

<https://allhackerranksolutionsbykaira.blogspot.com/search/label/HackerRank>

<https://awesomedatastructures.com/project/Java-aid/Hackerrank-Solutions#data-structures>

## MEANDERING ARRAY

```
// Java program to print the array in given order

import java.util.Arrays;

public class GFG {

    // Function which arrange the array.

    static void rearrangeArray(int arr[], int n)

    {

        // Sorting the array elements

        Arrays.sort(arr);

        int[] tempArr = new int[n]; // To store modified array

        // Adding numbers from sorted array to

        // new array accordingly

        int ArrIndex = 0;

        // Traverse from begin and end simultaneously

        for (int i = 0, j = n-1; i <= n / 2 || j > n / 2;

             i++, j--) {

            if(ArrIndex < n)

            {

                tempArr[ArrIndex] = arr[i];

                ArrIndex++;

            }

            if(ArrIndex < n)

            {

                tempArr[ArrIndex] = arr[j];

                ArrIndex++;

            }

        }

    }

}
```

```

}

// Modifying original array

for (int i = 0; i < n; i++)
    arr[i] = tempArr[i];

}

// Driver Code

public static void main(String args[])
{
    int arr[] = { 5, 8, 1, 4, 2, 9, 3, 7, 6 };

    int n = arr.length;

    rearrangeArray(arr, n);

    for (int i = 0; i < n; i++)
        System.out.print(arr[i] + " ");

}
}

```

### **DISTINCT DIGIT NUMBER**

```

// Java implementation of brute

// force solution.

import java.util.LinkedHashSet;

class GFG

{

    // Function to check if the given

    // number has repeated digit or not

    static int repeated_digit(int n)

    {

        LinkedHashSet<Integer> s = new LinkedHashSet<>();

        // Traversing through each digit

        while (n != 0)

        {

            int d = n % 10;

```

```
// if the digit is present
// more than once in the
// number
if (s.contains(d))
{
    // return 0 if the number
    // has repeated digit
    return 0;
}
s.add(d);
n = n / 10;
}

// return 1 if the number has
// no repeated digit
return 1;
}

// Function to find total number
// in the given range which has
// no repeated digit
static int calculate(int L, int R)
{
    int answer = 0;
    // Traversing through the range
    for (int i = L; i < R + 1; ++i)
    {
        // Add 1 to the answer if i has
        // no repeated digit else 0
        answer = answer + repeated_digit(i);
    }
    return answer;
}
```

```

// Driver Code

public static void main(String[] args)

{
    int L = 80, R = 120;

    // Calling the calculate

    System.out.println(calculate(L, R));
}
}

```

### TWIN STRING PROBLEM

```

import java.io.IOException;
import java.util.Scanner;

public class TwinStringProblem {
    //creating method for checking two string
    static boolean[] twins(String[] a, String[] b) {
        boolean[] result = new boolean[a.length];

        for (int i = 0; i < a.length; i++) {
            String aVal = a[i].toLowerCase();
            String bVal = b[i].toLowerCase();
            String[] aValArray = aVal.split("");
            String[] bValArray = bVal.split("");

            for (String s : aValArray) {
                for (int index = 0; index < aValArray.length; index++) {
                    if (bValArray[index].equals(s)) { // checking whether the
index match or not
                        if ((s.indexOf(s) % 2 == 0 && index % 2 == 0) || // ch
ecking even values
                            (s.indexOf(s) % 2 != 0 && index % 2 != 0)) {
                            result[i] = false;
                        } else if ((s.indexOf(s) % 2 == 0 && index % 2 != 0)
                            || (s.indexOf(s) % 2 != 0 && index % 2 == 0))
{
                            result[i] = true;
                            break;
                        }
                    }
                }
            }
        }
        return result;
    }
}

```

```
}

//main Method
public static void main(String[] args) throws IOException {
    Scanner in = new Scanner(System.in);

    int n = Integer.parseInt(in.nextLine().trim());
    String[] a = new String[n];
    for (int i = 0; i < n; i++) {
        a[i] = in.nextLine();
    }

    int m = Integer.parseInt(in.nextLine().trim());
    String[] b = new String[m];
    for (int i = 0; i < m; i++) {
        b[i] = in.nextLine();
    }

    // call twins function
    boolean[] results = twins(a, b);

    for (int i = 0; i < results.length; i++) {
        System.out.println(results[i] ? "Yes" : "No");
    }
}
```

## ARRAY GAME

```
20 |     * The function accepts INTEGER_ARRAY numbers as parameter.  
21 |     */  
22 |  
23 |     public static long countMoves(List<Integer> numbers) {  
24 |         // Write your code here  
25 |         long arrsum,smallest,arr_size=numbers.size();  
26 |         arrsum=0;  
27 |         smallest=(long) numbers.get(0);  
28 |         for(int i=0;i<arr_size;i++){  
29 |             long temp=(long) numbers.get(i);  
30 |             if(temp<smallest){  
31 |                 smallest=temp;  
32 |             }  
33 |             arrsum+=temp;  
34 |         }  
35 |         long minoperation = arrsum-arr_size*smallest;  
36 |         return minoperation;  
37 |     }  
38 | }  
39 | }  
40 | }
```

## PRODUCT SORT

```
10 |  
11 | class FreqComparator implements Comparator<Integer> {  
12 |     private final Map<Integer, Integer> freq;  
13 |     FreqComparator(Map<Integer, Integer> tempFreqMap)  
14 |     {  
15 |         this.freq = tempFreqMap;  
16 |     }  
17 |     @Override  
18 |     public int compare(Integer k1, Integer k2)  
19 |     {  
20 |         int freqCompare = freq.get(k1).compareTo(freq.get(k2));  
21 |         int valueCompare = k1.compareTo(k2);  
22 |         if (freqCompare == 0)  
23 |             return valueCompare;  
24 |         else  
25 |             return freqCompare;  
26 |     }  
27 | }  
28 |  
29 | class Result {  
30 |  
31 |     /* Complete the 'lensort' function below.  
32 | */  
33 | }
```

```
36     * The function accepts INTEGER_ARRAY,
37     */
38
39     /* public static List<Integer> itemsSort(List<Integer> items) {
40     // Write your code here
41     }*/
42
43     public static List<Integer> itemsSort(List<Integer> items){
44         HashMap<Integer, Integer> myMap = new HashMap<>();
45         List<Integer> output = new ArrayList<>();
46         Integer temp=0;
47         for(Integer curr:items){
48             //Integer count = map.getOrDefault(curr, temp);
49             // map.put(curr, count + 1);
50             //output.add(curr);
51             Integer res=myMap.get(curr);
52             if(res == null) myMap.put(curr, myMap.get(curr) + 1);
53             else myMap.put(curr,1);
54         }
55         FreqComparator comp=new FreqComparator(myMap);
56         Collections.sort(items,comp);
57         return items;
58     }
59
60
61 }
```

Xcode Results    Custom Input    Run Only    Run Tests    Submit

## ANOTHER SOLUTION



```
37 }  
38 else if(myMap.ContainsKey(item))  
39 {  
40     int values=myMap[item];  
41     values=values+1;  
42     myMap[item]=values;  
43 }  
44 var keysorted=from entry in myMap orderby entry.Key,entry.Value descending select entry;  
45 var sorted=from entry in keysorted orderby entry.Value select entry;  
46 foreach(var k in sorted)  
47 {  
48     for(int i=0;i<k.Value;i++)  
49     {  
50         output.Add(k.Key);  
51     }  
52 }  
53 ms =  
54 mS =  
55  
56 The  
57 occ  
58 The  
59 > class Solution...  
• The  
occ
```

Run Code

Run Tests

## **VOWELS**

## **RESTRUCTURED ARRAY**

(another solution also available)

The screenshot shows a Java code editor with the following code:

```
language Java 8 • Autocomplete Ready
```

```
1 > import java.io.*;
2 class Result {
3
4     /*
5      * Complete the 'getElements' function below.
6      *
7      * The function is expected to return an INTEGER_ARRAY.
8      * The function accepts following parameters:
9      * 1. INTEGER_ARRAY arr
10     * 2. 2D_INTEGER_ARRAY queries
11     */
12
13     public static List<Integer> getElements(List<Integer> arr, List<List<Integer>> queries) {
14         // Write your code here
15         int s;
16         List<Integer> row = new ArrayList<Integer>();
17         for(int i=0;i<queries.size();i++) {
18             s=((queries.get(i).get(0)-1)*arr.get(0))+queries.get(i).get(1);
19             row.add(arr.get(s));
20         }
21         return (row);
22     }
23 }
24
25 > public class Solution { ... }
```

## IS IT POSSIBLE

```
static LinkedList<Pair<Integer, Integer>> pairs = new LinkedList<Pair<Integer, Integer>>();
```

```
public static String isItPossible(Integer a, Integer b, Integer c, Integer d){

    pairs.addLast(new Pair<Integer, Integer>(a,b));

    while (!pairs.isEmpty()){

        Pair<Integer, Integer> pair = pairs.poll();

        Integer key = pair.getKey();

        Integer value = pair.getValue();

        if(key.equals(c) &&
```

```
    value.equals(d)){  
        return "YES";  
    }  
    int sum=key+value;  
    if (sum<=c){  
        pairs.addLast(new Pair<Integer, Integer>(sum,value));  
    }  
    if (sum<=d){  
        pairs.addLast(new Pair<Integer, Integer>(key,sum));  
    }  
}  
  
return "NO";  
}
```

## VOWELS

Language Java 8 Autocomplete Ready

```
1 > import java.io.*;
16
17 // Complete the hasVowels function below.
18 static List<Integer> hasVowels(List<String> strArr, List<String> query) {
19     List<Character> vowels = new ArrayList<>();
20     List<Integer> result = new ArrayList<>();
21     vowels.add('a');
22     vowels.add('e');
23     vowels.add('i');
24     vowels.add('o');
25     vowels.add('u');
26     int count,l,r;
27     Character firstChar, lastChar;
28     int[] vowelArray = new int[strArr.size()];
29     for(int i=0; i < strArr.size(); i++){
30         String str = strArr.get(i);
31         firstChar = str.charAt(0);
32         lastChar = str.charAt(str.length()-1);
33         if(vowels.contains(firstChar) && vowels.contains(lastChar)){
34             vowelArray[i] = 1;
35         }
36     }
37     else
38         vowelArray[i] = 0;
39 }
40 for(String qry: query) {
41     count = 0;
42     String[] lrStringArray = qry.split("-");
43     l = Integer.parseInt(lrStringArray[0]) -1;
44     r = Integer.parseInt(lrStringArray[1]) - 1;
45     // System.out.println(l+" "+ r)
46     for(int i =l;i<= r;i++){
47         count += vowelArray[i];
48     }
49     result.add(count);
50 }
51 return result;
52
53 }
54
55 > public static void main(String[] args) throws IOException { ... }
```

Test Results Custom Input

ANOTHER SOLUTION

```
java 8  ✓ A Factorial Ready
```

contains  $n$  elements, of lowercase English letters, in the format  $/r$ , how many strings start at index  $r$  and end at index  $r$ .

e'

dash delimited string, the start and end indices of the interval from 'bcb' to 'e'. The interval, from 'bcb' to 'e', contains 3 elements, of lowercase English letters, in the format  $/r$ , how many strings start at index  $r$  and end at index  $r$ .

'e', 'i', 'o', 'u').

'bcb', 'ece', 'aa', 'e']

'[2-5', '2-2']

represent two dash delimited strings, the start and end indices of the interval from 'bcb' to 'e'. The interval, from 'bcb' to 'e', contains 3 elements, of lowercase English letters, in the format  $/r$ , how many strings start at index  $r$  and end at index  $r$ .

The first and last character, 'e', 'i', 'o', 'u').

The third interval, from 'bcb' to 'e', contains 3 elements, of lowercase English letters, in the format  $/r$ , how many strings start at index  $r$  and end at index  $r$ .

The queries is [2, 3, 0].

```
1 > import java.io.*;
16
17     // Complete the hasVowels function below.
18     static List<Integer> hasVowels(List<String> strArr, List<String> query) {
19         List<Character> vowels=new ArrayList<>();
20         List<Integer> result=new ArrayList<>();
21         vowels.add('a');
22         vowels.add('e');
23         vowels.add('i');
24         vowels.add('o');
25         vowels.add('u');
26         int count,l,r;
27         Character firstChar, lastChar;
28         int[] vowelArray=new int[strArr.size()];
29         for(int i=0;i<strArr.size(); i++) {
30             String str=strArr.get(i);
31             firstChar=str.charAt(0);
32             lastChar=str.charAt(str.length()-1);
33             if(vowels.contains(firstChar)&& vowels.contains(lastChar)) {
34                 vowelArray[i]=1;
35             }
36             else
37                 vowelArray[i]=0;
38         }
39         for(String qry: query) {
40             count=0;
41             String str=strArr.get(i);
42             firstChar=str.charAt(0);
43             lastChar=str.charAt(str.length()-1);
44             if(vowels.contains(firstChar)&& vowels.contains(lastChar)) {
45                 vowelArray[i]=1;
46             }
47             else
48                 vowelArray[i]=0;
49         }
50         for(String qry: query) {
51             count=0;
52             String[] lrStringArray=qry.split("-");
53             l=Integer.parseInt(lrStringArray[0])-1;
54             r=Integer.parseInt(lrStringArray[1])-1;
55             for(int i=l;i<=r;i++) {
56                 count +=vowelArray[i];
57             }
58             result.add(count);
59         }
60         return result;
61     }
62
63 > public static void main(String[] args) throws IOException { -
```

## AUTOSCALE POLICY

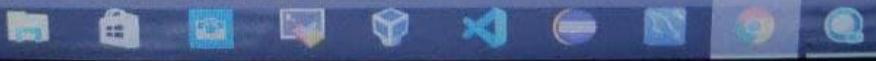
```
 29
30
31     public static int finalInstances(int instances, List<Integer> averageUtil){
32
33         for(int i=0; i<averageUtil.size(); i++){
34
35             if(averageUtil.get(i) < 25){
36
37                 if(instances != 1 && (instances & 1) == 0){
38
39                     instances = instances / 2;
40                     i = i + 10;
41
42                 }else if(instances != 1 && (instances & 1) == 1){
43
44                     instances = (instances / 2) + 1;
45                     i = i + 10;
46
47                 }else if(averageUtil.get(i) > 60){
48
49                     if( instances <= 100000000 ){
50                         instances = 2 * instances;
51                         i = i + 10;
52
53                     }
54
55                 }
56             }
57         }
58
59     }
60
61
62 }
```

## MINIMUM SWAPS

```
~ > import java.io.*;
4  class Result {
5
6      /*
7       * Complete the 'minimumSwaps' function below.
8       *
9       * The function is expected to return an INTEGER.
10      * The function accepts INTEGER_ARRAY popularity as parameter.
11      */
12
13
14      public static int minimumSwaps(List<Integer> popularity) {
15          // Write your code here
16          Collections.reverse(popularity);
17          int swapCounter=0;
18          for(int i=0; i<popularity.size(); i++){
19              if(popularity.get(i)==(i+1)){
20                  continue;
21              }
22              Collections.swap(popularity, i, popularity.get(i)-1);
23              swapCounter++;
24              i--;
25          }
26          return swapCounter;
27      }
28
29  }
30
31  > public class Solution {...
```

Test Results

Custom Input



## ANOTHER SOLUTION

The screenshot shows a code editor window with the following details:

- Language:** C#
- Autocomplete Ready:** Indicated by a checkmark.

```
1 > using System.CodeDom.Compiler;...
16 class Result
17 {
18
19     /*
20      * Complete the 'minimumSwaps' function below.
21      *
22      * The function is expected to return an INTEGER.
23      * The function accepts INTEGER_ARRAY popularity as parameter.
24     */
25
26     public static int minimumSwaps(List<int> popularity)
27     {
28         popularity.Reverse();
29         int swapCounter=0;
30         for(int i=0;i<popularity.Count;i++)
31         {
32             if(popularity[i]==(i+1))
33             {
34                 continue;
35             }
36             int temp = popularity[popularity[i]-1];
37             popularity[popularity[i]-1]=popularity[i];
38             popularity[i]=temp;
39             swapCounter++;
40             i--;
41         }
42         return swapCounter;
43     }
44 }
45 }
46 }
47 }
48 > class Solution...
```

## ARE ALMOST EQUIVALENT

### Test Equivalent Strings

are considered "almost equivalent" if the same length AND for each lowercase letter the number of occurrences of  $x$  in the two strings by no more than 3. There are two strings,  $s$  and  $t$ , that each contains  $n$  strings.  $s[i]$  and  $t[i]$  are the  $i^{th}$  pair of strings. They have the same length and consist of lowercase English letters. For each pair of strings, determine if they are almost equivalent.

```
ab', 'aaaaabb'
bc', 'abb']

Occurrences of 'a' in s[0] = 4 and in t[0] = 1,
a is 3
Occurrences of 'b' in s[0] = 2 and in t[0] = 4,
b is 2
Occurrences of 'c' in s[0] = 0 and in t[0] = 1,
c is 1
Number of occurrences of 'a', 'b', and 'c' in the
strings never differs by more than 3. This pair is
equivalent so the return value for this case
is 0
Occurrences of 'a' in s[1] = 5 and in t[1] = 1,
a is 4
Occurrences of 'b' in s[1] = 2 and in t[1] = 2,
b is 2
Occurrences of 'c' in s[1] = 0 and in t[1] = 1,
c is 1
```

```
public static List<String> areAlmostEquivalent(List<String> s, List<String> t) {
    // Write your code here
    List<String> result = new ArrayList<String>();
    for(int j=0; j<s.size();j++){
        String x=s.get(j);
        String y=t.get(j);
        if(x.length() != y.length()){
            result.add("NO");
        } else{
            Map<Character, Integer> mapx= new HashMap<Character, Integer>();
            Map<Character, Integer> mapy= new HashMap<Character, Integer>();
            for(int i = 0;i<x.length();i++){
                if(mapx.containsKey(x.charAt(i))){
                    mapx.put(x.charAt(i),mapx.get(x.charAt(i))+1);
                } else{
                    mapx.put(x.charAt(i), 1);
                }
            }
            for(int i=0;i<y.length();i++){
                if(mapy.containsKey(y.charAt(i))){
                    mapy.put(y.charAt(i),mapy.get(y.charAt(i))+1);
                } else{
                    mapy.put(y.charAt(i), 1);
                }
            }
            int count=0;
            for(char p : mapx.keySet()){
                if(mapy.containsKey(p)){
                    if(Math.abs(mapx.get(p)-mapy.get(p)) >= 4){
                        count++;
                    }
                }
            }
            for(char q : mapy.keySet()){
                if(mapx.containsKey(q)){
                    if(Math.abs(mapx.get(q)-mapy.get(q)) >= 4){
                        count++;
                    }
                } else{
                    if(mapy.get(q)>=4){
                        count++;
                    }
                }
            }
            result.add(count==0? "YES" : "NO");
        }
    }
    return result;
}
```

Line: 14 Col: 1

Run Code Run Tests Submit

Test Results Custom Input

### Test Equivalent Strings

are considered "almost equivalent" if the same length AND for each lowercase letter the number of occurrences of  $x$  in the two strings by no more than 3. There are two strings,  $s$  and  $t$ , that each contains  $n$  strings.  $s[i]$  and  $t[i]$  are the  $i^{th}$  pair of strings. They have the same length and consist of lowercase English letters. For each pair of strings, determine if they are almost equivalent.

```
'aaaaabb'
'abb'

Occurrences of 'a' in s[0] = 4 and in t[0] = 1,
a is 3
Occurrences of 'b' in s[0] = 2 and in t[0] = 4,
b is 2
Occurrences of 'c' in s[0] = 0 and in t[0] = 1,
c is 1
Number of occurrences of 'a', 'b', and 'c' in the
strings never differs by more than 3. This pair is
equivalent so the return value for this case
is 0
Occurrences of 'a' in s[1] = 5 and in t[1] = 1,
a is 4
Occurrences of 'b' in s[1] = 2 and in t[1] = 2,
b is 2
Occurrences of 'c' in s[1] = 0 and in t[1] = 1,
c is 1
```

Language Java 8 Autocomplete Ready

```
int count=0;
for(char p : mapx.keySet()){
    if(mapy.containsKey(p)){
        if(Math.abs(mapx.get(p)-mapy.get(p)) >= 4){
            count++;
        }
        break;
    }
}
else if(mapx.get(p)>=4){
    count++;
    break;
}
}
for(char q : mapy.keySet()){
    if(mapx.containsKey(q)){
        if(Math.abs(mapx.get(q)-mapy.get(q)) >= 4){
            count++;
        }
        break;
    }
}
else if(mapy.get(q)>=4){
    count++;
    break;
}
}
result.add(count==0? "YES" : "NO");
}
return result;
```

Line: 14 Col: 1

Run Code Run Tests Submit

Test Results Custom Input

## ARRANGING COINS

Language Java 8 Autocomplete Ready

```
24 public static void arrangeCoins(List<Long> coins) {  
25     // Write your code here  
26     for(Long a: coins){  
27         int res = (int)(Math.sqrt(8*a+1));  
28         Integer res = (int)((temp-1)/2);  
29         System.out.println(res);  
30     }  
31 }  
32  
33
```

Test Results Custom Input

## COUNT DUPLICATE ELEMENTS

The screenshot shows a Java code editor interface. At the top, there are tabs for "Activity Details - SumTotal" and "External Site". Below the tabs, the code language is set to "Java 8" and the status bar indicates "Autocomplete Ready". The main area contains the following Java code:

```
/*
 * The function is expected to return an INTEGER.
 * The function accepts INTEGER_ARRAY numbers as parameter.
 */

public static int countDuplicate(List<Integer> numbers) {
    // Write your code here
    int[] arr=new int[numbers.size()];
    for(int i=0;i<numbers.size();i++){
        arr[i]=numbers.get(i);
    }
    List<Integer> li=new ArrayList<Integer>();
    for(int i=0;i<arr.length;i++){
        for(int j=i+1;j<arr.length;j++){
            if(arr[i]==arr[j]){
                li.add(arr[i]);
            }
        }
    }
    List<Integer> listwithout=new ArrayList<>(new HashSet<>(li));
    return listwithout.size();
}
}

> public class Solution { ... }
```

At the bottom of the editor, there are buttons for "Test Results", "Custom Input", "Run Code" (which is highlighted in blue), and "Run Test". Below these buttons is a toolbar with various icons.

## ANOTHER SOLUTION

```
public static int countDuplicate(List<int> numbers)
{
    Dictionary<int,int> frequency = new Dictionary<int, int>();
    for(int i=0; i<numbers.Count;i++){

```

```
if(frequency.ContainsKey(numbers[i])){
    frequency[numbers[i]] = frequency[numbers[i]]+1;
}
else{
    frequency.Add(numbers[i],1);
}
}

List<int> result = new List<int>();
foreach(var keys in frequency.Keys){
    if(frequency[keys]>1){
        result.Add(keys);
    }
}
return result.Count;
}
```

```
75     * }
76     *
77     */
78
79     public static long getNumber(SinglyLinkedListNode binary) {
80         // Write your code here
81         long ans = 0;
82         SinglyLinkedListNode a = binary;
83         while(a!=null){
84             ans = (ans<<1)+(a.data);
85             a = a.next;
86         }
87         return ans;
88
89     }
90
91 }
```

## DISTINCT NUMBERS

```
static void countNumbers(List<List<Integer>> arr)
{
    int A[][]=new int[arr.size()][2];
    int x=0;
    for(List<Integer> item:arr)
    {
        for(int i=0;i<item.size();i++)
        {
            A[x][i]=item.get(i);
        }
        x++;
    }
    for(int i=0;i<arr.size();i++)
    {
        int l=A[i][0];
        int r=A[i][1];
        long count=0;
        for(int j=l;j<=r;j++)
        {
            int num=j;
            boolean visited[] = new boolean[10];
            while(num!=0)
            {
                if(visited[num%10])
                    break;
                visited[num%10]=true;
                num=num/10;
            }
            if(num==0)
                count++;
        }
        System.out.println(count);
    }
}
public static void main(String[] args) throws IOException {
```

```

25
26     public static int numPlayers(int k, List<Integer> scores) {
27         // Write your code here
28         if(k<=0)
29             return 0;
30         Collections.sort(scores, Collections.reverseOrder());
31         int rank=1;
32         int res=0;
33         for(int i=0;i<scores.size();i++){
34             if(i==0){
35                 rank=1;
36             }
37             else if(scores.get(i)!=scores.get(i-1)){
38                 rank=i+1;
39             }
40             if(rank<=k && scores.get(i)>0)
41                 res++;
42             else break;
43         }
44         return res;
45     }
46 }
47 }
48 > public class Solution {

```

## competitive gaming soln 2

3. Competitive Gaming

A group of friends are playing a video game together. During the game, each player earns a number of points. At the end of a round, players who achieve at least a certain rank get to "level up" their characters to gain increased abilities.

Given the scores of the players at the end of a round, how many players will be able to level up?

**Note:** Players with equal scores will have equal ranks, but the player with the next lower score will be ranked based on the position within the list of all players' scores. For example, if there are four players, and three players tie for first place, their ranks are 1, 1, 1, and 4.

**Note:** No player with a score of 0 can level up, regardless of rank.

Example  
Input  
Output

```

public static int numPlayers(int k, List<Integer> scores) {
    Collections.sort(scores, Collections.reverseOrder());
    int rank = 1;
    int res = 0;

    for(int i=0;i<scores.size();i++){
        if(i==0){
            rank = 1;
        }
        else if(scores.get(i)!=scores.get(i-1)){
            rank = i+1;
        }
        if(rank<=k && scores.get(i)>0) {
            res++;
        }
        else break;
    }
    return res;
}

```

Test Results   Custom Input   Run Code   Run Tests   2021-03-22 16:53

## Consolidated partitions

The screenshot shows a code editor window with the following details:

- Title Bar:** One Cognizant, Email - Das, Payel (...), iPRIMED LVDS: Log..., Cognizant, Chat | Microsoft Tea..., Attachments - One..., MyPractice, Other bookmarks, Reading list.
- Language:** Java 8
- Code Content:**

```
21  * The function accepts following parameters:
22  * 1. INTEGER_ARRAY used
23  * 2. INTEGER_ARRAY totalCapacity
24  */
25  public static int minPartitions(List<Integer> used, List<Integer> totalCapacity) {
26      // Write your code here
27      int siz = used.size();
28      int sum=0;
29      int count=0;
30      Collections.sort(totalCapacity);
31      Collections.reverse(totalCapacity);
32      int[] u= new int[siz];
33      int[] t=new int[siz];
34      for(int i=0; i<siz; i++){
35          u[i] = used.get(i);
36          t[i] = totalCapacity.get(i);
37          sum=sum+u[i];
38      }
39      for(int j=0; j<siz; j++){
40          sum=sum-t[j];
41          count++;
42          if(sum <= 0){
43              break;
44          }
45      }
46      return count;
47  }
48
49 }
50 > public class Solution { ... }
```
- Bottom Bar:** Run Code, Run Tests, Submit, Line: 14 Col: 1, ENG 16:12, Custom Input.

consolidate partitions soln 2

The screenshot shows a computer screen with a Java code editor open in a browser window. The URL is <https://www.hackerrank.com/test/4629r1s6c2h/questions/70cb1b7n4e8>. The code is for a problem titled "1. Consolidating Partitions". The code uses Java 7 syntax and includes imports for `java.util.List`, `java.util.stream.IntStream`, and `java.util.Collections`. It defines a method to calculate the minimum number of partitions needed given a list of used space units and a total capacity list.

```
15  /*Integer sum = used.stream()
16  .mapToInt(Integer::intValue).sum();*/
17  int partitions = used.size();
18  int counterTotal = 0;
19  int countTotal=0;
20  Collections.sort(totalCapacity);
21  Collections.reverse(totalCapacity);
22  Int[] arr1 = new Int[partitions];
23  Int[] arr2 = new Int[partitions];
24  for(int i=0;i<partitions;i++) {
25      arr1[i] = used.get(i);
26      arr2[i] = totalCapacity.get(i);
27      counterTotal=counterTotal+arr1[i];
28  }
29  for(int j=0;j<partitions;j++) {
30      counterTotal=counterTotal-arr2[j];
31      countTotal++;
32      if(counterTotal<=0) {
33          break;
34      }
35  }
36  return countTotal;
37 }
38 }
```

Below the code editor are tabs for "Test Results" and "Custom Input". At the bottom right, there are buttons for "Run Code", "Run Tests", and "Submit", along with a timestamp "2021/03/22 16:52". The status bar at the bottom of the screen shows "Line: 16 Col: 57" and a battery level of "45%".

## Construction management

HackerRank

1420/1420/1/questions/5e04d4c38

Management

They are building a new neighborhood, and they have the design. Each house will be built using materials (e.g., wood, brick, or concrete), but no two houses can be made of the same material. Because varying size and complexity, the cost of the houses varies. Given the cost of using each material, what is the minimum cost needed to build all houses?

There are  $n = 3$  houses to be built. Also,  $\text{cost} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{bmatrix}$ , denoting the cost of materials for each of the three houses. The minimum cost to build all the houses is 4, as seen in the figure below.

```
import java.util.*;  
class Result {  
    /*  
     * Complete the 'minCost' function below.  
     *  
     * The function is expected to return an INTEGER.  
     * The function accepts 2D_INTEGER_ARRAY cost as parameter.  
     */  
    public static int minCost(List<List<Integer>> cost) {  
        int len,min_cost;  
        Integer[][] array=new Integer[cost.size()][];  
        Integer[] array1=new Integer[0];  
        for(int i=0;i<cost.size();i++)  
            array[i]=cost.get(i).toArray(array1);  
        for(int i=1;i<array.length;i++){  
            array[i][0]=Math.min(array[i-1][1],array[i-1][2]);  
            array[i][1]=Math.min(array[i-1][0],array[i-1][2]);  
            array[i][2]=Math.min(array[i-1][0],array[i-1][1]);  
        }  
        len=array.length;  
        min_cost=Math.min(array[len-1][0],Math.min(array[len-1][1],array[len-1][2]));  
        return min_cost;  
    }  
}  
public class Solution {  
}
```

The cheapest material is the first one, which is material 1. For the first house, the materials cost the same as with the same material can't be used because the constraint. The next best option is the second material, which is the cheapest material for the third house is material 1. Therefore, the total cost to build all three houses is  $1 + 2 + 1 = 4$ .

minCost in the editor below.

String parameter

Test Results

Custom Input

SLOWEST KEY PRESS

**2. Slowest Key Press**

Engineers have redesigned a keypad used by ambulance drivers in urban areas. In order to determine which key takes the longest time to press, the keypad is tested by a driver. Given the results of that test, determine which key takes the longest to press.

**Example**  
`keyTimes = [[0, 2], [1, 5], [0, 9], [2, 15]]`

Elements in `keyTimes[i][0]` represent encoded characters in the range `ascii[a-z]` where `a = 0, b = 1, ..., z = 25`. The second element, `keyTimes[i][1]` represents the time the key is pressed since the start of the test. The elements will be given in ascending time order. In the example, keys pressed, in order are `0102encoded = abac` at times `2, 5, 9, 15`. From the start time, it took `2 - 0 = 2` to press the first key, `5 - 2 = 3` to press the second, and so on. The longest time it

```
public static char slowestKey(List<List<Integer>> keyTimes) {
    int arr[][] = new int[keyTimes.size()][2];
    int arr2[][] = new int[keyTimes.size()][2];

    for(int i=0;i<keyTimes.size();i++) {
        List<Integer> list = keyTimes.get(i);
        for(int j=0;j<2;j++) {
            arr[i][j] = list.get(j);
            arr2[i][j] = list.get(j);
        }
    }

    int prev = 0;
    for(int i=0;i<keyTimes.size();i++) {
        arr2[i][1] = arr[i][1]-prev;
        prev = arr[i][1];
    }

    int max = Integer.MIN_VALUE;
    int ans = 0;
    for(int i=0;i<keyTimes.size();i++) {
        int value = arr2[i][1];
        if(value>max) {
            ans = arr2[i][0];
            max=arr2[i][1];
        }
    }

    int result = 97+ans;
    char c= (char)result;
    return c;
}
```

Test Results   Custom Input   Run Code   Run Tests   Submit   Line: 18 Col: 15   2021/03/22 16:53   453 PM   22-Mar-21

**2. Slowest Key Press**

Engineers have redesigned a keypad used by ambulance drivers in urban areas. In order to determine which key takes the longest time to press, the keypad is tested by a driver. Given the results of that test, determine which key takes the longest to press.

**Example**  
`keyTimes = [[0, 2], [1, 5], [0, 9], [2, 15]]`

Elements in `keyTimes[i][0]` represent encoded characters in the range `ascii[a-z]` where `a = 0, b = 1, ..., z = 25`. The second element, `keyTimes[i][1]` represents the time the key is pressed since the start of the test. The elements will be given in ascending time order. In the example, keys pressed, in order are `0102encoded = abac` at times `2, 5, 9, 15`. From the start time, it took `2 - 0 = 2` to press the first key, `5 - 2 = 3` to press the second, and so on. The longest time it

```
int prev = 0;
for(int i=0;i<keyTimes.size();i++) {
    arr2[i][1] = arr[i][1]-prev;
    prev = arr[i][1];
}

int max = Integer.MIN_VALUE;
int ans = 0;
for(int i=0;i<keyTimes.size();i++) {
    int value = arr2[i][1];
    if(value>max) {
        ans = arr2[i][0];
        max=arr2[i][1];
    }
}

int result = 97+ans;
char c= (char)result;
return c;
```

> public class Solution { ... }

Test Results   Custom Input   Run Code   Run Tests   Submit   Line: 18 Col: 15   2021/03/22 16:53   453 PM   22-Mar-21

AREA OF THE BOX

The screenshot shows a Java code editor interface. At the top, there's a navigation bar with several icons and a dropdown menu set to "Java 8". Below the menu, the status bar displays "Complete Ready ⓘ" and "Line: 43". The main area contains the following Java code:

```
// write your code here
int n, m ;
int min ;
long res=0;
List<Long> fin = new ArrayList<>();

for(List<Integer> lst : queries) {
    n = lst.get(0);
    m = lst.get(1);
    min = Math.min(m,n);
    res = 0 ;

    for(int i=1; i<=min; ++i) {
        res += (n-i+1) * (m-i+1);
    }
    fin.add(res);
}

return fin;
}
```

The code implements a function that takes a list of queries (each query being a list of two integers) and returns a list of results. For each query, it calculates the sum of products of pairs of numbers from  $n$  down to  $i$  and from  $m$  down to  $i$ , where  $i$  ranges from 1 to  $\min(n, m)$ . The results are stored in a list named `fin`.

## CAR INHERITANCE

```
class WagonR extends Car{  
    int mileage;  
    public WagonR(Integer mileage){  
        super(false, "4");  
        this.mileage = mileage;  
    }  
  
    @Override  
    public String getMileage(){  
        String mil=Integer.toString(mileage);  
        return mileage+" kmpl";  
    }  
}
```

```
class HondaCity extends Car{  
    int mileage;  
    public HondaCity(Integer mileage){  
        super(true, "4");  
        this.mileage = mileage;  
    }  
  
    @Override  
    public String getMileage(){  
        String mil=Integer.toString(mileage);  
        return mil+" kmpl";  
    }  
}
```

```
class InnovaCrysta extends Car{  
    int mileage;  
    public InnovaCrysta(Integer mileage){  
  
        super(false, "6");  
        this.mileage = mileage;  
  
    }  
  
    @Override  
    public String getMileage(){  
        String mil=Integer.toString(mileage);  
        return mil+" kmpl";  
    }  
}
```

## PRISON BREAK

The screenshot shows a Java code editor with the following details:

- Language: Java 8
- Autocomplete Ready
- Code content (lines 1-57):

```
public static long prison(int n, int m, List<Integer> h, List<Integer> v) {
    // Write your code here
    boolean[] xBol=new boolean[n+1];
    Arrays.fill(xBol,true);
    boolean[] yBol=new boolean[m+1];
    Arrays.fill(yBol,true);
    for(int x:h){
        xBol[x]=false;
    }
    for(int y:v){
        yBol[y]=false;
    }
    int cx=0,xMax=Integer.MIN_VALUE,cy=0,yMax=Integer.MIN_VALUE;
    for(int i=1;i<=n;i++){
        if(xBol[i]){
            cx=0;
        }else{
            cx++;
            xMax=Math.max(xMax,cx);
        }
    }
    for(int i=1;i<=m;i++){
        if(yBol[i]){
            cy=0;
        }else{
            cy++;
            yMax=Math.max(yMax,cy);
        }
    }
    return (xMax+1)*(yMax+1);
}
```

- Test Results and Custom Input tabs are visible at the bottom.
- Run button is visible on the right.
- DELL logo is visible at the bottom center of the screen.

Check this also: <https://leetcode.com/discuss/interview-question/1002082/twillio-qa-prison-break>

## COUNT STRING PERMUTATIONS

The screenshot shows a Java code editor with the following code:

```
1 > import java.io.*;...
14
15 class Result {
16
17     /*
18      * Complete the 'countPerms' function below.
19      *
20      * The function is expected to return an INTEGER.
21      * The function accepts INTEGER n as parameter.
22      */
23
24     public static int countPerms(int n) {
25         // Write your code here
26         long[][] per=new long[n+1][5];
27         int MOD=(int)(1e9+7);
28         for(int i=0;i<5;i++){
29             per[1][i]=1;
30         }
31     }
32
33     int[][] relation=new int[][][]{
34         {1},{0,2},{0,1,3,4},{2,4},{0}
35     };
36
37     for(int i=1;i<n;i++){
38         for(int u=0;u<5;u++){
39             per[i+1][u]=0;
40             for(int v:relation[u]){
41                 per[i+1][u]+=per[i][v]%MOD;
42             }
43         }
44     }
45     long ans=0;
46     for(int i=0;i<5;i++){
47         ans=(ans+per[n][i])%MOD;
48     }
49     return (int)ans;
50 }
51
52 }
53
54 > public class Solution { ... }
```

The code implements a dynamic programming solution to calculate the number of permutations of a string of length n. It uses a 2D array 'per' where per[i][j] represents the number of permutations of a string of length i ending in character j. The base case is per[1][j] = 1 for all j from 0 to 4. The transition rule is per[i+1][u] = sum(per[i][v] for v in relation[u]). The final answer is the sum of per[n][j] for all j from 0 to 4.

### **LIST INHERITANCE**

<https://allhackerranksolutionsbykaira.blogspot.com/2020/08/list-inheritance-hackerrank-solution.html>

### **DEVICE NAME SYSTEM**

<https://www.homeworklib.com/question/1486956/in-java-6-device-name-system-suggested-problem>

### **SPORT INHERITANCE**

<https://www.goeduhub.com/2885/program-inherit-cricketplayer-footballplayer-hockeyplayer>

## MERGE 2 ARRAYS

Language: Java 8      Autocomplete Ready ⓘ

```
22     * 1. INTEGER_ARRAY a
23     * 2. INTEGER_ARRAY b
24     */
25
26     public static List<Integer> mergeArrays(List<Integer>
27         // Write your code here
28         List<Integer> out=new ArrayList<Integer>();
29         // List<integer> out1=new ArrayList<Integer>();
30         out.addAll(a);
31         out.addAll(b);
32         out.sort(Comparator.naturalOrder());
33         return out;
34     }
35
36 }
37
38 > public class Solution {...
```

### Test Results

### Custom Input

Compiled successfully. All available test cases passed

Test case 10	□
Test case 11	□
Test case 12	□
Test case 13	□
Test case 14	□
Test case 15	□
Test case 16	□

Your Output (stdout)

```
1 6
2 1
```

## CONDENSED LIST

```
77  */
78
79  public static SinglyLinkedListNode condense(SinglyLinkedListNode head) {
80      // Write your code here
81      HashSet<Integer> hSet = new HashSet<>();
82      SinglyLinkedListNode ptr1;
83      SinglyLinkedListNode ptr2;           I
84      ptr1 = head;
85      ptr2 = null;
86      while(ptr1!=null){
87          int val = ptr1.data;
88          if(hSet.contains(val))ptr2.next = ptr1.next;
89          else{
90              hSet.add(val);
91              ptr2 = ptr1;
92          }
93          ptr1 = ptr1.next;
94      }
95      return head;
96  }
```

## WAYS TO SUM

The screenshot shows a Java code editor with the following details:

- Title:** Ways to Sum
- Language:** Java 8
- Code Content:**

```
22  * 1. INTEGER total
23  * 2. INTEGER k
24  */
25
26  public static int ways(int total, int k) {
27      // Write your code here
28      int[][] b=new int[k+1][total+1];
29      for(int i=1;i<=total;i++){
30          b[1][i]=1;
31      }
32      for(int i=1;i<=k;i++){
33          b[i][0]=1;
34      }
35      for(int i=2;i<=k;i++){
36          for(int j=1;j<=total;j++){
37              if(j==i){
38                  b[i][j]=b[i][j-i]+b[i-1][j];
39              }else{
40                  b[i][j]=b[i-1][j];
41              }
42          }
43      }
44      return Math.abs(b[k][total]);
45  }
46
47
48
```
- Test Results:** The editor shows sample inputs and outputs:
  - Input: 8  
Output: 5
  - Input: 1  
Output: 1
  - Input: 2  
Output: 2
  - Input: 3  
Output: 4
  - Input: 4  
Output: 7
  - Input: 5  
Output: 11
  - Input: 6  
Output: 17
  - Input: 7  
Output: 25
  - Input: 8  
Output: 35
  - Input: 9  
Output: 51
  - Input: 10  
Output: 77
  - Input: 11  
Output: 108
  - Input: 12  
Output: 145
  - Input: 13  
Output: 187
  - Input: 14  
Output: 238
  - Input: 15  
Output: 295
  - Input: 16  
Output: 360
  - Input: 17  
Output: 435
  - Input: 18  
Output: 520
  - Input: 19  
Output: 615
  - Input: 20  
Output: 720
  - Input: 21  
Output: 835
  - Input: 22  
Output: 960
  - Input: 23  
Output: 1095
  - Input: 24  
Output: 1235
  - Input: 25  
Output: 1380
  - Input: 26  
Output: 1535
  - Input: 27  
Output: 1700
  - Input: 28  
Output: 1875
  - Input: 29  
Output: 2050
  - Input: 30  
Output: 2230
  - Input: 31  
Output: 2415
  - Input: 32  
Output: 2600
  - Input: 33  
Output: 2785
  - Input: 34  
Output: 2970
  - Input: 35  
Output: 3155
  - Input: 36  
Output: 3340
  - Input: 37  
Output: 3525
  - Input: 38  
Output: 3710
  - Input: 39  
Output: 3895
  - Input: 40  
Output: 4080
  - Input: 41  
Output: 4265
  - Input: 42  
Output: 4450
  - Input: 43  
Output: 4635
  - Input: 44  
Output: 4820
  - Input: 45  
Output: 5005
  - Input: 46  
Output: 5190
  - Input: 47  
Output: 5375
  - Input: 48  
Output: 5560
- Status:** Autocomplete Ready
- Line/Column:** Line: 14 Col: 1

## BALANCED ARRAY

The screenshot shows a programming environment with the following details:

- Header:** Practice, Java (1.8), Test against custom input.
- Problem Title:** Balanced Array
- Basic Statistics:** Accuracy: 54.59%, Submissions: 10939, Points: 1
- Description:** Given an array of even size N, task is to find minimum value that can be added to an element so that array become balanced. An array is balanced if the sum of the left half of the array elements is equal to the sum of right half.
- Example 1:**
  - Input:** N = 4, arr[] = {1, 5, 3, 2}
  - Output:** 1
  - Explanation:** Sum of first 2 elements is 1 + 5 = 6, Sum of last 2 elements is 3 + 2 = 5, To make the array balanced you can add 1.
- Example 2:** (not visible in the screenshot)
- Code Editor:** Java code for Solution class:

```
14 //User function Template for Java
15
16
17 class Solution
18 {
19     long minValueToBalance(long a[],int n)
20     {
21         long sum1=0;
22         long sum2=0;
23         for(int i=0;i<n/2;i++){
24             sum1+=a[i];
25         }
26         for(int j=n/2;j<n;j++){
27             sum2+=a[j];
28         }
29     }
30     return Math.abs(sum1-sum2);
31
32
33
34 }
35
36
37
38 // } Driver Code Ends
```

Buttons at the bottom: Sun icon, Compile & Run, Submit.

The screenshot shows a browser window with the following details:

- Title:** 4. Balanced Array
- Description:** Given an array of numbers, find the index of the smallest array element (the pivot), for which the sums of all elements to the left and to the right are equal. The array may not be reordered.
- Example:** arr=[1,2,3,4,6]
  - the sum of the first three elements, 1+2+3=6. The value of the last element is 6.
  - Using zero based indexing, arr[3]=4 is the pivot between the two subarrays.
  - The index of the pivot is 3.
- Function Description:** Complete the function `balancedSum` in the editor below.
- Code Editor:** Java code for `balancedSum` function:

```
18 /**
19 * Complete the 'balancedSum' function below.
20 *
21 * The function is expected to return an INTEGER.
22 * The function accepts INTEGER_ARRAY arr as parameter.
23 */
24
25
26 public static int balancedSum(List<Integer> arr)
27 {
28     int n=arr.Count;
29     int[] fsum=new int[n];
30     int[] bsum=new int[n];
31     fsum[0]=arr[0];
32     for(int i=1;i<n;i++)
33     {
34         fsum[i]=fsum[i-1]+arr[i];
35     }
36     bsum[n-1]=arr[n-1];
37     for(int j=n-2;j>=0;j--)
38     {
39         bsum[j]=bsum[j+1]+arr[j];
40     }
41     for(int k=1;k<n-1;k++)
42     {
43         if(fsum[k]==bsum[k])
44             return k;
45     }
46 }
47
48
49
50 }
51 > class Solution...
```

Buttons at the bottom: Input Format for Custom Testing, Test Results, Custom Input.

## EMPLOYEE IMPLEMENTATION

<https://github.com/vadym-usatiuk/Hackerrank-Employee-Implementation/blob/master/src/Solution.java>

## **WORK SCHEDULE**

```
1 > import java.io.*;
14 class Result {
15
16     /*
17      * Complete the 'findSchedules' function below.
18      *
19      * The function is expected to return a STRING_ARRAY.
20      * The function accepts following parameters:
21      * 1. INTEGER workHours
22      * 2. INTEGER dayHours
23      * 3. STRING pattern
24     */
25
26     static List<String> timeTable = new ArrayList<>();
27     public static List<String> createTable(String[] listPatterns, int idx, int variation, int
dayHours){
28         if(idx==listPatterns.length){
29             if(variation==0){
30                 timeTable.add(String.join("", listPatterns));
31             }
32             return timeTable;
33         }
34         if(listPatterns[idx].equals(String.valueOf("?"))){
35             for(int i =0;i<=dayHours;i++){
36                 String buffer = listPatterns[idx];
37
38                 public static List<String> construct(int workHours, int dayHours, String pattern){
39
40                     Integer worked_hrs = 0;
41                     int variation = 0;
42                     String[] list_pattern = pattern.split("");
43                     for(String s: list_pattern){
44                         if(!String.valueOf("?").equals(s)){
45                             worked_hrs+=Integer.valueOf(s);
46                         }
47                     }
48                     variation = workHours - worked_hrs;
49                     createTable(list_pattern, 0, variation, dayHours);
50                     return timeTable;
51
52                 }
53                 public static List<String> findSchedules(int workHours, int dayHours, String pattern) {
54                     // Write your code here
55                     return construct(workHours, dayHours, pattern);
56                 }
57             }
58         }
59     }
60
61     */
62 }
63
64
65
66
67
68 }
69 > public class Solution {
```

Test Results

Custom Input

Run Code

Run Tests

Submit

```
Language: Java 8 | Autocomplete Ready | Line: 65 Col: 12
```

```
34     if(listPatterns[idx].equals(String.valueOf("?"))){
35         for(int i = 0; i <= dayHours; i++){
36             String buffer = listPatterns[idx];
37             listPatterns[idx] = String.valueOf(i);
38             createTable(listPatterns, idx+1, variation-i, dayHours);
39             listPatterns[idx] = buffer;
40         }
41     } else{
42         createTable(listPatterns, idx+1, variation, dayHours);
43     }
44     return timeTable;
45 }
46 }
47 public static List<String> construct(int workHours, int dayHours, String pattern){
48
49     Integer worked_hrs = 0;
50     int variation = 0;
51     String[] list_pattern = pattern.split("");
52     for(String s: list_pattern){
53         if(!String.valueOf("?").equals(s)){
54             worked_hrs+=Integer.valueOf(s);
55         }
56     }
57     variation = workHours - worked_hrs;
58     createTable(list_pattern, 0, variation, dayHours);
```

## COUNT OPTIONS

```
 */
public static long countOptions(int n, int k) {
    // Write your code here
    if(n < k) return 0;
    int[][] dp = new int[n+1][k+1];
    for(int i=0; i <= n; i++)
        dp[i][1] = 1;
    dp[0][0] = 1;
    for(int i=1; i <= n; i++)
        for(int j=2; j <= k; j++){
            dp[i][j] = dp[i-1][j-1];
            if(i >= 2*j) dp[i][j] += dp[i-j][j];
        }
    return dp[n][k];
```

Count the number of ways to divide N in k groups incrementally

<https://www.geeksforgeeks.org/count-the-number-of-ways-to-divide-n-in-k-groups-incrementally/>

## PERFECT SUBSTRING

kerrank.com/test/25iher3t6te/questions/7o8lell38fr

Language Java 8 Autocomplete Ready

```
27     return false;
28 }
29 }
30 }
31 }
32 public static int perfectSubstring(String s, int k) {
33 // Write your code here
34 int res = 0;
35 for(int i = 0; i < s.length(); i++) {
36     int[] arr = new int[10];
37     for(int j = i; j < s.length(); j++) {
38         if(j > i + (10*k))
39             break;
40         char ch = s.charAt(j);
41         arr[ch - '0']++;
42         if(check(arr, k)) res++;
43     }
44 }
45 }
46 return res;
47 }
48 }
49 }
50 }
51 > public class Solution { ...
```

Line: 25

kerrank.com/test/25iher3t6te/questions/7o8lell38fr

Language Java 8 Autocomplete Ready

```
1 > import java.io.*;
14
15 class Result {
16
17     /*
18     * Complete the 'perfectSubstring' function below.
19     *
20     * The function is expected to return an INTEGER.
21     * The function accepts following parameters:
22     * 1. STRING s
23     * 2. INTEGER k
24     */
25 public static boolean check(int[] arr, int k) {
26     for(int val : arr) {
27         if(val != 0 && val != k)
28             return false;
29     }
30     return true;
31 }
32 public static int perfectSubstring(String s, int k) {
33 // Write your code here
34 int res = 0;
35 for(int i = 0; i < s.length(); i++) {
36     int[] arr = new int[10];
37     for(int j = i; j < s.length(); j++) {
38         if(j > i + (10*k))
39             break;
40         char ch = s.charAt(j);
41         arr[ch - '0']++;
42         if(check(arr, k)) res++;
43     }
44 }
45 }
46 return res;
47 }
```

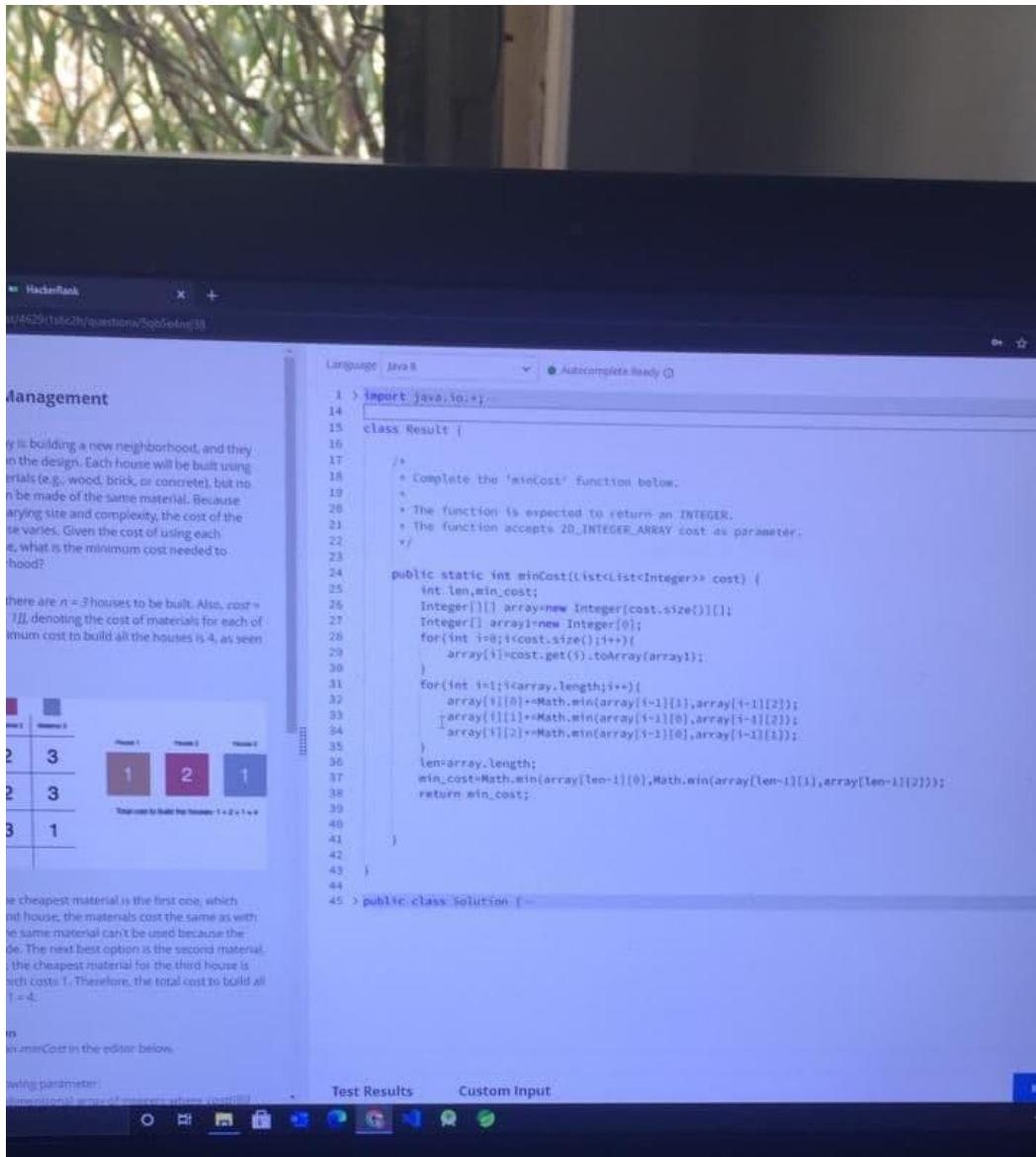
Line: 25 C

Test Results Custom Input Run Code Run Tests Submit

## LOAD BALANCING

<https://leetcode.com/discuss/interview-question/719447/wish-online-assessment-question>

## CONSTRUCTION MANAGEMENT



## IS POSSIBLE

The screenshot shows a Java code editor with the following details:

- Language: Java 8
- Autocomplete Ready
- Code content:

```
import java.io.*;
class Result {

    /**
     * Complete the 'isPossible' function below.
     *
     * The function is expected to return a STRING.
     * The function accepts following parameters:
     * 1. INTEGER a
     * 2. INTEGER b
     * 3. INTEGER c
     * 4. INTEGER d
     */
    public static String isPossible(int a, int b, int c, int d) {
        // Write your code here
        return gcd(a,b)==gcd(c,d) ? "Yes": "No";
    }

    public static long gcd(long p,long q){
        return q==0 ? p : gcd(q,p%q);
    }
}

> public class Solution {
```

## ANOTHER SOLUTION

```
static LinkedList<Pair<Integer,Integer>> pairs = new LinkedList<Pair<Integer, Integer>>();
```

```
public static String isItPossible(Integer a, Integer b, Integer c, Integer d){

    pairs.addLast(new Pair<Integer, Integer>(a,b));

    while (!pairs.isEmpty()){

        Pair<Integer, Integer> pair = pairs.poll();

        Integer key = pair.getKey();

        Integer value = pair.getValue();
```

```
if(key.equals(c) &&
   value.equals(d)){
    return "YES";
}

int sum=key+value;
if (sum<=c){
    pairs.addLast(new Pair<Integer, Integer>(sum,value));
}
if (sum<=d){
    pairs.addLast(new Pair<Integer, Integer>(key,sum));
}

}

return "NO";
}
```

## GROUP DIVISION

The screenshot shows a programming challenge titled "2. Group Division" on the HackerRank platform. The challenge involves grouping students based on their skill levels. The code editor contains a C# solution for the "groupDivision" function. The constraints and example input are also visible.

**Challenge Details:**

**ALL** 1h 22m left

**2. Group Division**

A university has admitted a group of  $n$  students with varying skill levels. To better accommodate the students, the university has decided to create classes tailored to the skill levels. A placement examination will return a skill level that will be used to group the students, where  $levels[i]$  represents the skill level of student  $i$ . All students within a group must have a skill level within  $maxSpread$ ; a specified range of one another. Determine the minimum number of classes that must be formed.

**Example**

$n = 5$   
 $levels = [1, 4, 7, 3, 4]$   
 $maxSpread = 2$

The students in any group must be within  $maxSpread - 2$  levels of each other. In this case, one optimal grouping is  $(1, 3), (4, 4)$ , and  $(7)$ . Another possible grouping is  $(1, 3, 4), (7)$ . There is no way to form fewer than 3 groups.

**Function Description**

Complete the function `groupDivision` in the editor below.

`groupDivision` has the following parameter(s):

- `int levels[n]`: the skill level for each student
- `int maxSpread`: the maximum allowed skill level difference between any two members of a group

Returns

- `int`: the minimum number of groups that can be formed

**Constraints**

<https://www.hackerrank.com/test/fesebq3k19o/questions/sf15042b28>

**Test Results** **Custom Input**

Language: C# Autocomplete Loading...

```
1 > using System.CodeDom.Compiler;...
16 class Result
17 {
18
19     /*
20     * Complete the 'groupDivision' function below.
21     *
22     * The function is expected to return an INTEGER.
23     * The function accepts following parameters:
24     * 1. INTEGER_ARRAY levels
25     * 2. INTEGER maxSpread
26
27
28     public static int groupDivision(List<int> levels, int maxSpread)
29     {
30         levels.Sort();
31         int result=1;
32         int previousOne=levels[0];
33         for(int i=1;i<levels.Count;i++)
34         {
35             if(previousOne+maxSpread<levels[i])
36             {
37                 previousOne=levels[i];
38                 result++;
39             }
40         }
41         return result;
42     }
43
44 }
45 > Close Solution...
```

## SORT AN ARRAY

**1. Sort an Array**

Given an array of integers, any array element can be moved to the end in one move. Determine the minimum number of moves required to sort the array, ascending.

**Example:**  
 $arr = [5, 1, 3, 2]$

- Move the value  $arr[2] = 3$  to the end to get  $arr = [5, 1, 2, 3]$ .
- Move  $arr[0] = 5$  to the end to achieve the sorted array,  $arr = [1, 2, 3, 5]$ .
- The minimum number of moves required to sort the array is 2.

**Function Description**  
Complete the function `getMinimumMoves` in the editor below.

`getMinimumMoves` has the following parameter:

- $int arr[]$ : an array of integers

Returns:

- $int$ : the minimum number of moves needed to sort the array in ascending order

**Constraints**

- $1 \leq n \leq 10^5$
- $0 \leq arr[i] \leq 10^9$
- array elements are distinct

```

25     List<int> sortedArray=new List<int>();
26
27     int temp=0;
28     int check=0;
29     int output=0;
30     foreach(int a in arr)
31     {
32         sortedArray.Add(a);
33     }
34     sortedArray.Sort();
35     if(arr.SequenceEqual(sortedArray))
36     {
37         return 0;
38     }
39     for(int i=0;i<sortedArray.Count;i++)
40     {
41         temp=sortedArray[i];
42         while(check<sortedArray.Count && arr[check]!=temp)
43         {
44             check=check+1;
45             output=output+1;
46         }
47         check=check+1;
48         if(check==sortedArray.Count)
49         {
50             break;
51         }
52     }
53     return output;
54 }
55 }
56 }
57 }
58 }
59 }
60 }
61 }
62 > class Solution {

```

## EFFICIENT SHIPPING

The screenshot shows a code editor interface with the following details:

- Language:** C#
- Status:** Autocomplete Ready

The code block contains the following content:

```
* Complete the 'getMaxUnits' function below.  
*  
* The function is expected to return a LONG_INTEGER.  
* The function accepts following parameters:  
* 1. LONG_INTEGER_ARRAY boxes  
* 2. LONG_INTEGER_ARRAY unitsPerBox  
* 3. LONG_INTEGER truckSize  
*/  
  
private static long maxvalue=long.MinValue;  
public static long getMaxUnits(List<long> boxes, List<long> unitsPerBox, long truckSize)  
{  
    long capt=0;  
  
    int boxsize=boxes.Count();  
    int j;  
    long endingmaxvalue=0;  
    for(j=0;j<boxsize;j++)  
    {  
        capt+=boxes[j];  
        if(capt<=truckSize)  
        {  
            endingmaxvalue+=boxes[j]*unitsPerBox[j];  
            if(maxvalue<endingmaxvalue){  
                maxvalue=endingmaxvalue;  
            }  
        }  
    }  
    return maxvalue;  
}
```

At the bottom of the code editor, there are tabs for "Results" and "Custom Input", and a blue "Run Code" button.

## THE RESTRUCTURED ARRAY

```
23     * The function is expected to return an INTEGER_ARRAY.
24     * The function accepts following parameters:
25     * 1. INTEGER_ARRAY arr
26     * 2. 2D_INTEGER_ARRAY queries
27 */
28
29     public static List<int> getElements(List<int> arr, List<List<int>> queries)
30     {
31
32         List<int> new_array = new List<int>();
33         int n;
34         for(int i =0;i<queries.Count;i++){
35             n = ((queries[i][0]-1)*arr[0])+queries[i][1];
36             new_array.Add(arr[n]);
37         }
38         return new_array;
39     }
40
41 }
```

### PRODUCT OF MAXIMUM AND MINIMUM IN A DATA SET

```
28
29     public static List<long> maxMin(List<string> operations, List<
30     {
31         List<long> parray=new List<long>();
32         List<long> elements=new List<long>();
33         long max=long.MinValue;
34         long min=long.MaxValue;
35
36         for(int i=0;i<operations.Count;i++)
37         {
38             if(operations[i].Equals("push"))
39             {
40                 elements.Add((long)x[i]);
41                 if(max<x[i])
42                 {
43                     max=x[i];
44                 }
45                 if(min>x[i])
46                 {
47                     min=x[i];
48                 }
49                 parray.Add(min*max);
50             }
51             else
52             {
53                 elements.Remove((long)x[i]);
54                 min=elements.Min();
55                 max=elements.Max();
56                 parray.Add(min*max);
57             }
58         }
59     }
60 }
```

```
for(int i=0;i<operations.Count;i++)
{
    if(operations[i].Equals("push"))
    {
        elements.Add((long)x[i]);
        if(max<x[i])
        {
            max=x[i];
        }
        if(min>x[i])
        {
            min=x[i];
        }
        parray.Add(min*max);
    }
    else
    {
        elements.Remove((long)x[i]);
        min=elements.Min();
        max=elements.Max();
        parray.Add(min*max);
    }
}
return parray;
}
}
}
> class Solution ...
```

# Frequency sort

```
s[] size n =  
Language Java 8 Autocomplete Ready  
20 * The function accepts INTEGER_ARRAY items as parameter.  
21 */  
22  
23 public static List<Integer> itemsSort(List<Integer> items) {  
24 // Write your code here  
25 Map<Integer, Integer> map = new HashMap<>();  
26  
27 for (int i = 0; i < items.size(); i++) {  
28 map.put(items.get(i), map.getOrDefault(items.get(i), 0) + 1);  
29 }  
30  
31 Collections.sort(items, (n1, n2) ->{  
32 int freq1 = map.get(n1);  
33 int freq2 = map.get(n2);  
34  
35 if (freq1 != freq2) {  
36 return freq1 - freq2;  
37 }  
38  
39 return n1 - n2;  
40 });  
41  
42 return items;  
43 }  
44 }
```

## Twin Strings

```
public static List<string> twins(List<string> a, List<string> b)  
{  
    var res = new string[a.Count];  
    for(var i=-1; ++i <a.Count;)  
    {  
        string aitem = a[i],bitem=b[i];  
        var twinis = aitem.Length==bitem.Length;  
        if(twinis)  
        {  
            var tomap = new int['z' - 'a'+1,2];  
            for(var j=-1; ++j<bitem.Length;)  
            {  
                ++tomap[bitem[j] -'a',j&1];  
            }  
            for(var j=-1; ++j<bitem.Length;)  
            {  
                if(--tomap[aitem[j]-'a',j&1]<0)  
                {  
                    twinis = false;  
                    break;  
                }  
            }  
        }  
        res[i]=twinis ? "Yes" : "No";  
    }  
    return res.ToList();  
}
```

# Competitive gaming

```
public static int numPlayers(int k, List<int> scores)
{
    var ranks = new List<int>();
    var groups = scores.Where(f => f != 0).OrderByDescending(f => f).GroupBy(g => g).ToList();
    var rank = 1;
    foreach (var g in groups)
    {
        var count = g.Count();
        for (var j = 0; j < count; j++)
        {
            ranks.Add(rank);
        }
        rank = rank + count;
    }
    return ranks.Count(r => r <= k);
}
```

## 1.construction management

```
public static int minCost(List<List<int>> cost)
{
    for(int i=1;i<cost.Count;i++){
        cost[i][0]+=Math.Min(cost[i-1][1],cost[i-1][2]);
        cost[i][1]+=Math.Min(cost[i-1][0],cost[i-1][2]);
        cost[i][2]+=Math.Min(cost[i-1][0],cost[i-1][1]);
    }
    int len=cost.Count;
    int min_cost=Math.Min(cost[len-1][0],Math.Min(cost[len-1][1],cost[len-1][2]));
    return min_cost;
}
```

## 2.Area of the box

```
class Result {
```

```
    public static List<Long> numberOfWays(List<List<Integer>> queries) {
        // Write your code here
        int n,m;
        int min;
        long res=0;
        List<Long>fin=new ArrayList<>();
        for(List<Integer> lst:queries){
            n=lst.get(0);
            m=lst.get(1);
            min=Math.min(m,n);
            res=0;
            for(int i=1;i<=min;++i){
                res+=(n-i+1)*(m-i+1);
            }
            fin.add(res);
        }
        return fin;
    }
}
```