Homework #3

Red Correction Date: 2021/11/26 16:40
Blue Correction Date: 2021/11/28 14:00
Purple Correction Date: 2021/12/6 00:00
Brown Correction Date: 2021/12/8 14:00
Due Time: 2021/12/16 14:20
Contact TAs: `ada-ta@csie.ntu.edu.tw`

Instructions and Announcements

- There are **four programming problems** and **two hand-written problems**.
- **Scoring policy.** Although the total score in this homework is 102, you'll get 100 if your original score exceeds 100.
- **Programming.** The judge system is located at `https://ada-judge.csie.ntu.edu.tw`. Please login and submit your code for the programming problems (i.e., those containing "Programming" in the problem title) by the deadline. NO LATE SUBMISSION IS ALLOWED.
- **Hand-written.** For other problems (also known as the "hand-written problems"), you should upload your answer to **Gradescope** as demonstrated in class. NO LATE SUBMISSION IS ALLOWED.
- **Collaboration policy.** Discussions with others are strongly encouraged. However, you should write down your solutions **in your own words**. In addition, for **each and every** problem you have to specify the references (e.g., the URL of the website you consulted or the people you discussed with) on the first page or comment in code of your solution to that problem. You may get zero points due to the lack of references.
- **Tips for programming problems.** Since the input files for some programming problems may be large, please add
  - `std::ios_base::sync_with_stdio(false);`
  - `std::cin.tie(nullptr);`

  to the beginning of the main function if you are using `std::cin`.

Problem 1 - ADA Adventurer (Programming) (11 points)

Problem Description

ADA Road, a prosperous country, was established by the ancestors' wisdom long time ago. This country consists of N cities indexed from 1 to N and $(N-1)$ roads. Each road connects two different cities, and all cities are connected. More formally, for any two different cities, a path exists from one to another along the given roads, and the distance between two cities is defined as the number of roads on this path.

To become a valiant ADA adventurer, you decided to visit this country. After arriving at the ADA Road, you discovered that the distance of some city pairs is too large, so villages need to pay lots of effort to reach one from another. To help the villages and get the experience points, you would like to help them resolve this problem. The only stuff allowed is to rebuild at most one road. That is, you can only remove at most one road from this country, and find a new pair of cities to build a road between them. The target is to minimize the farthest distance after rebuilding the road. Notice that you should avoid making some pairs of cities disconnected. Can you find the best way to complete it to win villages' reliance?

Input

The first line only contains one integer N, describing the number of cities in ADA Road.

In the next $N-1$ lines, each line contains two different integers a, b $(1 \le a, b \le N)$, describing a road connecting city a and b.

It is guaranteed that there exists a path from one city to another one for any pair of cities.

Test Group 0 (0 %)

- Sample Input.

Test Group 1 (20 %)

- $2 \le N \le 50$.

Test Group 2 (30 %)

- $2 \le N \le 1000$.

Test Group 3 (50 %)

- $2 \le N \le 2 \times 10^5$.

Output

Print one integer in a line, indicating the farthest distance after rebuilding the road.

Sample Input 1

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

Sample Output 1

```
3
```

Sample Input 2

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

Sample Output 2

```
3
```

Sample Input 3

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

Sample Output 3

```
4
```

Problem 2 - P kingdom (13 points + Bonus 2 points)

Problem Description

P kingdom, a prosperous country, was established by the wisdom of the ancestors a long time ago.

This country consists of N cities indexed from 1 to N and M directed roads. Each road e_i connects two different cities u_i and v_i, indicating that you can visit city v_i from city u_i.

As a resident in the P kingdom, you find that there are specific cities where residents may not be able to visit all other cities in the P kingdom.

In order to meet everyone's daily needs, you have to solve this problem by building new directed roads. Now you can propose a list of roads that need to be built and minimize the number of roads on the list.

Input

The first line contains two integers T, representing the number of test cases, and $flag$ ($flag \in \{0, 1\}$), which will be described in the output section.

Each test case consists of two parts: the first line contains two space-separated integers N, M, representing the number of cities and the number of roads, respectively.

Each line of the following M contains two space-separated integers u_i and v_i, indicating a road connection from u_i to v_i.

- $1 \leq N \leq 1000$
- $0 \leq M \leq \min(1000, N \times (N-1))$
- $1 \leq u_i \leq N$
- $1 \leq v_i \leq N$
- $u_i \neq v_i$

It is guaranteed that the sum of N does not exceed 3000 and the sum of M does not exceed 3000.

Test Group 0 (0 %)

- Sample Input.

Test Group 1 (30 %)

- $flag = 0$
- $u_i < v_i$

Test Group 2 (30 %)

- $flag = 0$

Test Group 3 (20 %)

- $u_i < v_i$

Test Group 4 (20 %)

- No additional constraints.

Bonus Group (2 points)

- $N, M \leq 100000$
- The sum of N does not exceed 100000 and the sum of M does not exceed 100000.

Output

For each testcase, print an integer on the first line representing the number of roads K that need to be built.

If the $flag$ mentioned in the input section is equal to 1, please furthermore print the following K lines: each line contains s_i and t_i, representing building a road from s_i to t_i in the list. If there are multiple ways to match this condition, you can print any of them.

Sample Input 1

```
1 0
4 2
1 2
3 4
```

Sample Output 1

```
2
```

Sample Input 2

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

Sample Output 2

```
3
2 3
4 5
5 1
0
```

Problem 3 - I Want It All!!! (Programming) (13 points)

Problem Description

Xiao Feng and Baluteshih are learning graph theory together from **Waynetu**, a legendary grandmaster in graph theory. To make sure that they have good learning efficiency, **Waynetu** decided to assign homework to them. The homework is described as follow:

Given a weighted connected undirected graph, please find a spanning tree of it.

Xiao Feng and Baluteshih are working on this homework together. Xiao Feng loves shortest-path trees, so he would like to find a shortest path tree starting from vertex 1. However, Baluteshih loves minimum spanning trees, so he would like to find a minimum spanning tree of the given graph. Please help Xiao Feng and Baluteshih to find a spanning tree that satisfies both of their preferences simultaneously or determine that it is impossible.

Input

The first line contains two integers N, M, describing the number of the vertices and the edges of the graph. The vertices are indexed from 1 to N.

In the next M lines, the i^{th} line contains three integers a_i, b_i, c_i, describing an undirected edge with index i which connects the vertices a_i and b_i with weight c_i.

It is guaranteed that the input forms a connected graph.

Test Group 0 (0 %)

- Sample Input.

Test Group 1 (40 %)

- $1 \leq N \leq 10^5$.
- $0 \leq M \leq 3 \times 10^5$.
- $1 \leq a_i, b_i \leq N, a_i \neq b_i$.
- $1 \leq c_i \leq 10^9$.
- All c_i are distinct.

Test Group 2 (60 %)

- $1 \leq N \leq 10^5$.
- $0 \leq M \leq 3 \times 10^5$.
- $1 \leq a_i, b_i \leq N, a_i \neq b_i$.
- $1 \leq c_i \leq 10^9$.

Output

If Xiao Feng and Baluteshih cannot find the same spanning tree of the input graph, please print "`No`" (without quotes) in the first line.

Otherwise, please print "`Yes`" (without quotes) in the first line. In the second line, please print $n-1$ integers, describing indices of the chosen edges of the spanning tree. If there are multiple choices of the spanning trees, you may print any of them.

Sample Input 1

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

Sample Output 1

```
Yes
1 2 3 4
```

Sample Input 2

```
7 9
5 7 3
3 2 4
1 4 18
3 4 17
2 5 10
1 5 49
1 3 35
3 5 14
6 5 19
```

Sample Output 2

```
Yes
1 2 3 4 5 9
```

Sample Input 3

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

Sample Output 3

```
No
```

Sample Input 4

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

Sample Output 4

```
Yes
1 2 4
```

Hint

1. You may want to find “a shortest-path tree that forms a minimum spanning tree” instead of finding “a minimum spanning tree that forms a shortest-path tree”.

2. Test Group 1 can only help you verify the correctness of your shortest-path tree and minimum spanning tree. Try not to regard it as a hint for full credit.

Problem 4 - Cats! (Programming) (13 points)

Problem Description

BB loves cats! There are N cats in his garden, and the cats are labeled from 1 to N. BB's garden can be seen as a straight line, and the i^{th} cat is initially located at x-coordinate x_i.

Every once in a while, BB will place some food somewhere in the garden, which attracts some of the cats nearby. More precisely, there are Q sequential events, the j^{th} of which is of one of the two types:

- BB placed some food at x-coordinate p_j, and attracted all cats within radius r_j. That is, every cat within $[p_j - r_j, p_j + r_j]$ moved to p_j.
- The ${t_j}^{\text{th}}$ cat moved to x-coordinate x'_j

Unfortunately, BB feels like his garden might be a bit too crowded after some of the events. The cats might feel uncomfortable due to this. To try resolving this problem, BB would like to first know the *crowdedness* of his garden. **He define *crowdedness* be the number of pairs (i, j), $i < j$, such that the i^{th} cat and the j^{th} cat are at the same location.**

Please help BB find out the *crowdedness* after each event.

Input

The first line of the input contains two positive integers N and Q, denoting the number of cats on BB's garden and the number of events, respectively.

The second line contains N space-separated integers $x_1, x_2, \ldots, x_N$, denoting the initial x-coordinate of the cats.

Then Q lines follows. The j-th line of which is of one of the following two forms:

- `1` p_j r_j: BB placed some food at p_j with radius r_j.
- `2` t_j x'_j: The ${t_j}^{\text{th}}$ cat moved to x'_j.

Note that the events happen one by one (i.e., the j^{th} event happens under the effect of all previous $j - 1$ events).

- $1 \leq N, Q \leq 10^5$
- $0 \leq x_i, p_j, r_j, x'_j \leq 10^9$
- $1 \leq t_j \leq N$

Output

Please output Q lines, where the j^{th} line denotes the *crowdedness* of BB's garden after the j^{th} event.

Test Group 0 (0 %)

- Sample Input

Test Group 1 (10 %)

- $N, Q \leq 100$

Test Group 2 (20 %)

- $N, Q \leq 5000$

Test Group 3 (30 %)

- There are only events of type 1.

Test Group 4 (40 %)

- No Additional Constraint

Sample Input 1

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

Sample Output 1

```
1
3
1
3
2
10
```

Sample Input 2

```
8 5
1 1 2 3 5 8 13 21
1 10 4
1 3 1
1 3 2
1 8 4
1 12 9
```

Sample Output 2

```
2
3
11
11
28
```

Hint

1. STL is your good friend:
   - Use `std::map` or `std::set` to maintain the cats in ascending order of their x-coordinate.
   - Use `std::vector` or `std::list` when you need arbitrary length array.
   - Go to Cpp Reference if you don't know how to use an STL container or function.
2. Be careful **NOT** to use `std::endl` since it might be very slow. Use `'\n'` instead.
3. Let `c` be an `std::map` or `std::set`. Be careful **NOT** to use `std::lower_bound(c.begin(), c.end(), x)`. Use `c.lower_bound(x)` instead.
4. You should try to prove the time complexity of your code.
5. The test data in Test Group 1 might be a lot weaker than other test groups. Passing this test group does not guaranteed your code is bug-free.

Problem 5 - Shi-Hei Robots (Hand-Written) (20 points)

Note: In this problem, you ***don't*** *have to write pseudocode as long as you clearly explain your method. If there are mistakes in your pseudocode, you will receive an additional penalty. You don't have to prove the time complexity of your method unless the problem description asks you to do so.*

Stanley, Wang Jung Tsan (王榮燦), a famous video game streamer and a previous world champion of an online game, has always been accused of purchasing robots to boost his number of stream viewers. In fact, there are some "robots" watching him streaming — they are so-called "Shi-Hei robots." "Shi-Hei robots (史黑機器人)" is a group of evil villains whose ultimate goal is to defame Stanley by convincing people that Stanley is buying robots. To be convincing, they always comment robot emojis in the chatroom so that they look like real robots. One day, you accidentally discover a secret of "Shi-Hei robots": if a group member sees another group member that he/she knows to spam in the chatroom, he/she will continue commenting robot emojis, and eventually, they will be unstoppable. However, simply banning all of them is not feasible since Stanley is merciful and does not want to ban many people. As a MOD (moderator) of Stanley's streaming channel, you must decide which member of "Shi-Hei robots" should be banned so that the villain group can be split into as many sub-groups as you can. If you can find out which member should be banned, you will receive a precious reward from Stanley — "Stanley Smile (王榮燦笑)".

More precisely, we use $G(V, E)$, an undirected graph, to denote the relationship among members of "Shi-Hei robots". The goal is to find out the single vertex $v \in V$ whose removal from the graph split the graph into as many different connected components as possible.

In the following part of this problem, we will use $s(v, G)$ to denote the number of connected components in the graph G after removing vertex v from G. You may assume that the initial graph G (before removing vertex) is connected, has at least two vertices, and is represented using the adjacency lists. Recall that you can output the neighbors of v in $O(deg(v))$ time and $deg(v)$ in $O(1)$ time for any vertex $v \in V$ when using the adjacency-list representation.

Here are the definitions of some terminologies used in this problem: For a tree $T(V, E)$ with root vertex r, "the *ancestors* of vertex v" denotes all vertices on the path from r to v, and "the *descendants* of vertex v" denotes any vertex x such that v is an *ancestor* of x. Every vertex is both *ancestor* and *descendant* of itself. "*Child* of vertex v" denotes any *descendant* x of v such that edge $(x, v) \in E$.

(1) **(3 points)** Let T be a DFS tree of G with root $r \in V$. That is, $T = (V, E_T)$, where E_T is a subset of E, connecting all vertices in G. Given T, please briefly describe how to find $s(r, T)$ in $O(1)$ time complexity and **prove the correctness**.

(2) **(4 points)** Let v be a vertex other than the root r (i.e. $v \neq r$). Suppose that there is no edge $\{u, w\} \in E$ where $u, w \neq v$, and u is an *ancestor* of v in T, and w is a *descendant* of v in T. Please briefly describe how to find $s(v, G)$ in $O(1)$ time complexity and **prove the correctness**.

(3) **(5 points)** Let $D_T(w)$ denotes in tree T, the set of *descendants* of vertex $w \in V$ including w itself. Also, for a set $S \subseteq V$, let $N_G(S)$ denotes in graph $G(V, E)$, the set of "neighbors of vertices in set S", i.e. $N_G(S) = \{y \in V : \exists x \in S \text{ s.t. } \{x, y\} \in E\}$.

Define $\mathsf{up}_T(w) := \min_{y \in N_G(D_T(w))} \mathsf{depth}_T(y)$. That is, $\mathsf{up}_T(w)$ denotes in tree T, the minimum depth of "any neighbor in G of any *descendant* of w in T".

Suppose v is an arbitrary vertex that is not the root in T, with *children* $w_1, w_2, \cdots, w_k$. Please show how to compute $s(v, G)$ as a function of $k, \mathsf{up}_T(w_1), \mathsf{up}_T(w_2), \cdots, \mathsf{up}_T(w_k)$, and $\mathsf{depth}(v)$. Please **prove the correctness** of your method as well.

(4) **(8 points)** Given G, please design an algorithm that computes $s(v, G)$ for **all** vertices $v \in V$ with time complexity $O(|V| + |E|)$. Analyze its time complexity and **prove the correctness**.

Problem 6 - Lost in Pekoland (Hand-Written) (30 points)

Note: In this problem, if you're asked to provide an algorithm, you should prove both its correctness and its time complexity, and you ***don't*** *have to write pseudocode as long as you can clearly explain your method. You can directly use any algorithm taught in class*

Pekoland, a place full of hopes and dreams, is an amusement park constructed by "Usada Kensetsu," a construction corporation founded by its CEO, Usada Pekora. The park comprises numerous celebrated landmarks, including a magnificent Castle, the grand Pekora Gundam, some formidable roller coasters, and a creepy monument of a mysterious figure, Yagoo. With these features, Pekoland has long been an appealing tourist attraction around the continent.

One day, Pekora wants to build some railways so that it can be more convenient to commute between the sites in her park. Suppose that the Pekoland's layout G looks like a set of sites denoted as V connected by a set of roads denoted as E, each road connects exactly two different sites with both of its two endpoints and can be traveled from both directions, every two sites are connected by at most one road, and for any two different sites a and b, one is always able to go from a to b by walking through a series of roads in E.

Part I

Pekora decides to transform some of these roads into railways, but she notices that each road e has its own width $w(e)$, suppose a *path* is a series of roads $(v_0, v_1)(v_1, v_2)\dots(v_{n-1}, v_n)$ where all $(v_i, v_{i+1}) \in E$, we define the width of a path as the *minimum* width of the roads it consists of. Pekora hopes that after the railway construction, for any pair of sites s and t, one can travel from s to t entirely through railways by some of the widest path between them, that is, if $E^{'}$ is the subset of roads being transformed, then at least one of the widest path between s and t on $G(V, E)$ is still included in $G^{'}(V, E^{'})$, but since they only have limited funds, she hopes $|E^{'}|$ can be as small as possible. Please help her find the set $E^{'}$ that meets her needs.

(1) **(3 points)** Give a $O(E \log V)$ algorithm that finds the *maximum* spanning tree of G.

(2) **(5 points)** Prove that for any pair of sites s and t, the maximum spanning tree of G contains some of their widest path. Conclude that Pekora can find the required set $E^{'}$ in $O(E \log V)$ time complexity.

Part II

Months later, Usada Kensetsu became so much richer and transformed all roads into railways. Now every road e has its length $d(e)$, which is a positive real number. To get to any site from her castle as quickly as possible, Pekora wants to find a shortest path from her castle s (which also belongs to V) to every other site. Luckily, some of her employees had learned some algorithms and know how to find the shortest path from s to all the other sites for her. However, she finds out that sometimes when a road gets reconstructed, its length can be lengthened or shortened, and it may influence the shortest distance of a site from s. She becomes curious that at a moment, which roads have the property that the change of its length would alter the shortest path distance from s to some sites $v \in V$?

We define that a road e is *upwards critical* if increasing $d(e)$ by any positive real number ε would increase the shortest path distance from s to at least one site $v \in V$, and is *downwards critical* if decreasing $d(e)$ by any positive real number ε would decrease the shortest path distance from s to at least one site $v \in V$.

(3) **(5 points)** Prove that a road $(u, v) \in E$ is downwards critical if and only if there is a shortest path from s to u or v that ends at (u, v).

(4) **(5 points)** Referring to the claim above, make a claim of the if and only if condition of being upwards critical and prove your claim.

(5) **(6 points)** Using the above claims, give an $O(E \log V)$ time complexity algorithm that determines all upwards critical roads and all downwards critical roads respectively.

Part III

Since that Pekora is very naughty and likes to prank others, she has set up several traps in her amusement park. As a result, Pekoland is also considered to be a dangerous place to go by many. One day Pekora notices that every site $v \in V$ is graded by a *dangerous value* $k(v) \geq 0$ by travelers. It comes to her that it may be fun to build a new roller coaster that sets off from some $v \in V$, traveling through some roads of E then finally ends at the starting point v, and she would like the sum of all the dangerous value of the sites on the cycle to be high.

Thus she asks Moona Hoshinova, the chief technical officer of her company, to conduct the plan, but since building a roller coaster is tiring and costly, she replies that if the total length of the cycle is too large, she would not be willing to take over the task. After some negotiation, they define a cycle $v_1, v_2, \ldots, v_n, v_{n+1} = v_1$ on G, where all $v_i \in V$ and all $(v_i v_{i+1}) \in E$, as an *exciting cycle* if it satisfies that

$$\frac{\sum_{i=1}^{n} k(v_i)}{\sum_{i=1}^{n} d(v_i, v_{i+1})} > K$$

where K is a constant given by Moona, and they meet the consensus that if there is an exciting cycle on G, then Moona would agree to build the roller coaster on this cycle, but if there is not such a cycle, then she would object to the plan. In this problem(and only this problem), we suppose the graph is directed and strongly connected, also, for any $u, v \in V$ if $uv \in E$, then $vu \in E$, but it's possible that $d(u, v) \neq d(v, u)$ since that to travel from two direction of a road may not be as difficult as each other.

(6) **(6 points)** Please provide an algorithm running in $O(VE)$ time that finds if there exists an *exciting cycle.*

Hint: you can try re-weighting the graph.