MAC 115 – Introdução à Computação – IF Noturno

(*Versão quase final*)

Um Minotauro Perdido & Percolação

Exercício-Programa 4 (EP4) Data de entrega: **1/12/2014 (23:55)**

1 O labirinto e o minotauro perdido

Considere um labirinto como na Figura 1.

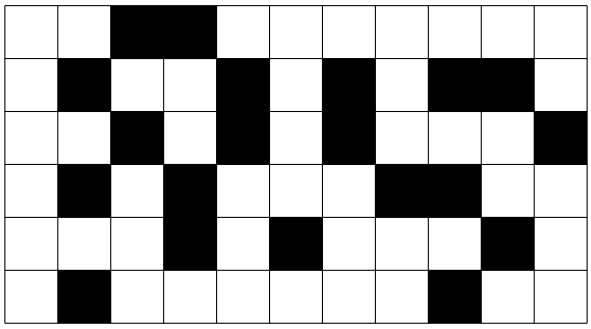


Figura 1: Um labirinto

Um labirinto desses pode ser representado por uma matriz retangular L, cujo elemento L_{ij} vale 0 ou -1 conforme a casa correspondente do labirinto seja uma passagem livre ou uma parede, respectivamente.

Suponha agora que em alguma posição desse labirinto haja um *minotauro perdido*, e também uma porta (de entrada/saída). Queremos ajudar o minotauro a encontrar a porta.

Um método geral para resolver esse problema consiste em marcar com o número k ($k = 1, 2, \ldots$) as casas livres que estejam a exatamente $k - 1$ passos de distância da porta, pelo caminho mais curto possível. Suponha que, a cada passo, o minotauro possa se deslocar de apenas uma casa na vertical ou na horizontal. Então, rotula-se inicialmente a posição da porta com 1 e para cada $k \geq 2$ examinam-se certas casas livres do labirinto, marcando-se com k aquelas ainda não marcadas e que sejam vizinhas a alguma casa marcada com $k - 1$.

A marcação continua até ser atingido um valor de k (28 no exemplo abaixo) tal que nenhuma casa esteja em condições de ser marcada. Supondo que a $L = [1\,..\,6, 1\,..\,11]$,[1] e que a porta esteja na posição $[6, 11]$, ao final da marcação teremos a matriz na Figura 2.

[1] Esta notação deve ser intuitiva o suficiente.

26	27	−1	−1	12	11	10	9	10	11	12
25	−1	0	0	−1	12	−1	8	−1	−1	13
24	25	−1	0	−1	13	−1	7	6	5	−1
23	−1	21	−1	15	14	15	−1	−1	4	3
22	21	20	−1	16	−1	16	17	18	−1	2
23	−1	19	18	17	18	17	18	−1	2	1

Figura 2: A matriz rotulada

Tendo feito esta marcação, um caminho mais curto do minotauro até a porta (se existir) pode ser determinado partindo-se da posição do minotauro e passando a cada etapa para uma casa vizinha cuja numeração seja menor do que a atual.

Por exemplo, se o minotauro estiver na posição $[3, 2]$, ele precisará percorrer pelo menos 25 casas para chegar à porta:

$$[3,2]\ \ [3,1]\ \ [4,1]\ \ [5,1]\ \ [5,2]\ \ [5,3]\ \ \ldots\ \ [4,10]\ \ [4,11]\ \ [5,11]\ \ [6,11] \tag{1}$$

2 Seu programa

Faça um programa em C para resolver o problema descrito a seguir.

São dados dois inteiros positivos m e n, uma matriz $L = [1\,..\,m, 1\,..\,n]$, representando um labirinto como descrito na Seção 1, a posição $[m1, m2]$ de um minotauro e a posição $[p1, p2]$ da porta desse labirinto. Queremos

(i) determinar se existe ou não um caminho para o minotauro chegar à porta do labirinto e

(ii) se houver caminhos como em (i), determinar um tal caminho mais curto.

2.1 Funções auxiliares

Seu programa deve implementar as funções descritas abaixo (pelo menos).

- Uma função com o protótipo

```
void marque(int L[][MAXCOL], int m, int n, int p1, int p2);
```

que tem como parâmetros um labirinto, isto é, uma matriz de inteiros $L = [1\,..\,m, 1\,..\,n]$, suas dimensões m e n, e dois inteiros $p1$, $p2$ que indicam a posição $[p1, p2]$ da porta nesse labirinto.

Essa função deve efetuar a marcação da matriz L, como explicado na Seção 1.

- Uma função com o protótipo

```
int ha_caminho(int L[][MAXCOL], int m, int n, int m1, int m2);
```

que tem como parâmetros uma matriz de inteiros $L[1\,..\,m, 1\,..\,n]$ já marcada, as dimensões m e n dessa matriz, e dois inteiros $m1$ e $m2$ indicando a posição $[m1, m2]$ do minotauro nessa matriz.

Esta função deve devolver 1 caso haja um caminho que leva o minotauro à porta e deve devolver 0 caso contrário.

- Uma função com o protótipo

```
int imprima_caminho(int L[][MAXCOL], int m, int n, int m1, int m2);
```

que tem os mesmos parâmetros que `ha_caminho()` e que encontra um caminho de comprimento mínimo da porta até a posição $[m1, m2]$, onde está o minotauro (esta função deve ser chamada somente no caso de haver um caminho). Esta função deve devolver o comprimento (número de "casas") no caminho encontrado e também deve imprimir o caminho encontrado. A saída deve ser como no exemplo (1) (com todas as casas que constituem o caminho, isto é, sem as reticências).

- Uma função com o protótipo

```
void gere_lab(int L[][MAXCOL], int m, int n, double p);
```

que recebe inteiros positivos m e n e um número real p, com $0 \leq p \leq 1$, e que gera um labirinto aleatório com m linhas e n colunas, com cada "casa" do labirinto desbloqueada com probabilidade p, independentemente de todas as outras casas. Este labirinto aleatório gerado deve ser devolvido em L.

- Uma função com protótipo

```
void gere_psms(int L[][MAXCOL], int m, int n,
               int *p1, int *p2, int *m1, int *m2);
```

que recebe um labirinto L com m linhas e n colunas e gera uma posição aleatória $[*p1, *p2]$ para a porta e uma posição aleatória $[*m1, *m2]$ para o minotauro. Note que não deve haver conflito entre as posições geradas e L, isto é, a porta só pode estar em uma posição desbloqueada de L, assim como o minotauro só pode estar em uma posição desbloqueada de L.

2.2 Formas de execução do programa

Seu programa deve ter pelo menos três formas de execução. A forma de execução é regida pelo formato da entrada. O primeiro caractere da entrada define a forma de execução.

2.2.1 Forma basica

Nesta forma, a entrada deve ser, por exemplo, como segue:

```
b
10 20
  0  0  0 -1 -1  0  0 -1  0  0  0 -1 -1 -1 -1  0 -1 -1  0  0
  0  0  0  0  0 -1  0 -1 -1  0 -1  0  0  0  0  0  0  0  0 -1
 -1  0  0 -1  0  0  0  0  0  0 -1  0 -1  0  0  0  0 -1  0  0
  0  0  0 -1  0 -1  0 -1  0 -1  0  0  0 -1 -1  0  0  0 -1  0
  0 -1  0  0  0 -1  0  0 -1 -1  0  0 -1 -1 -1  0  0  0 -1 -1
 -1  0  0  0  0 -1  0 -1 -1  0  0  0  0  0  0  0  0  0 -1  0
 -1  0  0 -1  0  0 -1  0  0 -1  0  0  0  0  0  0  0  0  0  0
  0  0  0 -1  0  0  0  0  0 -1 -1  0  0  0 -1  0  0 -1 -1  0
  0  0 -1 -1  0  0  0  0  0 -1 -1  0  0  0  0  0  0  0  0  0
  0 -1 -1  0 -1  0 -1 -1  0  0  0 -1  0  0  0  0  0  0  0  0
8 7
1 3
```

O caractere `b` indica que estamos nesta forma "básica" de execução. A seguir, são dados o número de linhas e o número de colunas do labirinto e o labirinto em si. Depois são dadas as posições da porta e do minotauro.

Com as informações dadas nessa entrada, seu programa deve imprimir o labirinto "marcado" (como discutido na Seção 1), deve imprimir um caminho de comprimento mínimo para o minotauro chegar à porta e, finalmente, deve imprimir uma representação gráfica do labirinto e do caminho encontrado, como na saída abaixo:

```
Matriz marcada:
 14 13 12 -1 -1 13 12 -1 16 15 16 -1 -1 -1 -1  0 -1 -1  0  0
 13 12 11 10  9 -1 11 -1 -1 14 -1  0  0  0  0  0  0  0  0 -1
 -1 11 10 -1  8  9 10 11 12 13 -1  0 -1  0  0  0  0 -1  0  0
 11 10  9 -1  7 -1 11 -1 13 -1  0  0  0 -1 -1  0  0  0 -1  0
 12 -1  8  7  6 -1 12 13 -1 -1  0  0 -1 -1 -1  0  0  0 -1 -1
 -1  8  7  6  5 -1 13 -1 -1  0  0  0  0  0  0  0  0  0 -1  0
 -1  9  8 -1  4  3 -1  3  4 -1  0  0  0  0  0  0  0  0  0  0
 11 10  9 -1  3  2  1  2  3 -1 -1  0  0  0 -1  0  0 -1 -1  0
 12 11 -1 -1  4  3  2  3  4 -1 -1  0  0  0  0  0  0  0  0  0
 13 -1 -1  0 -1  4 -1 -1  5  6  7 -1  0  0  0  0  0  0  0  0

Ha caminho.
[ 1,  3] [ 2,  3] [ 2,  4] [ 2,  5] [ 3,  5] [ 4,  5] [ 5,  5]
[ 6,  5] [ 7,  5] [ 7,  6] [ 8,  6] [ 8,  7]
Comprimento: 12
```

```
----------------------------------------
     M X X     X       X X X X   X X
     * * * X   X X   X                 X
 X     X *           X   X         X
       X * X   X   X       X X       X
   X     * X     X X     X X X       X X
 X       * X   X X                   X
 X     X * * X     X
       X   * P     X X       X     X X
     X X           X X
   X X   X   X X       X
----------------------------------------
```

2.2.2 Labirinto aleatório

Nesta forma, a entrada deve começar, por exemplo, como segue:

```
r
8 15 0.7 31415
```

O caractere `r` indica que estamos nesta forma "aleatória" de execução. A seguir, são dados o número de linhas e o número de colunas do labirinto. Também é dada uma probabilidade p ($p = 0.7$ no exemplo acima) e um inteiro que deve ser usado como a semente do gerador de números aleatórios (use `srandom()` e `random()` em seu programa). Seu programa deve então imprimir um labirinto aleatório com probabilidade de cada casa estar desbloqueada igual a p. Após ver o labirinto gerado, o usuário irá fornecer as posições da porta e do minotauro (o usuário irá escolher posições desbloqueadas do labirinto).

Seu programa deve então imprimir o labirinto "marcado", deve imprimir um caminho de comprimento mínimo para o minotauro chegar à porta e, finalmente, deve imprimir uma representação gráfica do labirinto e do caminho encontrado. Segue um exemplo:

```
r
8 15 0.7 31415

Labirinto gerado:
  0 -1  0  0  0  0 -1  0  0  0 -1  0 -1  0 -1
 -1  0 -1 -1 -1 -1  0 -1 -1  0 -1  0  0  0  0
  0 -1  0 -1  0  0  0  0  0  0  0  0 -1  0 -1
 -1  0  0  0  0  0  0  0  0  0  0  0 -1 -1  0  0
  0 -1  0  0  0  0  0  0  0 -1 -1 -1 -1 -1  0
 -1  0  0 -1 -1  0 -1  0  0 -1  0  0 -1  0  0
 -1  0  0  0  0  0 -1  0  0 -1  0 -1  0  0  0
 -1 -1  0  0 -1  0  0  0  0  0  0  0  0  0 -1

Porta:
3 3    /* fornecido pelo usuario  */
```

```
Minotauro:
7 15   /* fornecido pelo usuario  */

Matriz marcada:
  0 -1  0  0  0  0 -1 14 13 12 -1 14 -1 16 -1
 -1  0 -1 -1 -1 -1  8 -1 -1 11 -1 13 14 15 16
  0 -1  1 -1  5  6  7  8  9 10 11 12 -1 16 -1
 -1  3  2  3  4  5  6  7  8  9 10 -1 -1 17 18
  0 -1  3  4  5  6  7  8  9 -1 -1 -1 -1 -1 19
 -1  5  4 -1 -1  7 -1  9 10 -1 16 17 -1 19 20
 -1  6  5  6  7  8 -1 10 11 -1 15 -1 17 18 19
 -1 -1  6  7 -1  9 10 11 12 13 14 15 16 17 -1

Ha caminho.
[ 7, 15] [ 7, 14] [ 7, 13] [ 8, 13] [ 8, 12] [ 8, 11]
[ 8, 10] [ 8,  9] [ 7,  9] [ 6,  9] [ 5,  9] [ 4,  9]
[ 4,  8] [ 4,  7] [ 4,  6] [ 4,  5] [ 4,  4] [ 4,  3]
[ 3,  3]
Comprimento: 19

------------------------------
   X         X       X   X   X
 X   X X X X   X X   X
   X P X                 X   X
 X   * * * * * * *     X X
   X             * X X X X X
 X     X X   X   * X     X
 X           X   * X   X * * M
 X X     X       * * * * *   X
------------------------------
```

2.2.3 Instâncias integralmente aleatórias

Nesta forma, a entrada deve ser, por exemplo, como segue:

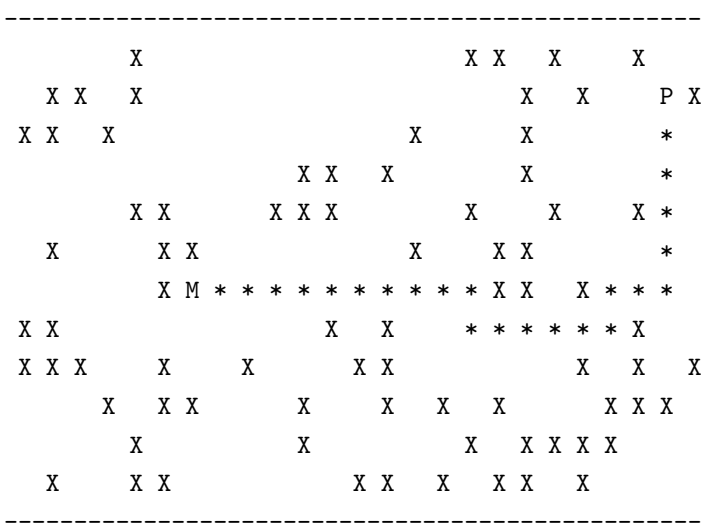
```
R
12 25 0.7 2020
```

O caractere `R` indica que estamos nesta forma "integralmente aleatória" de execução. A seguir, são dados o número de linhas e o número de colunas do labirinto. Também é dada uma probabilidade p ($p = 0.7$ no exemplo acima) e um inteiro que deve ser usado como a semente do gerador de números aleatórios (use `srandom()` e `random()` em seu programa). Seu programa deve então gerar um labirinto aleatório com probabilidade de cada casa estar desbloqueada igual a p. Seu programa deve também gerar uma posição aleatória para a porta e para o minotauro. Seu programa deve imprimir o labirinto gerado e as posições da porta e minotauro.

Seu programa deve então imprimir o labirinto "marcado", deve imprimir um caminho de

comprimento mínimo para o minotauro chegar à porta e, finalmente, deve imprimir uma representação gráfica do labirinto e do caminho encontrado. Segue um exemplo:

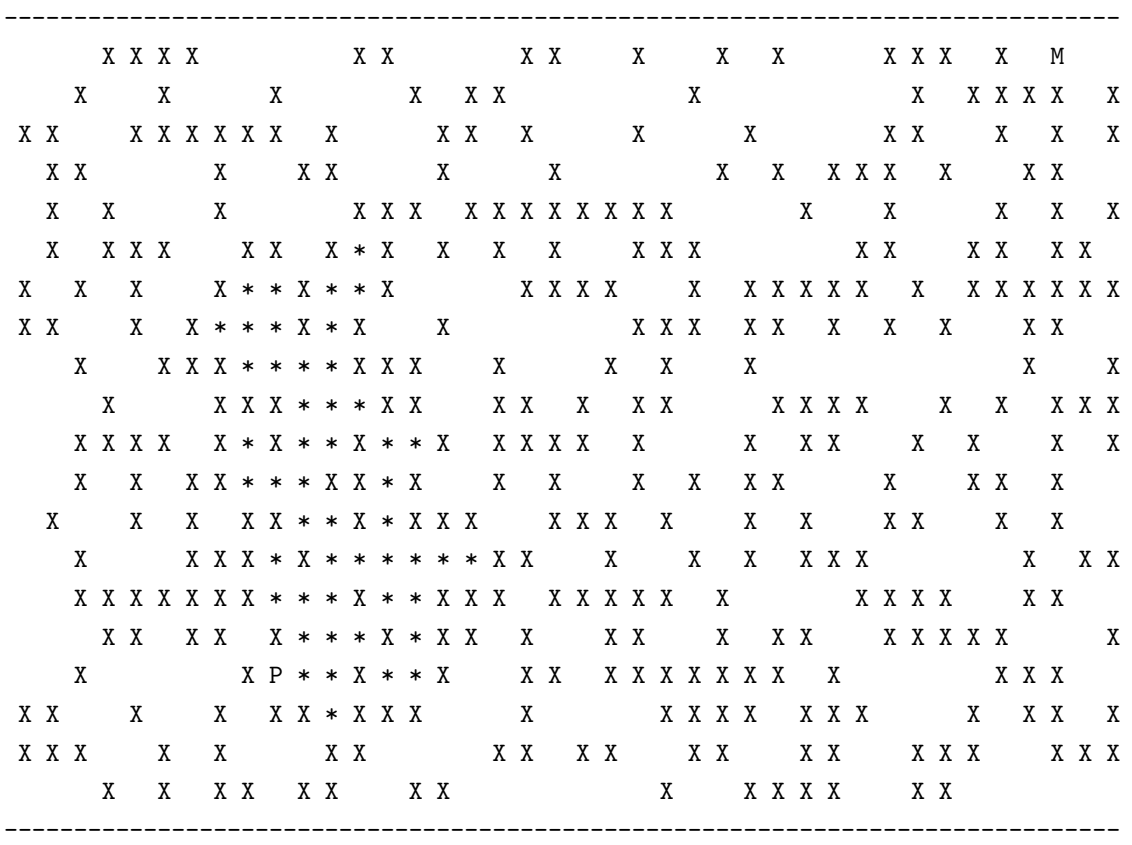
```
Labirinto gerado:
  0  0  0  0 -1  0  0  0  0  0  0  0  0  0  0  0 -1 -1  0 -1  0  0 -1  0  0
  0 -1 -1  0 -1  0  0  0  0  0  0  0  0  0  0  0  0  0 -1  0 -1  0  0  0 -1
 -1 -1  0 -1  0  0  0  0  0  0  0  0  0  0 -1  0  0  0 -1  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0 -1 -1  0 -1  0  0  0  0 -1  0  0  0  0  0  0
  0  0  0  0 -1 -1  0  0  0 -1 -1 -1  0  0  0  0 -1  0  0 -1  0  0 -1  0  0
  0 -1  0  0  0 -1 -1  0  0  0  0  0  0  0 -1  0  0 -1 -1  0  0  0  0  0  0
  0  0  0  0  0 -1  0  0  0  0  0  0  0  0  0  0  0 -1 -1  0 -1  0  0  0  0
 -1 -1  0  0  0  0  0  0  0  0  0 -1  0 -1  0  0  0  0  0  0  0  0  0 -1  0  0
 -1 -1 -1  0  0 -1  0  0 -1  0  0  0 -1 -1  0  0  0  0  0  0 -1  0 -1  0 -1
  0  0  0 -1  0 -1 -1  0  0  0 -1  0  0 -1  0 -1  0 -1  0  0  0 -1 -1 -1  0
  0  0  0  0 -1  0  0  0  0  0 -1  0  0  0  0  0 -1  0 -1 -1 -1 -1  0  0  0
  0 -1  0  0 -1 -1  0  0  0  0  0  0 -1 -1  0 -1  0 -1 -1  0 -1  0  0  0  0

Porta:
2 24
Minotauro:
7 7

Labirinto marcado:
  0  0  0  0 -1 32 31 30 29 28 27 26 25 24 23 22 -1 -1  0 -1  5  4 -1  2  3
  0 -1 -1  0 -1 31 30 29 28 27 26 25 24 23 22 21 22 23 -1  7 -1  3  2  1 -1
 -1 -1 33 -1 31 30 29 28 27 26 25 24 23 24 -1 20 21 22 -1  6  5  4  3  2  3
 34 33 32 31 30 29 28 27 26 27 -1 -1 22 -1 20 19 20 21 -1  7  6  5  4  3  4
 35 34 33 32 -1 -1 27 26 25 -1 -1 -1 21 20 19 18 -1 22 23 -1  7  6 -1  4  5
 34 -1 32 31 30 -1 -1 25 24 23 22 21 20 19 -1 17 16 -1 -1  9  8  7  6  5  6
 33 32 31 30 29 -1 25 24 23 22 21 20 19 18 17 16 15 -1 -1 10 -1  8  7  6  7
 -1 -1 30 29 28 27 26 25 24 23 22 -1 20 -1 16 15 14 13 12 11 10  9 -1  7  8
 -1 -1 -1 30 29 -1 27 26 -1 24 23 24 -1 -1 17 16 15 14 13 12 -1 10 -1  8 -1
  0  0  0 -1 30 -1 -1 27 26 25 -1 23 22 -1 18 -1 16 -1 14 13 14 -1 -1 -1  0
  0  0  0  0 -1 30 29 28 27 26 -1 22 21 20 19 20 -1  0 -1 -1 -1 -1  0  0  0
  0 -1  0  0 -1 -1 28 27 26 25 24 23 -1 -1 20 -1  0 -1 -1  0 -1  0  0  0  0

Ha caminho.
[ 7,  7] [ 7,  8] [ 7,  9] [ 7, 10] [ 7, 11] [ 7, 12] [ 7, 13]
[ 7, 14] [ 7, 15] [ 7, 16] [ 7, 17] [ 8, 17] [ 8, 18] [ 8, 19]
[ 8, 20] [ 8, 21] [ 8, 22] [ 7, 22] [ 7, 23] [ 7, 24] [ 6, 24]
[ 5, 24] [ 4, 24] [ 3, 24] [ 2, 24]
Comprimento: 25
```



2.3 Casos em que nenhum caminho existe

Em qualquer um dos modos de execução descritos na Seção 2.2, pode ocorrer de não haver um caminho como procurado. Nesse caso, seu programa deve imprimir uma mensagem indicando esse fato. Além disso, seu programa deve ter como saída um diagrama que mostra a posição da porta (`P`), do minotauro (`M`), e as casas acessíveis a partir da porta (`*`), isto é, as casas a partir das quais o minotauro poderia chegar à porta. Segue um exemplo.

```
Nao ha caminho.  Posicoes com acesso a porta:
```



3 Percolação

Seu algoritmo para marcar matrizes como discutido na Seção 1 pode ser usado para simular processos de *percolação*. Agora consideramos que nossas matrizes L representam um meio poroso, pelo qual um líquido pode atravessar ("percolar"). Supomos que há um líquido na borda superior de L e que a gravidade faz com que este líquido desça através das casas desbloqueadas de L. Podem ocorrer duas coisas: ou o líquido chega à borda inferior de L ou o líquido fica preso, sem chegar à borda inferior. No primeiro caso, dizemos que o líquido *percolou* e no segundo caso dizemos que a percolação não ocorreu.

Os diagramas a seguir ilustram dois meios porosos. No primeiro caso, ocorreu percolação e no segundo não ocorreu (as casas marcadas com * foram ocupadas pelo líquido).

```
------------------------------------------------------------
X X * X X X * X * X * * * X * X X X X * * X * * * X X * * *
X * * * X * * * X X   X X * * * * * X * * * X * * * X * * X * X
* * X X   X X   X X X X * * X X X X * X   X * X X * * * * * X
X X               X     X * * * * * X * * X * * * X * * * * * X
X       X X   X   X X X * * * * * X X * X   X * * * * * X X * *
  X   X   X X   X     X X * X * * X X X X X X X * X X X X X
    X     X   X   X   X X * X X * * * X   X   X X X X
        X     X     X X * * X * * X * * X X         X   X
    X X           X * * * * * X * * X X X       X X X X   X X
    X     X       X * X * * X * * X     X   X X     X   X X
------------------------------------------------------------

------------------------------------------------------------
X X * X X X * X * X * * * X * X X X X * * X * * * X X * * *
X * * * X * * * X X   X X * * * * * X     X X * * * * X * * X X X
* * X X   X X   X X X X * * X X X X X   X   X * X X * * * * * X
X X               X     X X       X     X * * * X * X * * X
X       X X   X   X X X         X X   X   X * * * * * X X * *
  X   X   X X   X     X X   X   X X X X X X X X * X X X X X
    X     X   X   X   X X   X X       X   X   X X X X
        X     X     X X     X     X     X X         X   X
    X X           X         X     X X X       X X X X   X X
    X     X       X   X     X     X     X   X X   X X   X X
------------------------------------------------------------
```

Para simular este processo de percolação, basta suas matrizes marcadas serem como no exemplo abaixo.

```
 -1 -1  1 -1 -1 -1  1 -1  1 -1  1  1  1 -1  1 -1 -1 -1 -1  1  1 -1  1  1  1 -1 -1  1  1  1
 -1  3  2  3 -1  3  2 -1 -1  0 -1 -1  2  3  2  3 -1  0  0 -1 -1  3  2  2 -1 11 12 -1 -1 -1
  5  4 -1 -1  0 -1 -1  0 -1 -1 -1 -1  3  4 -1 -1 -1 -1  0 -1  0 -1  3 -1 -1 10 11 12 13 -1
 -1 -1  0  0  0  0  0  0  0 -1  0  0 -1 -1  0  0  0 -1  0  0 -1  5  4  5 -1  9 -1 13 14 -1
 -1  0  0  0 -1 -1  0 -1  0 -1 -1 -1  0  0  0  0 -1 -1  0 -1  0 -1  5  6  7  8 -1 -1 15 16
  0 -1  0 -1  0 -1 -1  0 -1  0  0 -1 -1  0 -1  0 -1 -1 -1 -1 -1 -1 -1 -1  8 -1 -1 -1 -1 -1
  0  0 -1  0  0 -1  0 -1  0 -1  0 -1 -1  0 -1 -1  0  0  0 -1  0 -1  0 -1 -1 -1 -1  0  0  0
  0  0  0  0 -1  0  0 -1  0  0 -1 -1  0  0 -1  0  0 -1  0  0 -1 -1  0  0  0  0 -1  0 -1  0
  0  0 -1 -1  0  0  0  0  0 -1  0  0  0  0 -1  0  0 -1 -1 -1  0  0  0 -1 -1 -1 -1  0 -1 -1
  0  0 -1  0  0 -1  0  0  0 -1  0 -1  0  0 -1  0  0 -1  0  0 -1  0 -1 -1  0 -1 -1  0 -1 -1
```

3.1 Dois novos modos de execução

Seu programa pode também ter os seguintes dois modos de execução.

3.1.1 Instâncias de percolação

Nesta forma, a entrada deve ser, por exemplo, como segue:

```
p
10 20 .55 20142014
```

O caractere `p` indica que queremos simular um processo de percolação. A seguir, são dados o número de linhas e o número de colunas da matriz L, que deve ser gerada aleatoriamente. Em L, cada posição deve estar independentemente desbloqueada com uma probabilidade p, dada pelo usuário (no exemplo acima, $p = 0.55$). Finalmente, o usuário deve fornecer uma semente para o gerador de números aleatórios (20142014 no exemplo acima).

Seu programa deve então imprimir uma representação gráfica do meio poroso, indicando as posições atingidas pelo líquido e uma mensagem dizendo se ocorreu ou não percolação:

```
------------------------------------------------------------
 X * * X * * X X X * * * * * * * X * * * X X X X * * * X * * *
 X X * X * X       X * * * X X X X * X * X X X X X X X X * *
 X X X   X   X X     X X X X X   X * * * * * * * X X X X X X
     X   X         X X   X     X X X * * * X * * X * * * * X
               X   X X     X X       X * * * * X * X * * * *
   X X X X   X         X X   X     X * * * X * * * * * X * X
       X   X         X   X   X X X X * X X X * * * X X X X X
   X   X     X   X X         X X   X X X X X * * X       X X
 X         X   X       X X             X X * * * * X   X
 X   X     X X X       X   X         X X X X X * X       X
------------------------------------------------------------
Ocorreu percolacao.
```

Eis outra possível saída:

```
------------------------------------------------------------
 X * * X * * X X X * * * * * * * X X * * X X X X * * * X X * *
 X X * X * X       X * * * X X X X * X * X X X X X X X X * *
 X X X   X   X X     X X X X X   X * * * * * * * X X X X X X
     X   X         X X   X     X X X * * * X * * X         X
               X X X X X     X X       X * X * * X   X   X
   X X X X   X           X X   X X   X * * * X * X       X   X
       X   X           X   X   X X X X X X X X X     X X X X X
   X   X     X   X X         X X   X X X X X X   X       X X
 X           X X X       X X             X X         X   X
 X   X       X X X       X   X         X X X X X   X       X
------------------------------------------------------------
Nao ocorreu percolacao.
```

3.1.2 Probabilidade crítica de percolação

Neste caso, a entrada deve ter a forma exemplificada abaixo:

```
P
400 500 .59 20142015 2000
```

O caractere `P` indica que queremos simular o processo de percolação várias vezes (2000 vezes no exemplo acima). Como no caso do modo `p` (Seção 3.1.1), são dados o número de linhas e o número de colunas da matriz L (400 e 500 no exemplo acima), que deve ser gerada aleatoriamente, com cada posição independentemente desbloqueada com uma probabilidade p, dada pelo usuário (no exemplo acima, $p = 0.59$). Como antes, o usuário deve fornecer uma semente para o gerador de números aleatórios (20142015 no exemplo acima). Seu programa deve ter como saída a razão entre o número de vezes que ocorreu a percolação e o número total. Essa razão pode ser interpretada como a probabilidade de ocorrer percolação em um meio poroso de "porosidade" p ($p = 0.59$ no exemplo acima). A saída poderia ser como segue:

```
Probabilidade de percolacao em um meio de porosidade 0.59: 0.4155
```

Experimente variar a porosidade p para entender como varia a probabilidade de percolação correspondente. Use seu programa para estimar o que podemos chamar de a "porosidade crítica" p_c.

4 Observações

Este é um enunciado quase completo desse exercício-programa. Inicialmente, concentre sua atenção no modo de execução `b` (Seção 2.2.1) e no requisito descrito na Seção 2.3. Várias partes desse EP serão opcionais, valendo bônus.