DSA PRACTICE SET 6

B Abhimanyu(22CB006)

1) Bubble Sort

CODE

```
class Main{
  public static void main(String[] args){
    int[] arr = {4,3,6,2,5};
    int n = arr.length;
    for(int i=0; i<n-1; i++){
      for(int j = 0; j<n-i-1; j++){
        if(arr[j]>arr[j+1]){
          int temp = arr[j];
          arr[j] = arr[j+1];
          arr[j+1] = temp;
        } }}
    for(int i: arr){
      System.out.print(i+" ");
    }}}
```

OUTPUT

```
C:\Users\abhim\Desktop\java>java Main
2 3 4 5 6
C:\Users\abhim\Desktop\java>
```

Time Complexity: O(n^2)

Space Complexity: O(1)

2) Selection Sort

CODE

```
class Main {
  static void swap(int arr[], int i, int j) {
    int temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
  }
  static void quickSort(int arr[], int low, int high) {
    if (low < high) {
      int p = partition(arr, low, high);
      quickSort(arr, low, p - 1);
```

```
        quickSort(arr, p + 1, high);
      }
    }
    static int partition(int arr[], int low, int high) {
      int pivot = arr[high];
      int i = low - 1;
      for (int j = low; j < high; j++) {
        if (arr[j] < pivot) {
          i++;
          swap(arr, i, j);
        }
      }
      swap(arr, i + 1, high);
      return i + 1;
    }
    public static void main(String[] args) {
      int[] arr = {10, 7, 8, 9, 1, 5};
      System.out.println("Original Array:");
      printArray(arr);
      quickSort(arr, 0, arr.length - 1);
      System.out.println("\nSorted Array:");
      printArray(arr);
    }
    static void printArray(int[] arr) {
      for (int i = 0; i < arr.length; i++) {
        System.out.print(arr[i] + " ");
      }
      System.out.println();
    }
}
```

OUTPUT

```
C:\Users\abhim\Desktop\java>java Main
Original Array:
10 7 8 9 1 5

Sorted Array:
1 5 7 8 9 10

C:\Users\abhim\Desktop\java>
```

Time Complexity: O(nlogn)

Space Complexity: O(logn)

3) Non Repeating Characters

CODE

```
import java.util.*;
class Main {
  static char nonRepeatingChar(String s) {
    Map<Character, Integer> map = new HashMap<>();
    for (int i = 0; i < s.length(); i++) {
      map.put(s.charAt(i), map.getOrDefault(s.charAt(i), 0) + 1);
    }
    for (int i = 0; i < s.length(); i++) {
      if (map.get(s.charAt(i)) == 1) {
        return s.charAt(i);
      } }
    return '$';
  }
  public static void main(String[] args) {
    String s1 = "geeksforgeeks";
    System.out.println("First non-repeating character in \"" + s1 + "\": " +
nonRepeatingChar(s1));
  }}
```

OUTPUT

```
C:\Users\abhim\Desktop\java>java Main
First non-repeating character in "geeksforgeeks": f

C:\Users\abhim\Desktop\java>
```

Time Complexity: O(n)

Space Complexity: O(n)

4) Edit Distance

CODE

```
class Main {
  public int editDistance(String s1, String s2) {
    int[][] mat = new int[s1.length() + 1][s2.length() + 1];
    for (int i = 0; i <= s1.length(); i++) {
      mat[i][0] = i;
    }
```

```
    for (int i = 0; i <= s2.length(); i++) {
      mat[0][i] = i;
    }
    for (int i = 1; i <= s1.length(); i++) {
      for (int j = 1; j <= s2.length(); j++) {
        if (s1.charAt(i - 1) == s2.charAt(j - 1)) {
          mat[i][j] = mat[i - 1][j - 1];
        } else {
          mat[i][j] = 1 + Math.min(mat[i - 1][j],
                    Math.min(mat[i][j - 1],
                          mat[i - 1][j - 1]));
        }
      }
    }
    return mat[s1.length()][s2.length()];
  }
  public static void main(String[] args) {
    Main solution = new Main();
    String s1 = "geek";
    String s2 = "gesek";
    System.out.println("Edit Distance between \"" + s1 + "\" and \"" + s2 + "\": " +
solution.editDistance(s1, s2));
  }
}
```

OUTPUT

```
C:\Users\abhim\Desktop\java>java Main
Edit Distance between "geek" and "gesek": 1

C:\Users\abhim\Desktop\java>
```

Time Complexity: O(n^2)
Space Complexity: O(m*n)

5) Kth Largest Elements

CODE

```
import java.util.*;

class Main {
  static List<Integer> kLargest(int arr[], int k) {
    Arrays.sort(arr);
```

```
    List<Integer> ans = new ArrayList<>();
    for (int i = arr.length - 1; i >= arr.length - k; i--) {
        ans.add(arr[i]);
    }
    return ans;
  }
  public static void main(String[] args) {
    int[] arr1 = {1, 23, 12, 9, 30, 2, 50};
    int k1 = 3;
    System.out.println("K Largest Elements: " + kLargest(arr1, k1));
  }
}
```

OUTPUT

```
C:\Users\abhim\Desktop\java>java Main
K Largest Elements: [50, 30, 23]

C:\Users\abhim\Desktop\java>
```

Time Complexity: O(nlogn)
Space Complexity: O(n)

6) Form Largest Number

CODE

```
import java.util.*;
class Solution {
  String printLargest(String[] arr) {
    Arrays.sort(arr, (a, b) -> (b + a).compareTo(a + b));
    if (arr[0].equals("0")) {
        return "0";
    }
    StringBuilder result = new StringBuilder();
    for (String s : arr) {
        result.append(s);
    }
    return result.toString();
  }
  public static void main(String[] args) {
    Solution solution = new Solution();
    String[] arr1 = {"3", "30", "34", "5", "9"};
    System.out.println("Largest Number: " + solution.printLargest(arr1));
```

```
  }
}
```

OUTPUT

```
C:\Users\abhim\Desktop\java>java Main
Largest Number: 9534330

C:\Users\abhim\Desktop\java>
```

Time Complexity: O(nlongn)

Space Complexity: O(n)