6206 Reduce the Maintenance Cost

There are n towns (numbered from 1 to n) in a strange city and the maintenance costs of the towns are not necessarily same. There are bi-directional roads in the city and a road connects two towns. The merchants use the roads to trade amongst towns. If a road becomes unusable for some reason, it may affect the trading between some towns.

So, the mayor of the city has planned that every road will be maintained by one of its connecting towns. That means if a road connects town a and b, then either a or b (not both) will be given the responsibility to maintain the road.

The cost of maintaining a road is $= N * L$. Here:
N = The number of town pairs that cannot trade if this road is damaged,
L = the length of the road

If a town is given the responsibility to maintain some roads then the maintenance cost of the town will be increased by the summation of maintenance costs of the roads it'd be maintaining. Now, you are given the information of the towns, your task is to assign the roads to towns such that the maximum maintenance cost of a town is as small as possible.

Input

Input starts with an integer T ($T \leq 30$), denoting the number of test cases.

Each case starts with a blank line. Next line contains two integers n ($2 \leq n \leq 10000$) and m ($0 \leq m \leq 20000$), where n denotes the number of towns and m denotes the number of roads respectively. The next line contains n space separated integers between 1 and 10000 (inclusive), where the i-th integer denotes the maintenance cost of the i-th town. Each of the next m lines contains three integers u v w ($1 \leq u, v \leq n$, $1 \leq w \leq 10000$, $u \neq v$), denoting that there is a road between town u and v whose length is w. There is at most one road between two towns.

Output

For each case, print the case number and the result.

Note: Dataset is huge, use faster I/O methods.

For case 2, if the road between 1 and 4 goes down the town pairs that will be affected are: (1, 4), (1, 6), (2, 4), (2, 6), (3, 4), (3, 6), (4, 5), (5, 6).

Sample Input

```
3

2 1
5 10
1 2 10

6 6
10 20 30 40 50 60
1 2 1
2 3 1
1 3 1
```

```
1 4 6
1 5 6
4 6 2

3 1
10 20 30
2 3 10
```

Sample Output

```
Case 1: 15
Case 2: 80
Case 3: 30
```