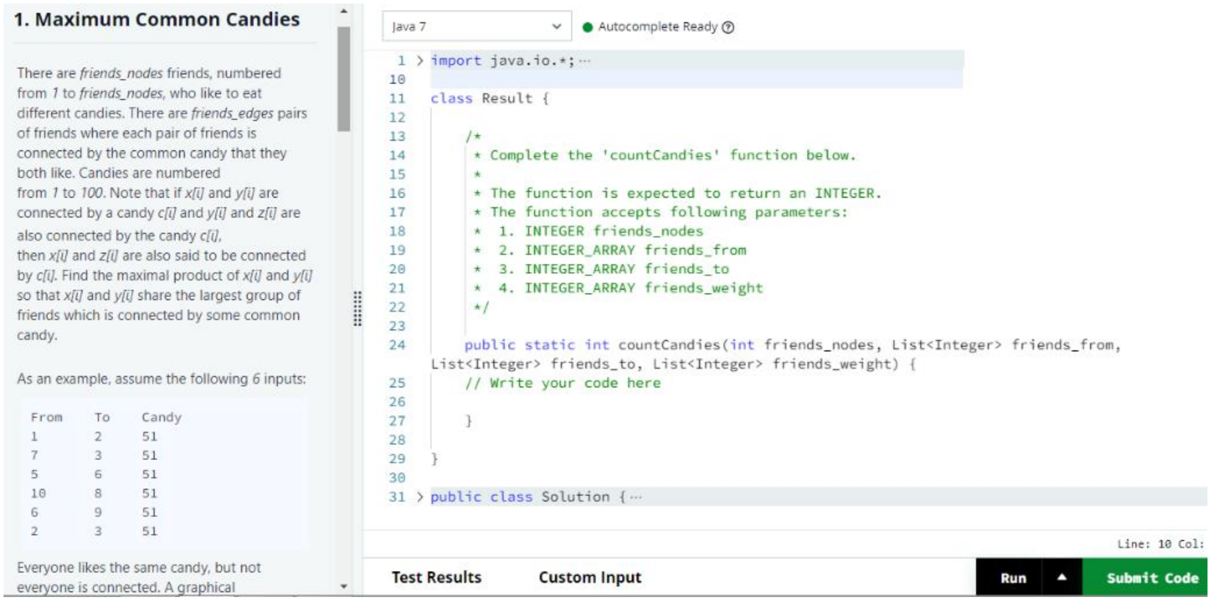
Virtusa



**1. Maximum Common Candies**

There are *friends_nodes* friends, numbered from *1* to *friends_nodes*, who like to eat different candies. There are *friends_edges* pairs of friends where each pair of friends is connected by the common candy that they both like. Candies are numbered from *1* to *100*. Note that if *x[i]* and *y[i]* are connected by a candy *c[i]* and *y[i]* and *z[i]* are also connected by the candy *c[i]*, then *x[i]* and *z[i]* are also said to be connected by *c[i]*. Find the maximal product of *x[i]* and *y[i]* so that *x[i]* and *y[i]* share the largest group of friends which is connected by some common candy.

As an example, assume the following *6* inputs:

```
From    To     Candy
1       2      51
7       3      51
5       6      51
10      8      51
6       9      51
2       3      51
```

Everyone likes the same candy, but not everyone is connected. A graphical

```
> import java.io.*;...

class Result {

    /*
     * Complete the 'countCandies' function below.
     *
     * The function is expected to return an INTEGER.
     * The function accepts following parameters:
     *  1. INTEGER friends_nodes
     *  2. INTEGER_ARRAY friends_from
     *  3. INTEGER_ARRAY friends_to
     *  4. INTEGER_ARRAY friends_weight
     */

    public static int countCandies(int friends_nodes, List<Integer> friends_from,
List<Integer> friends_to, List<Integer> friends_weight) {
    // Write your code here

    }

}

> public class Solution {...
```

```
package org.hackerrank;

import java.io.*;
import java.util.*;


class Result {

    /*
     * Complete the 'countCandies' function below.
     *
     * The function is expected to return an INTEGER.
     * The function accepts following parameters:
     *  1. INTEGER friends_nodes
     *  2. INTEGER_ARRAY friends_from
     *  3. INTEGER_ARRAY friends_to
     *  4. INTEGER_ARRAY friends_weight
     */

    public static int countCandies(int friends_nodes,
                                   List<Integer>
friends_from,
                                   List<Integer>
friends_to,
                                   List<Integer>
friends_weight) {

        // Write your code here
        int[] friends_from1=friends_from.stream()
```

```
                .mapToInt(Integer::intValue)
                .toArray();
        int[] friends_to1=friends_to.stream()
                .mapToInt(Integer::intValue)
                .toArray();
        int[] friends_weight1=friends_weight.stream()
                .mapToInt(Integer::intValue)
                .toArray();

        int res = 0, max_weight = 0;

        // loop to traverse all weight
        for (int i = 0; i < friends_nodes; i++) {
            // check if weight is more than max weight
            if (friends_weight1[i] > max_weight) {
                // calculate res as multiply of from
and to
                res = friends_from1[i] *
friends_to1[i];
                // set max weight
                max_weight = friends_weight1[i];
            }
        }
        // return result
        return res;


    }

}

public class Solution {
    public static void main(String[] args) throws
IOException {
        BufferedReader bufferedReader = new
BufferedReader(new InputStreamReader(System.in));
        BufferedWriter bufferedWriter = new
BufferedWriter(new
FileWriter(System.getenv("OUTPUT_PATH")));

        int friends_nodes =
Integer.parseInt(bufferedReader.readLine().trim());

        int friends_fromCount =
Integer.parseInt(bufferedReader.readLine().trim());
```

```java
        List<Integer> friends_from = new ArrayList<>();

        for (int i = 0; i < friends_fromCount; i++) {
            int friends_fromItem =
Integer.parseInt(bufferedReader.readLine().trim());
            friends_from.add(friends_fromItem);
        }

        int friends_toCount =
Integer.parseInt(bufferedReader.readLine().trim());

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

        for (int i = 0; i < friends_toCount; i++) {
            int friends_toItem =
Integer.parseInt(bufferedReader.readLine().trim());
            friends_to.add(friends_toItem);
        }

        int friends_weightCount =
Integer.parseInt(bufferedReader.readLine().trim());

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

        for (int i = 0; i < friends_weightCount; i++) {
            int friends_weightItem =
Integer.parseInt(bufferedReader.readLine().trim());
            friends_weight.add(friends_weightItem);
        }

        int result = Result.countCandies(friends_nodes,
friends_from, friends_to, friends_weight);

        bufferedWriter.write(String.valueOf(result));
        bufferedWriter.newLine();

        bufferedReader.close();
        bufferedWriter.close();
    }
}
```