Конкурс роботів

Дослідники зі Шегединського університету організовують змагання з програмування роботів. Твій друг, Ганга, вирішив взяти участь у цьому конкурсі. Метою є написання програми для створення Пулібота, який би віддавав належну шану великому інтелекту відомої угорської породи пастушого собаки, Пулі.

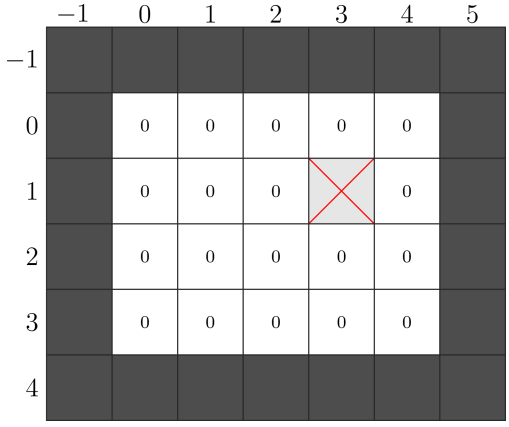
Пулібот буде протестовано на лабіринті, що складається з сітки розміром $(H + 2) \times (W + 2)$ клітинок. Рядки сітки нумеруються від -1 до H зверху вниз, а стовпці сітки нумеруються від -1 до W зліва направо. Ми називаємо клітинку, розташовану у рядку r та стовпці c сітки ($-1 \leq r \leq H$, $-1 \leq c \leq W$), клітинкою (r, c).

Розглянемо клітинку (r, c) таку, що $0 \leq r < H$ і $0 \leq c < W$. Є 4 клітинки **суміжні** з клітинкою (r, c):

- клітинка $(r, c - 1)$ називається **лівою** відносно клітинки (r, c);
- клітинка $(r + 1, c)$ називається **нижньою** відносно клітинки (r, c);
- клітинка $(r, c + 1)$ називається **правою** відносно клітинки (r, c);
- клітинка $(r - 1, c)$ називається **верхньою** відносно клітинки (r, c).

Клітинка (r, c) називається **граничною** клітинкою лабіринту, якщо виконується $r = -1$ або $r = H$ або $c = -1$ або $c = W$. Кожна клітинка, яка не є граничною клітинкою лабіринту, є або клітинкою **перешкоди**, або **порожньою** клітинкою. Крім того, кожна порожня клітинка має **колір**, представлений невід'ємним цілим числом від 0 до Z_{MAX} включно. Спочатку колір кожної порожньої клітинки дорівнює 0.

Наприклад, розглянемо лабіринт з $H = 4$ і $W = 5$, що містить одну клітинку перешкод $(1, 3)$:



Єдина клітинка перешкоди позначена хрестиком. Граничні клітинки лабіринту заштриховані. Числа, написані в кожній порожній клітинці, позначають відповідні кольори.

Шлях довжиною ℓ ($\ell > 0$) від клітинки (r_0, c_0) до клітинки (r_ℓ, c_ℓ) це послідовність попарно різних *порожніх* клітинок $(r_0, c_0), (r_1, c_1), \ldots, (r_\ell, c_\ell)$ у якій для кожного i ($0 \le i < \ell$) клітинки (r_i, c_i) і (r_{i+1}, c_{i+1}) суміжні.

Зверніть увагу, що шлях довжиною ℓ містить рівно $\ell + 1$ клітинок.

На конкурсі дослідники створили лабіринт, у якому існує принаймні один шлях від клітинки $(0, 0)$ до клітинки $(H - 1, W - 1)$. Зауважте, що це означає, що клітинки $(0, 0)$ і $(H - 1, W - 1)$ гарантовано порожні.

Ханга не знає, які клітини лабіринту порожні, а які - перешкоди.

Ваше завдання - допомогти Hanga запрограмувати Pulibot так, щоб він міг знайти *найкоротший шлях* (тобто шлях мінімальної довжини) від клітинки $(0, 0)$ до клітинки $(H - 1, W - 1)$ у невідомому лабіринті, створеному дослідниками. Специфікація Pulibot і правила конкурсу описані нижче.

Зверніть увагу, що останній розділ цього завдання надає опис інструменту для візуалізації Pulibot.

Специфікація Полібота

Визначте **стан** клітинки (r, c) для кожного $-1 \le r \le H$ і $-1 \le c \le W$ як ціле число, так:

- якщо клітинка (r, c) є граничною клітинкою, то її стан -2;
- якщо клітинка (r, c) є клітинкою перешкоди, то її стан -1;
- якщо клітинка (r, c) є порожньою клітинкою, то її станом є колір клітинки.

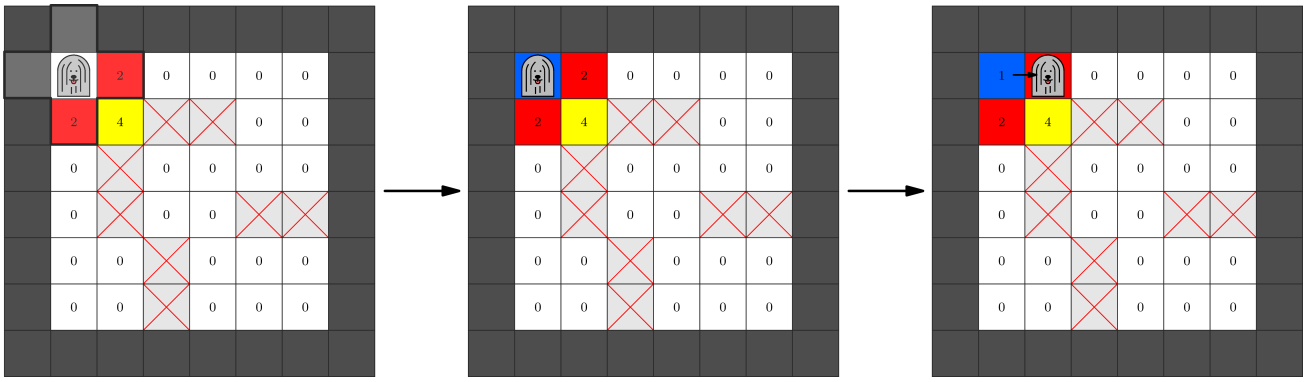
Програма Pulibot являє собою послідовність кроків. На кожному кроці Pulibot розпізнає стан сусідніх клітинок, а потім виконує інструкцію. Інструкції, які він виконує, визначаються на основі визначених станів. Далі наведено більш точний опис.

Припустімо, що на початку поточного кроку Pulibot знаходиться в клітинці (r, c), яка є порожньою. Крок виконується наступним чином:

1. По-перше, Pulibot розпізнає поточний **масив станів**, тобто масив $S = [S[0], S[1], S[2], S[3], S[4]]$, що складається зі стану клітинки (r, c) і всіх суміжних клітинок:
    - $S[0]$ — стан клітинки (r, c).
    - $S[1]$ — стан клітинки зліва.
    - $S[2]$ — стан клітинки знизу.
    - $S[3]$ — стан клітинки справа.
    - $S[4]$ — стан клітинки зверху.

2. Потім Pulibot визначає **інструкцію** (Z, A), яка відповідає отриманому масиву станів.
3. Нарешті Pulibot виконує цю інструкцію: він змінює колір клітинки (r, c) на колір Z а потім виконує дію A, яка є однією з таких дій:
    - *залишитися* в клітинці (r, c);
    - *перемістити* до однієї з 4 суміжних клітинок;
    - *завершити програму*.

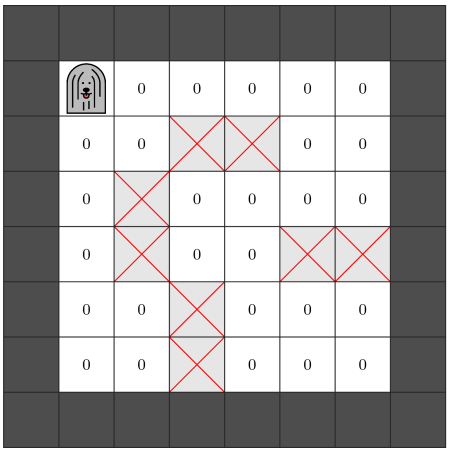
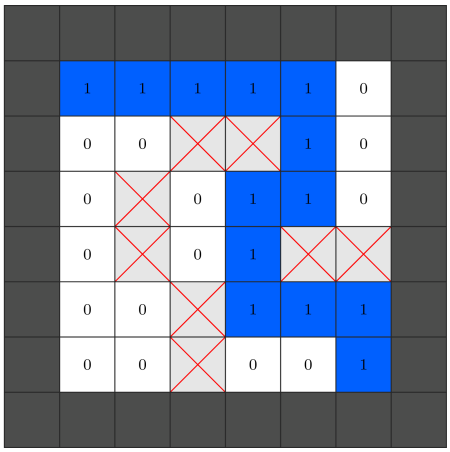
Для прикладу розглянемо сценарій, показаний зліва наступного малюнка. Pulibot зараз знаходиться в клітинці $(0, 0)$ кольору 0. Pulibot розпізнає масив станів $S = [0, -2, 2, 2, -2]$. Pulibot може мати програму, яка, розпізнавши цей масив, встановлює колір поточної клітинки на $Z = 1$, а потім рухається вправо, як показано на малюнку посередині та праворуч:



Правила конкурсу роботів

- На початку Pulibot поміщається в клітинку $(0, 0)$ і починає виконувати свою програму.
- Pulibot заборонено переходити в непорожню клітинку.
- Програма Pulibot має припинити роботу після не більше ніж $500\,000$ кроків.
- Після завершення роботи програми Pulibot порожні клітинки в лабіринті мають бути забарвлені таким чином, щоб:
    - Існує найкоротший шлях від $(0, 0)$ до $(H - 1, W - 1)$, для якого колір кожної клітинки в цьому шляху дорівнює 1.
    - Колір кожної іншої порожньої клітинки – 0.
- Pulibot може завершити своє виконання на будь-якій порожній клітинці.

Наприклад, на наступному малюнку показано можливий лабіринт із $H = W = 6$. Початкова конфігурація відображається ліворуч, а можливе правильне забарвлення порожніх клітинок після завершення відображається праворуч:

Деталі імплементації

Вам слід імплементувати наступну функцію:

```
void program_pulibot()
```

- Ця функція повинна створити програму Pulibot. Ця програма має працювати правильно для всіх значень H і W і будь-якого лабіринту, який відповідає обмеженням завдання.
- Ця функція викликається рівно один раз для кожного тесту.

Ця функція може викликати наступну функцію для створення програми Pulibot:

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

- S: масив довжиною 5, що описує масив стану.
- Z: невід'ємне ціле число, що представляє колір.
- A: один символ, що представляє дію Pulibot наступним чином:
  - `H`: залишитися;
  - `W`: рух вліво;
  - `S`: рух вниз;
  - `E`: рух вправо;
  - `N`: рух вверх;
  - `T`: завершити програму.
- Виклик цієї функції повідомляє Pulibot, що після розпізнавання стану S він повинен виконати інструкцію (Z, A).

Виклик цієї функції кілька разів з одним і тим же станом S призведе до вердикту `Output isn't correct`.

Не потрібно викликати `set_instruction` з кожним можливим масивом станів S. Однак, якщо пізніше Pulibot розпізнає стан, для якого не було встановлено інструкцію, ви отримаєте вердикт `Output isn't correct`.

Після завершення `program_pulibot` градер запускає програму Pulibot для одного або кількох лабіринтів. Ці виклики *не* зараховуються до ліміту часу для вашого рішення. Градер *не* є адаптивним, тобто набір лабіринтів попередньо визначений у кожному тестовому випадку.

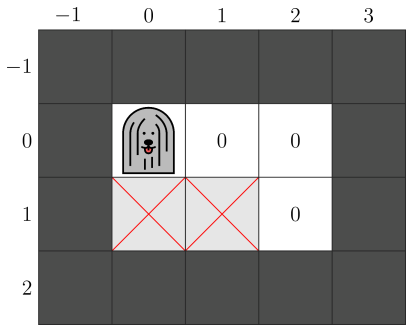
Якщо Pulibot порушить будь-яке з Правил конкурсу роботів до завершення своєї програми, ви отримаєте вердикт `Output isn't correct`.

Приклад

Функція `program_pulibot` може здійснювати виклики `set_instruction` наступним чином:

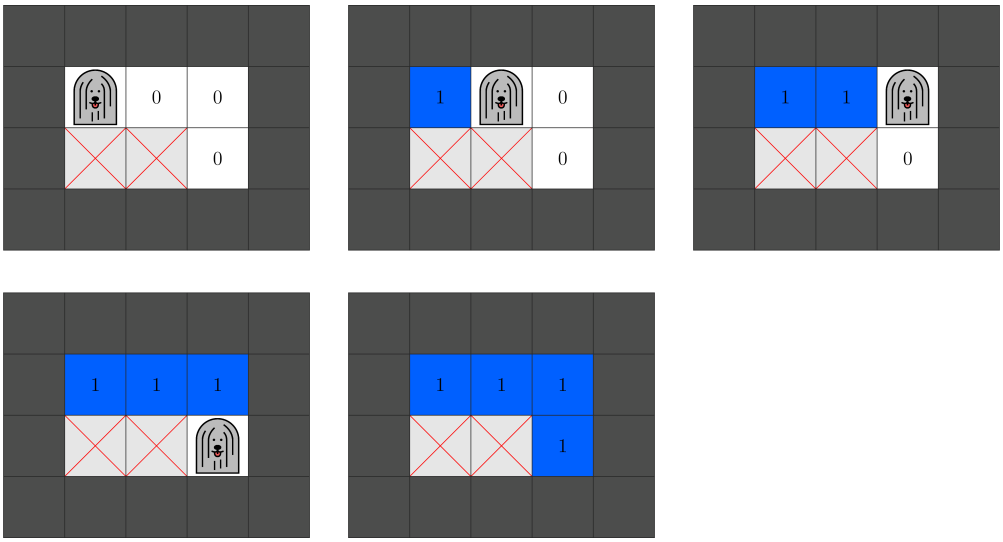
Виклик	Інструкція для масиву станів S
`set_instruction([0, -2, -1, 0, -2], 1, E)`	Встановіть колір 1 і рухайтесь праворуч
`set_instruction([0, 1, -1, 0, -2], 1, E)`	Встановіть колір 1 і рухайтесь праворуч
`set_instruction([0, 1, 0, -2, -2], 1, S)`	Встановіть колір 1 і рухайтесь вниз
`set_instruction([0, -1, -2, -2, 1], 1, T)`	Встановіть колір 1 і завершіть програму

Розглянемо сценарій, де $H = 2$ і $W = 3$, і лабіринт показано на наступному малюнку.



Для цього конкретного лабіринту програма Pulibot виконується в чотири етапи. Стани, які Пулібот визначає і інструкції, які він виконує, відповідають чотирьом зробленим викликам `set_instruction`, описаним вище, по порядку. Остання з цих інструкцій завершує програму.

На наступному малюнку показано лабіринт перед кожним із чотирьох кроків і його кінцевий фарбування після завершення.



Однак зверніть увагу, що ця програма з 4 інструкцій може не знайти найкоротший шлях в інших дійсних лабіринтах. Таким чином, якщо його буде подано, він отримає вердикт `Output isn't correct`.

Обмеження

$Z_{MAX} = 19$. Отже, Pulibot може використовувати кольори від 0 до 19 включно.

Для кожного лабіринту, який використовується для тестування Pulibot:

- $2 \leq H, W \leq 15$
- Є принаймні один шлях від клітинки $(0, 0)$ до клітинки $(H - 1, W - 1)$.

Підзадачі

1. (6 балів) У лабіринті немає клітинок-перешкод.
2. (10 балів) $H = 2$
3. (18 балів) Між кожною парою порожніх клітинок є рівно один шлях.
4. (20 балів) Кожний найкоротший шлях від клітинки $(0, 0)$ до клітинки $(H - 1, W - 1)$ має довжину $H + W - 2$.
5. (46 балів) Без додаткових обмежень.

Якщо в будь-якому з тестових випадків виклики процедури `set_instruction` або програми Pulibot під час її виконання не відповідають обмеженням, описаним у Деталях імплементації, оцінка вашого рішення для цієї підзадачі буде 0.

У кожній підзадачі ви можете отримати частковий бал, створивши майже правильне розфарбування.

Формально:

- Розв’язання тесту вважається **повним**, якщо остаточне забарвлення порожніх клітинок відповідає правилам конкурсу роботів.
- Розв’язок тесту є **частковим**, якщо кінцеве розфарбування виглядає наступним чином:
  - Існує найкоротший шлях від $(0, 0)$ до $(H - 1, W - 1)$, для якого колір кожної клітинки в цьому шляху дорівнює 1.
  - У сітці немає інших порожніх клітинок кольору 1.
  - Деякі порожні клітинки в сітці мають колір, відмінний від 0 і 1.

Якщо ваше розв’язання тесту не є ані повним, ані частковим, ваш бал за відповідний результат тестовий приклад буде 0.

У підзадачах 1-4, правильне рішення отримає 100% балів, а частково правильні рішення отримають 50% балів за відповідну підзадачу.

У підзадачі 5 ваш бал залежить від кількості кольорів, використаних у програмі Pulibot. Точніше, позначте через $Z^\star$ максимальне значення Z для всіх викликів, здійснених до `set_instruction`. Оцінка тесту розраховується згідно з наступною таблицею:

Умова	Оцінка (повна)	Оцінка (часткова)
$11 \leq Z^\star \leq 19$	$20 + (19 - Z^\star)$	$12 + (19 - Z^\star)$
$Z^\star = 10$	31	23
$Z^\star = 9$	34	26
$Z^\star = 8$	38	29
$Z^\star = 7$	42	32
$Z^\star \leq 6$	46	36

Оцінка за кожну підзадачу є мінімальною кількістю балів за тестові приклади у підзадачі.

Приклад градера

Градер зчитує вхідні дані в такому форматі:

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

Тут m — це масив з H масивів з W цілих чисел, що описують неграничні клітинки лабіринту. $m[r][c] = 0$, якщо клітинка (r, c) є порожньою клітинкою, і $m[r][c] = 1$, якщо клітинка (r, c) є клітинкою перешкодою.

Градер спочатку викликає `program_pulibot()`. Якщо градер виявляє порушення протоколу, він виводить `Protocol Violation: <MSG>` і завершується, де `<MSG>` є одним із таких повідомлень про помилку:

- `Invalid array`: $-2 \le S[i] \le Z_{MAX}$ не виконується для деякого i або довжина S не дорівнює 5.
- `Invalid color`: $0 \le Z \le Z_{MAX}$ не виконується.
- `Invalid action`: символ A не є одним із `H`, `W`, `S`, `E`, `N` або `T`.
- `Same state array`: `set_instruction` викликано з тим самим масивом S принаймні двічі.

В іншому випадку, коли `program_pulibot` завершиться, градер виконує програму Pulibot у лабіринті, описаному вхідними даними.

Градер дає два результати.

По-перше, градер записує журнал дій Pulibot у файл `robot.bin` у робочому каталозі. Цей файл слугує вхідними даними для інструмента візуалізації, описаного в наступному розділі.

По-друге, якщо програма Pulibot не завершується успішно, градер виводить одне з таких повідомлень про помилку:

- `Unexpected state`: Pulibot розпізнав масив стану, з яким не було викликано `set_instruction`.
- `Invalid move`: виконання дії призвело до того, що Pulibot перемістився до непорожньої клітинки.
- `Too many steps`: Pulibot виконав кроки на суму $500\,000$, не завершуючи свою програму.

В іншому випадку нехай $e[r][c]$ буде станом клітинки (r, c) після завершення програми Pulibot. Градер друкує рядки H у такому форматі:

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

Візуалізатор

Пакет вкладень для цього завдання містить файл із назвою `display.py`. Після виклику цей скрипт на Python відображає дії Pulibot у лабіринті, описаному вхідними даними градера. Для цього бінарний файл `robot.bin` повинен бути присутнім у робочому каталозі.

Щоб викликати скрипт, виконайте наступну команду.

```
python3 display.py
```

З'являється простий графічний інтерфейс. Основні особливості такі:

- Ви можете спостерігати за станом усього лабіринту. Поточне місце розташування Pulibot виділено прямокутником.

- Ви можете переглядати кроки Pulibot, натискаючи кнопки зі стрілками або гарячі клавіші. Ви також можете перейти до певного кроку.
- Майбутній крок у програмі Pulibot показано внизу. Він показує поточний масив стану та інструкцію, яку він виконуватиме. Після останнього кроку відображається або одне з повідомлень про помилку градера, або `Terminated`, якщо програма завершується успішно.
- Кожному числу, яке представляє колір, можна призначити візуальний колір фону, а також текст для відображення. Відображуваний текст – це короткий рядок, який слід записати в кожну клітинку одного кольору. Ви можете призначити кольори фону та відображення тексту одним із наведених нижче способів.
    - Встановіть їх у діалоговому вікні після натискання кнопки `Colors`.
    - Відредагуйте вміст файлу `colors.txt`.
- Щоб перезавантажити `robot.bin`, скористайтеся кнопкою `Reload`. Це корисно, якщо вміст `robot.bin` змінився.