機器人競賽

塞格德大學的AI研究人員正在舉辦一場機器人編程競賽。 你的朋友Hanga決定參加這場比賽。目標是編寫一個稱為*Pulibot*的終極程式，讚美著著名的匈牙利牧羊犬品種Puli的偉大智慧。

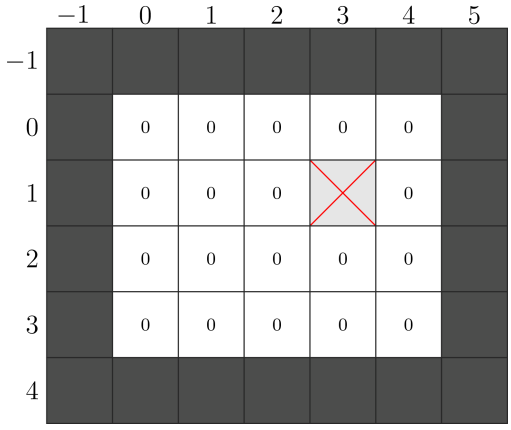
Pulibot將在一個由 $(H+2) \times (W+2)$ 個單元格組成的迷宮中進行測試。 網格的行從北到南編號為 -1 到 H，列從西到東編號為 -1 到 W。 我們將網格中位於第 r 行第 c 列的單元格($-1 \leq r \leq H$, $-1 \leq c \leq W$)稱為單元格 (r, c)。

考慮一個滿足 $0 \leq r < H$ 且 $0 \leq c < W$ 的單元格 (r, c)。單元格 (r, c) 有 4 個**相鄰**單元格:

- 單元格 $(r, c-1)$ 稱為單元格 (r, c) 的**西側**單元格；
- 單元格 $(r+1, c)$ 稱為單元格 (r, c) 的**南側**單元格；
- 單元格 $(r, c+1)$ 稱為單元格 (r, c) 的**東側**單元格；
- 單元格 $(r-1, c)$ 稱為單元格 (r, c) 的**北側**單元格。

如果 $r = -1$ 或 $r = H$ 或 $c = -1$ 或 $c = W$，則單元格 (r, c) 稱為迷宮的**邊界**單元格。 迷宮中不是邊界單元格的每個單元格都是**障礙物**單元格或**空白**單元格。 此外，每個空白單元格都有一個**顏色**，由介於 0 和 Z_{MAX} 之間的非負整數來表示。 最初，每個空白單元格的顏色都是 0。

例如，考慮一個迷宮，其中 $H = 4$ 且 $W = 5$，包含一個障礙物單元格 $(1, 3)$ ：



唯一的障礙物單元格用 X 表示。迷宮的邊界單元格被陰影覆蓋。 每個空白單元格內的數字表示它的顏色。

從單元格 (r_0, c_0) 到單元格 (r_ℓ, c_ℓ) 的長度為 ℓ ($\ell > 0$) 的**路徑**是一個由兩兩不同的*空白*單元格 $(r_0, c_0), (r_1, c_1), \ldots, (r_\ell, c_\ell)$ 組成的序列， 其中對於每個 i ($0 \leq i < \ell$)，單元格 (r_i, c_i) 和 (r_{i+1}, c_{i+1}) 是相鄰的。

注意，長度為 ℓ 的路徑包含恰好 $\ell + 1$ 個單元格。

在比賽中，研究人員設置了一個迷宮，從單元格 $(0, 0)$ 到單元格 $(H-1, W-1)$ 至少存在一條路徑。 注意，這意味著單元格 $(0, 0)$ 和 $(H-1, W-1)$ 保證是空白的。

Hanga不知道迷宮中的哪些單元格是空白的，哪些是障礙物。

你的任務是幫助Hanga編寫Pulibot的程式，使其能夠在研究人員設置的未知迷宮中找到從單元格 $(0, 0)$ 到單元格 $(H-1, W-1)$ 的*最短路徑*（即最小長度的路徑)。 下面描述了Pulibot的規格和比賽的規則。

Pulibot的規則

對於每個 $-1 \leq r \leq H$ 和 $-1 \leq c \leq W$，將單元格 (r, c) 的**狀態**定義為一個整數，使得：

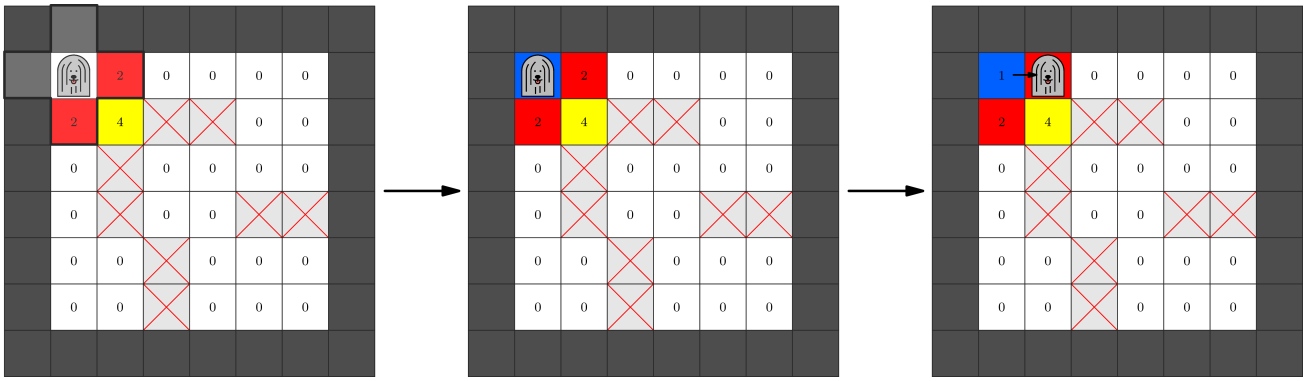
- 如果單元格 (r, c) 是邊界單元格，則其狀態為 -2 ；
- 如果單元格 (r, c) 是障礙物單元格，則其狀態為 -1 ；
- 如果單元格 (r, c) 是空白單元格，則其狀態為該單元格的顏色。

Pulibot的程式被執行為一系列步驟。 在每個步驟中，Pulibot識別附近單元格的狀態，然後執行一個指令。 它執行的指令由識別到的狀態確定。 以下是更詳細的描述。

假設在當前步驟的開始時，Pulibot位於一個空白單元格 (r, c)。 步驟執行如下：

1. 首先，Pulibot識別當前的**狀態數組**，即由單元格 (r, c) 和所有相鄰單元格的狀態組成的數組 $S = [S[0], S[1], S[2], S[3], S[4]]$ ：
    - $S[0]$ 是單元格 (r, c) 的狀態。
    - $S[1]$ 是西側單元格的狀態。
    - $S[2]$ 是南側單元格的狀態。
    - $S[3]$ 是東側單元格的狀態。
    - $S[4]$ 是北側單元格的狀態。
2. 然後，Pulibot確定對應於識別到的狀態數組的**指令** (Z, A)。
3. 最後，Pulibot執行該指令：將單元格 (r, c) 的顏色設置為顏色 Z，然後執行動作 A，該動作是以下動作之一：
    - 在單元格 (r, c) 上*停留*；
    - 移動到4個相鄰單元格之一；
    - *終止程式*。

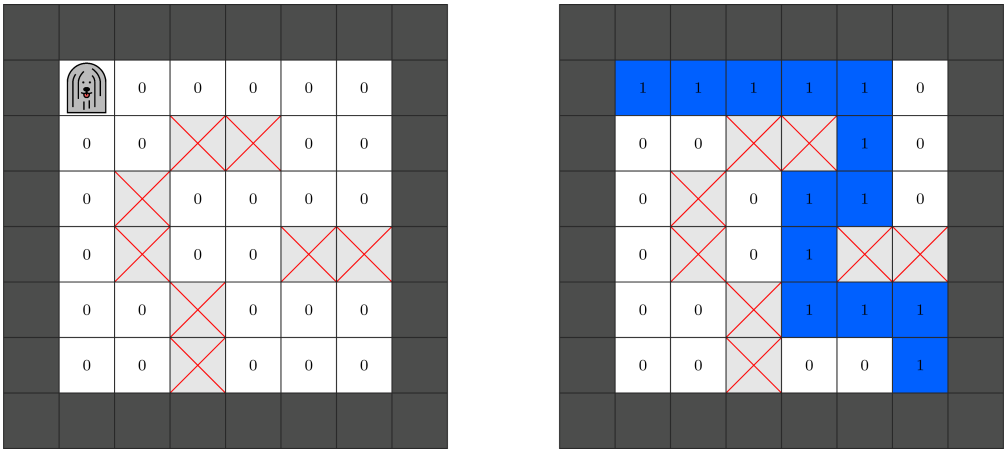
例如，考慮下圖左側顯示的情景。 Pulibot目前位於顏色為 0 的單元格 $(0, 0)$。 Pulibot識別到的狀態數組為 $S = [0, -2, 2, 2, -2]$。 Pulibot可能有一個程式，在識別到這個數組時，將當前單元格的顏色設置為 $Z = 1$，然後向東移動，如圖中間和右側所示：



機器人比賽規則

- 開始時，Pulibot被放置在單元格 $(0, 0)$ 並開始執行其程式。
- Pulibot不允許移動到非空的單元格。
- Pulibot的程式必須在最多 $500,000$ 步後終止。
- 在Pulibot的程式終止後，迷宮中的空單元格應該被著色，使得：
  - 從 $(0, 0)$ 到 $(H - 1, W - 1)$ 存在一條最短路徑，該路徑上每個單元格的顏色都是 1。
  - 其他所有空單元格的顏色都是 0。
- Pulibot可以在任何空單元格終止其程式。

例如，下圖顯示了一個可能的迷宮，其中 $H = W = 6$。左側顯示了起始配置，右側顯示了一個終止後可接受的空單元格的著色：



實現細節

您應該實現以下程序。

```
void program_pulibot()
```

- 這個程序應該生成Pulibot的程式。該程式應該對所有的 H 和 W 的值以及滿足任務限制的任何迷宮都能正確運行。
- 對於每個測試用例，這個程序只會被調用一次。

這個程序可以調用以下程序來生成Pulibot的程式：

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

- S： 長度為 5 的描述狀態數組。
- Z： 表示顏色的非負整數。
- A： 表示Pulibot動作的單個字符，如下所示：
  - `H`：停留；
  - `W`：向西移動；
  - `S`：向南移動；
  - `E`：向東移動；
  - `N`：向北移動；
  - `T`：終止程式。
- 調用此程序指示Pulibot在識別到狀態 S 時執行指令 (Z, A)。

使用相同的狀態數組 S 來多次調用`set_instruction`將導致輸出不正確的判決。

不需要對每個可能的狀態數組 S 都調用`set_instruction`。然而，如果Pulibot稍後識別到一個未設置指令的狀態，則會得到輸出不正確的判決。

在`program_pulibot`完成後，評分機將在一個或多個迷宮上調用Pulibot的程式。 這些調用不計入解決方案的時間限制。 評分機不是自適應的，也就是說，每個測試用例中的迷宮集合是預先定義的。

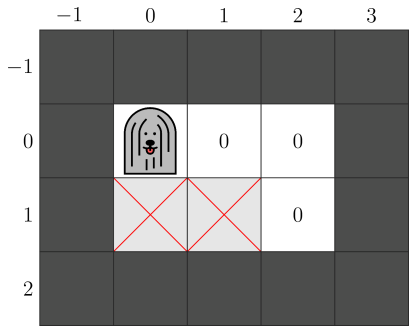
如果Pulibot在終止其程式之前違反了任何機器人比賽規則，則會得到輸出不正確的判決。

樣例

`program_pulibot`程序可以如下調用`set_instruction`：

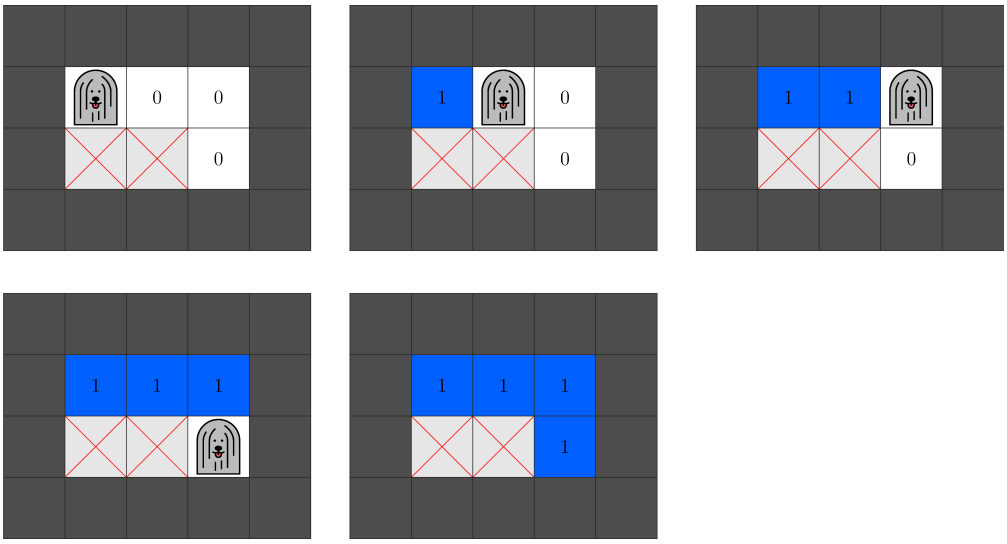
調用	狀態數組 S 的指令
`set_instruction([0, -2, -1, 0, -2], 1, E)`	設置顏色為 1 並向東移動
`set_instruction([0, 1, -1, 0, -2], 1, E)`	設置顏色為 1 並向東移動
`set_instruction([0, 1, 0, -2, -2], 1, S)`	設置顏色為 1 並向南移動
`set_instruction([0, -1, -2, -2, 1], 1, T)`	設置顏色為 1 並終止程式

考慮一個場景，其中 $H = 2$ 且 $W = 3$，迷宮顯示在下圖中。



對於這個特定的迷宮，Pulibot的程式運行了四個步驟。Pulibot識別到的狀態數組和執行的指令與上面的四次`set_instruction`調用完全對應。其中最後一個指令終止了程式。

下圖顯示了每個步驟之前迷宮的狀態以及終止後的最終著色。



但是，請注意，這個包含 4 個指令的程式可能無法在其他有效的迷宮中找到最短路徑。 因此，如果提交，它將收到輸出不正確的判決。

約束條件

$Z_{MAX} = 19$。因此，Pulibot可以使用從0到19的顏色。

對於用於測試Pulibot的每個迷宮：

- $2 \leq H, W \leq 15$
- 從單元格 $(0, 0)$ 到單元格 $(H - 1, W - 1)$ 至少存在一條路徑。

子任務

1. (6分) 迷宮中沒有障礙單元格。
2. (10分) $H = 2$
3. (18分) 每對空單元格之間恰好存在一條路徑。
4. (20分) 從單元格 $(0, 0)$ 到單元格 $(H - 1, W - 1)$ 的每條最短路徑的長度爲 $H + W - 2$。
5. (46分) 沒有額外的約束條件。

如果在任何測試用例中，對set_instruction過程的調用或Pulibot的程序在執行過程中不符合實現細節中描述的約束條件，則該子任務的解決方案得分爲 0。

在每個子任務中，您可以通過生成幾乎正確的着色來獲得部分分數。

具體而言：

- 如果空單元格的最終着色滿足機器人競賽規則，則測試用例的解決方案是**完整的**。
- 如果最終着色如下所示，則測試用例的解決方案是**部分的**：
  - 存在一條從 $(0, 0)$ 到 $(H - 1, W - 1)$ 的最短路徑，路徑中每個單元格的顏色爲 1。
  - 網格中沒有其他顏色爲 1 的空單元格。
  - 網格中的某個空單元格的顏色不是 0 和 1。

如果您對測試用例的解決方案既不完整也不部分，則相應測試用例的得分爲 0。

在子任務1-4中，全解的得分為100%，而部份解的得分是該子任務分數的50%。

在子任務5中，您的得分取決於Pulibot程序中使用的顏色數量。 更具體地說，設 $Z^\star$ 是對set_instruction的所有調用中 Z 的最大值。 根據以下表格計算測試用例的得分：

條件	得分（完整）	得分（部分）
$11 \leq Z^\star \leq 19$	$20 + (19 - Z^\star)$	$12 + (19 - Z^\star)$
$Z^\star = 10$	31	23
$Z^\star = 9$	34	26
$Z^\star = 8$	38	29
$Z^\star = 7$	42	32
$Z^\star \leq 6$	46	36

每個子任務的得分是該子任務中測試用例的最低得分。

樣例評分器

樣例評分器按以下格式讀取輸入：

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

這裏， m 是一個包含 H 個 W 個整數數組的數組，描述了迷宮的非邊界單元格。 如果單元格 (r, c) 是空單元格，則 $m[r][c] = 0$，如果單元格 (r, c) 是障礙單元格，則 $m[r][c] = 1$。

樣例評分器首先調用program_pulibot()。 如果樣例評分器檢測到協議違規，樣例評分器會打印Protocol Violation: <MSG>並終止，其中<MSG>是以下錯誤消息之一：

- `Invalid array`：對於某些 i，$-2 \leq S[i] \leq Z_{MAX}$ 未滿足，或者 S 的長度不是 5。
- `Invalid color`：$0 \leq Z \leq Z_{MAX}$ 未滿足。
- `Invalid action`：字符 A 不是`H`、`W`、`S`、`E`、`N`或`T`。
- `Same state array`：`set_instruction`至少兩次使用相同的數組 `$S`。

否則，當`program_pulibot`完成時，樣例評分器在輸入描述的迷宮中執行Pulibot的程序。

樣例評分器生成兩個輸出。

首先，樣例評分器將Pulibot的動作日誌寫入工作目錄中的文件`robot.bin`。 該文件作爲下一節中描述的可視化工具的輸入。

其次，如果Pulibot的程序沒有成功終止，樣例評分器會打印以下錯誤消息之一：

- `Unexpected state`：Pulibot識別到一個`set_instruction`未使用的狀態數組。
- `Invalid move`：執行一個動作導致Pulibot移動到一個非空單元格。
- `Too many steps`：Pulibot執行了$500\,000$步而沒有終止其程序。

否則，設$e[r][c]$是Pulibot程序終止後單元格(r, c)的狀態。 樣例評分器以以下格式打印H行：

- 第`1 + r`行（$0 \leq r < H$）：$e[r][0]$ $e[r][1]$ ... $e[r][W-1]$

顯示工具

此任務的附件包含一個名為 `display.py` 的文件。 當調用時，這個 Python 腳本會在由樣本評分器的輸入描述的迷宮中顯示 Pulibot 的動作。 為此，二進制文件 `robot.bin` 必須存在於工作目錄中。

要調用腳本，執行以下命令。

```
python3 display.py
```

一個簡單的圖形界面會出現。主要功能如下：

- 您可以觀察整個迷宮的狀態。Pulibot 的當前位置會用一個矩形突出顯示。
- 您可以通過點擊箭頭按鈕或按下它們的熱鍵來瀏覽 Pulibot 的步驟。您還可以跳轉到特定的步驟。
- Pulibot 程序中即將到來的步驟顯示在底部。 它顯示當前狀態數組和它將執行的指令。 在最後一步之後，它會顯示評分器的錯誤消息之一，或者如果程序成功終止則顯示 `Terminated`。
- 對於表示顏色的每個數字，您可以分配一個背景顏色以及一個顯示文字。該顯示文字應為一個短字符串，並應該出現在每個具有相同顏色的單元格中。您可以以以下任一方式分配背景顏色和顯示文本：
  - 在點擊 `Colors` 按鈕後，在對話窗口中設置它們。
  - 編輯 `colors.txt` 文件的內容。
- 要重新加載 `robot.bin`，請使用 `Reload` 按鈕。如果 `robot.bin` 的內容已更改，這很有用。