ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Algorithms Lab HS18
Department of Computer Science
Prof. Dr. A. Steger, Prof. Dr. E. Welzl, Prof. Dr. P. Widmayer
cadmo.ethz.ch/education/lectures/HS18/algolab

Threefold Problem Sheet – Domino Magic

This sheet presents three problems based on a common storyline. The three problems share several characteristics and as a result they appear quite similar in nature. But this does not necessarily mean that these problems can be solved using the same strategies and techniques. In fact, the whole idea of these threefold sheets is to highlight how possibly subtle differences and changes in a problem formulation can have a great impact on how the resulting problem can be solved. We believe that studying these sheets is an excellent practice of how to approach the modeling aspects of solving a problem.

Read through the problems and try to work out by yourself possible lines of modeling and then algorithmically solving them. Give reasons of why or why not certain approaches appear plausible. Ideally, substantiate your ideas by sketching corresponding proof ideas.

This sheet was handed out during the tutorial on November 21, 2018, and solved and discussed there as well.

Main Story

Domino Magic has been a producer for stylish domino game pieces for several years. Their stylish designs and quality materials have always ensured them a safe position on the market. Recently however, due to the global financial crisis, the demand for luxurious domino games has drastically dropped. In order to stay alive, the company has decided to expand into the garden floor tiling business (producing rectangular tiles of a length to width ratio of 2:1).

Exercise – *Domino Snake*

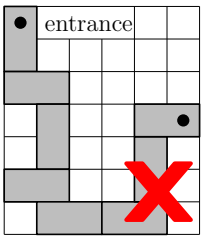
Domino Magic's garden tile division has quickly managed to break even, so its CEO decides to manufacture a batch of garden tile dominoes in his favourite colour, *bilious green*. This in turn catches the attention of Hissing House, the world's leading zoo on dangerous snakes.

Hissing house would like to guide their visitors through the future brand new snake house from its entrace to its exit by a *domino snake* on the floor. A domino snake consists of consecutively tiled bilious green dominoes.

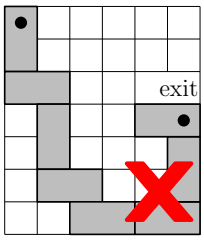
Problem Write a program to help Domino Magic to decide whether they can satisfy the requirements by Hissing House for a variety of floor plans. More specifically, each floor plan has a grid structure, where squares belong either to the floor (and thus can be tiled) or to a snake cage. The floor plan should be evaluated for several pairs of entrance and exit locations.

Each domino of the domino snake has to cover exactly two horizontally or vertically adjacent squares. The snake needs to start on the entrance and end on the exit. Furthermore the first square of every domino needs to touch the second square of its preceeding domino (see the figure for invalid snakes).

You can assume that the supply of green bilious dominos is sufficiently large.



dominos misaligned



finishes incorrectly

Input The first line contains $1 \leqslant t \leqslant 30$, the number of testcases. Each of the t testcases is described as follows:

- It starts with a single line that contains three integers `h w p`, separated by a space, specifying the height h and the width w of the floor plan at hand ($1 \leqslant h, w \leqslant 1'000$), and the number p of entrance-exit pairs which should be tested ($0 \leqslant p \leqslant 1'000$).
- The next p lines each describe one entrance-exit pair.
  The positions of the entrance (q, r) and the exit (s, t) are given by four space-separated integers `q r s t` (where $1 \leqslant q, s \leqslant h$, $1 \leqslant r, t \leqslant w$, e.g. $(1, w)$ is the top right corner).
- The last h lines each describe one row of the floor plan, ordered from top to bottom.
  Each such line consists of w characters: '`1`' if that square can be tiled and '`0`' if it belongs to a snake cage. The surrounding walls of Hissing House are built from snake cages as well, hence the border cells of the input consist of `0`'s.

Output For every test case output a single line consisting of p characters "y" or "n" (one letter per entrance/exit pair). Output the letter "y" if a domino snake can be tiled and "n" otherwise.

Points There are two test sets:

1. For the first set, worth 40 points, you may assume that $p = 1$.
2. For the second set, worth 60 points, there are no additional constraints.

Sample Input

```
2
6 6 4
2 3 4 5
3 2 3 5
5 2 5 5
3 2 5 3
000000
001100
011110
011110
001100
000000
5 6 2
4 2 2 5
2 3 4 4
000000
001010
011000
011110
000000
```

Sample Output

```
nyny
ny
```

Exercise – *New Tiles*

Problem Domino Magic started releasing brand new 2×2 tiles because everybody is already tired of the 1×2 domino tiles. To make a long story short, you are given a rectangular floor plan with some cells which you are not allowed to place tiles on, and your goal is to maximize the number of new 2×2 tiles you can put in this rectangle without overlapping.

Input The first line contains $1 \leqslant t \leqslant 20$, the number of testcases. Each of the t testcases is described as follows:

- It starts with a single line that contains two integers `h w`, separated by a space, specifying specify the height h and the width w of the floor plan at hand ($1 \leqslant h \leqslant 100$, $1 \leqslant w \leqslant 17$).
- The following h lines each describe one row of the floor plan, ordered from top to bottom. Each such line consists of w characters: '`1`' if that square can be tiled and '`0`' if it cannot be used. You may assume that the floor plan is surrounded by a wall, so the border cells of the input consist of `0`'s.

Output For each test case output a single line with the maximum number of new 2x2 tiles you can place on the grid without overlapping.

Points There are two test sets:

1. For the first set, worth 50 points, you may assume that $w \leqslant 10$.
2. For the second set, worth 50 points, there are no additional constraints.

Sample input

```
2
5 5
00000
00110
01110
01110
00000
5 6
000000
001100
011110
011110
000000
```

Sample output

```
1
2
```

Exercise – *Snakes strike back*

You get an emergency call from Hissing House, because a child walking on the Domino Snake from the first exercise has been bitten by a snake. Tours for visitors have been cancelled and the floor of the zoo has to be remodeled.

Hissing House would like to keep the cage layout. In a first step, the path on the floor should (as before) be at least 1m wide everywhere (i.e. at least as wide as the size of a square on the floor plan). *Additionally* the path should not get closer than 0.5m to any snake cage. In a second step, they would like to test whether these values can be doubled. Finally, they want to consider an arbitrary path width of p meters (with a security margin of $p/2$ meters).

Problem To make it easier for you to decide whether this is possible, you are given one entrance and one exit. Furthermore you are not required to use dominoes for the path; it may bend and snake its way through the zoo as long as it keeps a minimum width of p meters (i.e. for every point on the edge of the path the closest point on the other side of the path must have distance $\geqslant p$ meters).

Input The first line contains $1 \leqslant t \leqslant 20$, the number of testcases. Each of the t testcases is described as follows:

- It starts with a single line that contains three integers `h w p`, separated by a space, specifying the height h and the width w of the floor plan at hand ($1 \leqslant h, w \leqslant 500$), as well as the path width p ($1 \leqslant p \leqslant 30$, all values in meters).
- The next line defines the entrance (q, r) and the exit (s, t), given by four space-separated integers `q r s t` (where $1 \leqslant q, s \leqslant h$ and $1 \leqslant r, t \leqslant w$, e.g. $(1, w)$ corresponds to the top right corner).
- The last h lines each describe one row of the floor plan, ordered from top to bottom. Each such line consists of w characters: '`1`' if the full $1m \times 1m$ square belongs to the floor and '`0`' if it belongs completely to a snake cage.

It is guaranteed that the $(2p+1) \times (2p+1)$ rectangles centered at the entrance and at the exit do not contain any snake cage (i.e. you don't need to worry about boundary conditions at the entrance and exit). Furthermore the surrounding walls of Hissing House are built from snake cages as well, hence the border cells of the input consist of `0`'s.

Output For every test case output a single line containing the word 'yes' if the given floor plan allows for a path from the entrance to the exit which has width $\geqslant p$ meters and keeps clear of all snake cages by at least $p/2$ meters, and 'no' otherwise.

Points There are three test sets:

1. For the first set, worth 30 points, you may assume that $p = 1$.
2. For the second set, worth 30 points, you may assume that $p = 2$.
3. For the third set, worth 40 points, there are no additional constraints.

Sample Input

```
3
8 8 1
3 3 6 6
00000000
01111110
01111010
01111110
01111110
01011110
01111110
00000000
7 7 1
3 3 5 5
0000000
0111110
0111000
0111110
0001110
0111110
0000000
11 11 2
4 4 8 8
00000000000
01111100000
01111100000
01111110000
01111111000
01111111110
00011111110
00001111110
00000111110
00000111110
00000000000
```

Sample Output

```
yes
no
yes
```

Briefly sketch your approach for each of the three exercises.

Exercise 1) — *Domino Snakes*

Method(s): ☐ Brute force ☐ Greedy ☐ Backtracking ☐ Dynamic programming ☐ Sorting ☐ Sliding window ☐ Binary search ☐ Graph components ☐ Minimum spanning tree ☐ BFS/DFS ☐ Shortest paths (Dijkstra) ☐ Maximum matching ☐ Network flows ☐ MinCost flows ☐ Delaunay triangulation ☐ Minimum enclosing circle ☐ LP ☐ QP

Time Complexity:

Reasoning:

Exercise 2) — *New Tiles*

Method(s): ☐ Brute force ☐ Greedy ☐ Backtracking ☐ Dynamic programming ☐ Sorting ☐ Sliding window ☐ Binary search ☐ Graph components ☐ Minimum spanning tree ☐ BFS/DFS ☐ Shortest paths (Dijkstra) ☐ Maximum matching ☐ Network flows ☐ MinCost flows ☐ Delaunay triangulation ☐ Minimum enclosing circle ☐ LP ☐ QP

Time Complexity:

Reasoning:

Exercise 3) — *Snakes strike back*

Method(s): ☐ Brute force ☐ Greedy ☐ Backtracking ☐ Dynamic programming ☐ Sorting ☐ Sliding window ☐ Binary search ☐ Graph components ☐ Minimum spanning tree ☐ BFS/DFS ☐ Shortest paths (Dijkstra) ☐ Maximum matching ☐ Network flows ☐ MinCost flows ☐ Delaunay triangulation ☐ Minimum enclosing circle ☐ LP ☐ QP

Time Complexity:

Reasoning:

Please provide some feedback on the *BGL* tutorials

What is the "muddiest point" of the BGL lectures / Which part of the BGL tutorials is the most unclear to you? Be *very specific*.

Your feedback on this threefold problem set:

Your feedback on the BGL tutorials: