We will check if a given graph is bipartite and output the minimum spanning tree of the graph. For the extra credit part, you will write a simple algorithm to perform the topological sorting of a given graph. We encourage you to use as much code (data structures and algorithms) as possible from the C++ standard library for the C++17 standard.

**Problem Description**

Create a program that checks if a given **undirected graph** is bipartite and also outputs the minimum spanning tree of the graph. The graph configuration will be given in a file using an edge list format and the file name will be passed as an argument to the program.

The input file will have the following format:

* The first line will contain 2 numbers separated by space, N M, where N is the number of nodes in the undirected graph, and M is the number of edges.
* The next M lines will contain 3 numbers, the two nodes between which a connection exists, and the weight of that connections.

The program will write the result into the result.txt file:

* On the first line, the program will write Yes if the given graph is bipartite or No otherwise. (50 points)
* On the subsequent lines, the program will write the graph edges, one per line, that are part of the minimum spanning tree in the ascending order by their weight. The edges are written as a pair of two nodes which they connect. (50 points)

Note, all numbers are positive integers and the two parts of the program are checked independently.

**Example**

Executing the following command ./lab4b.out network.txt where the content of network.txt is:

6 7

3 1 0

3 5 3

1 0 1

0 5 2

0 2 5

5 4 3

4 2 4

This is a bipartite graph with 6 nodes and 7 edges. The two partitions contain the nodes [3, 0, 4] and [1, 5, 2]. The result.txt file should have the following content:

Yes

3 1

1 0

0 5

5 4

4 2

Note, if the graph had an additional edge between node 4 and 0 of weight 6 then the graph would no longer be bipartite, however, the minimum spanning tree would have remained the same.

**Problem Description**

Create a program that outputs the nodes of a given **directed graph** in topological order. The graph configuration will be given in a file using an edge list format and the file name will be passed as an argument to the program.

The input file will have the following format:

* The first line will contain 2 numbers separated by space, N M, where N is the number of nodes in the undirected graph, and M is the number of edges.
* The next M lines will contain 2 numbers, the source and destination nodes between which a connection exists.

The program will write the result into the result.txt file as a list of nodes in topological order separated by space. (50 points). An additional 50 points will be awarded if the nodes follow the topological and lexicographical order.

**Example**

Executing the following command ./lab4b.out network.txt where the content of network.txt is:

5 4

2 1

3 1

1 0

1 4

The result.txt should contain one of the possible results, e.g., 3 2 1 4 0 or 2 3 1 0 4 which is also in lexicographical order.