to j<n-i-1 do the following
3. If A[i] > A[j] swap A[i] and A[j]
4. End for loop1
5. End for loop2

Code: -

```cpp
// C++ program for implementation of Bubble sort
#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: \n";
    printArray(arr, n);
    return 0;
}
```

Output: -

```
Sorted array:
11 12 22 25 34 64 90
PS C:\Users\hp\Desktop\C++ DSA\Practice2>
```

Time Complexity: -

$O(n^2)$ as there are two nested loops.

Space Complexity: -

O(1) as we are not using any space.