

DSA Lab 5 - (53457) M. Abdullah

- **Problem:** Write two functions, one for bubbleSort and one for insertionSort. Each function should take an array of integers and sort it in ascending order.
- **Input:** An array of integers, e.g., {5, 2, 9, 1, 5, 6}
- **Output:** Sorted array, e.g., {1, 2, 5, 5, 6, 9}
- **Challenge:** Implement both algorithms in-place, without using additional arrays.
- **Problem:** Modify the bubble sort and insertion sort algorithms to sort an array in **descending order** instead of ascending.
- **Input:** An array of integers, e.g., {3, 1, 4, 1, 5, 9}
- **Output:** Sorted array in descending order, e.g., {9, 5, 4, 3, 1, 1}
- **Hint:** In both algorithms, change the comparison condition to reverse the order.
- **Problem:** Modify the bubble sort algorithm to count the number of swaps it performs while sorting an array.
- **Input:** An array of integers, e.g., {5, 3, 2, 1}
- **Output:** Sorted array and the number of swaps, e.g., Sorted array: {1, 2, 3, 5}, Number of swaps: 5
- **Hint:** Increment a counter each time a swap operation occurs.
- **Problem:** Optimize insertion sort by using **binary search** to find the correct position for each element being inserted. Implement this modified insertion sort.
- **Input:** An array of integers, e.g., {7, 3, 8, 2, 9}
- **Output:** Sorted array, e.g., {2, 3, 7, 8, 9}

- **Hint:** Use binary search in the sorted part of the array to find the insertion point, reducing the number of comparisons in each insertion step.

Code

```
#include<iostream>
using namespace std;
class Sorting {
public:
    //bubble sort
    void bubbleSort(int arr[], int n) { //O(n^2)
        int temp = arr[0];
        for(int i=0; i<n; i++) {
            for(int j=i+1; j<n; j++) {
                if(arr[i]>arr[j]) {
                    temp = arr[i];
                    arr[i] = arr[j];
                    arr[j] = temp;
                }
            }
        }
        cout<<"Sorted Array: ";
        for(int i=0; i<n; i++) {
            cout<<arr[i]<<"\t";
        }
    }
    //insertion sort
    void insertionSort(int arr[], int n) { //O(n^2)
        for (int i = 1; i < n; i++) {
            int key = arr[i];
            int j = i - 1;

            while (j >= 0 && arr[j] > key) {
                arr[j + 1] = arr[j];
```

```

        j--;
    }
    arr[j + 1] = key;
}

cout<<"Sorted Array: ";
for(int i=0;i<n;i++) {
    cout<<arr[i]<<"\t";
}
}

//insertion sort in descending order
void insertionSortDescending(int arr[], int n) { //O(n^2)
    for (int i = 1; i < n; i++) {
        int key = arr[i];
        int j = i - 1;

        while (j >= 0 && arr[j] < key) {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = key;
    }

    cout<<"Sorted Array in Descending: ";
    for(int i=0;i<n;i++) {
        cout<<arr[i]<<"\t";
    }
}

// bubble sort descending order
void bubbleSortDescending(int arr[], int n) { //O(n^2)
    int temp = arr[0];
    for(int i=0;i<n;i++) {
        for(int j=i+1;j<n;j++) {
            if(arr[i]<arr[j]) {
                temp = arr[i];

```

```

        arr[i] = arr[j];
        arr[j] = temp;
    }
}
}
cout<<"Sorted Array in Descending: ";
for(int i=0;i<n;i++) {
    cout<<arr[i]<<"\t";
}
}
// Bubble Sort Modified
void bubbleSortModified(int arr[], int n) { // O(n^2)
    int temp = arr[0];
    int counter=0;
    for(int i=0;i<n;i++) {
        for(int j=i+1;j<n;j++) {
            if(arr[i]>arr[j]) {
                temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
                counter++;
            }
        }
    }
    cout<<"Number of Swaps to Sort the Array: "<<counter;
}
// Insertion Sort Optimized
void insertionSortOptimized(int arr[], int n) { // O(n log n)
    for (int i = 1; i < n; i++) {
        int key = arr[i];
        int tempArr[i];
        for (int k=0;k<i;k++) {
            tempArr[k] = arr[k];
        }
        int low = 0;
        int high = i-1;
    }
}

```

```

int index;
int mid;
while(low<=high) {
    mid = (low+high)/2;
    if(key==tempArr[mid]) {
        index = mid+1;
        break;
    }
    else if(key>tempArr[mid]) {
        low = mid+1;
    }
    else {
        high = mid-1;
    }
}
if(low>mid) {
    index = low-1;
}
else {
    index = high+1;
}

// shifting and swapping
int shifts = i-index;
for(int j=0;j<shifts;j++) {
    arr[i-j] = arr[shifts-j-1];
}
//Assigning key to its index
arr[index] = key;
}

cout<<"Sorted Array: ";
for(int z=0;z<n;z++) {
    cout<<arr[z]<<"\t";
}

```

```

    }
};

int main() {
    int arr[5] = {64, 34, 25, 12, 22};
    Sorting sort;
    cout<<"Unsorted Array: ";
    for(int i=0;i<5;i++) {
        cout<<arr[i]<<"\t";
    }
    cout<<endl;
    sort.bubbleSort(arr,5);
    cout<<endl;
    cout<<"Unsorted Array: ";
    int arr1[6] = {21, 13, 27, 54,0,5};
    for(int i=0;i<6;i++) {
        cout<<arr1[i]<<"\t";
    }
    cout<<endl;
    sort.insertionSort(arr1,6);

    int arr2[6] = {54, 12, 2, -1,23,102};
    cout<<"Unsorted Array: ";
    for(int i=0;i<6;i++) {
        cout<<arr2[i]<<"\t";
    }
    cout<<endl;
    sort.bubbleSortDescending(arr2,5);
    cout<<endl;
    cout<<"Unsorted Array: ";
    int arr3[6] = {21, 13, 27, 54,0,5};
    for(int i=0;i<6;i++) {
        cout<<arr3[i]<<"\t";
    }
    cout<<endl;

```

```

sort.insertionSortDescending(arr3,6);
cout<<endl;
// Bubble sort modified
sort.bubbleSortModified(arr1,6);
cout<<endl;
// Insertion Sort Optimized
cout<<"Unsorted Array: ";
int arr5[6] = {21, 13, 27, 54,10,5};
for(int i=0;i<6;i++) {
    cout<<arr5[i]<<"\t";
}
cout<<endl;
sort.insertionSortOptimized(arr5,6);
cout<<endl;
return 0;
}

```

Output

```

PS D:\IT-Study-Resources>
PS D:\IT-Study-Resources> & 'c:\Users\hp\.vscode\extensions\ms-vscode.cpptools-1.22.10-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-pbx54fp1.dqb' '--stdout=Microsoft-MIEngine-Out-cwkr5wp.hnr' '--stderr=Microsoft-MIEngine-Error-byitvha.tfx' '--pid=Microsoft-MIEngine-Pid-4cfo2csj.qop' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
Unsorted Array: 64 34 25 12 22
Sorted Array: 12 22 25 34 64
Unsorted Array: 21 13 27 54 0 5
Sorted Array: 0 5 13 21 27 54
Unsorted Array: 54 12 2 -1
Sorted Array: 23 102
Sorted Array in Descending: 54 23 12 2 -1
Unsorted Array: 21 13 27 54 0 5
Sorted Array in Descending: 54 27 21 13 5 0
Number of Swaps to Sort the Array: 0
Unsorted Array: 21 13 27 54 10 5
Sorted Array: 5 10 13 27 54 13
PS D:\IT-Study-Resources>

```

