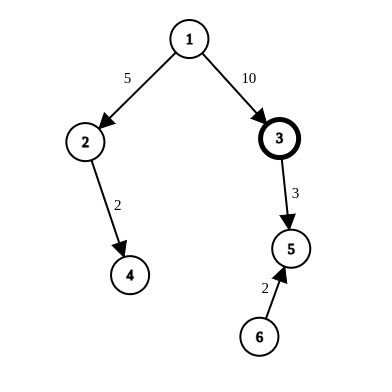
You are given a tree consisting of *n* nodes. Out of these *n* nodes, you may choose any node as root from where you will start the path. Now, you may go from your current node to any node to which it is directly connected via an edge. Each edge i of the tree is given in input as three integers u[i], v[i] and w[i] which denotes an edge connecting u[i] and v[i]. Via edge {u[i], v[i] , w[i]}, you can go both ways( either from u[i] to v[i] or v[i] to u[i]) but going from u[i] to v[i] will cost you w[i] units while going from v[i] to u[i] will cost you **0** units. You can not go back to the node you came from and will finally stop at some leaf node. Consider all possible paths from the root to each of the leaf nodes. Then the cost of each path is the sum of costs encountered at each edge as per the rules given above. The maximum among all of these path costs is the **leaf path cost** for that particular **root**.

Your task is to choose a root(starting point) such that your leaf path cost is as minimum as possible. The solution will output the minimum value of leaf path cost considering every node as root.

Example:

Consider input n: 6  
edges:  
[[1,2,5],  
[1,3,10],  
[2,4,2],  
[3,5,3],  
[6,5,2]]

Expected Output: 7



In the above figure, choosing 3 as root will give the minimum leaf path cost. Taking 3 as root, only 4 and 6 are leaf nodes. Hence, there are two paths in total, one from 3 to 4 and one from 3 to 6. The cost of path from 3 to 4 is 0 + 5 + 2 = 7 and for 3 to 6 is 3 + 0 = 3. Hence, the leaf path cost when 3 is root is max(7, 3) = 7.

Note that choosing 1 as root will give leaf path cost as 13.

* **[execution time limit] 1 seconds (cpp)**
* **[input] integer n**

Number of nodes in the given tree.  
2<=n<=10^5

* **[input] array.array.integer edges**

A matrix of dimensions n-1 x 3. Each row i contains u[i], v[i] and w[i] in order.  
1<=u,v,w<=10^4

* **[output] integer**

**[C++] Syntax Tips**

**// Prints help message to the console**

**// Returns a string**

**string helloWorld(string name) {**

**cout << "This prints to the console when you Run Tests" << endl;**

**return "Hello, " + name;**

**}**

**main.cpp**

C++

1

2

3

4

int solution(int n, vector<vector<int>> edges) {

}





**TESTS**

CUSTOM TESTS

**RUN TESTS**

**MORE**

Problem 2:

You are given a binary string of length n.

You have to find a minimum length substring of the above given string such that the count of subsequences of this substring which equal "01" is atleast k.

Your solution will return the length of the minimum length substring. If no such string exists then return -1.

Example:

Consider input n: 8  
k: 2  
inp: "11011001"

Expected Output: 3

Then out of all substrings of string "11011001", "011" is the substring of min. length which has number of "01" subsequences atleast 2(exactly 2 in this case).

* **[execution time limit] 4 seconds (py3)**
* **[input] integer n**

The length of the given string.  
2<=n<=2\*10^5

* **[input] integer k**

The value k.  
0<=k<=10^9

* **[input] string inp**

The binary string.

* **[output] integer**

An integer denoting length of minimum length substring that has atleast k subsequences equal to "01".

**[Python 3] Syntax Tips**

**# Prints help message to the console**

**# Returns a string**

**def helloWorld(name):**

**print("This prints to the console when you Run Tests")**

**return "Hello, " + name**

problem 1:

You are given n pairs of elements where each i'th pair has two values a[i] and b[i]. You have to select exactly k pairs out of the given n pairs. You can also reorder the selected k pairs in any possible way. Let's call the reordered pair indices as p1, p2, ... pk. Then the beauty **B** of an ordering is defined as:

https://codesignal.s3.amazonaws.com/uploads/1647063608962/unnamed.png

You have to find the maximum possible value of Beauty **B**

Example:

Consider input n: 5  
k: 3  
inp:  
[[4,2],  
[2,3],  
[3,4],  
[1,5],  
[2,7]]

For this case, the pairs [a[i], b[i]] in order are [4,2], [2,3], [3,4], [1,5], [2,7] and k=3. Let us pick 3 pairs such that indices p1 = 3, p2 = 4 and p3 = 5. Then the beauty for this ordering = (3 - 1 + 2)\*min(4,5,7) = 16.

* **[execution time limit] 4 seconds (py3)**
* **[input] integer n**

The total number of pairs.  
1<=n<=10^5

* **[input] integer k**

Number of pairs to be selected.  
1<=k<=1000

* **[input] array.array.integer inp**

An n X 2 matrix where inp[i][0] represents value of a in i'th pair and inp[i][1] represents value of b in i'th pair.

1<=a,b<=1000

* **[output] integer**

**[Python 3] Syntax Tips**

**# Prints help message to the console**

**# Returns a string**

**def helloWorld(name):**

**print("This prints to the console when you Run Tests")**

**return "Hello, " + name**