Connected Sum

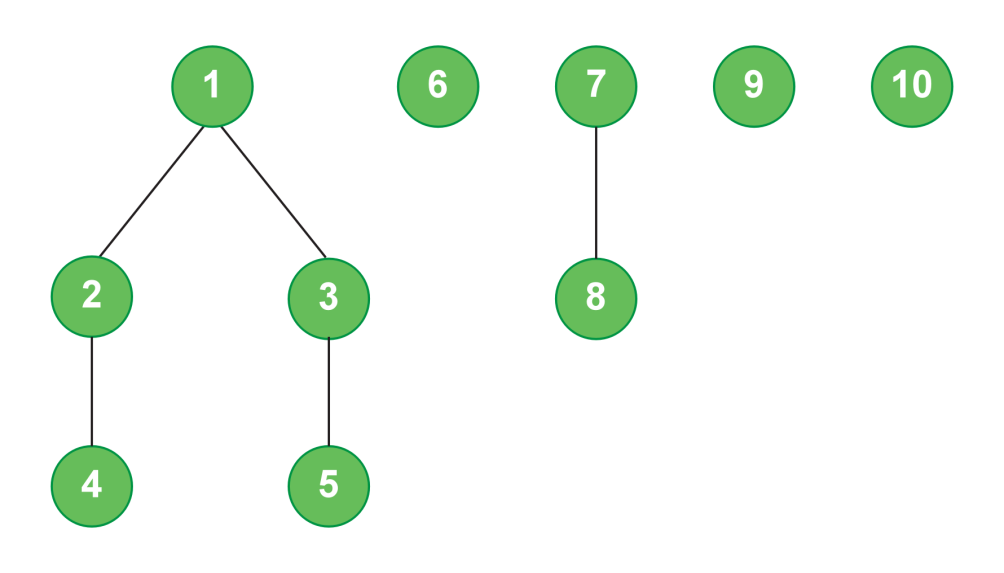
Given a number of nodes and a list of connected pairs, determine the weights of each isolated set of nodes assuming each node weighs *1* unit. Then for each weight calculated, sum the ceiling of its square root and return the final sum.

****Example****

*graph\_nodes = 10*

*graph\_from = [1, 1, 2, 3, 7]*

*graph\_to = [2, 3, 4, 5, 8]*



There are *graph\_edges = 5* edges to consider. There are *2* isolated sets with more than one node, *{1, 2, 3, 4, 5}* and *{7, 8}.*  The ceilings of their square roots are *51/2 ≅ 2.236* and *ceil(2.236) = 3, 21/2 ≅ 1.414* and *ceil(1.414) = 2.*  The other three isolated nodes are separate and the square root of their weights is *11/2 = 1* respectively.  The sum is *3 + 2 + (3 \* 1) = 8.*

****Function Description****

Complete the function *connectedSum* in the editor below.

connectedSum has the following parameter(s):

*int graph\_nodes:* the number of nodes

*int* *graph\_from[graph\_edges]:*  an array of integers that represent one end of an edge  
    *int* *graph\_to[graph\_edges]:*  an array of integers that represent the other end of an edge

****Returns****:

*int*: an integer that denotes the sum of the values calculated

****Constraints****

* *2 ≤ graph\_nodes ≤ 105*
* *1 ≤ graph\_edges ≤ 105*
* *1 ≤ graph\_from[i], graph\_to[i] ≤ n*
* *graph\_from[i] ≠ graph\_to[i]*

Input Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.

The first line contains two space-separated integers, *graph\_nodes*, the number of nodes, and *graph\_edges*, the number of edges.

Each of the next *m* lines contains two space-separated integers, *graph\_from[i]*and *graph\_to[i],* two nodes connected by an edge.

Sample Case 0

****Sample Input 0****

STDIN     Function

-----     ---------

4 2   →  graph\_nodes = 4 nodes, graph\_edges = 2 edges

1 2 → graph\_from[] = [1, 1], graph\_to[] = [2, 4]

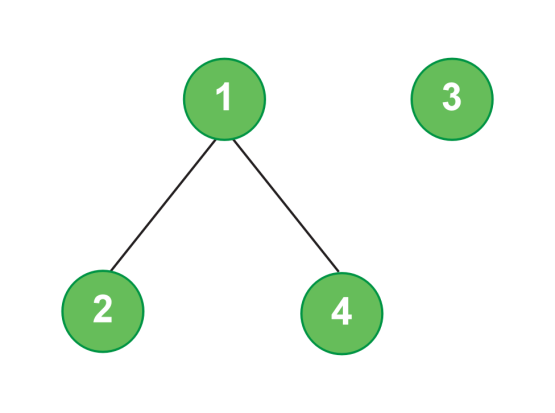
1 4

****Sample Output 0****

3

****Explanation 0****

The diagram below shows the groups of nodes:



The values to sum are:

1. Set *{1, 2, 4}: c = ceil(sqrt(3)) = 2*
2. Set *{3}: c = ceil(sqrt(1)) = 1*

*2 + 1 = 3*

Sample Case 1

****Sample Input 1****

STDIN     Function

-----    ---------

8 4  →  graph\_nodes = 8 nodes, graph\_edges = 4 edges

8 1 → graph\_from[] = [8, 5, 7, 8], graph\_to[] = [1, 8, 3, 6]

5 8

7 3

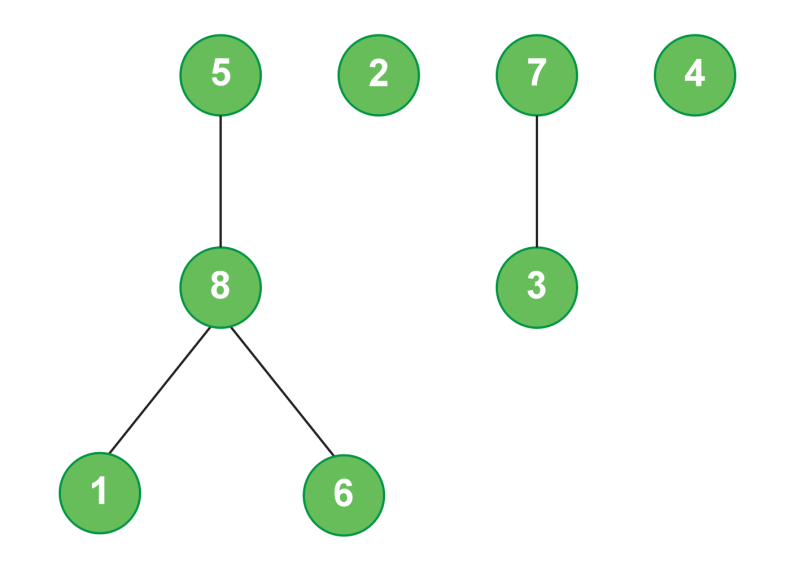
8 6

****Sample Output 1****

6

****Explanation 1****

The diagram below shows the groups of nodes:



The values to sum for each group are:

1. Set *{2}: c = ceil(sqrt(1)) = 1*
2. Set *{4}: c = ceil(sqrt(1)) = 1*
3. Set *{1, 5, 6, 8}: c = ceil(sqrt(4)) = 2*
4. Set *{3, 7}: c = ceil(sqrt(2)) = 2*

*1 + 1 + 2 + 2 = 6*

*import java.io.\*;*

*import java.math.\*;*

*import java.security.\*;*

*import java.text.\*;*

*import java.util.\*;*

*import java.util.concurrent.\*;*

*import java.util.function.\*;*

*import java.util.regex.\*;*

*import java.util.stream.\*;*

*import static java.util.stream.Collectors.joining;*

*import static java.util.stream.Collectors.toList;*

*class Result {*

*/\**

*\* Complete the 'connectedSum' function below.*

*\**

*\* The function is expected to return an INTEGER.*

*\* The function accepts UNWEIGHTED\_INTEGER\_GRAPH graph as parameter.*

*\*/*

*/\**

*\* For the unweighted graph, <name>:*

*\**

*\* 1. The number of nodes is <name>Nodes.*

*\* 2. The number of edges is <name>Edges.*

*\* 3. An edge exists between <name>From[i] and <name>To[i].*

*\**

*\*/*

*public static int connectedSum(int graphNodes, List<Integer> graphFrom, List<Integer> graphTo) {*

*}*

*}*

*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")));*

*String[] graphNodesEdges = bufferedReader.readLine().replaceAll("\\s+$", "").split(" ");*

*int graphNodes = Integer.parseInt(graphNodesEdges[0]);*

*int graphEdges = Integer.parseInt(graphNodesEdges[1]);*

*List<Integer> graphFrom = new ArrayList<>();*

*List<Integer> graphTo = new ArrayList<>();*

*IntStream.range(0, graphEdges).forEach(i -> {*

*try {*

*String[] graphFromTo = bufferedReader.readLine().replaceAll("\\s+$", "").split(" ");*

*graphFrom.add(Integer.parseInt(graphFromTo[0]));*

*graphTo.add(Integer.parseInt(graphFromTo[1]));*

*} catch (IOException ex) {*

*throw new RuntimeException(ex);*

*}*

*});*

*int result = Result.connectedSum(graphNodes, graphFrom, graphTo);*

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

*bufferedWriter.newLine();*

*bufferedReader.close();*

*bufferedWriter.close();*

*}*

*}*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**