Problem E. Flow

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

One of *Pang*'s research interests is the maximum flow problem.

A directed graph G with n vertices is *universe* if the following condition is satisfied:

- G is the union of k vertex-independent simple paths from vertex 1 to vertex n of the same length.

A set of paths is vertex-independent if they do not have any internal vertex in common.

A vertex in a path is called internal if it is not an endpoint of that path.

A path is simple if its vertices are distinct.

Let G be a *universe* graph with n vertices and m edges. Each edge has a non-negative integral capacity. You are allowed to perform the following operation any (including 0) times to make the maximum flow from vertex 1 to vertex n as large as possible:

Let e be an edge with positive capacity. Reduce the capacity of e by 1 and increase the capacity of another edge by 1.

Pang wants to know what is the minimum number of operations to achieve it?

Input

The first line contains two integers n and m ($2 \le n \le 100000, 1 \le m \le 200000$).

Each of the next m lines contains three integers x, y and z, denoting an edge from x to y with capacity z ($1 \le x, y \le n$, $0 \le z \le 1000000000$).

It's guaranteed that the input is a *universe* graph without multiple edges and self-loops.

Output

Output a single integer — the minimum number of operations.

Examples

standard input	standard output
4 3 1 2 1 2 3 2 3 4 3	1
4 4 1 2 1 1 3 1 2 4 2 3 4 2	1