|  |  |
| --- | --- |
| **Problem Statement** | |
|  | You work for an electric company, and the power goes out in a rather large apartment complex with a lot of irate tenants. You isolate the problem to a network of sewers underneath the complex with a step-up transformer at every junction in the maze of ducts. Before the power can be restored, every transformer must be checked for proper operation and fixed if necessary. To make things worse, the sewer ducts are arranged as a tree with the root of the tree at the entrance to the network of sewers. This means that in order to get from one transformer to the next, there will be a lot of backtracking through the long and claustrophobic ducts because there are no shortcuts between junctions. Furthermore, it's a Sunday; you only have one available technician on duty to search the sewer network for the bad transformers. Your supervisor wants to know how quickly you can get the power back on; he's so impatient that he wants the power back on the moment the technician okays the last transformer, without even waiting for the technician to exit the sewers first.  You will be given three int[]'s: **fromJunction**, **toJunction**, and **ductLength** that represents each sewer duct. Duct i starts at junction (**fromJunction[i]**) and leads to junction (**toJunction[i]**). **ductlength[i]** represents the amount of minutes it takes for the technician to traverse the duct connecting **fromJunction[i]** and **toJunction[i]**. Consider the amount of time it takes for your technician to check/repair the transformer to be instantaneous. Your technician will start at junction 0 which is the root of the sewer system. Your goal is to calculate the minimum number of minutes it will take for your technician to check all of the transformers. You will return an int that represents this minimum number of minutes. |
| **Definition** | |
|  | |  |  | | --- | --- | | Class: | PowerOutage | | Method: | estimateTimeOut | | Parameters: | int[], int[], int[] | | Returns: | int | | Method signature: | int estimateTimeOut(int[] fromJunction, int[] toJunction, int[] ductLength) | | (be sure your method is public) | | |
| **Limits** | |
|  | |  |  | | --- | --- | | Time limit (s): | 2.000 | | Memory limit (MB): | 64 | |
| **Constraints** | |
| - | **fromJunction** will contain between 1 and 50 elements, inclusive. |
| - | **toJunction** will contain between 1 and 50 elements, inclusive. |
| - | **ductLength** will contain between 1 and 50 elements, inclusive. |
| - | **toJunction**, **fromJunction**, and **ductLength** must all contain the same number of elements. |
| - | Every element of **fromJunction** will be between 0 and 49 inclusive. |
| - | Every element of **toJunction** will be between 1 and 49 inclusive. |
| - | **fromJunction[i]** will be less than **toJunction[i]** for all valid values of i. |
| - | Every (**fromJunction[i]**,**toJunction[i]**) pair will be unique for all valid values of i. |
| - | Every element of **ductlength** will be between 1 and 2000000 inclusive. |
| - | The graph represented by the set of edges (**fromJunction[i]**,**toJunction[i]**) will never contain a loop, and all junctions can be reached from junction 0. |
| **Examples** | |
| 0) |  |
|  | |  |  |  |  | | --- | --- | --- | --- | | |  | | --- | | {0} | | {1} | | {10} | | | Returns: 10 | | |  | | --- | | The simplest sewer system possible. Your technician would first check transformer 0, travel to junction 1 and check transformer 1, completing his check. This will take 10 minutes. | | |
| 1) |  |
|  | |  |  |  |  | | --- | --- | --- | --- | | |  | | --- | | {0,1,0} | | {1,2,3} | | {10,10,10} | | | Returns: 40 | | |  | | --- | | Starting at junction 0, if the technician travels to junction 3 first, then backtracks to 0 and travels to junction 1 and then junction 2, all four transformers can be checked in 40 minutes, which is the minimum. | | |
| 2) |  |
|  | |  |  |  |  | | --- | --- | --- | --- | | |  | | --- | | {0,0,0,1,4} | | {1,3,4,2,5} | | {10,10,100,10,5} | | | Returns: 165 | | |  | | --- | | Traveling in the order 0-1-2-1-0-3-0-4-5 results in a time of 165 minutes which is the minimum. | | |
| 3) |  |
|  | |  |  |  |  | | --- | --- | --- | --- | | |  | | --- | | {0,0,0,1,4,4,6,7,7,7,20} | | {1,3,4,2,5,6,7,20,9,10,31} | | {10,10,100,10,5,1,1,100,1,1,5} | | | Returns: 281 | | |  | | --- | | Visiting junctions in the order 0-3-0-1-2-1-0-4-5-4-6-7-9-7-10-7-8-11 is optimal, which takes (10+10+10+10+10+10+100+5+5+1+1+1+1+1+1+100+5) or 281 minutes. | | |
| 4) |  |
|  | |  |  |  |  | | --- | --- | --- | --- | | |  | | --- | | {0,0,0,0,0} | | {1,2,3,4,5} | | {100,200,300,400,500} | | | Returns: 2500 | |  | |

This problem statement is the exclusive and proprietary property of TopCoder, Inc. Any unauthorized use or reproduction of this information without the prior written consent of TopCoder, Inc. is strictly prohibited. (c)2003, TopCoder, Inc. All rights reserved.