Robot Contest

Исследователи ИИ в университете Сегеда проводят соревнование роботов. Ваша подруга, Ханга, решила принять участие в соревновании. Цель соревнования — запрограммировать лютейшего *Пулибота*, который назван так в честь знаменитой очень умной венгерской породы собак Пули.

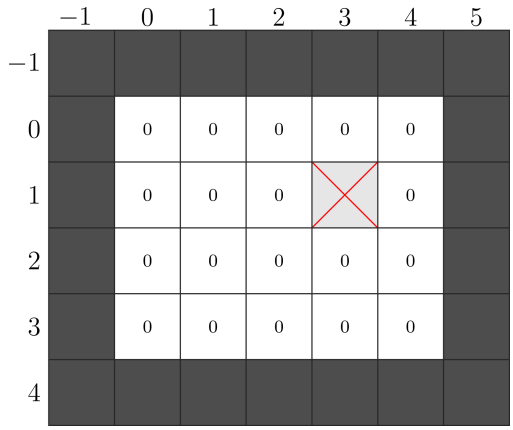
Пулибот будет протестирован на клетчатом лабиринте, состоящем из $(H+2) \times (W+2)$ клеток. Строки лабиринта пронумерованы от -1 до H с севера на юг, столбцы пронумерованы от -1 до W с запада на восток. Будем обозначать клетку в строке r, столбце c лабиринта ($-1 \leq r \leq H$, $-1 \leq c \leq W$) как (r, c).

Рассмотрим клетку (r, c), такую что $0 \leq r < H$ и $0 \leq c < W$. Есть 4 клетки, **соседние** с клеткой (r, c):

- клетка $(r, c-1)$ — это клетка на **запад** от клетки (r, c);
- клетка $(r+1, c)$ — это клетка на **юг** от клетки (r, c);
- клетка $(r, c+1)$ — это клетка на **восток** от клетки (r, c);
- клетка $(r-1, c)$ — это клетка на **север** от клетки (r, c).

Клетка (r, c) называется **граничной**, если $r = -1$, или $r = H$, или $c = -1$, или $c = W$. Каждая клетка, которая не граничная, может быть либо **пустой**, либо содержать **препятствие**. Кроме того, каждая пустая клетка имеет **цвет**, который задается неотрицательным целым числом от 0 до Z_{MAX}, включительно. Исходно цвет каждой пустой клетки равен 0.

Например, рассмотрим лабиринт с $H = 4$ и $W = 5$, содержащий единственное препятствие в клетке $(1, 3)$:



Единственная клетка, содержащая препятствие, обозначена крестиком. Граничные клетки обозначены темным цветом. Номера, написанные в клетках, обозначают их цвет.

Путь длины ℓ ($\ell > 0$) из клетки (r_0, c_0) в клетку (r_ℓ, c_ℓ) — это последовательность попарно различных *пустых* клеток $(r_0, c_0), (r_1, c_1), \ldots, (r_\ell, c_\ell)$, в которой для каждого i ($0 \le i < \ell$) клетки (r_i, c_i) и (r_{i+1}, c_{i+1}) являются соседними.

Обратите внимание, что путь длины ℓ содержит ровно $\ell + 1$ клетку.

Во время соревнования жюри использует лабиринт, который содержит хотя бы один путь из клетки $(0, 0)$ в клетку $(H - 1, W - 1)$. Обратите внимание, что это означает, что гарантируется, что клетки $(0, 0)$ и $(H - 1, W - 1)$ пустые.

Ханга не знает, какие клетки лабиринта пустые, а какие содержат препятствия.

Ваша задача — помочь Ханге запрограммировать Пулибота, чтобы он мог находить *кратчайший путь* (то есть путь минимальной длины) из клетки $(0, 0)$ в клетку $(H - 1, W - 1)$ в неизвестном лабиринте, который будет использован для соревнований. Спецификация Пулибота и правила соревнований приведены в следующих разделах.

Обратите внимание, что в последнем разделе описывается визуализатор для этой задачи.

Pulibot's Specification

Определим **состояние** клетки (r, c) для каждого $-1 \le r \le H$ и $-1 \le c \le W$ как целое число:

- если клетка (r, c) является граничной, то ее состояние равно -2;
- если клетка (r, c) содержит препятствие, то ее состояние равно -1;
- если клетка (r, c) является пустой, то ее состояние равно ее цвету.

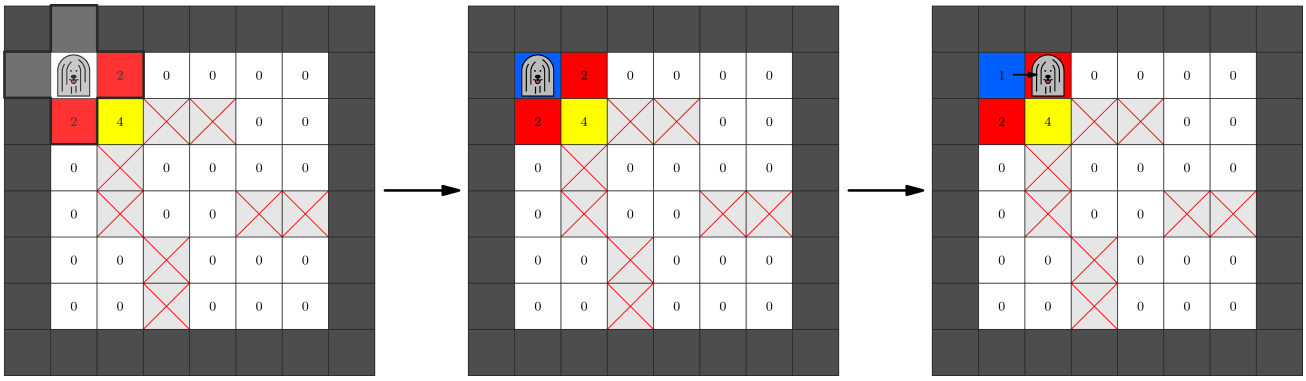
Программа Пулибота исполняется как последовательность шагов. На каждом шаге Пулибот смотрит на цвета клетки, где он находится, и соседних клеток и выполняет инструкцию. Инструкция, которую он выполняет, зависит от того, в каком состоянии находятся клетки, на которые он посмотрел. А именно:

Пусть в начале текущего шага Пулибот находится в пустой клетке (r, c). Шаг выполняется следующим образом:

1. Сначала Пулибот формирует **массив состояний**, а именно, массив $S = [S[0], S[1], S[2], S[3], S[4]]$, включающий состояние клетки (r, c) и всех соседних клеток:
    - $S[0]$ — это состояние клетки (r, c).
    - $S[1]$ — это состояние клетки на запад.
    - $S[2]$ — это состояние клетки на юг.
    - $S[3]$ — это состояние клетки на восток.
    - $S[4]$ — это состояние клетки на север.

2. Затем Пулибот определяет **инструкцию** (Z, A), которая соответствует сформированному массиву состояний.
3. Наконец, Пулибот исполняет инструкцию: он делает цвет клетки (r, c) равным Z, а затем выполняет действие A, одно из следующих:
    - *остаться* в клетке (r, c);
    - *переместиться* в одну из 4 соседних клеток;
    - *завершить работу программы*.

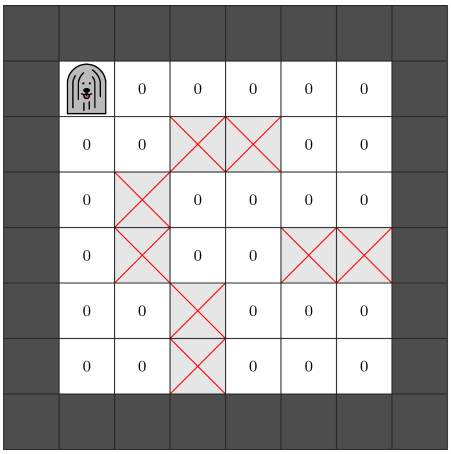
Например, рассмотрим сценарий, изображенный слева на следующем рисунке. Пулибот находится в клетке $(0, 0)$, цвет которой 0. Пулибот формирует массив состояний $S = [0, -2, 2, 2, -2]$. Пусть в программе Пулибота для данного массива состояний указана инструкция поменять цвет текущей клетки на $Z = 1$ и переместиться на восток, это происходит, как показано посередине и справа на рисунке:

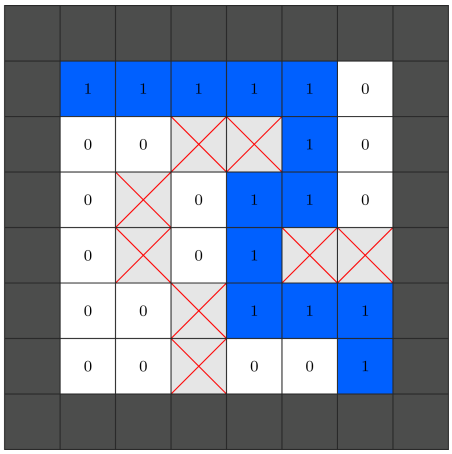


Robot Contest Rules

- В начале Пулибот находится в клетке $(0, 0)$ и начинает исполнять программу.
- Пулибот не может перемещаться в клетку, которая не является пустой.
- Программа Пулибота должна завершиться не более чем за $500\,000$ шагов.
- После завершения программы Пулибота пустые клетки лабиринта должны иметь такие цвета, чтобы выполнялись следующие условия:
    - Существует кратчайший путь из клетки $(0, 0)$ в клетку $(H - 1, W - 1)$, для которого цвет каждой клетки на этом пути равен 1.
    - Цвет каждой другой пустой клетки равен 0.
- Пулибот может завершить выполнение программы на любой пустой клетке.

Например, следующий рисунок показывает возможный лабиринт с $H = W = 6$. Стартовая конфигурация показана слева, а одна из возможных раскрасок пустых клеток после завершения программы показана справа:





Implementation Details

Вам следует реализовать следующую функцию.

```
void program_pulibot()
```

- Эта функция должна создать программу для Пулибота. Программа должна корректно работать для всех значений H и W, и всех лабиринтов, которые подходят под условие задачи.
- Функция будет вызывана ровно один раз для каждого теста.

Функция может сделать вызовы следующей функции, чтобы создать программу для Пулибота:

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

- S: массив длины 5, задающий массив состояний.
- Z: неотрицательное целое число, задающее цвет.
- A: символ, задающий действие Пулибота в следующем формате:
  - `H`: остаться на текущей клетке;
  - `W`: переместиться на запад;
  - `S`: переместиться на юг;
  - `E`: переместиться на восток;
  - `N`: переместиться на север;
  - `T`: завершить программу.
- Вызов этой функции добавляет в программу Пулибота , что в случае массива состояний S он должен выполнить действия (Z, A).

Если вы сделаете несколько вызовов этой функции с одним и тем же массивом состояний S, проверка завершится с результатом `Output isn't correct`.

Вы не должны вызывать функцию `set_instruction` с каждым возможным массивом состояний S. Однако, если Пулибот в какой-то момент сформирует массив состояний, для которого у него нет инструкции, результат проверки будет `Output isn't correct`.

После завершения функции `program_pulibot` грейдер запустит Пулибота с выведенной программой на одном или нескольких лабиринтах. Эти запуски *не считаются* при определении времени работы вашего решения. Грейдер *не адаптивный*, то есть набор лабиринтов заранее определен для каждого теста.

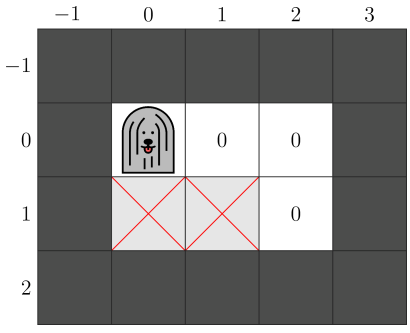
Если Пулибот нарушит правила, описанные в разделе Robot Contest Rules, результат проверки будет `Output isn't correct`.

Example

Функция `program_pulibot` может вызвать функцию `set_instruction` следующим образом:

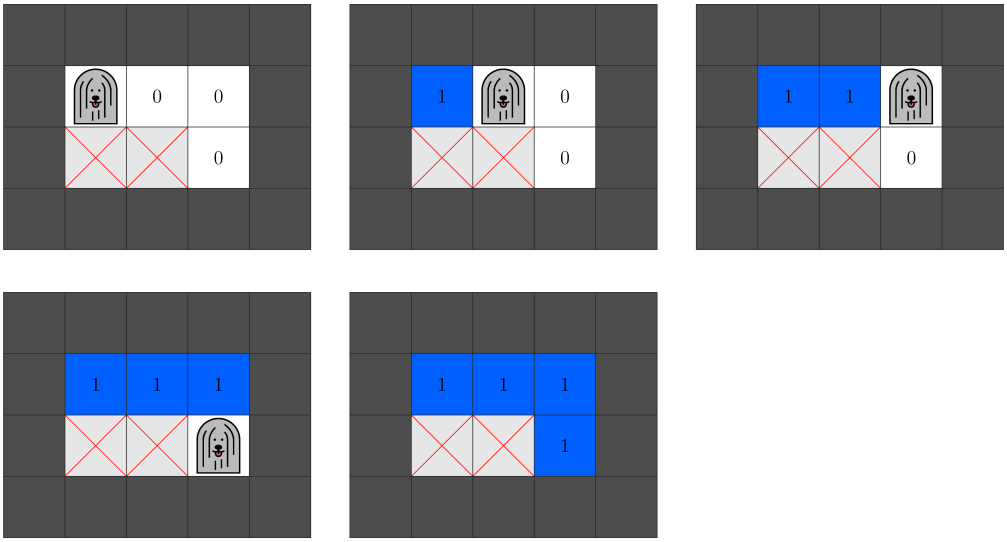
Вызов	Инструкция для массива состояний S
`set_instruction([0, -2, -1, 0, -2], 1, E)`	Установить цвет в 1 и переместиться на восток
`set_instruction([0, 1, -1, 0, -2], 1, E)`	Установить цвет в 1 и переместиться на восток
`set_instruction([0, 1, 0, -2, -2], 1, S)`	Установить цвет в 1 и переместиться на юг
`set_instruction([0, -1, -2, -2, 1], 1, T)`	Установить цвет в 1 и завершить выполнение программы

Рассмотрим сценарий в котором $H = 2$ и $W = 3$, и лабиринт изображен на следующем рисунке.



На этом конкретном лабиринте при исполнении программы Пулибот выполнит четыре шага. Массивы состояний, которые сформирует Пулибот, и выполненные им инструкции, как раз соответствуют четырем вызовам `set_instruction`, сделанным выше, в этом порядке. Последняя из этих инструкций завершит программу.

Следующий рисунок показывает лабиринт перед каждым из четырех шагов и окончательный вид лабиринта перед завершением работы.



Заметим, тем не менее, что эта программа из 4 инструкций не всегда сможет найти кратчайший путь в других корректных лабиринтах. Таким образом, если такое решение отправить в тестирующую систему, оно получит вердикт `Output isn't correct`.

Constraints

$Z_{MAX} = 19$. Таким образом, Пулибот может использовать цвета от 0 до 19, включительно.

Для каждого лабиринта, на котором Пулибот будет запускаться, выполнено:

- $2 \leq H, W \leq 15$
- Есть хотя бы один путь от клетки $(0, 0)$ до клетки $(H - 1, W - 1)$.

Subtasks

1. (6 баллов) В лабиринте нет клеток с препятствиями.
2. (10 баллов) $H = 2$
3. (18 баллов) Есть ровно один путь между любой парой пустых клеток.
4. (20 баллов) Длина кратчайшего пути из клетки $(0, 0)$ в клетку $(H - 1, W - 1)$ равна $H + W - 2$.
5. (46 баллов) Нет дополнительных ограничений.

Если в любом из тестов подзадачи вызов функции `set_instruction` или поведение Пулибота во время исполнения выведенной программы не соответствует правилам, описанным в разделе Implementation Details, баллы за эту подзадачу будут равны 0.

В каждой подзадаче вы можете получить частичный балл, если получившаяся раскраска лабиринта является почти корректной.

Формально:

- Решение для теста является **полным**, если финальная раскраска пустых клеток удовлетворяет правилам, описанным в разделе Robot Contest Rules.
- Решение для теста является **частичным**, если финальная раскраска удовлетворяет следующим условиям:
  - Существует путь из $(0, 0)$ до $(H - 1, W - 1)$, для которого цвет каждой клетки на пути равен 1.
  - Никакая другая пустая клетка не имеет цвет 1.
  - Некоторые клетки лабиринта имеют цвет, отличный и от 0, и от 1.

Если ваше решение для теста не является ни полным, ни частичным, ваш балл на этом тесте равен 0.

В подзадачах 1-4 вы получите 100% баллов подзадачи за тест, если ваше решение является полным, и 50% баллов подзадачи за тест, если ваше решение является частичным.

В подзадаче 5 ваш балл зависит от числа цветов, использованных программой для Пулибота. Точнее, обозначим как $Z^\star$ максимальное значение Z среди всех вызовов `set_instruction`. Баллы за тест вычисляются в соответствии со следующей таблицей:

Условие	Баллы (полное решение)	Баллы (частичное решение)
$11 \leq Z^\star \leq 19$	$20 + (19 - Z^\star)$	$12 + (19 - Z^\star)$
$Z^\star = 10$	31	23
$Z^\star = 9$	34	26
$Z^\star = 8$	38	29
$Z^\star = 7$	42	32
$Z^\star \leq 6$	46	36

Баллы за каждую подзадачу равны минимуму баллов за тесты этой подзадачи.

Sample Grader

Грейдер читает входные данные в следующем формате:

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

Здесь m представляет собой H массивов из W целых чисел, описывающих негранничные клетки лабиринта. $m[r][c] = 0$, если клетка (r, c) является пустой, $m[r][c] = 1$, если клетка (r, c) содержит препятствие.

Грейдер вызывает `program_pulibot()`. Если грейдер обнаруживает, что протокол нарушен, он выводит `Protocol Violation: <MSG>` и завершается, здесь `<MSG>` — одно из следующих сообщений:

- `Invalid array`: $-2 \leq S[i] \leq Z_{MAX}$ не выполнено для некоторого i, или длина массива S не равна 5.
- `Invalid color`: $0 \leq Z \leq Z_{MAX}$ не выполнено.
- `Invalid action`: символ A — не один из `H`, `W`, `S`, `E`, `N` или `T`.
- `Same state array`: `set_instruction` была вызвана с одним и тем же массивом состояний S хотя бы два раза.

Иначе функция `program_pulibot` завершается и грейдер выполняет полученную программу для Пулибота на описанном во вводе лабиринте.

Грейдер формирует два результата.

Во-первых он записывает лог всех действий Пулибота в файл `robot.bin` в рабочем каталоге. Этот файл может быть использован визуализатором, который описан в следующем разделе.

Во-вторых, если программа для Пулибота не завершается успешно, грейдер выводит одно их следующих сообщений об ошибке:

- `Unexpected state`: Пулибот сформировал массив состояний, для которого инструкция не была определена вызовом `set_instruction`.
- `Invalid move`: в результате выполнения инструкции Пулибот оказался в непустой ячейке.
- `Too many steps`: Пулибот выполнил $500\,000$ шагов и не завершил выполнение программы.

Иначе пусть $e[r][c]$ является состоянием клетки (r, c) после завершения программы Пулибота. Грейдер выводит H строк в следующем формате:

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

Display Tool

В приложении к этой задаче в тестирующей системе есть файл `display.py`. Если запустить эту программу на питоне, она выведет действия Пулибота на лабиринте, переданном грейдеру. Для этого используется бинарный файл `robot.bin`, который должен находиться в рабочем каталоге.

Для запуска визуализатора выполните следующую команду.

```
python3 display.py
```

Программа запустится и откроет окно с графическим интерфейсом. Его основные возможности следующие:

- Можно посмотреть на состояние массива. Текущее положение Пулибота выделено прямоугольником.
- Вы можете отслеживать действия Пулибота, нажимая на стрелки, или соответствующие им горячие клавиши. Также можно перейти к конкретному шагу.
- Очередной шаг Пулибота показывается снизу. Он показывает текущий массив состояний и инструкцию, которая ему соответствует. После последнего шага он показывает либо сообщение об ошибке от грейдера, либо `Terminated`, если программа успешно завершилась.
- Для каждого числа, задающего цвет, вы можете выбрать цвет, используемый визуализатором, а также текст, которым будут помечаться клетки такого цвета. Вы можете изменять цвета или текст одним из двух способов:
  - В диалоговом окне, нажав на кнопку `Colors` .
  - Изменив содержание файла `colors.txt`.
- Для перезагрузки файла `robot.bin` используется кнопка `Reload`. Это может быть полезным, если содержимое файла `robot.bin` изменилось.