Business Flow

You own a toy shop that has been very successful. Part of your business model is that you have a factory that produces all of the toys for your shop, and a warehouse which stores the toys temporarily until they are ready to ship out to customers or retailers. You also already have a nice network of trucks, vans, etc. which are used to transport your toys from the factory to the warehouse on a regular basis.

In expanding your operation though, you have made a possibly concerning decision. Instead of expanding into selling more toys, you have decided to expand into entirely new product lines (e.g., soda, video games, sports equipment, etc.) and you cannot manufacture these new items at the same factory. In fact, each new line of business you are purchasing has its own factory that produces the products, and its own warehouse where those products will be stored. You also do not want to purchase anymore trucks, etc. if you can avoid it. Thus, you want to figure out if there is a way to support the transport of the new product factories to the respective warehouses with the transport capacity that you already have.

Write a program that, given the product factories and their respective warehouses, the capacity of transporting products throughout your network, and the desired amount of each product to move, determines whether or not it is possible to move every product from its factory to its warehouse.

Your transportation network will be modeled as a directed graph $G = (V, E)$. Each node in G will either be a single product's factory, a single product's warehouse, or an intersection node. Intersection nodes are places where trucks can meet up and rearrange the products that are being held in each truck. For example, you might drive two trucks to an intersection node, where a third truck is waiting. Then you might spread the products out among all three trucks and send them each on a different path from there. Each edge in your graph will have a capacity (the amount of product that can be moved along that edge). To model this, each edge $e = (u, v) \in E$ contains a total capacity $c(u, v) \geq 0$. Your network does not have any anti-parallel edges (i.e., $e = (u, v) \in E \to e' = (v, u) \notin E$). Your business is comprised of p different products $P_1, P_2, ..., P_p$. Each product has a unique factory, warehouse, and demand. More formally, for each product i you have $P_i = (f_i, w_i, d_i) \mid f_i, w_i \in V, d_i \in \mathbb{N}$ (f_i is the node of the graph that is the factory for product i, w_i is the node that is the warehouse for product i, and d_i is the demand, how much product to move from factory to warehouse, for product i). You must also follow the following constraints:

- For each product i, the total we define the variable f_{iuv} to be the total amount of product i that is being moved from node u to node v.
- The aggregate flow from node u to node v is $f_{uv} = \sum_{i=1}^{p} f_{iuv}$. In other words, the aggregate flow is the sum of all of the products that are flowing along one particular edge.
- The aggregate flow must be less than or equal to the capacity of each edge: $\forall_{u,v} f_{uv} \leq c(u, v)$.
- All flows $f_{iuv} \geq 0$ (you can't push negative product through an edge)
- The aggregate flow going into a node must exactly equal the aggregate flow coming out of that node (conservation of flow).
- A network is feasible if there is a solution such that each product moves its demand exactly from each respective factory to each respective warehouse.

In solving this problem, we would like you to formulate the problem as a *Linear Program*. Keep the following in mind as you work to do so:

- You are not expected to implement a linear programming algorithm from scratch yourself. Research different algorithms for linear programming. Find library calls that solve linear programs in your language of interest. For this assignment only, you may ask generative AI to generate an implementation of a linear programming solver (ONLY for that component of the assignment).
- Spend time formulating this problem as a linear program. How will you pass the input into the linear programming algorithms you chose above? What form does the input need to take? Notice that the objective function here is actually nothing (we are just looking for any feasible solution). So...what can you set your objective function to to model this aspect of the problem?
- Note that all inputs to this problem will be integers (edge capacities, demands, etc.) but the solution (actual flow values) can be any real number. The company has determined that if your code produces a non-integer solution, they will review that by hand and decide what to do in a separate meeting.

Some notes on language, LP libraries, and efficiency

We recommend using **Python** for this assignment. This is primarily because the *SciPy* library is pre-configured on Gradescope (tested and working) and contains a linear programming subroutine you can use. We have a working solution in Python as well.

Regarding efficiency, this is not the primary focus of this assignment, so the time limit on the autograder is very generous and you should not need to worry about meeting the time limit if you have a correct working solution (unless you have an egregious inefficiency in your code).

If you use other tools for your *linear programming* implementation, consider the following:

1. If you need another Python module installed (via Pip install), you need to contact Floryan and he can configure the autograder so that it contains your module.
2. If your linear programming algorithm is contained within a Java jar file (or multiple ones), you should upload the jar as part of your solution and make sure your Makefile uses the -cp flag to build with the jar files.
3. If you use your own implementation (or one generated by AI), then you should just upload this implementation as part of your source code.
4. If you library is difficult to include or configure, then we reserve the right to reject your request and ask you to use another approach.

A note on grading

For this assignment, it is required that you use *linear programming* to solve the assignment instead of a custom algorithm you design yourself. The autograder will award up to 10 points for passing the test cases as usual. However, the TAs and I will also be awarding 10 points for using Linear Programming as intended (we will look at your code to ensure this). If you do not use Linear Programming for this assignment, then your max grade is a 50 percent.

Input

The first line of input will contain the number of nodes in the network $v \leq 100$ (implicitly indexed from 0 to $n-1$), the number of edges in the network $e \leq 5000$, and the number of products $p \leq 40$.

The next p lines will each contain the values f_i, w_i, and d_i (the index of the factory node, warehouse node, and demand of product i respectively). Note that $0 \leq f_i, w_i \leq n-1$ and $d_i \in \mathbb{N}$. Note that it is possible for multiple products to share the same factory or warehouse (or both), though it is also possible that they are all unique.

The next e lines will each contain the start index, end index, and capacity of an edge. Note that for all capacities, $c(u,v) \in \mathbb{N}$.

Output

Simply output the string "Yes" if it is possible to meet the demand of every product exactly without violating any other constraints of the problem. Output "No" if it is not possible to satisfy the demand of all products.

Sample Input

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

Sample Output

```
Yes
```