Road Reorganization (roads)

The Roman Army has recently gotten new carriages, which can carry significantly more people. Unfortunately, this also means that the carriages are too big and often get stuck when there's traffic from the opposite direction. In order to prevent this, the Emperor decides to make every road in the Empire one-way.

However, the Emperor likes to go to his hometown quite often, and wants to make sure that when he is there, he can quickly return to Rome if something important comes up. To ensure this, he wants to make sure the shortest path from his hometown to Rome does not become longer than it currently is. Additionaly, he also wants to make sure he is able to return to his hometown, but this does not have to be as fast as possible.

The Roman Empire contains N cities and there are M bi-directional roads. The Emperor's hometown is city 0 and Rome is city $N-1$. Formally, he wants to choose a direction for each road. He wants to make sure the shortest path from city 0 to city $N-1$ is not longer than it was before the roads became one-way roads. Additionally, he wants to make sure there is still a path from city $N-1$ back to city 0. Your task is to orient the roads in such a way, or let the Emperor know that this is impossible.

> ⇨ It is guaranteed that any city in the Roman Empire is reachable from any other city. Note that this does not have to be the case after you orient the roads.

Note that orienting the roads such that you can go from 0 to $N-1$ and back, without necessarily minimizing the length of one of the trips, will still give you points.

Implementation

You will have to submit a single `.cpp` source file.

> ⚠ Among this task's attachments you will find a template `roads.cpp` with a sample implementation.

You will have to implement the following function:

C++	`vector<pair<int, int>> orient_roads(int N, int M, vector<int> A, vector<int> B, vector<int> W)`

- Integer N represents the number of cities.
- Integer M represents the number of roads.
- The arrays A, B and W describe the roads of the Roman Empire: There is a road between city $A[i]$ and city $B[i]$ with length $W[i]$.
- If there is a way to orient the roads in such a way, the function should return a vector of pairs of length M. Here, the i-th pair $\{F, T\}$ signifies that there is a road oriented from F to T.
- If there is no way to orient the roads, the function should return an empty vector.

It is guaranteed there is at most one road between two cities and that for all i there is $A[i] \neq B[i]$. If the function does not return a vector of length 0 or M, this will count as a wrong answer. All roads

mentioned in the input should be present in the output vector exactly once and no other roads should be present. The order of the roads in the output vector does not matter.

Sample Grader

Among this task's attachments you will find a simplified version of the grader used during evaluation, which you can use to test your solutions locally. The sample grader reads data from `stdin`, calls the function `orient_roads` and writes back on `stdout` using the following format.

The input is made up of $M + 1$ lines, containing:

- Line 1: the integers N and M.
- Line $2 + i$ $(0 \leq i < M)$: the integers A_i, B_i and W_i.

The output is made up of either 1 or $M + 1$ lines, containing:

- Line 1: `Yes` or `No`, indicating if it is possible to orient the roads in such a way.
- If the answer is `Yes`: Line $2 + i$ $(0 \leq i < M)$: F_i and T_i, indicating that the road between these two cities should be oriented from F_i to T_i.

In the output, the road orientations are printed in the order they are returned from the function.

Constraints

- $3 \leq N \leq 100\,000$.
- $N - 1 \leq M \leq 100\,000$.
- $0 \leq A_i, B_i \leq N - 1$ and $A_i \neq B_i$.
- $1 \leq W_i \leq 1\,000\,000\,000$.
- It is guaranteed that any city in the Roman Empire is reachable from any other city.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the full score of a subtask, your program needs to correctly solve all of its test cases. If the solution is valid but not optimal, you will obtain half of the points for this subtask.

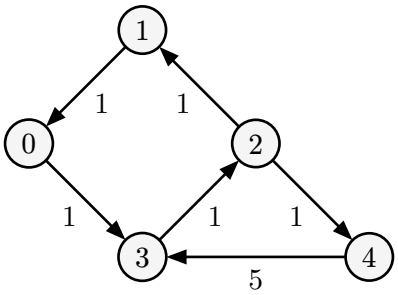
- **Subtask 0** [**0 points**]: Sample test cases.
- **Subtask 1** [**6 points**]: $\max(N, M) \leq 20$.
- **Subtask 2** [**11 points**]: $M \leq N$.
- **Subtask 3** [**17 points**]: It is guaranteed there is a direct road between city 0 and city $N - 1$.
- **Subtask 4** [**18 points**]: $\max(N, M) \leq 2000$.
- **Subtask 5** [**17 points**]: $W_i = 1$.
- **Subtask 6** [**31 points**]: No additional constraints.

⇨ You can get half of the points for a testcase if the shortest path from city 0 to $N - 1$ after the roads were directed is not the same length as the shortest path from city 0 to $N - 1$ before the roads were directed, even if there was no way to orient the roads that fully satisfied the Emperor.

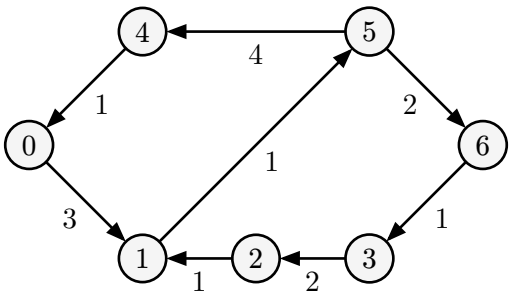
Examples

stdin	stdout
5 6 0 1 1 1 2 1 0 3 1 3 2 1 2 4 1 3 4 5	Yes 1 0 2 1 0 3 3 2 2 4 4 3
7 8 0 1 3 1 2 1 2 3 2 3 6 1 0 4 1 4 5 4 5 6 2 1 5 1	Yes 0 1 2 1 3 2 6 3 4 0 5 4 5 6 1 5
8 9 0 1 3 1 2 4 2 4 2 0 3 6 3 4 1 4 5 3 5 6 1 6 7 2 5 7 3	No

Explanation



In the first example, we can get from city 0 to 4 via $0 \to 3 \to 2 \to 4$. This is a path with length 3, which is also the minimum length before the roads were directed. We can also still get back via $4 \to 3 \to 2 \to 1 \to 0$. It does not matter how long this path is.



In the second example, we can get from 0 to 6 with route $0 \to 1 \to 5 \to 6$ and we can get back using $6 \to 3 \to 2 \to 1 \to 5 \to 4 \to 0$.

In the third example, it can be shown there is no orientation of roads that would make the Emperor happy.