Problem 1 - ADA Treasure (Programming) (15 points)

Problem Description

One day, YP and BB found an ancient map showing the information of an underground world in the ADA kingdom. The underground world contains N underground cities connected via M **directed** roads, where the i-th **directed** road connects the city u_i to the city v_i, indicating that people can travel from u_i to v_i via this road.

Moreover, there are treasures everywhere in the underground world: the i-th city has s_i units of treasures and the j-th road has w_j units of treasures.

As adventurers, YP and BB carry a bag with unlimited capacity to the underground world and want to pick up as many treasures as possible. They can start their journey at any city, visit an unlimited number of cities and roads, and go back to the ADA Kingdom via a spaceship at any city. Note that they always travel together. Once they visit a city or a road, they automatically pick up all treasures in the city or on the road; there will be no treasure if they visit the same city or the same road again.

Please write a program to help YP and BB compute the maximum units of treasures they can get.

Input

The first line of the input contains two integers N and M indicating the number of underground cities and roads in the underground world, respectively.

The next line contains N space-separated integers $s_1, s_2, \ldots, s_N$, where the i-th integer indicates the units of treasures in the city i.

In the next M lines, each line contains three integers u_i, v_i, w_i, indicating the road i connecting the city u_i to v_i with w_i treasures.

- $1 \leq N \leq 5 \times 10^5$
- $0 \leq M \leq 5 \times 10^5$
- $1 \leq u_i, v_i \leq N, u_i \neq v_i$
- $0 \leq s_i \leq 10^9$
- $0 \leq w_i \leq 10^9$

Test Group 0 (0 %)

- Sample Input

Test Group 1 (20 %)

- $M = N - 1$
- $u_i \leq i, v_i = i + 1$

Test Group 2 (20 %)

- You cannot find a sequence $c_0, c_1, \ldots, c_{Y-1}$ such that city c_i is connected to city $c_{(i+1) \mod Y}$ for all $i = 0 \cdots Y - 1$.

Test Group 3 (20 %)

- $1 \leq N, M \leq 5000$

Test Group 4 (40 %)

- No additional constraints.

Output

Output an integer indicating the maximum units of treasure that YP and BB can collect in one line.

Sample Input 1

```
7 6
3 6 2 6 7 0 1
1 2 1
2 3 4
2 4 2
1 5 7
3 6 0
6 7 1
```

Sample Output 1

```
18
```

Sample Input 2

```
7 8
2 7 5 6 4 3 3
1 4 9
5 4 6
4 6 5
4 6 1
3 7 5
2 6 2
5 2 7
1 6 3
```

Sample Output 2

```
25
```

Sample Input 3

```
8 10
4 8 0 5 9 2 4 1
1 3 5
2 1 0
4 1 0
4 2 6
3 4 1
1 4 7
8 2 3
7 1 5
3 5 2
4 6 8
```

Sample Output 3

```
56
```

Sample Input 4

```
10 16
3 1 4 1 5 9 2 6 5 3
1 2 5
2 3 8
3 1 9
4 5 7
5 4 9
6 7 3
7 8 2
8 6 3
9 10 8
10 9 4
3 6 6
2 7 2
3 10 6
4 7 4
5 7 3
4 9 3
```

Sample Output 4

```
61
```

Hint

1. Please refer to the hint note for hints.
2. The samples above might be helpful for debugging.
3. GL & HF (Good Luck and Have Fun).

Problem 2 - ADA Kingdom Revision (Programming) (15 points)

Problem Description

This is the ADA kingdom again!!! YP is the king of the ADA kingdom, and his responsibility (and his interest) is to protect the kingdom. The ADA kingdom has n cities connected by $n-1$ bi-directional roads. Each road has an importance value. It is guaranteed that for any two cities, there exist a sequence of roads such that one can move from one city to the other city, by walking along these roads.

There are k soldiers in the military. YP would like to allocate these soldiers to the cities to protect the roads. A road is defined as *protected* if there exist two soldiers such that one of the soldiers must pass through this road in order to arrive in the city guarded by the other soldier.

Please help YP allocate soldiers so that the sum of importance values over all protected roads is maximized.

Input

The first line contains an integer n ($2 \leq n \leq 5 \times 10^5$), indicating the number of cities in the ADA kingdom. In the next $n-1$ lines, each line contains three integers u, v, d ($1 \leq u, v \leq n, 1 \leq d \leq 10^9$), indicating that there exists a road connecting cities u and v with an importance value d.

Test Group 0 (0 %)

- Sample Input

Test Group 1 (20 %)

- $n \leq 100$

Test Group 2 (30 %)

- $n \leq 5000$

Test Group 3 (50 %)

- No additional constraints

Output

The output should contain n integers, the k-th of which represents the maximum sum of the importance values of all protected roads if there are k soldiers in the military. K starts from 1.

Sample Input 1

```
5
4 2 4
5 4 1
2 3 5
1 4 8
```

Sample Output 1

```
0 17 18 18 18
```

Sample Input 2

```
6
4 3 6
3 1 5
1 2 4
5 4 9
6 4 1
```

Sample Output 2

```
0 24 25 25 25 25
```

Sample Input 3

```
4
1 3 7
2 1 9
4 1 10
```

Sample Output 3

```
0 19 26 26
```

Problem 3 - ADA Roads (Programming) (15 points)

Problem Description

In another parallel universe, there are N cities and M bi-directional roads in the ADA kingdom, where the i-th road connects the city u_i and the city v_i with length w_i. Next month, people are coming to join a party held in the ADA city.

Due to some accidents, all roads were broken. So now, you, a person living in the ada city (different from the ADA city) is worried about how to attend the party next month.

In order to restore the transportation among cities as soon as possible, the government of the ADA kingdom dispatch many teams to find the solution. To save time, each team independently proposes a set of roads to be repaired such that after repairing every city can reach each other and the total length of the repaired roads is minimized.

You realize that there may exist many possible road sets that satisfy the requirement. You are now wondering: what is the shortest path between the ada city S and the ADA city T such that each edge on that path belongs to some of the potential road sets. Note that the edges can be in **different** road sets.

Input

The first line contains four space-separated integers N, M, S and T, which represent the number of cities, the number of roads, the ada city number, and the ADA city number respectively.

Each of the following M lines contains three space-separated integers u_i, v_i, and w_i, indicating there is a road connecting the city u_i and city v_i with length w_i.

It is guaranteed that if you repair all roads, every city can reach each other.

- $1 \leq N, M \leq 3 \times 10^5$
- $1 \leq u_i, v_i \leq N$
- $u_i \neq v_i$
- $0 \leq w_i \leq 10^9$

Test Group 0 (0 %)

- Sample Input

Test Group 1 (20 %)

- Every w_i is equal.

Test Group 2 (40 %)

- Every w_i is distinct.

Test Group 3 (20 %)

- $1 \leq N, M \leq 3000$

Test Group 4 (20 %)

- No other constraints.

Output

The output should contain one integer as the answer to this problem.

Sample Input 1

```
6 6 1 6
1 2 1
2 3 1
2 4 1
3 5 1
4 5 1
5 6 1
```

Sample Output 1

```
4
```

Sample Input 2

```
6 6 1 6
1 2 1
2 3 2
2 4 3
3 5 4
4 5 5
5 6 6
```

Sample Output 2

```
13
```

Sample Input 3

```
6 7 1 6
1 2 10
2 3 3
2 4 1
2 5 5
3 5 3
4 5 4
5 6 10
```

Sample Output 3

```
26
```

Sample Input 4

```
6 7 1 6
1 2 10
2 3 3
2 4 1
2 5 3
3 5 3
4 5 4
5 6 10
```

Sample Output 4

```
23
```

Problem 4 - Yet Another ADA Party (Programming) (10 points)

Problem Description

Tonight, people are coming to join a party held in the ADA city.

Since these people may have different backgrounds, such as coming from different countries, using different numbers of spaces to indent codes, placing the opening brackets of an if-statement at different locations, and so on, we can categorize them into N different types. During the party time, people chat with each other and persuade other people to become the same type as them.

To simplify the task, we assume that the party attendees form a line, and people join the party at either the front end or the back end of the line. Initially there is no attendee in the party, and then there are Q groups of people joining the party one by one. The i-th group contains c_i people with the type t_i (they all belong to the same type), and they join the party at the s_i end (where s_i is either "front" or "back"). When joining the party, they also persuade exactly k_i people beside them to become the same type as them. That is, now, $(c_i + k_i)$ people at the front (or the back) are all of the type t_i.

Let $C_{i,j}$ be the number of party attendees belonging the type j after the first i groups joined the party. Now, please find out the *popularity* for each of the N types. We define the popularity of type t as $C_{1,t} + C_{2,t} + \cdots + C_{Q,t}$.

Input

The first line of the input contains two integers N and Q, indicating the number of types and the number of groups joining the party, respectively. There are then Q more lines in the input, the i-th line of which contains one string and three integers s_i, c_i, t_i, k_i, indicating where the new group joined, the number of people in the new group, the type of the people in the group, and the number of attendees persuaded by the new group.

- $1 \le t_i \le N \le Q \le 2 \times 10^5$
- $s_i \in \{$"front", "back"$\}$
- $1 \le c_i \le 10^5$
- $0 \le k_i \le c_1 + c_2 + \cdots + c_{i-1} \le 2 \times 10^{10}$

Test Group 0 (0 %)

- Sample Input.

Test Group 1 (20 %)

- $N \le Q \le 8000$
- $c_i = 1$

Test Group 2 (40 %)

- $s_i =$ "back"

Test Group 3 (40 %)

- No other constraints

Output

Output one line containing N space-separated integers. The i-th integer is the popularity of the type i.

Sample Input 1

```
4 5
front 3 1 0
front 2 2 1
back 1 3 4
back 2 4 0
back 1 1 3
```

Sample Input 2

```
4 6
back 1 3 0
back 1 1 0
back 1 3 0
back 1 1 0
back 1 3 3
back 1 4 5
```

Sample Input 3

```
4 4
front 1 1 0
back 1 2 1
front 1 3 2
back 1 4 3
```

Sample Output 1

```
9 6 14 2
```

Sample Output 2

```
4 0 11 6
```

Sample Output 3

```
1 2 3 4
```

Hint

- Because the input files are large, please add the following lines to the beginning of the main function if you are using `std::cin`, and do not mix it with `cstdio` IO functions.
  - `std::ios_base::sync_with_stdio(false);`
  - `std::cin.tie(nullptr);`
- Below is what the party looks like after each group of people joined in **Sample Input 1**:
  - `1 1 1`
  - `2 2 2 1 1`
  - `2 3 3 3 3 3`
  - `2 3 3 3 3 3 4 4`
  - `2 3 3 3 3 1 1 1 1`
- Below is what the party looks like after each group of people joined in **Sample Input 2**:
  - `3`
  - `3 1`
  - `3 1 3`
  - `3 1 3 1`
  - `3 3 3 3 3`
  - `4 4 4 4 4 4`
- Below is what the party looks like after each group of people joined in **Sample Input 3**:
  - `1`
  - `2 2`
  - `3 3 3`
  - `4 4 4 4`

- You may need to use 64 bit integers to store the input and the output.

Problem 5 - Amortized Analysis (Hand-Written) (22 points)

In problem 5, you should prove the time complexity using the method specified in each problem statement.

(1) (5pts) Consider this problem: given an array $a[1..N]$ of length N, for each index i, please find $dp[i]$, the maximum index j such that $j < i$ and $a[j] > a[i]$. If no such j exists, define $dp[i] = 0$. The following algorithm provides a recursive solution to compute the answer of the above problem. Please derive the tightest amortized time complexity of the algorithm and prove it using the *potential method*.

Algorithm 1 booooooooook's nearest greater element

```
1: function H(i, x)
2:     if i = 0 ∨ a[i] > x then
3:         return i
4:     else
5:         return H(dp[i], x)
6:     end if
7: end function
8: dp[1] ← 0
9: for i ← 2  N do
10:    dp[i] ← H(i − 1, a[i])
```

(2) (5pts) Please briefly explain the algorithm you used in the Problem 4, and prove the amortized time complexity using the *accounting method*.

(3) (6pts) An α-weight-balanced binary search tree is a type of self-balancing binary search tree such that for every node, $\max(size(node.left), size(node.right)) \leq \alpha \cdot size(node)$, where $0.5 < \alpha < 1$ is a constant, $size(node)$ denotes the number of nodes in the subtree rooted at *node*, and $node.left$ and $node.right$ denote the left child and the right child of *node* respectively. For simplicity, we consider $size(node)$ is a $O(1)$ function.

The following provides one possible implementation of α-weight-balanced binary search tree. After an ordinary binary search tree insertion, if there is a node on the insertion path violating the α-weight-balance condition, say *node*, then we rebuild the whole subtree of *node* to make it perfectly balanced in $O(size(node))$ time. If there is more than one node violating the α-weight-balance condition, we choose the highest one to rebuild. Please prove that the amortized time complexity of the insertion operation is $O(\log n)$ using the *aggregate method*.

(4) (6pts) Consider this problem: given an array A of length N (indexed from 1) consisting of unsigned 64-bit integers, we maintain the **range bitwise AND operation** of every interval **online**. That is, we will scan the array from left to right. When encountering the index i, we will maintain for all $j \leq i$, the value of $A[j]$ & $A[j+1]$ & ... & $A[i]$, by some modification from the previous result. The following provides a simple algorithm in $O(N)$ total amortized time. Please briefly explain its correctness and then prove the time complexity of the algorithm using the *accounting method*.

Algorithm 2 range bitwise and

```
1: initialize B[1..N] with 0
2: for i ← 1 to N do
3:     B[i] ← A[i]
4:     for j ← i − 1 to 1 do
5:         if A[i] & B[j] = B[j] then
6:             break
7:         end if
8:         B[j] ← B[j] & A[i]
9:     end for
10: end for=0
```

Problem 6 - Robert the Archaeologist (Hand-Written) (28 points + 8 bonus points)

*In problem 6, please **briefly** explain your solution in text. **Do not** use pseudo code, or you will receive huge penalty.*

Robert is a legendary archaeologist! He is currently leading a large-scale systematic excavation project in Giverland. A basketful of papyrus rolls from 3 BC to 5 AD have been unearthed, restored, and deciphered by his great team. As the antiques have commanded extravagant prices in the market since the renaissance of Altertumswissenschaft brought about by the success of Robert's team, more and more robbers comes to plunder the unearthed papyri. Therefore, Robert, though being extraordinarily handsome, is quite vexed about the integrity of the excavated antiquities. To solve this problem, he decides to train some Crathvas, magical amorphous creatures found in the Convolutional Random Forest, to protect their finds. When a box of papyri is dug out, the Crathvas will guard and transport it to the closest town with vaults. (The closeness is defined by the transportation costs of the paths, which will be explained later.) If the town where the papyri are excavated already has vaults, then we have no need to transport the boxes.

Because Robert is also enthusiastic about optimization, he want to lower the cost spent training and feeding the Crathvas. However, there is no expert who can solve this problem in his team, so he ask you for help and offer you a bounty of $10,000 ADA dollars. You, talented at operations research and rapacious for the substantial amount of money, accept this challenge. Robert then happily emails you a map of the excavation sites in Giverland by RFC 1149.

This huge excavation project is being launched at all towns in Giverland. In Giverland, there are exactly R roads, and at the intersection of every pair of roads stands a town. The roads are therefore divided into totally E sections by the towns. There are $V \geq 127$ towns in total, and all of the towns are connected by the roads. The transportation cost $w_i \in \mathbb{Z}^+$ of each road section has been precisely foreseen by Arvin, the greatest prophet in the universe, and the transportation cost of a path is the sum of the transportation costs of all the road sections on the path. It is also known that there is no pair of road section connecting the same pair of towns, and there is no road section the two ends of which are the same town.

To put it simply, the map of Giverland can be represented by a graph satisfying the following conditions.

- The graph has $V \geq 127$ vertices and E edges.
- The graph is simple, undirected, connected, and weighted.
- The edges intersect only at their endpoints.
- All the edge costs are positive integers.
- The cost of a path π is $\sum_{e \in \pi} \text{cost}(e)$.

And Robert's problem can be paraphrased like this:

- **Robert's problem**
  Given a graph satisfying the aforementioned conditions. Some of the vertices are marked as secure. Find $\min_{\pi \in \text{paths starting from } V \text{ and ending at a secure vertex}} \text{cost}(\pi)$ for every vertex V. If there is no secure vertex, this value is defined as $+\infty$.

(1) (6pts) You, an international grandmaster in graph theory, are thinking about what special property this kind of graphs may have so that you can solve this problem more efficiently. It occurs to you that you can use a magic formula you have once dreamed of: $V - E + F = 2$, where V denotes the number of vertices, E denotes the number of edges, and F denotes the number of faces. (A face is either a region surrounding by some edges or the region outside

the connected component.) Although you deeply believe that this formula is undoubtedly true, being a computer scientist, you still have to give a proof before using it. Please write down a proof of the formula using mathematical induction. Besides, you think the graph is somewhat sparse. That is, the graph cannot have too many edges. Thus, please prove that $E = O(V)$ succinctly.

(2) (6pts) Now, you are confident enough to solve Robert's problem! You are given the map of Giverland, which is represented by a graph meeting the properties described by the problem statement. The towns with vaults are specified on the map as well. Please design an $O(V \log V)$-time algorithm to calculate the lowest transportation cost of each town. Briefly prove the correctness and time complexity of your algorithm.

Robert, the sagacious merchant, is earning more and more profits from his brilliant project. Hence, he can buy more vaults to secure the precious relics, and the number of towns with vaults are increasing. If we recalculate the minimum transportation costs of the towns when a new town is furnished with vaults, the time complexity of your algorithm will be too disappointing to Robert, who is very demanding about efficiency of algorithms. But you cannot come up with an improvement after racking your brains for months. You feel so distressed that you start to draw some ugly lines on the map of Giverland which don't intersect each other and anything on the original graph. By doing so, you accidentally making the graph **Kalos** ($\kappa\alpha\lambda\acute{o}\varsigma$), which means that every face on the graph is surrounded by **exactly 3 edges**.

One day, you encounter Arvin, the greatest prophet in the universe. Eyeing your nonplus, he pulls out some tarot cards, throws them into the heavenly fire, and stares at the luminous smoke dancing in the firmament. He then writes down several esoteric hieroglyphs on a crystalline diorite and dashes into the ocean unexpectedly. You are completely perplexed by this miracle. After decipherment, you find that the hieroglyphs on the rock represents "Saiko Psycho Cycle", which means a cycle **on a Kalos graph** satisfying the following three conditions.

1. The number of vertices strictly outside the cycle is at most $\frac{2}{3}V$.
2. The cycle comprises at most $4k$ vertices, where k is $\lceil \sqrt{V+1} \rceil$.
3. The number of vertices strictly inside the cycle minus the number of vertices strictly outside the cycle is minimized.

Besides, Arvin's script also prognosticates some mysterious claims about Saiko Psycho Cycle.

- **Claim 1**
  If the number of vertices strictly inside the cycle is greater than $\frac{2}{3}V$, then for every pair of vertices u, v in the cycle, the shortest distance between u and v on the subgraph comprised of the vertices on the cycle equals the shortest distance between u and v on the subgraph comprised of the vertices on the cycle and those inside the cycle.
- **Claim 2**
  If the number of vertices strictly inside the cycle is greater than $\frac{2}{3}V$, then the number of vertices on the cycle is exactly $4k$.

You are not sure if these claims are true, so you decide to prove them.

Because you are an international grandmaster in graph theory, when proving Arvin's claims, you can directly use the following theorems without proving them.

- **Menger's theorem**
  In a finite graph, the maximum number of internally vertex-disjoint paths between two nonadjacent vertices a and b equals the size of the minimum $a - b$ vertex cut.
- **Jordan curve theorem**
  Let C be a simple closed curve in the plane $\mathbb{R}^2$. $\mathbb{R}^2 \setminus C$ consists of exactly two connected components, the bounded interior and the unbounded exterior, and C is the boundary of each

component.
- You can make any graph satisfying the properties described in the problem statement to be Kalos in $O(V)$ time.

(3) (5pts) Please prove Claim 1.

(4) (5pts) Please prove Claim 2.

(5) (6pts) Based on Claim 1 and Claim 2, please show that a Saiko Psycho Cycle can never have $\frac{2}{3}V$ vertices or more inside it.

After completing these proofs, you realize that any Saiko Psycho Cycle on a Kalos graph is a vertex cut with no more than $4k = O(\sqrt{V})$ vertices that divides the graph into two or more disconnected parts, and each of them consists of no more than $\frac{2}{3}V$ vertices. Besides, you find it obvious that every Kalos graph has a Saiko Psycho Cycle.

Arvin emerges from the void again. He begins chanting an euphonious epic in dactylic hexameter, telling you that a Saiko Psycho Cycle can be found in $O(V)$ time. You are allowed to use this fact without proving it in the following problem.

(6) **(Bonus 8pts)** Now, full of eastern mystic power from Arvin the Unspeakable, your seven chakras are completely open. You reach an ultimate state called **Nirvana**, and you are capable of solving any problems in the blink of an eye. Please provide an algorithm to solve Robert's new problem. You are again given the map of Giverland, and there are no towns with vaults initially. There are two types of events: furnish a town with vaults, or a new box of papyri is excavated and we want to find out the minimum transportation cost at that time. Assume that there are Q events in total. Your algorithm should run in $O(V^{1.5} \log V + Q\sqrt{V})$ time. You should explain your algorithm in detail, and prove both the correctness and the time complexity.