max planck institut informatik

Competitive Programming

Saarland University — Summer Semester 2020

Julian Baldus, Markus Bläser, Karl Bringmann, Marian Dietz, Simon Schwarz,
Christoph Weidenbach, Dominic Zimmer

Assignments Week 8

Deadline: **June 30, 2020 at 16:00 sharp**

Please submit solutions to the problems in our judge system, available at
`https://compro.mpi-inf.mpg.de/`.
You can find your credentials on your personal status page in our CMS.

Problem	`biglineup`		`bankrobbery`	`farmers`	`tilecut`
Points	3	3*	3	3	3
Difficulty					
Time Limit	3s	3s	10s	5s	3s
Memory Limit	2 GB	2 GB	2 GB	2 GB	2 GB

Please note:

- *The problem* `biglineup` *contains a bonus subtask. The 3 points available there do not count towards the number of points required for the admission to the exam.*
- Later, we will reopen the judge for the problems of the last weeks. However, you won't get any points for submissions of these problems.
- In the judge you can switch between the exercises of different weeks in the top-right corner.
- Your solution will be judged immediately after submitting. This may take some time, depending on the current server load.
- You can submit as many times as you want. However, don't abuse the server or try to extract the secret test cases.
- If your solution is **accepted**, you will receive the points specified in the table above.
- If you get **another verdict**, you will receive 0 points.

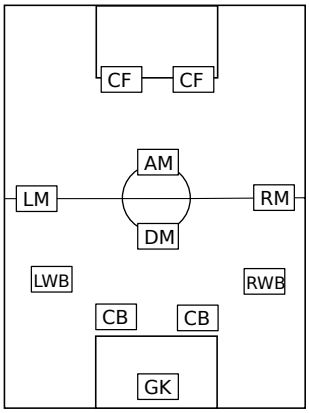
Big Lineup

Problem ID: biglineup

You may remember this problem from week 1. The only things that have changed are the size of the soccer team and a new constraint on the strengths of the players.

After first discussions of a resumption of the Bundesliga, you, as the first coach need to decide on a lineup for your first game. Unfortunately, some of your players have been infected, and therefore you can not use the usual lineup you used before. Luckily for you, exactly n players of your team seem not infected. Now you want to form the strongest team with the players that are left.

You have already decided on the tactical formation you wish to use, so now you need to select the players who should fill each of the n positions in the team. Since there are only n players left, it should be clear who is playing, but this still leaves the question of where to put which player.



Most players have a favoured position on the field where they are strongest, but some players are proficient in different positions. Your assistant has rated the playing strength of each of your n players in each of the n available positions in your formation, where a score of 1 means that this is an ideal position for the player and a score of 0 means that the player is not that proficient on this positions Find the lineup which maximises the sum of the playing strengths of your players for the positions you assigned them. Each position must be occupied by exactly one player.

Bonus Task (3 bonus points)

Note that the basic 2-chilli task appears as `biglineup1` *in the judge, and the bonus task as* `biglineup2`.

In this bonus task, the constraint $0 \leq s_{ij} \leq 1$ is replaced by $0 \leq s_{ij} \leq 100$ (that is, the only difference to `lineup` from week 1 is that n is not fixed to 11).

The solution is *not* a simple application of the algorithms from the lecture, although it matches thematically. As always, you are allowed to consult external sources under the condition that you cite them in your code.

Input

The input consists of n ($1 \leq n \leq 500$) lines, one for each player, where the i-th line contains n integer numbers s_{ij} between 0 and 1. s_{ij} describes the i-th player's strength on the j-th position.

Output

Print x, the maximum of the sum of player strengths over all possible lineups.

Sample Inputs

The first two samples may occur in both the basic and the bonus task. The third sample is valid for in the bonus task only.

Sample Input 1	Sample Output 1
4 0 1 1 1 1 1 0 0 0 1 0 0 1 0 0 0	3

Sample Input 2

```
4
1 1 0 0
0 1 1 0
0 1 1 0
0 0 1 1
```

Sample Output 2

```
4
```

Sample Input 3

```
11
100 0 0 0 0 0 0 0 0 0 0
0 80 70 70 60 0 0 0 0 0 0
0 40 90 90 40 0 0 0 0 0 0
0 40 85 85 33 0 0 0 0 0 0
0 70 60 60 85 0 0 0 0 0 0
0 0 0 0 0 95 70 60 60 0 0
0 45 0 0 0 80 90 50 70 0 0
0 0 0 0 0 40 90 90 40 70 0
0 0 0 0 0 0 50 70 85 50 0
0 0 0 0 0 0 66 60 0 80 80
0 0 0 0 0 0 50 50 0 90 88
```

Sample Output 3

```
970
```

Bank Robbery

Problem ID: bankrobbery

Dieter simply got no luck in the last few weeks. After a few promising dates, no women who was interested in him was wealthy enough to cover all of his debt. The hitman of the mafia is already on his way to Dieter (the deadline for delivering the money to the Mafia is at next Tuesday, 16:00). Therefore, Dieter has to take drastic measures. He decides that robbing the federal bank of the Saarland is the last way how he can collect enough money in this short timeframe.

The first part of his plan went great. He has sacked a large amount of money, and is just on his way out. However, after evading the police the last week, this time it won't be that easy. The police commander has ordered all available forces to block roads to catch Dieter.

Unfortunately, the police only has a very limited amount of personnel, but they want to be absolutely sure to catch Dieter. To catch him, they can build roadblocks on a few roads and check every person coming along those roads. Since some roads are large and some small, they all need different amounts of policemen to be blocked.

Dieter is currently at node 1 in the graph. The border of the Saarland is at node n. Therefore, Dieter wants to travel from node 1 to node n. Dieter can not travel through any road that has been blocked.

Dieter knows the exact amount of policemen that are stationed in the Saarland. Can you tell Dieter if the police is able to completely block all of his paths to the border?[1]

Input

The first line of the input contains an integer t. t test cases follow.

Each test case begins with a line containing three integers l, the amount of available policemen, n, the amount of intersections and m, the amount of roads. m lines follow, each consisting of three integers i, j, k, specifying a road from intersection i to j with k being the amount of policemen it takes to construct a roadblock on it. All roads are useable in both directions. Dieter always starts at intersection 1 and wants to get to the border (intersection n). Every test case ends with a blank line.

Output

For each test case, print a line containing "Case #i: t" with i being the number of the test case, starting at 1, and t being "yes" if the minimal amount of policemen it takes to construct at least one road block on every path from intersection 1 to n is smaller or equal to l, "no" otherwise.

Constraints

- $1 \leq t \leq 20$
- $1 \leq l \leq 10000$
- $2 \leq n \leq 400$
- $1 \leq m \leq 100000$
- $1 \leq i, j \leq n$
- $1 \leq k \leq 10000$
- There will always be at least one possible path from 1 to n.

[1] Then he could surrender and hope for a pardon

Sample Input 1	Sample Output 1
3 10 2 3 2 2 3 1 2 4 1 1 3 2 3 3 1 3 5 1 2 5 3 3 5 9 2 1 2 1 10	Case #1: yes Case #2: no Case #3: no

Sample Input 2

```
6
8 2 3
2 2 4
1 2 5
1 1 4

2 5 5
1 5 1
5 4 1
4 2 1
5 3 1
1 2 1

1 9 11
1 4 1
4 8 1
8 2 1
2 7 1
2 3 1
7 5 1
1 6 1
6 9 1
2 8 1
2 9 1
5 7 1

2 7 9
1 6 1
1 5 1
1 4 1
4 3 1
1 2 1
3 7 1
2 6 1
4 5 1
6 7 1

1 3 3
1 2 1
1 3 1
1 3 1

2 3 2
1 3 1
3 2 1
```

Sample Output 2

```
Case #1: yes
Case #2: yes
Case #3: no
Case #4: yes
Case #5: no
Case #6: yes
```

Farmers

Problem ID: farmers

You are the administrator of a big, fertile area. You area has size $n \times m$ and consists of $n \cdot m$ squares. Some squares of your land are, however, obstructed by rocks. They are infertile and it is impossible to farm on those squares.

Just a year ago, everything was fine. In each fertile square, exactly one farmer had its farm. However, a new deadly virus has killed exactly half of your farmers. To stay at the same productivity, you now want to assign 2 squares to each remaining farmer. Because tractors are yet to be invented, farmers would appreciate if their new property would consist of two *adjacent* squares (squares that are connected either horicontally or vertically). Note that diagonal connections do not count as adjacent. And, of course: You do not want to assign infertile land to any farmer.

Can you determine if it is possible to assign the fertile squares to the farmers such that each farmer gets exactly two adjacent squares?

Input

The first line of the input contains two numbers n and m ($1 \leq n, m \leq 100$), the size of the land you administer. The next n lines each contain m characters which are either `.`, representing a square of fertile land, or `#`, representing an infertile square of land.

Output

Print `YES` if it is possible to assign the fertile squares in pairs, and `NO` otherwise. Each fertile square must be part of exactly one pair, and the two squares in each pair must be adjacent.

Sample Input 1

```
2 4
..#.
#...
```

Sample Output 1

```
YES
```

Sample Input 2

```
3 3
.#.
.#.
.##
```

Sample Output 2

```
NO
```

Sample Input 3

```
6 3
#.#
...
#..
#..
...
#.#
```

Sample Output 3

```
NO
```

Tile Cut

Problem ID: tilecut

When Frodo, Sam, Merry, and Pippin are at the Green Dragon Inn drinking ale, they like to play a little game with parchment and pen to decide who buys the next round. The game works as follows:

Given an $n \times m$ rectangular tile with each square marked with one of the incantations `W`, `I`, and `N`, find the maximal number of triominoes that can be cut from this tile such that the triomino has `W` and `N` on the ends and `I` in the middle (that is, it spells `WIN` in some order). Of course the only possible triominoes are the one with three squares in a straight line and ell-shaped ones. The Hobbit that is able to find the maximum number wins and chooses who buys the next round. Your job is to find the maximal number.

Side note: Sam and Pippin tend to buy the most rounds of ale when they play this game, so they are lobbying to change the game to Rock, Parchment, Sword (RPS)!

Input

The first line of the input contains an integer t ($1 \leq t \leq 20$). t test cases follow, each of them separated by a blank line.

The first line of each test case consists of the two integers m and n ($1 \leq m, n \leq 30$). Then, n lines with m characters each follow, representing the rectangular grid containing only the letters `W`, `I`, and `N`.

Output

For each test case, print a line containing "Case #i: x" where i is its number, starting at 1, and x is the maximum total number of tiles that can be formed. Each line of the output should end with a line break.

Sample Input 1	Sample Output 1
3 4 4 WIIW NNNN IINN WWWI 5 5 NINWN INIWI WWWIW NNNNN IWINN 3 1 NIW	Case #1: 5 Case #2: 5 Case #3: 1