Overview In this lab, you will write a program that applies the max flow algorithm. You may write your assignment in any programming language that you choose. You may work alone or in pairs, and only per person per group needs to submit their code on blackboard. Make sure to include both participants in the comments of anything you submit or in a separate text file in your submission.

Problem Say we are playing a sportsball tournament where each team plays each other many times, and we are partway through the season. The problem here is to decide if a particular team has any possible chance of finishing in (or tieing for) first place or not. Here's a hypothetical situation with 5 teams, with each team playing a total of 40 games over the season.

Team	Current Win-Loss	Remaining	A	B	C	D	E
A	9-14	15	-	5	4	4	2
B	22-1	17	5	-	5	4	3
C	21-2	17	4	5	-	2	5
D	15-8	17	4	4	2	-	7
E	13-10	17	2	3	5	7	-

If team A wins all 15 of its remaining games, they would have 24 wins total, which makes it seem like they might have a shot at eventually tieing for first. However, lets work out the logic for what must happen with the other teams.

Notice that teams B and C have 5 remaining games with each other, and the only way for A to tie for first is if team B wins two of it's games against C, and C wins the remaing 3, and both B and C lose all games against all the other teams.

But then since B and C have to lose all their other games against D and E, teams D and E will end up with 21 wins each from those games. Thus the final 7 games between D and E will leave one of them with at least 25 wins by the pidgeonhole principle. Thus team A can never come in first!

In an real world situation with a large number n teams, finding a logical argument like the above to show that a team can possibly tie for first or can never win can get very complex, and sports writers regularly get it wrong.

Instead of finding a logical argument, we will design and implement an algorithm that can decide if a particular team can possibly win or not using max flow. Your final solution should not take much longer if all given numbers were multiplied by 1000.

Hint: In this example, the most possible wins that Team A can finish with is 24 wins (if E wins all remaining games). Thus, team A is still in the running for first place iff there is a way for all the *other* teams to end with at most 24 wins. In other words, there would need to be a way for team B to get at most 2 additional wins, and C to get at most 3, and D at most 9, etc. Try to create a flow graph where each "win" is represented by a unit of flow.

Your Assignment Describe a solution that solves the above problem, and implement it. You may write your algorithm in any language that you choose (but let me know if you plan to use something other than java, C++, or python). You may also import any code that you like from the internet. Implementations of Max Flow in java, C++, or python are easily found with a quick google search. Take some time to find one that has a built in graph implementation that you would like to interface with. Cite your sources for any code.

You should submit your written explanation of your solution below, as well as your code.

Input/Output Your code will read input from standard input given to you in the following format.

The first line of input is n the total number of teams

The next line of input contains n integers which specify the current number of wins for each team.

The next n lines of input represent the 2-D array of remaining games to be played. The jth integer of the ith row will represent the number of remaining games to be played between i and j. This input array will be symmetric, and will have 0s along the diagonal.

Your output should be information whether the first team listed can win/tie for first place or not. Simply print out "yes" or "no".

Example Input:

```
5
9 22 21 15 13
0 5 4 4 2
5 0 5 4 3
4 5 0 2 5
4 4 2 0 7
2 3 5 7 0
```

Example Output:

```
no
```

After/while reading all this information from standard in, you should create create an appropriate graph that you will eventually run the max-flow algorithm on.

Solution Who worked on this submission?

Solution:

Where did you source your implementation of Max Flow?

Solution:

Explain your construction and why it works (a formal proof isn't needed).

Solution:

Bonus (Up to 40 points)

(10 points each) Create an account on the ICPC programming contest archive. Show me that you've submitted working code for any of the following problems. 2016 - Waif After Dark 2017 - Transportation Delegation

(20 points) Describe and implement an algorithm that can *simultaneously* find all teams that can eventually tie/win in first place using *only one* call to max flow.