机器人比赛 （robot）

塞格德大学的人工智能研究人员正在举办一场机器人编程竞赛。 你的朋友 Hanga 决定参加比赛。由于著名的匈牙利牧羊犬品种 Puli 非常聪明，所以该比赛的目标定为编程实现顶级的 Pulibot。

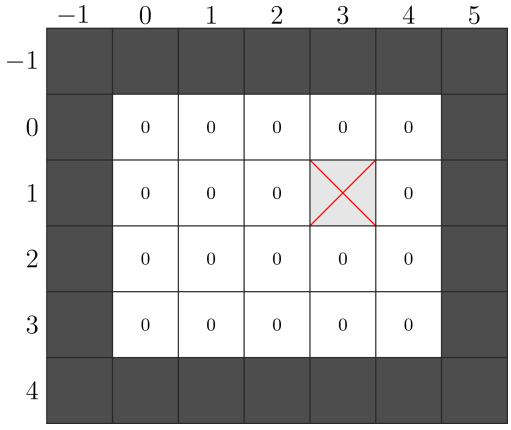
Pulibot 将在由 $(H+2) \times (W+2)$ 网格组成的迷宫中进行测试。 网格的行从北到南编号为 -1 到 H，网格的列从西到东编号为 -1 到 W。 我们将位于网格的第 r 行和第 c 列的单元格（$-1 \leq r \leq H$，$-1 \leq c \leq W$）称为单元格 (r, c)。

考虑一个单元格 (r, c) （$0 \leq r < H$，$0 \leq c < W$），和它**相邻**的有 4 个单元格。

- 单元格 $(r, c-1)$ 被称为单元格 (r, c) 的**西邻**；
- 单元格 $(r+1, c)$ 被称为单元格 (r, c) 的**南邻**；
- 单元格 $(r, c+1)$ 被称为单元格 (r, c) 的**东邻**；
- 单元格 $(r-1, c)$ 被称为单元格 (r, c) 的**北邻**。

如果 $r = -1$ 或 $r = H$ 或 $c = -1$ 或 $c = W$ 成立，则单元格 (r, c) 称为迷宫的**边界**。每个不是迷宫边界的单元格要么是**障碍**，要么是**空的**。 此外，每个空单元格都有一个**颜色**，由 0 和 Z_{MAX} 之间的非负整数表示，包括 0 和 Z_{MAX}。最初，每个空单元格的颜色为 0。

例如，考虑一个迷宫，$H = 4$， $W = 5$， 包含一个障碍单元格 $(1, 3)$。



唯一的障碍单元格用 $\times$ 表示。迷宫的边界单元格被阴影覆盖。 每个空单元格中的数字表示它的颜色。

从单元格 (r_0, c_0) 到单元格 (r_ℓ, c_ℓ) 的长度为 ℓ （$\ell > 0$） 的**路径** 是一个**空**单元格序列 $(r_0, c_0), (r_1, c_1), \ldots, (r_\ell, c_\ell)$，序列中的空单元格两两不同。其中对于每个 i （$0 \leq i < \ell$），单元格 (r_i, c_i) 和 (r_{i+1}, c_{i+1}) 是相邻的。

注意长度为 ℓ 的路径正好包含 $\ell + 1$ 个单元格。

在比赛中，研究人员设置了一个迷宫，其中至少有一条从单元格 $(0,0)$ 到单元格 $(H-1,W-1)$ 的路径。注意，这意味着单元格 $(0,0)$ 和 $(H-1,W-1)$ 保证为空。

Hanga 不知道迷宫中哪些单元格是空的，哪些单元格是障碍。

你的任务是帮助 Hanga 对 Pulibot 进行编程，使其能够在研究人员设置的未知迷宫中找到从单元格 $(0,0)$ 到单元格 $(H-1,W-1)$ 的**最短路径**（即长度最小的路径）。 Pulibot 的说明和比赛规则如下所述。

注意，在题面的最后一部分描述了一个显示工具，该工具可以用于可视化 Pulibot。

Pulibot 说明

对每个单元格 (r,c)（$-1 \le r \le H$，$-1 \le c \le W$），其**状态**定义为一个整数，具体如下：

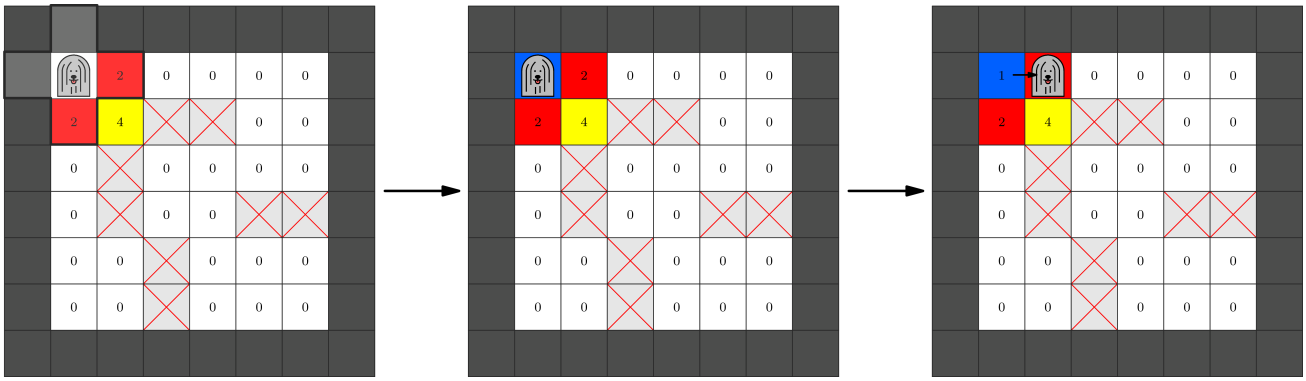
- 如果单元格 (r,c) 是边界，则其状态为-2；
- 如果单元格 (r,c) 是障碍，则其状态为-1；
- 如果单元格 (r,c) 是空的，那么它的状态就是单元格的颜色。

Pulibot 的程序是按一系列步骤执行的。在每一步中，Pulibot 都会识别附近单元格的状态，然后执行一条指令。它执行的指令由识别的状态决定。以下是更准确的描述。

假设在当前步骤开始时，Pulibot位于单元格 (r,c)，这是一个空单元格。该步骤执行如下：

1. 首先，Pulibot 识别当前**状态数组**，即数组 $S = [S[0], S[1], S[2], S[3], S[4]]$， 它包含单元格 (r,c) 及其所有相邻单元格的状态：
    - $S[0]$ 表示单元格 (r,c) 的状态。
    - $S[1]$ 表示西邻的状态。
    - $S[2]$ 表示南邻的状态。
    - $S[3]$ 表示东邻的状态。
    - $S[4]$ 表示北邻的状态。
2. 然后，Pulibot 确定与所识别的状态数组相对应的**指令** (Z, A)。
3. 最后，Pulibot 执行这条指令：它将单元格 (r,c) 的颜色设置为 Z，然后它执行动作 A, A 是以下动作之一：
    - **停留** 在单元格 (r,c)；
    - **移动** 到 4 个邻居之一；
    - **终止程序**。

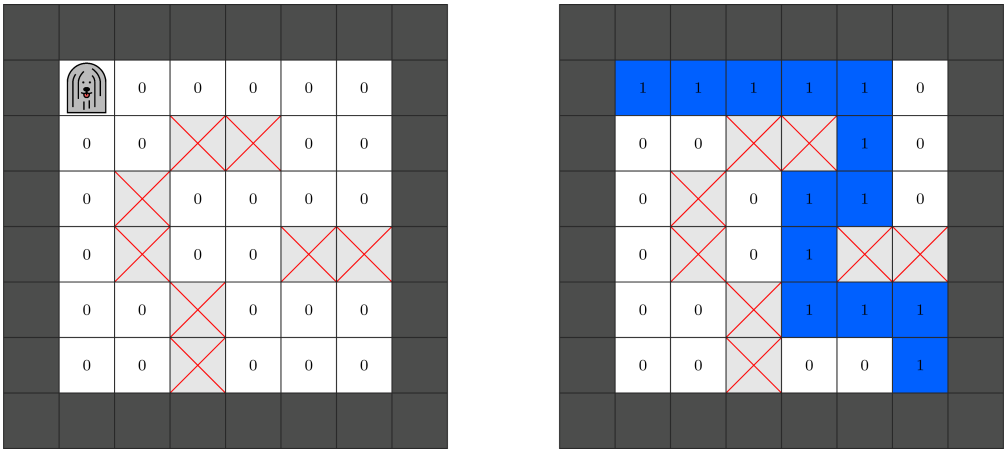
例如，考虑下图左侧显示的场景。Pulibot 当前位于单元格$(0,0)$，颜色为0。 Pulibot 识别出状态数组 $S = [0, -2, 2, 2, -2]$。Pulibot 可能有一个程序，该程序根据所识别的数组，将当前单元格的颜色设置为 $Z = 1$，然后向东移动，如图的中间和右侧所示：



机器人比赛规则

- 在开始时，Pulibot 被放置在单元格 $(0, 0)$ 并开始执行其程序。
- 不允许 Pulibot 移动到非空单元格。
- Pulibot 的程序必须在最多 $500\,000$ 步后终止。
- 在 Pulibot 的程序终止后，迷宫中的空单元格的着色满足以下要求：
  - 存在从 $(0, 0)$ 到 $(H - 1, W - 1)$ 的最短路径，路径中包括的每个单元格的颜色为 1。
  - 所有其他空单元格的颜色为 0。
- Pulibot 可以在任何空单元格终止其程序。

例如，下图显示了一个可能的迷宫，其中$H = W = 6$。左侧显示了初始配置，右侧显示了程序终止后空单元格的一种可以接受的着色：



实现细节

你要实现以下函数：

```
void program_pulibot()
```

- 这个函数应该产生 Pulibot 的程序。对于所有 H 和 W 的取值以及满足题目约束条件的任何迷宫，该程序应该都能正确工作。
- 对于每个测试用例，此函数只调用一次。

此函数可以调用以下函数来生成 Pulibot 的程序：

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

- S: 长度为 5 的数组，用来描述状态数组
- Z: 表示颜色的非负整数
- A: 表示 Pulibot 动作的单个字符，具体如下:
  - `H`: 停留;
  - `W`: 移动到西邻;
  - `S`: 移动到南邻;
  - `E`: 移动到东邻;
  - `N`: 移动到北邻;
  - `T`: 终止程序。
- 调用此函数指示 Pulibot 在识别状态数组 S 时应执行指令 (Z, A)。

用相同的状态数组 S 多次调用该函数将导致 `Output isn't correct` 的判定结果。

不需要对每个可能的状态数组 S 调用 `set_instruction`。 但是，如果 Pulibot 后来识别出未设置指令的状态数组，你将得到 `Output isn't correct` 的判定结果。

`program_pulibot` 完成后，评测程序会在一个或多个迷宫上调用 Pulibot 的程序。 这些调用**不**计入解决方案的时间限制。 评测程序**不**是自适应的，也就是说，每个测试用例的迷宫集合都是预先确定的。

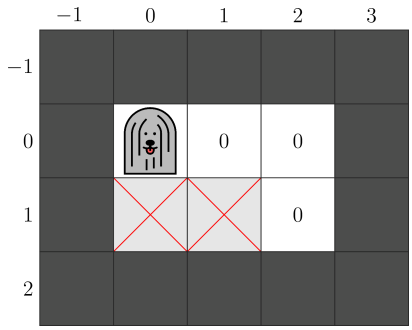
如果 Pulibot 在终止程序之前违反了任何机器人比赛规则，你将得到 `Output isn't correct` 的判定结果。

例子

函数 `program_pulibot` 可以调用 `set_instruction` 如下：

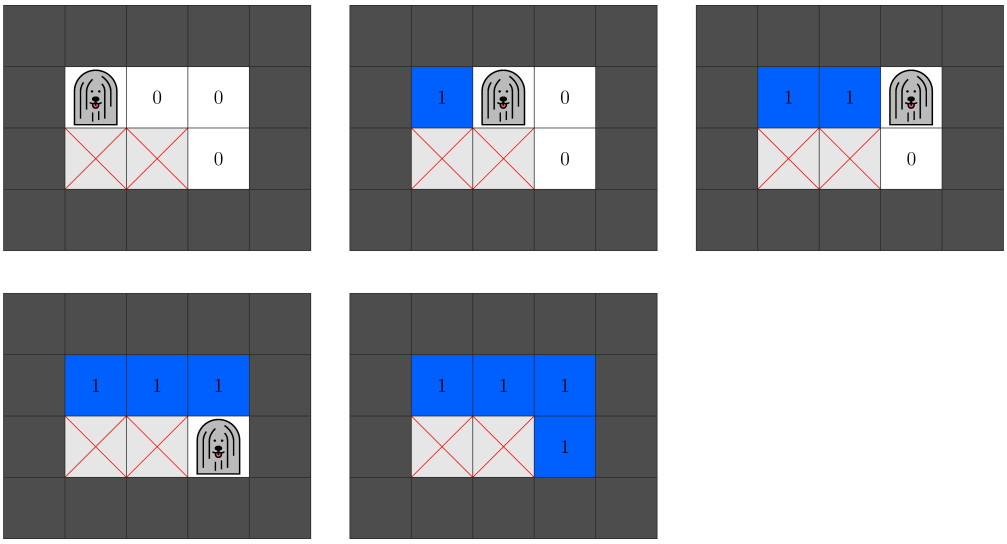
调用	对应状态数组 S 的指令
`set_instruction([0, -2, -1, 0, -2], 1, E)`	着色 1 并且东移
`set_instruction([0, 1, -1, 0, -2], 1, E)`	着色 1 并且东移
`set_instruction([0, 1, 0, -2, -2], 1, S)`	着色 1 并且南移
`set_instruction([0, -1, -2, -2, 1], 1, T)`	着色 1 并且终止程序

考虑一个场景，$H = 2$， $W = 3$，迷宫如下图所示。



对于这个特定的迷宫，Pulibot 的程序分四个步骤运行。 Pulibot 识别的状态数组和它执行的指令正好依次对应上述对“set_instruction”的四次调用。 这些指令的最后一条指令终止程序。

下图展示了四个步骤每一步之前的迷宫以及终止后的最终颜色。



但是，注意这个由 4 条指令构成的程序有可能在其他合法的迷宫中找不到最短路径。 所以，如果这个程序被提交，它会收到 `Output isn't correct` 的判定结果。

约束条件

$Z_{\mathrm{MAX}} = 19$。因此，Pulibot 可以使用 0 到 19 的颜色，包含 0 和 19。

对于每个用来测试Pulibot的迷宫：

- $2 \leq H, W \leq 15$
- 至少有一条从单元格 $(0, 0)$ 到 $(H - 1, W - 1)$ 的路径。

子任务

1. （6 分）迷宫中没有障碍单元格。
2. （10 分）$H = 2$
3. （18 分）任意两个空单元格之间恰好有一条路径。
4. （20 分）从单元格 $(0, 0)$ 到 $(H - 1, W - 1)$ 的最短路径的长度为 $H + W - 2$。
5. （46 分）无额外约束条件。

如果在任何测试用例中，对函数 `set_instruction` 的调用或 Pulibot 程序的执行不符合“实现细节”中所描述的限制条件，则该子任务的解决方案得分将为 0。

在每个子任务中，你可以通过生成几乎正确的着色来获得部分分数。

形式化地说：

- 如果空单元格的最终颜色满足机器人竞赛规则，则测试用例的解决方案是**完整**的。
- 如果最终着色如下所示，则测试用例的解决方案是**部分**的：
  - 存在一条从 $(0, 0)$ 到 $(H - 1, W - 1)$ 的最短路径，路径中包含的每个单元格的颜色为 1。
  - 网格中没有其他颜色为 1 的空单元格。
  - 网格中的某些空单元格的颜色既不是 0 也不是 1。

如果你对某个测试用例的解决方案既不完整也不部分，则该测试用例的得分将为 0。

在子任务 1-4 中，对每个测试用例来说，完整解决方案的计分为该子任务分数的 100%，部分解决方案的计分为该子任务分数的 50%。

在子任务 5 中，你的分数取决于 Pulibot 程序中所使用颜色的数量。 更准确地说，用 $Z^\star$ 表示对 `set_instruction` 进行的所有调用中 Z 的最大值。 测试用例上的得分按下表计算：

条件	分数 (完整)	分数 (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

每个子任务的得分是该子任务中所有测试用例上计分的最小值。

评测程序示例

评测程序示例按照以下格式读取输入：

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

其中，m 是一个 H 行 W 列的二维整数数组，描述迷宫中非边界单元格。 如果单元格 (r, c) 是空的，$m[r][c] = 0$；如果单元格 (r, c) 是障碍， $m[r][c] = 1$。

评测程序示例首先调用 `program_pulibot()`。如果评测程序示例检测到违反规则的行为，则会打印 `Protocol Violation: <MSG>` 并终止，其中 `<MSG>` 是以下错误消息之一：

- `Invalid array`：$-2 \leq S[i] \leq Z_{\mathrm{MAX}}$ 对某些 i 不成立或者 S 的长度不是 5。
- `Invalid color`：$0 \leq Z \leq Z_{\mathrm{MAX}}$ 不成立。
- `Invalid action`：字符 A 不是 `H`, `W`, `S`, `E`, `N` 或 `T`。
- `Same state array`：用相同的 S 调用 `set_instruction` 两次或以上。

否则，当 `program_pulibot` 完成时，评测程序示例将在输入所描述的迷宫中执行 Pulibot 的程序。

评测程序示例产生两个输出。首先，评测程序示例将 Pulibot 动作记录写入工作目录中的文件 `robot.bin`。该文件用作下一节中描述的可视化工具的输入。

其次，如果 Pulibot 的程序未成功终止，评测程序示例将打印以下错误消息之一：

- `Unexpected state`：Pulibot 识别出一个无法调用“set_instruction”的状态数组。
- `Invalid move`：执行一个动作，导致 Pulibot 移动到一个非空单元格。
- `Too many steps`：Pulibot 执行了 $500\,000$ 步没有终止程序。

否则，令 $e[r][c]$ 为 Pulibot 程序终止后单元格 (r, c) 的状态。评测程序示例按以下格式打印 H 行：

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

显示工具

此任务的附件包含有一个名为 `display.py` 的文件。调用时，此 Python 脚本会显示 Pulibot 在由评测程序示例的输入所描述的迷宫中的操作。为此，工作目录中要有二进制文件 `robot.bin`。

要调用该脚本，请执行以下命令。

```
python3 display.py
```

一个简单的图形界面将会出现，主要特性如下：

- 你可以观察整个迷宫的状态。Pulibot 的当前位置以矩形突出显示。
- 你可以通过单击箭头按钮或按热键来浏览 Pulibot 的步骤。你还可以跳转到特定步骤。
- Pulibot 程序中即将进行的步骤显示在底部。它显示当前状态数组及将要执行的指令。在最后一步之后，它或者会显示评测程序的错误消息之一，或者在程序成功终止时显示 `Terminated`。
- 对于代表颜色的每个数字，你可以指定视觉背景颜色以及显示的文本。显示的文本是一个短字符串，应出现在每个具有那个颜色的单元格。你可以通过以下任一方式指定背景颜色和显示的文本：
  - 单击 `Colors` 按钮后在对话框窗口中设置它们。
  - 编辑 `colors.txt` 文件的内容。
- 要重新加载 `robot.bin`，请使用 `Reload` 按钮。这可以用来处理 `robot.bin` 的内容发生更改的情况。