

DSA Practice 6 (18-11-24)

By Jashwanth SA

CSE-C, 22CS066

1. Bubble sort:

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

Output:

Bubble Sort

Difficulty: Easy Accuracy: 59.33% Submissions: 236K+ Points: 2

Given an array, `arr[]`. Sort the array using bubble sort algorithm.

Examples :

Input: `arr[] = [4, 1, 3, 9, 7]`
Output: `[1, 3, 4, 7, 9]`

Output Window

Compilation Results Custom Input Y.O.G.I. (AI Bot)

Problem Solved Successfully

Suggest Feedback

Test Cases Passed: **1115 / 1115**

Attempts: Correct / Total: **1 / 1**

Accuracy: 100%

Custom Input Compile & Run Submit

Time complexity: $O(n)$

2. Quick sort:

```
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 p=partition(arr,low,high);
            quickSort(arr,low,p-1);
            quickSort(arr,p+1,high);
        }
    }

static void swap(int[] arr, int i,int j){
    int temp=arr[i];
    arr[i]=arr[j];
    arr[j]=temp;
}

static int partition(int arr[], int low, int high) {
    // your code here
    int pivot=arr[high],i=low-1;
    for(int j=low;j<=high-1;j++){
        if(arr[j]<pivot){
            i++;
            swap(arr,i,j);
        }
    }
    swap(arr,i+1,high);
    return i+1;
}
}

```

Output:

Quick Sort

Difficulty: Medium Accuracy: 55.23% Submissions: 235K+ Points: 4

Implement Quick Sort, a Divide and Conquer algorithm, to sort an array, `arr[]` in ascending order. Given an array, `arr[]`, with starting index `low` and ending index `high`, complete the functions `partition()` and `quickSort()`. Use the last element as the pivot so that all elements less than or equal to the pivot come before it, and elements greater than the pivot follow it.

Note: The `low` and `high` are inclusive.

Examples:

Output Window

Compilation Results Custom Input Y.O.G.I. (AI Bot)

Problem Solved Successfully

Suggest Feedback

Test Cases Passed	Attempts : Correct / Total
1120 / 1120	1 / 1
	Accuracy : 100%

```

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 p=partition(arr,low,high);
            quickSort(arr,low,p-1);
            quickSort(arr,p+1,high);
        }
    }

    static void swap(int[] arr, int i,int j){
        int temp=arr[i];
        arr[i]=arr[j];
        arr[j]=temp;
    }

    static int partition(int arr[], int low, int high) {
        // your code here
        int pivot=arr[high],i=low-1;
        for(int j=low;j<=high-1;j++){
            if(arr[j]<pivot){
                i++;
                swap(arr,i,j);
            }
        }
        swap(arr,i+1,high);
        return i+1;
    }
}

```

Time complexity: $O(n \log n)$

3. Non repeating character:

```

class Solution {
    // Function to find the first non-repeating character in a string.

```

```

static char nonRepeatingChar(String s) {
    // Your code here
    HashMap<Character,Integer> dict=new HashMap<>();
    for(char c:s.toCharArray()){
        dict.put(c,dict.getOrDefault(c,0)+1);
    }
    for(char c:s.toCharArray()){
        if(dict.get(c)==1){return c;}
    }
    return '$';
}
}

```

Output:

Non Repeating Character

Difficulty: Easy Accuracy: 40.43% Submissions: 230K+ Points: 2

Given a string *s* consisting of lowercase Latin Letters. Return the first non-repeating character in *s*. If there is no non-repeating character, return '\$'.
Note: When you return '\$' driver code will output -1.

Examples:

Input: *s* = "geeksforgeeks"

Output: \$

Compilation Results Custom Input Y.O.G.I. (AI Bot)

Problem Solved Successfully

Suggest Feedback

Test Cases Passed	Attempts: Correct / Total
1130 / 1130	1 / 1
	Accuracy: 100%

```

// User function Template for Java
class Solution {
    // Function to find the first non-repeating character in a string.
    // Your code here
    static char nonRepeatingChar(String s) {
        HashMap<Character,Integer> dict=new HashMap<>();
        for(char c:s.toCharArray()){
            dict.put(c,dict.getOrDefault(c,0)+1);
        }
        for(char c:s.toCharArray()){
            if(dict.get(c)==1){return c;}
        }
        return '$';
    }
}

```

Time complexity: $O(n)$

4. Edit Distance:

```

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

```

```

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

```

Output:

The screenshot shows the LeetCode interface for the 'Edit Distance' problem. On the left, the problem description states: 'Given two strings s1 and s2. Return the minimum number of operations required to convert s1 to s2. The possible operations are permitted: 1. Insert a character at any position of the string. 2. Remove any character from the string. 3. Replace any character from the string with any other character.' Below this, the 'Output Window' shows 'Problem Solved Successfully' with '1115 / 1115' test cases passed and '1 / 1' attempts correct. On the right, the Java code for the solution is displayed, implementing a dynamic programming table 'dp' to calculate the edit distance between two strings.

Time complexity: $O(n^2)$

5. K largest elements:

```

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> res=new ArrayList<>();
        int n=arr.length;
        for(int i=n-1;i>=n-k;i--){
            res.add(arr[i]);
        }
        return res;
    }
}

```

Output:

k largest elements

Difficulty: Medium Accuracy: 53.56% Submissions: 163K+ Points: 4

Given an array `arr[]` of positive integers and an integer `k`, Your task is to return `k` largest elements in decreasing order.

Examples

Input: `arr[] = [12, 5, 787, 1, 23]`, `k = 2`
Output: `[787, 23]`

Output Window

Compilation Results Custom Input Y.O.G.I. (AI Bot)

Problem Solved Successfully

Test Cases Passed	Attempts : Correct / Total
1111 / 1111	1 / 1
	Accuracy : 100%

```

1 // User function Template for Java
2
3 class Solution {
4
5     // Function to find the first negative integer in every window of size k
6     static List<Integer> kLargest(int arr[], int k) {
7         // write code here
8         Arrays.sort(arr);
9         List<Integer> res=new ArrayList<>();
10        int n=arr.length;
11        for(int i=n-1;i>=n-k;i--){
12            res.add(arr[i]);
13        }
14        return res;
15    }
16 }
17
18 // Driver code ends
19
20
21
22
23

```

Time complexity: $O(n)$

6. Form the largest number:

```

class Solution {
    String printLargest(int[] arr) {
        // code here
        String[] res=Arrays.stream(arr).mapToObj(String::valueOf).toArray(String[]::new);
        Arrays.sort(res,(a,b)->(b+a).compareTo(a+b));
        return String.join("",res);
    }
}

```

Output:

Form the Largest Number

Difficulty: Medium Accuracy: 37.82% Submissions: 162K+ Points: 4

Given an array of integers `arr[]` representing non-negative integers, arrange them so that after concatenating all of them in order, it results in the **largest** possible number. Since the result may be very large, return it as a string.

Examples:

Input: `arr[] = [3, 30, 34, 5, 9]`

Output Window

Compilation Results Custom Input Y.O.G.I. (AI Bot)

Problem Solved Successfully

Test Cases Passed	Attempts : Correct / Total
1111 / 1111	1 / 1
	Accuracy : 100%

```

1 // User function Template for Java
2
3 class Solution {
4     String printLargest(int[] arr) {
5         // code here
6         String[] res=Arrays.stream(arr).mapToObj(String::valueOf).toArray(String[]::new);
7         Arrays.sort(res,(a,b)->(b+a).compareTo(a+b));
8         return String.join("",res);
9     }
10 }

```

Time complexity: $O(n)$