

# DSA PRACTICE -6

Name : Tharani Kumar K

Dept : IT

Date : 18/11/2024

## 1) Bubble Sort

**Code:**

```
import java.util.Scanner;
public class Main {
    public static void bubblesort(int[] arr){
        int n = arr.length;
        for(int i=n-1;i>=0;i--){
            for(int j=0;j<i;j++){
                if(arr[j]>arr[j+1]){
                    int temp = arr[j];
                    arr[j] = arr[j+1];
                    arr[j+1] = temp;
                }
            }
        }
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int[] arr = new int[n];
        for(int i=0;i<n;i++){
            arr[i] = sc.nextInt();
        }
        bubblesort(arr);
        for(int i:arr) {
            System.out.print(i + " ");
        }
    }
}
```

Output :

```
C:\Users\thara\.jdk\openjdk-23\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA
6
98 67 54 102 34 56
34 54 56 67 98 102
```

Time Complexity :  $O(n^2)$

## 2) Quick Sort

**Code :**

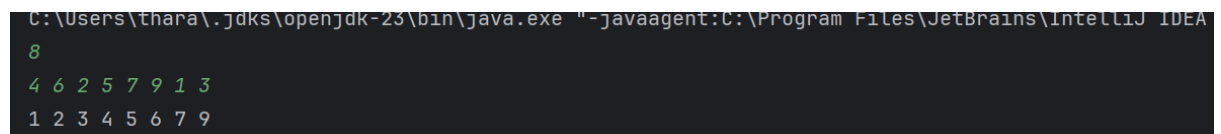
```
import java.util.*;
public class Main {
    public static void qs(int[] arr,int low,int high){
        if(low<high){
            int p = pivot(arr,low,high);
            qs(arr,low,p-1);
            qs(arr,p+1,high);
        }
    }
    public static int pivot(int[] arr,int low,int high){
        int pele = arr[low];
        int i=low,j=high;
        while(i<j){
            while(i<=high && arr[i]<=pele){
                i++;
            }
            while(j>low && arr[j]>pele){
                j--;
            }
            if(i<j){
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
        return i;
    }
}
```

```

    }
}
int temp = arr[low];
arr[low] = arr[j];
arr[j] = temp;
return j;
}
public static void main(String[] args){
    Scanner sc = new Scanner(System.in);
    int n = sc.nextInt();
    int[] arr = new int[n];
    for(int i=0;i<n;i++){
        arr[i] = sc.nextInt();
    }
    qs(arr,0,n-1);
    for(int i:arr){
        System.out.print(i+" ");
    }
}
}

```

#### Output:



```

C:\Users\thara\.jdk\openjdk-23\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA
8
4 6 2 5 7 9 1 3
1 2 3 4 5 6 7 9

```

Time Complexity :  $O(n^2)$

Space Complexity :  $O(1)$

### 3) Non Repeating Character

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:** 'f'

**Explanation:** In the given string, 'f' is the first character in the string which does not repeat.

**Input:** s = "racecar"

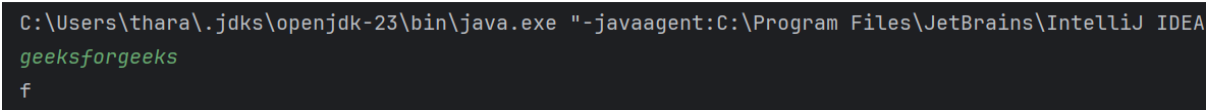
**Output:** 'e'

**Explanation:** In the given string, 'e' is the only character in the string which does not repeat.

**Code:**

```
import java.util.*;
public class Main {
    public static char helper(String s){
        HashMap<Character,Integer> hp = new HashMap<>();
        for(char ch : s.toCharArray()){
            hp.put(ch, hp.getOrDefault(ch,0)+1);
        }
        for(char ch : s.toCharArray()){
            if(hp.get(ch) == 1){
                return ch;
            }
        }
        return '$';
    }
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        String s = sc.next();
        if(helper(s) == '$'){
            System.out.print(-1);
        }
        else {
            System.out.print(helper(s));
        }
    }
}
```

**Output:**



```
C:\Users\thara\.jdk\openjdk-23\bin\java.exe -javaagent:C:\Program Files\JetBrains\IntelliJ IDEA
geeksforgeeks
e
```

Time Complexity : O(n)

Space Complexity : O(n)

#### 4) K largest Elements

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]

**Explanation:** 1st largest element in the array is 787 and second largest is 23.

**Input:** arr[] = [1, 23, 12, 9, 30, 2, 50], k = 3

**Output:** [50, 30, 23]

**Explanation:** Three Largest elements in the array are 50, 30 and 23.

**Code:**

```
import java.util.*;
public class Main {
    public static List<Integer> helper(int[] arr,int k){
        PriorityQueue<Integer> pq = new PriorityQueue<>();
        for(int i:arr){
            pq.add(i);
            if(pq.size()>k){
                pq.poll();
            }
        }
        List<Integer> ans = new ArrayList<>();
        while(!pq.isEmpty()) {
            ans.add(pq.poll());
        }
        Collections.sort(ans, new Comparator<Integer>() {
            public int compare(Integer a, Integer b) {
                return b - a; // Descending order
            }
        });
        return ans;
    }
    public static void main(String[] args){
```

```

Scanner sc = new Scanner(System.in);
int n = sc.nextInt();
int[] arr = new int[n];
for(int i=0;i<n;i++){
    arr[i] = sc.nextInt();
}
int k = sc.nextInt();
List<Integer> ans = helper(arr,k);
System.out.print(ans);
}
}

```

Output:

```

C:\Users\thara\.jdk\openjdk-23\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA
5
12 5 787 1 23
2
[787, 23]

```

Time Complexity :  $O(n \log k)$

Space Complexity :  $O(k)$  ( $K \rightarrow$  represents the no. of largest elements)

## 5) Form Largest Number

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

Note: There are no leading zeros in each array element.

**Examples:**

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

**Output:** "9534330"

**Explanation:** Given numbers are {"3", "30", "34", "5", "9"}, the arrangement "9534330" gives the largest value

**Input:** `arr[] = ["54", "546", "548", "60"]`

**Output:** "6054854654"

**Explanation:** Given numbers are {"54", "546", "548", "60"}, the arrangement "6054854654" gives the largest value.

**Code:**

```
import java.util.*;
public class Main {
    public static String helper(String[] arr){
        Arrays.sort(arr, new Comparator<String>() {
            public int compare(String a,String b){
                return (b+a).compareTo(a+b);
            }
        });
        StringBuilder sb = new StringBuilder();
        for(String i:arr){
            sb.append(i);
        }
        while(sb.charAt(0) == '0' && sb.length()>1){
            sb.deleteCharAt(0);
        }
        return sb.toString();
    }
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        String[] arr = new String[n];
        for(int i=0;i<n;i++){
            arr[i] = sc.next();
        }
        String ans = helper(arr);
        System.out.print(ans);
    }
}
```

### Output:

```
C:\Users\thara\.jdk\openjdk-23\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA
5
3 30 34 5 9
9534330
```

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