Competição de Robôs

Pesquisadores de IA da Universidade de Szeged estão organizando uma competição de programação de robôs. Seu amigo, Hanga, decidiu participar da competição. O objetivo é programar o melhor *Pulibot*, cujo nome é uma homenagem à grande inteligência dos cães da raça húngara, chamada Puli.

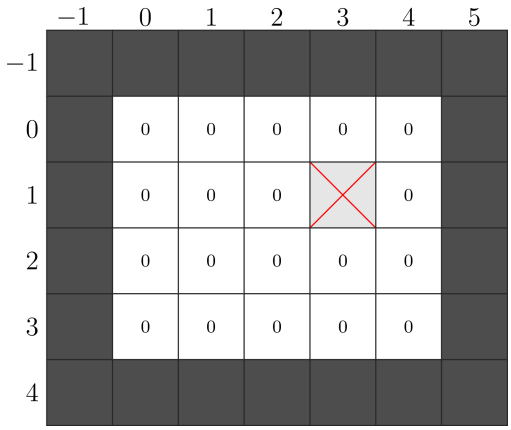
Pulibot será testado num labirinto no formato de uma grade de $(H+2) \times (W+2)$ células. As linhas da grade são numeradas de -1 até H do norte para o sul e as colunas são numeradas de -1 até W do oeste para o leste. Vamos nos referir à célula localizada na linha r e coluna c da grade ($-1 \leq r \leq H$, $-1 \leq c \leq W$) como célula (r, c).

Considere a célula (r, c) tal que $0 \leq r < H$ e $0 \leq c < W$. Há 4 células **adjacentes** à célula (r, c):

- célula $(r, c-1)$ é a célula a **oeste** da célula (r, c);
- célula $(r+1, c)$ é a célula ao **sul** da célula (r, c);
- célula $(r, c+1)$ é a célula a **leste** da célula (r, c);
- célula $(r-1, c)$ é a célula ao **north** da célula (r, c).

A célula (r, c) é chamada de célula **de borda** do labirinto se $r = -1$ ou $r = H$ ou $c = -1$ ou $c = W$. Cada célula que não é uma célula de borda do labirinto é ou uma célula **obstáculo** ou uma célula **vazia**. Adicionalmente, cada célula vazia tem uma **cor**, representada por um inteiro não negativo entre 0 e Z_{MAX}, inclusive. Inicialmente, a cor de cada célula vazia é 0.

Por exemplo, considere um labirinto com $H = 4$ e $W = 5$, contendo uma única célula obstáculo $(1, 3)$:



A única célula obstáculo é denotada por uma cruz. Células de borda do labirinto são sombreadas. O número escrito em cada célula vazia representa sua cor.

Um **caminho** de comprimento ℓ ($\ell > 0$) da célula (r_0, c_0) para a célula (r_ℓ, c_ℓ) é uma sequencia de células *vazias* $(r_0, c_0), (r_1, c_1), \ldots, (r_\ell, c_\ell)$, todas distintas entre si, em que para i ($0 \leq i < \ell$) as células (r_i, c_i) e (r_{i+1}, c_{i+1}) são adjacentes.

Note que um caminho de comprimento ℓ contém exatamente $\ell + 1$ células.

Na competição, os pesquisadores montaram um labirinto no qual existe ao menos um caminho da célula $(0, 0)$ para a célula $(H - 1, W - 1)$. Note que isso implica que as células $(0, 0)$ e $(H - 1, W - 1)$ são garantidamente vazias.

Hanga não sabe quais células do labirinto são vazias e quais são obstáculos.

Sua tarefa é ajudar Hanga a programar Pulibot de tal maneira que ele seja capaz de encontrar algum *menor caminho* (isto é, um caminho de comprimento mínimo) da célula $(0, 0)$ para a $(H - 1, W - 1)$ no labirinto desconhecido montado pelos pesquisadores. A especificação do Pulibot e as regras da competição são descritas abaixo.

Note que a última seção deste enunciado descreve uma ferramenta que você pode usar para visualizar Polibot.

Especificação do Pulibot

Defina o **estado** da célula (r, c) para cada $-1 \leq r \leq H$ e $-1 \leq c \leq W$ como um inteiro tal que:

- se a célula (r, c) é uma célula de borda então seu estado é -2;
- se a célula (r, c) é uma célula obstáculo então seu estado é is -1;
- se a célula (r, c) é uma célula vazia então seu estado é a cor da célula.

O programa Pulibot é executado como uma sequência de passos. Em cada passo, Pulibot reconhece os estados das células vizinhas e então executa uma instrução. A instrução que ele executa é determinada pelos estados reconhecidos. Uma descrição mais precisa segue.

Suponha que no início do passo atual Pulibot está na célula (r, c), que é uma célula vazia. O passo é executado da seguinte maneira:

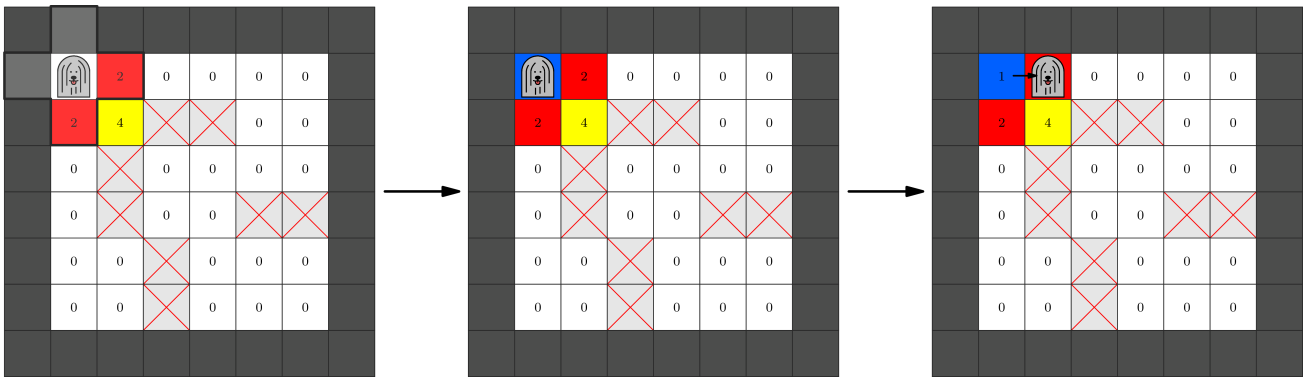
1. Primeiro, Pulibot reconhece o **vetor de estado** atual, isto é, o vetor $S = [S[0], S[1], S[2], S[3], S[4]]$, composto pelos estados da célula (r, c) e de todas as células adjacentes:
    - $S[0]$ é o estado da célula (r, c).
    - $S[1]$ é o estado da célula a oeste.
    - $S[2]$ é o estado da célula ao sul.
    - $S[3]$ é o estado da célula a leste.
    - $S[4]$ é o estado da célula ao norte.
2. Então, Pulibot determina a **instrução** (Z, A) que corresponde ao vetor de estado reconhecido.

3. Finalmente, Pulibot executa essa instrução: ele pinta a célula (r, c) com a cor Z e então executa a ação A, que é uma das seguintes ações:
    - *ficar parado* na célula (r, c);
    - *mover-se* para uma das 4 células adjacentes;
    - *terminar programa*.

Por exemplo, considere o cenário mostrado à esquerda da figura a seguir.

Pulibot está no momento na célula $(0, 0)$, que tem cor 0. Pulibot reconhece o vetor de estado $S = [0, -2, 2, 2, -2]$.

Pulibot pode ter um programa que, ao reconhecer esse vetor, pinta a célula atual com a cor $Z = 1$ e então move-se para o leste, como mostrado no meio e à direita da figura:

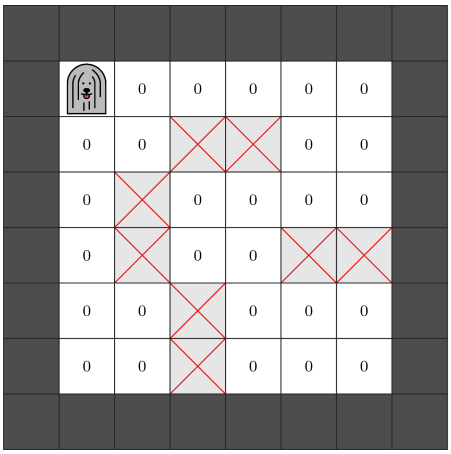


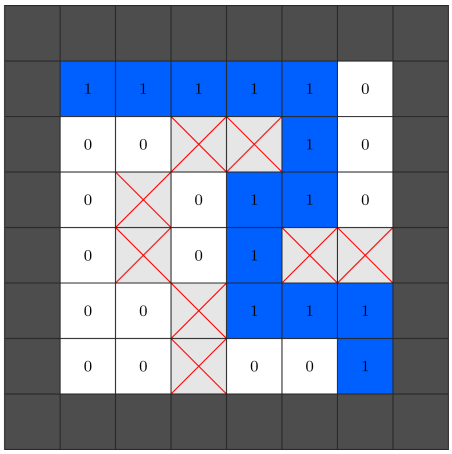
Regras da Competição de Robôs

- No início, Pulibor é colocado na célula $(0, 0)$ e começa a executar seu programa.
- Pulibot não pode mover-se para uma célula que não seja vazia.
- O programa de Pulibot deve terminar após no máximo $500\,000$ passos.
- Após o término do programa de Pulibot, células vazias no labirinto devem estar pintadas de tal maneira que:
    - Existe um caminho mínimo de $(0, 0)$ até $(H - 1, W - 1)$ para o qual cada célula incluída no caminho está pintada com a cor 1.
    - Todas as outras células vazias estão pintadas com a cor 0.
- Pulibot pode terminar seu programa em qualquer célula vazia.

Por exemplo, a seguinte figura mostra um possível labirinto com $H = W = 6$.

A configuração inicial é mostrada à esquerda e uma coloração aceitável das células vazias após o término é mostrado à direita:





Detalhes de implementação

Você deve implementar o seguinte procedimento.

```
void program_pulibot()
```

- Este procedimento deve produzir o programa de Pulibot. O programa deve funcionar corretamente para todos os valores de H e W que satisfaçam as restrições da tarefa.
- Este procedimento é chamado exatamente uma vez para cada caso.

Para produzir o programa de Pulibot este procedimento pode fazer chamadas para o seguinte procedimento:

```
void set_instruction(int[] S, int Z, char A)
```

- S: vetor de comprimento 5 descrevendo um vetor de estado.
- Z: um inteiro não negativo representando uma cor.
- A: um único caractere representando a ação da Pulibot, como a seguir:
  - `H`: ficar parado;
  - `W`: mover-se para a oeste;
  - `S`: mover-se para a sul;
  - `E`: mover-se para a leste;
  - `N`: mover-se para a norte;
  - `T`: terminar o programa.
- A chamada a este procedimento instrui Pulibot para que ao reconhecer o vetor de estado S ele execute a instrução (Z, A).

Chamar este procedimento múltiplas vezes com o mesmo vetor de estado S resultará no veredito `Output isn't correct`.

Não é necessário chamar `set_instruction` com cada possível vetor de estado S. No entanto, se Pulibot em algum momento reconhecer um vetor de estado para o qual uma instrução não foi

descrita, você receberá o veredito `Output isn't correct`.

Após `program_pulibot` terminar, o corretor exemplo invoca o programa de Pulibot com um ou mais labirintos. Essas invocações *não* contam no limite de tempo para sua solução. O corretor exemplo *não* é adaptivo, isto é, o conjunto de labirintos é predefinido em cada caso de teste.

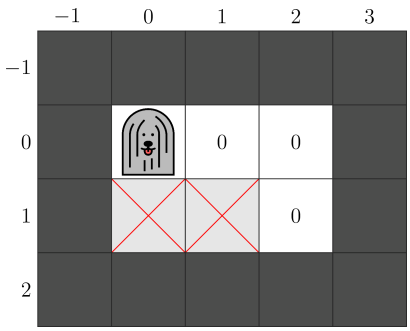
Se Pulibot violar qualquer das Regras da Competição de Robôs antes de terminar o seu programa, você receberá o veredito `Output isn't correct`.

Exemplo

O procedimento `program_pulibot` pode fazer chamadas a `set_instruction` como a seguir:

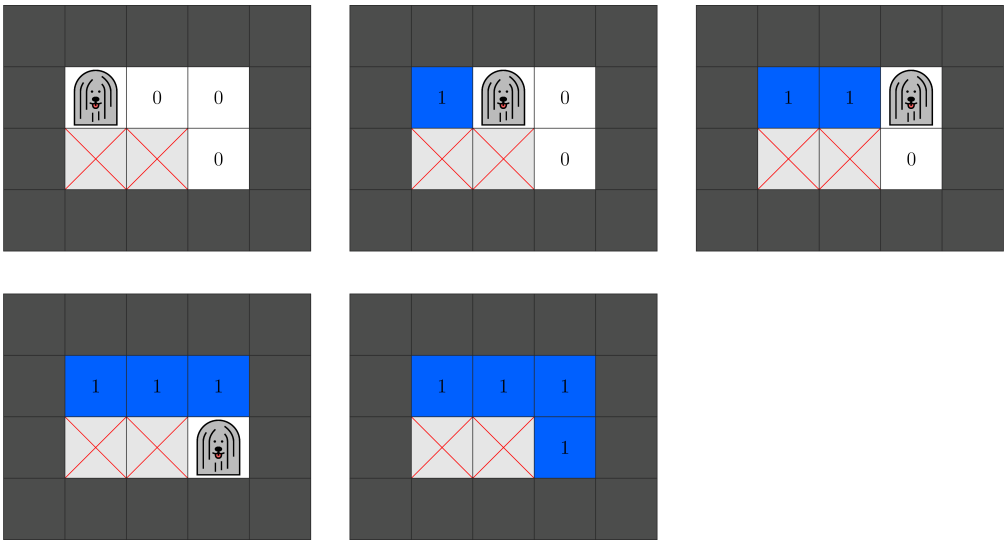
Chamada	Instrução para o vetor de estado S
`set_instruction([0, -2, -1, 0, -2], 1, E)`	Pintar com cor 1 e mover-se para o leste
`set_instruction([0, 1, -1, 0, -2], 1, E)`	Pintar com cor 1 e mover-se para o leste
`set_instruction([0, 1, 0, -2, -2], 1, S)`	Pintar com cor 1 e mover-se para o sul
`set_instruction([0, -1, -2, -2, 1], 1, T)`	Pintar com cor 1 e terminar o programa

Considere um cenário em que $H = 2$ and $W = 3$, e o labirinto é mostrado na seguinte figura.



Para este labirinto o programa de Pulibot executa em quatro passos. Os vetores de estado que Pulibot reconhece e as instruções que ele executa correspondem exatamente às quatro chamadas para `set_instruction` feitas acima, em ordem. A última dessas instruções termina o programa.

A figura seguinte mostra o labirinto antes de cada um dos quatro passos e as cores finais após o término.



Entrentanto, note que este programa de 4 instruções poderia não encontrar um caminho mínimo para outros labirintos válidos. Portanto, se submetido, ele receberá o veredito `Output isn't correct`.

Restrições

$Z_{MAX} = 19$. Portanto, Pulibot pode usar cores de 0 ate 19, inclusive.

Para cada labirinto usado para testar Pulibot:

- $2 \leq H, W \leq 15$
- Existe ao menos um caminho da célula $(0, 0)$ para a célula $(H - 1, W - 1)$.

Subtarefas

1. (6 pontos) Não há células obstáculo no labirinto.
2. (10 pontos) $H = 2$
3. (18 pontos) Existe exatamente um caminho entre cada par de células vazias.
4. (20 pontos) Todos os menores caminhos da célula $(0, 0)$ para a célula $(H - 1, W - 1)$ tem comprimento $H + W - 2$.
5. (46 pontos) Nenhuma restrição adicional

Se, em qualquer dos casos de teste, as chamadas ao procedimento `set_instruction` ou ao programa de Pulibot durante a execução não respeitarem as condições estabelecidas em Detalhes de Implementação, a pontuação de sua solução para a subtarefa será 0.

Em cada subtarefa, você pode obter uma pontuação parcial produzindo uma coloração que é quase correta.

Formalmente:

- A solução de um caso de teste é **completa** se a coloração final das células vazias satisfazem as Regras da Competição de Robôs.
- A solução de um caso de teste é **parcial** se a coloração final é como se segue:
  - Existe um caminho mínimo de $(0, 0)$ até $(H - 1, W - 1)$ para o qual a cor de cada célula incluída no caminho é 1.
  - Não existe célula vazia colorida com 1 no labirinto.
  - Alguma célula vazia no labirinto têm cores diferentes de 0 e 1.

Se sua solução para um caso de teste não é nem completa nem parcial, sua pontuação para esse caso de teste será 0.

Nas subtarefas 1-4, a pontuação para uma solução completa é 100% e a pontuação parcial para um caso de teste é 50% dos pontos para essa subtarefa.

Na subtarefa 5, sua pontuação depende do número de cores usadas pelo programa de Pulibot. Mais precisamente, denote por $Z^\star$ o valor máximo de Z considerando todas as chamadas feitas a `set_instruction`. A pontuação do caso de teste é calculada de acordo com a seguinte tabela:

Condição	Pontuação (completa)	Pontuação (parcial)
$11 \leq Z^\star \leq 19$	$20 + (19 - Z^\star)$	$12 + (19 - Z^\star)$
$Z^\star = 10$	31	23
$Z^\star = 9$	34	26
$Z^\star = 8$	38	29
$Z^\star = 7$	42	32
$Z^\star \leq 6$	46	36

A pontuação para cada subtarefa é o mínimo de pontos para os casos de teste na subtarefa.

Corretor Exemplo

O corretor exemplo lê a entrada no seguinte formato:

- linha 1: H W
- linha $2 + r$ ($0 \leq r < H$): $m[r][0]$ $m[r][1]$ $\ldots$ $m[r][W - 1]$

Aqui, m é um vetor de H vetores de W inteiros, descrevendo as células que não são células de borda do labirinto. $m[r][c] = 0$ se a célula (r, c) é uma célula vazia e $m[r][c] = 1$ se a célula (r, c) é uma célula obstáculo.

O corretor exemplo primeiramente chama `program_pulibot()`. Se o corretor exemplo detecta uma violação de protocolo, ele imprime `Protocol Violation: <MSG>` e termina, onde `<MSG>` é uma das seguintes mensagens de erro:

- `Invalid array`: $-2 \leq S[i] \leq Z_{MAX}$ não é obedecido para algum i ou o comprimento de S não é 5.
- `Invalid color`: $0 \leq Z \leq Z_{MAX}$ não é obedecido.
- `Invalid action`: caractere A não é um entre `H`, `W`, `S`, `E`, `N` ou `T`.
- `Same state array`: `set_instruction` foi chamada com o mesmo vetor de estado S pelo menos duas vezes.

Caso contrário, quando `program_pulibot` completa, o corretor exemplo executa o programa de Pulibot no labirinto descrito na entrada.

O corretor exemplo produz duas saídas.

Primeiramente, o corretor exemplo escreve um relatório das ações de Pulibot no arquivo `robot.bin` no diretório corrente. Este arquivo serve como entrada para a ferramenta de visualização descrita na seção seguinte.

Depois, se o programa de Pulibot não termina com sucesso, o corretor exemplo imprime uma entre as seguintes mensagens de erro:

- `Unexpected state`: Pulibot reconheceu um vetor de estado para o qual `set_instruction` não foi chamado.
- `Invalid move`: a execução de uma ação fez com que Pulibot se movesse para uma célula não vazia.
- `Too many steps`: Pulibot executou $500\,000$ passos sem terminar seu programa.

Caso contrário, seja $e[r][c]$ o estado da célula (r, c) após o programa de Pulibot terminar.

O corretor exemplo imprime H linhas no seguinte formato:

- Lihna $1 + r$ ($0 \leq r < H$): $e[r][0]$ $e[r][1]$ $\ldots$ $e[r][W-1]$

Ferramenta de Visualização

O pacote para esta tarefa contém um arquivo de nome `display.py`. Quando chamado, esse script Python mostra as ações de Pulibot no labirinto descrito pela entrada do corretor exemplo. Para isso, o arquivo binário `robot.bin` deve estar presente no diretório de trabalho.

Para invocar o script, execute o seguinte comando.

```
python3 display.py
```

Uma interface gráfica simples aparece. Os principais recursos são os seguintes:

- Você pode observar o status do labirinto completo. A localização corrente de Pulibot é enfatizada por um retângulo.

- Você pode navegar pelos passos de Pulibot clicando os botões de setas ou pressionando as teclas correspondentes. Você pode também pular para um passo específico.
- O próximo passo do programa de Pulibot é mostrado na parte de baixo. Ele mostra o estado corrente e a instrução que será executada. Após o passo final, ele mostra uma das mensagens de erro do corretor exemplo, ou `Terminated` se o programa termina com sucesso.
- Para cada número que representa uma cor, você pode atribuir uma cor de fundo, bem como um texto. O texto é uma pequena cadeia de caracteres que aparecerá na célula tendo aquela cor. Você pode atribuir cores de fundo e textos em qualquer das seguintes maneiras:
    - Usando uma janela de diálogo clicando o botão `Colors`.
    - Editando o conteúdo do arquivo `colors.txt`.
- Para recarregar `robot.bin`, use o botão `Reload`. Ele é útil se o conteúdo do arquivo `robot.bin` foi alterado.