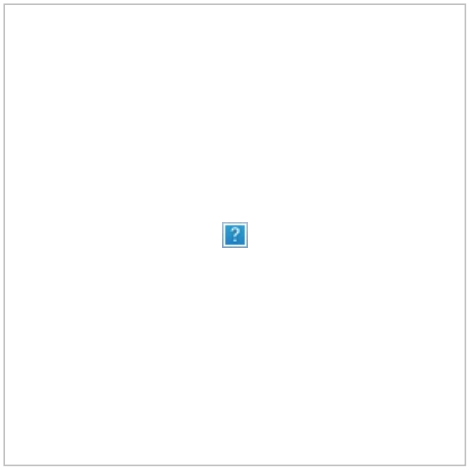
English version | [ver versão em português]

[PI042] Pizza Quest



There’s a city-wide blackout, and the automatic doors of the local supermarket just opened for a limited time. Julie Tribbiani is on a mission: rescue the last pizza before it’s gone forever.

But there’s a catch — the supermarket has turned into a maze of toppled shelves and blocked aisles, and Julie’s sense of direction isn’t exactly award-winning. Starting from his current position, he needs to figure out if he can navigate through the maze to reach the pizza.

The Supermarket Maze

The layout of the supermarket is given as a 2D grid:

- 'J' marks Julie's starting point.
- 'P' marks the precious pizza.
- '.' marks a walkable path (clear floor).
- '#' marks a wall or obstacle.

Julie can move up, down, left, or right — but not diagonally (he’s wearing flip-flops, after all).

The Problem

For each supermarket layout, determine whether there exists a valid path from Julie to the pizza, using only walkable cells ('.').

Input

The first line contains an integer `N (1 ≤ N ≤ 20)` — the number of test cases.

Each test case consists of:

- A line with two integers `R` and `C` (`1 ≤ R, C ≤ 100`) — the number of rows and columns in the grid.
- Followed by `R` lines, each with `C` characters representing the map.

There will always be exactly one `'J'` and one `'P'` in each map.

Output

Print `N` lines — one per test case. For each, print:

- `yes`, if Julie can reach the pizza
- `no`, if there's no path to it

Example Input 1

```
2
5 10
..#.......
###...#...
..#...#.PS
.J#...####
......#...
4 4
J..#
..#.
.#..
#..P
```

Example Output 1

```
yes
no
```

On the first case Julie can reach all the positions in green, which include reaching the pizza (hence, the answer is "yes"):

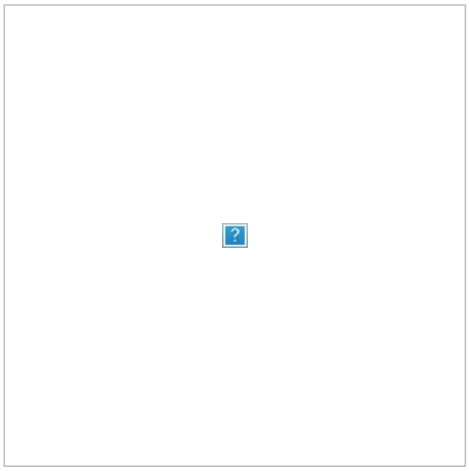
```
..#.......
###...#...
..#...#..P
.J#...####
......#...
```

On the second case Julie can reach all the positions in green, but has no way to reach the pizza (hence, the answer is "no"):

```
J..#
..#.
.#..
#..P
```

Versão em Português | [see english version]

[PI042] Missão Pizza



Há um apagão geral na cidade, e as portas automáticas do supermercado local abriram por tempo limitado. Julie Tribbiani está em uma missão: resgatar a última pizza antes que ela desapareça para sempre.

Mas há um problema — o supermercado virou um labirinto de prateleiras tombadas e corredores bloqueados, e o senso de direção da Julie não é dos melhores. Partindo da sua posição atual, ela precisa descobrir se consegue navegar pelo labirinto até chegar à pizza.

O Labirinto do Supermercado

A planta do supermercado é representada por uma grelha 2D:

- 'J' marca a posição inicial da Julie.
- 'P' marca a localização da preciosa pizza.
- '.' representa um caminho livre (chão desimpedido).
- '#' representa uma parede ou obstáculo.

Julie pode mover-se para cima, baixo, esquerda ou direita — mas não na diagonal (afinal, ela está de chinelos).

O Problema

Para cada planta do supermercado, determine se existe um caminho válido de Julie até a pizza, usando apenas células caminháveis ('.').

Entrada

A primeira linha contém um inteiro `N (1 ≤ N ≤ 20)` — o número de casos de teste.

Cada caso de teste consiste em:

- Uma linha com dois inteiros `R` e `C` (`1 ≤ R, C ≤ 100`) — o número de linhas e colunas da grelha.
- Seguidas por `R` linhas, cada uma com `C` caracteres representando o mapa.

Haverá sempre exatamente um `'J'` e um `'P'` em cada mapa.

Saída

Imprima `N` linhas — uma por caso de teste. Para cada uma, imprima:

- `yes`, se Julie consegue alcançar a pizza
- `no`, se não há caminho até ela

Exemplo de Entrada 1

```
2
5 10
..#.......
###...#...
..#...#.PS
.J#...####
......#...
4 4
J..#
..#.
.#..
#..P
```

Exemplo de Saída 1

```
yes
no
```

No primeiro caso, Julie consegue alcançar todas as posições em verde, incluindo a pizza (portanto, a resposta é "yes"):

```
..#.......
###...#...
..#...#..P
.J#...####
......#...
```

No segundo caso, Julie consegue alcançar todas as posições em verde, mas não tem como chegar até a pizza (portanto, a resposta é "no"):

```
J..#
..#.
.#..
#..P
```
