

# DSA Practice

Date: 18/11/2024

**Problems:**

**1).Bubble sort**

**2).Quick sort**

**3).Non repeating characters**

**4).Edit distance**

**5).k largest elements**

**6).form the largest number**

**Problem1:**

```
class Solution {  
    // Function to sort the array using bubble sort algorithm.  
    public static void bubbleSort(int arr[]) {  
        // code here  
        int n = arr.length;  
        int j=0;  
        while(j<n){  
            for(int i=0;i<n-1;i++){  
                if(arr[i]>arr[i+1]){  
                    int temp = arr[i];  
                    arr[i] = arr[i+1];  
                    arr[i+1] = temp;  
                }  
            }  
            j++;  
        }  
    }  
}
```

}

## Output:

The screenshot shows a web browser window with the URL <https://www.geeksforgeeks.org/problems/bubble-sort/1>. The page displays the 'Output Window' for a problem titled 'Bubble Sort'. The status is 'Problem Solved Successfully'. The test cases passed are 1115 / 1115. The attempts are 1 / 1, and the accuracy is 100%. The points scored are 2 / 2, and the time taken is 0.59. The 'Solve Next' section shows buttons for 'Selection Sort', 'Insertion Sort', and 'Counting Sort'. The code editor on the right shows a Java solution for the bubble sort problem. The code is as follows:

```
1 // Driver Code Ends
2 // User function Template for Java
3
4 class Solution {
5     // Function to sort the array using bubble sort algorithm.
6     public static void bubbleSort(int arr[]) {
7         // code here
8         int n = arr.length;
9         int j = 0;
10        while(j < n){
11            for(int i = 0; i < n - 1 - j; i++){
12                if(arr[i] > arr[i + 1]){
13                    int temp = arr[i];
14                    arr[i] = arr[i + 1];
15                    arr[i + 1] = temp;
16                }
17            }
18            j++;
19        }
20    }
21 }
22 // Driver Code Ends
```

## Problem2:

```
class Solution {
```

```
    // Function to sort an array using quick sort algorithm.
```

```
    static void quickSort(int arr[], int low, int high) {
```

```
        // code here
```

```
        if(low < high){
```

```
            int pivotindex = partition(arr, low, high);
```

```
            quickSort(arr, low, pivotindex - 1);
```

```
            quickSort(arr, pivotindex + 1, high);
```

```
        }
```

```
    }
```

```
    static int partition(int arr[], int low, int high) {
```

```
        // your code here
```

```
int pivot = arr[low];
int left = low+1;
int right = high;
while(left<=right){
    while(left<=high && arr[left]<=pivot){
        left++;
    }
    while(right>=left && arr[right]>pivot){
        right--;
    }
    if(left<right){
        int temp = arr[left];
        arr[left] = arr[right];
        arr[right] = temp;
    }
}
int temp = arr[low];
arr[low] = arr[right];
arr[right] = temp;
return right;
}
```

**Output:**

The screenshot shows the GeeksforGeeks website interface. On the left, the 'Output Window' indicates a successful submission for the 'Quick Sort' problem. It shows that all 1120 test cases were passed, 4 out of 4 points were scored, and the time taken was 0.59 seconds. The right pane displays the Java code for the quick sort algorithm. The bottom of the page shows a Windows taskbar with various application icons and system status.

### Problem3:

```
class Solution {
```

```
    // Function to find the first non-repeating character in a string.
```

```
    static char nonRepeatingChar(String s) {
```

```
        // Your code here
```

```
        Map<Character,Integer> map = new HashMap<>();
```

```
        for(char ch : s.toCharArray()){
```

```
            map.put(ch,map.getOrDefault(ch,0)+1);
```

```
        }
```

```
        for(char ch : s.toCharArray()){
```

```
            if(map.get(ch)==1){
```

```
                return ch;
```

```
            }
```

```
        }
```

```
        return '$';
```

```
    }
```

}

## Output:

The screenshot shows the GeeksforGeeks website interface. On the left, the 'Output Window' displays the following information:

- Problem Solved Successfully** (with a green checkmark)
- Test Cases Passed:** 1130 / 1130
- Attempts:** Correct / Total: 1 / 2
- Accuracy:** 50%
- Points Scored:** 2 / 2
- Time Taken:** 0.6
- Your Total Score:** 81 (with an upward arrow)
- Solve Next:** Reverse Words, Longest substring with distinct characters, Balloon Everywhere
- Kick start your career with GFG 160!** (with a right arrow)

On the right, the code editor shows a Java solution for the 'Non Repeating Character' problem. The code is as follows:

```
1 // Driver Code Ends
29
30 // User function Template for Java
31
32 class Solution {
33     // Function to find the first non-repeating character in a string.
34     static char nonRepeatingChar(String s) {
35         // Your code here
36         Map<Character, Integer> map = new HashMap<>();
37         for(char ch : s.toCharArray()){
38             map.put(ch, map.getOrDefault(ch, 0) + 1);
39         }
40         for(char ch : s.toCharArray()){
41             if(map.get(ch) == 1){
42                 return ch;
43             }
44         }
45         return '$';
46     }
47 }
48
49
```

## Problem4:

```
class Solution {
    public int editDistance(String s1, String s2) {
        // Code here
        int n = s1.length();
        int m = s2.length();
        int[][] dp = new int[n+1][m+1];
        for(int i=0; i<=n; i++){
            for(int j=0; j<=m; j++){
                if(i==0){
                    dp[i][j]=j;
                }else if(j==0){
                    dp[i][j]=i;
                }else if(s1.charAt(i-1)==s2.charAt(j-1)){
```

```

        dp[i][j] = dp[i-1][j-1];
    }else{
        dp[i][j] = 1+Math.min(dp[i-1][j-1],Math.min(dp[i-1][j],dp[i][j-1]));
    }
}
}
return dp[n][m];
}
}

```

## Output:

The screenshot shows the GeeksforGeeks interface for the 'Edit Distance' problem. The left sidebar indicates a successful solution with 1115/1115 test cases passed, 8/8 points scored, and a time taken of 0.25. The right pane displays the following Java code:

```

1 // Driver Code Ends
2
3
4
5
6
7
8
9
10
11
12
13
14 class Solution {
15     public int editDistance(String s1, String s2) {
16         // Code here
17         int n = s1.length();
18         int m = s2.length();
19         int[][] dp = new int[n+1][m+1];
20         for(int i=0; i<=n; i++){
21             for(int j=0; j<=m; j++){
22                 if(i==0){
23                     dp[i][j]=j;
24                 }else if(j==0){
25                     dp[i][j]=i;
26                 }else if(s1.charAt(i-1)==s2.charAt(j-1)){
27                     dp[i][j] = dp[i-1][j-1];
28                 }else{
29                     dp[i][j] = 1+Math.min(dp[i-1][j-1],Math.min(dp[i-1][j],dp[i][j-1]));
30                 }
31             }
32         }
33         return dp[n][m];
34     }
35 }
36
37
38
39
40
41
42
43
44
45

```

## Problem5:

```

class Solution {
    // Function to find the first negative integer in every window of size k
    static List<Integer> kLargest(int arr[], int k) {
        // write code here
        Arrays.sort(arr);

```

```

List<Integer> ls = new ArrayList<>();

for(int i=arr.length-1;i>=arr.length-k;i--){

    ls.add(arr[i]);

}

return ls;

}

}

```

## Output:

The screenshot shows the GeeksforGeeks interface for the 'k largest elements' problem. The left sidebar indicates the problem was solved successfully with 1111/1111 test cases passed, 100% accuracy, 4/4 points scored, and a time taken of 0.5 seconds. The right pane displays the Java code for the solution, which uses a list to store the k largest elements and returns it.

## Problem6:

```

class Solution {

    String printLargest(int[] arr) {

        // code here

        Integer[] arr1 = Arrays.stream(arr).boxed().toArray(Integer[]::new);

        Arrays.sort(arr1,(a,b)->{

            String temp1 = Integer.toString(a);

            String temp2 = Integer.toString(b);

```

```

        return (temp1 + temp2).compareTo(temp2 + temp1);
    });

    StringBuilder sb = new StringBuilder();

    for(int no : arr1){
        sb.append(Integer.toString(no));
    }

    return sb.toString();
}
}

```

## Output:

The screenshot shows the GeeksforGeeks website interface for the problem "Form the Largest Number". The left sidebar displays the "Output Window" with the following details:

- Problem Solved Successfully** (with a green checkmark icon)
- Test Cases Passed:** 1111 / 1111
- Attempts:** Correct / Total: 1 / 1
- Accuracy:** 100%
- Points Scored:** 4 / 4
- Time Taken:** 1.18
- Your Total Score:** 97 (with an upward arrow icon)
- Solve Next:** Max sum in the configuration, Maximum Index, Maximize Number of 1's
- Kick start your career with GFG 160!** (with a right arrow icon)

The right sidebar shows the code editor with the following Java code:

```

1 // Driver Code Ends
28
29
30 // User function Template for Java
31
32 class Solution {
33     String printLargest(int[] arr) {
34         // code here
35         Integer[] arr1 = Arrays.stream(arr).boxed().toArray(Integer[]::new);
36         Arrays.sort(arr1, (a, b) -> {
37             String temp1 = Integer.toString(a);
38             String temp2 = Integer.toString(b);
39             return (temp2 + temp1).compareTo(temp1 + temp2);
40         });
41         StringBuilder sb = new StringBuilder();
42         for(int no : arr1){
43             sb.append(Integer.toString(no));
44         }
45         return sb.toString();
46     }
47 }

```

The bottom of the screenshot shows the Windows taskbar with the date and time: 18-11-2024, 22:40.