Problem Statement

You ran into some trouble when playing your favorite video game. There are **N** towns on the map, numbered from 0 to **N**-1. For each pair of distinct towns, there is exactly one way to get from one to the other by following one or more roads. Your enemy has just occupied some of the towns, and your first attempt at blocking him is to remove some of the roads in order to block off communication between every pair of distinct occupied towns. You would like to do this with as little effort as possible.

You will be given a String[] **roads** describing the roads between towns. Each element of **roads** is in the form "a b e" (quotes for clarity only), meaning that there is a bidirectional road between towns a and b, and the effort required to destroy the road is e. You will also be given a int[] **occupiedTowns** containing the towns occupied by your enemy. Return the minimum total effort required to destroy enough roads so that no two distinct occupied towns have a path (direct or indirect) between them.

Definition

Class:	BlockEnemy
Method:	minEffort
Parameters:	int, String[], int[]
Returns:	int
Method signature:	int minEffort(int N, String[] roads, int[] occupiedTowns)
(be sure your method is public)	

Constraints

- **N** will be between 2 and 50, inclusive.
- **roads** will contain between 1 and 50 elements, inclusive.
- Each element of **roads** will contain between 5 and 50 characters, inclusive.
- Each element of **roads** will be in the form "a b e" (quotes for clarity) where a and b are distinct integers between 0 and N-1, inclusive, and e is an integer between 1 and 1,000,000, inclusive.
- The integers in **roads** will not contain extra leading zeroes.
- There will be exactly one way to get from one town to any other, by following one or more roads.
- **occupiedTowns** will contain between 1 and 50 elements, inclusive.
- Each element of **occupiedTowns** will be between 0 and N-1, inclusive.
- The elements of **occupiedTowns** will be distinct.

Examples

0)

```
5
{"1 0 1", "1 2 2", "0 3 3", "4 0 4"}
{3, 2, 4}
Returns: 4
```

To make the communication between town 2 and the other occupied towns impossible, we must destroy one of the first two roads. We choose the first one as it requires less effort. Similarly, we choose between the last two roads in order to disconnect towns 3 and 4. The total cost is therefore 4.

1)

```
5
{"1 0 1", "1 2 2", "0 3 3", "4 0 4"}
{3}
Returns: 0
```

This is the same map as above. Since there is only one occupied town, we don't need to destroy any road here.

2)

```
12
{"0 1 3", "2 0 5", "1 3 1", "1 4 8", "1 5 4", "2 6 2",
 "4 7 11", "4 8 10", "6 9 7", "6 10 9", "6 11 6"}
{1, 2, 6, 8}
Returns: 13
```

Towns 1 and 2 are on the path from 6 to 8. We have to destroy the sixth road to disconnect town 6 from 2. We must destroy one of the first two roads to disconnect towns 1 and 2 and we pick the first one. Also, we should destroy one of the roads on the path from 1 to 8 and we choose the fourth road. The total cost is 2 + 3 + 8 = 13.

3)

```
12
{"0 1 3", "2 0 5", "1 3 1", "1 4 8", "1 5 4", "2 6 2",
 "4 7 11", "4 8 10", "6 9 7", "6 10 9", "6 11 6"}
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11}
Returns: 66
```

We have to destroy all roads this time.