Robot Contest

Các nhà nghiên cứu AI tại Đại học Szeged đang tổ chức một cuộc thi lập trình robot. Một người bạn của bạn tên là Hanga đã quyết định tham gia cuộc thi. Mục tiêu là lập trình *Pulibot* tối thượng, ngưỡng mộ trí thông minh tuyệt vời của Puli, giống chó chăn gia súc nổi tiếng của Hungary.

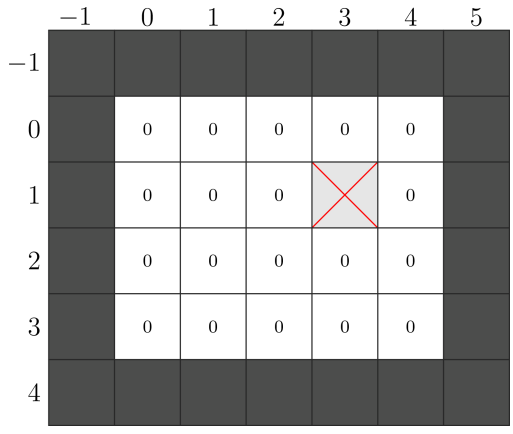
Pulibot sẽ được thử nghiệm trên một mê cung bao gồm một lưới kích thước $(H+2) \times (W+2)$ các ô. Các hàng của lưới được đánh số từ -1 đến H từ bắc tới nam và các cột của lưới được đánh số từ -1 đến W từ tây sang đông. Ta gọi ô nằm ở hàng r và cột c của lưới ($-1 \le r \le H$, $-1 \le c \le W$) là ô (r, c).

Xét một ô (r, c) với $0 \le r < H$ và $0 \le c < W$. Có 4 ô **liền kề** với ô (r, c):

- ô $(r, c-1)$ được gọi là ô **phía tây** của ô (r, c);
- ô $(r+1, c)$ được gọi là ô **phía nam** của ô (r, c);
- ô $(r, c+1)$ được gọi là ô **phía đông** của ô (r, c);
- ô $(r-1, c)$ được gọi là ô **phía bắc** của ô (r, c).

Ô (r, c) được gọi là ô **biên giới** của mê cung nếu $r = -1$ hoặc $r = H$ hoặc $c = -1$ hoặc $c = W$. Mỗi ô không phải là ô biên giới của mê cung là ô **chướng ngại vật** hoặc ô **trống**. Ngoài ra, mỗi ô trống có một **màu**, được biểu thị bởi một số nguyên không âm nằm trong khoảng từ 0 đến Z_{MAX}, kể cả 0 và Z_{MAX}. Ban đầu, màu của mỗi ô trống là 0.

Ví dụ, xét một mê cung có $H = 4$ và $W = 5$, chứa một ô chướng ngại vật $(1, 3)$:



Ô chướng ngại vật duy nhất được biểu thị bằng hình chữ X. Các ô biên giới của mê cung được tô đậm. Số được ghi vào mỗi ô trống đại diện cho màu của chúng.

Một **đường đi** có độ dài ℓ ($\ell > 0$) từ ô (r_0, c_0) đến ô (r_ℓ, c_ℓ) là một chuỗi các ô *trống* đôi một khác nhau $(r_0, c_0), (r_1, c_1), \ldots, (r_\ell, c_\ell)$ trong đó với mỗi i ($0 \le i < \ell$) các ô (r_i, c_i) và (r_{i+1}, c_{i+1}) là liền kề.

Lưu ý là đường đi có độ dài ℓ chứa đúng $\ell + 1$ ô.

Tại cuộc thi, các nhà nghiên cứu đã thiết lập một mê cung trong đó tồn tại ít nhất một đường đi từ ô $(0, 0)$ đến ô $(H - 1, W - 1)$. Lưu ý rằng điều này ngụ ý là các ô $(0, 0)$ và $(H - 1, W - 1)$ được đảm bảo là các ô trống.

Hanga không biết những ô nào của mê cung là trống và những ô nào là chướng ngại vật.

Nhiệm vụ của bạn là giúp Hanga lập trình Pulibot để nó có thể tìm ra *đường đi ngắn nhất* (nghĩa là đường đi có độ dài nhỏ nhất) từ ô $(0, 0)$ đến ô $(H - 1, W - 1)$ trong mê cung chưa biết trước do các nhà nghiên cứu thiết lập. Thông số kỹ thuật của Pulibot và các quy tắc của cuộc thi được mô tả dưới đây.

Lưu ý rằng phần cuối cùng của đề bài này mô tả một công cụ hiển thị mà bạn có thể sử dụng để trực quan hóa Pulibot.

Thông số kỹ thuật của Pulibot

Định nghĩa **trạng thái** của một ô (r, c) cho mỗi $-1 \le r \le H$ và $-1 \le c \le W$ là một số nguyên sao cho:

- nếu ô (r, c) là một ô biên giới thì trạng thái của nó là -2;
- nếu ô (r, c) là một ô chướng ngại vật thì trạng thái của nó là -1;
- nếu ô (r, c) là một ô trống thì trạng thái của nó là màu của ô đó.

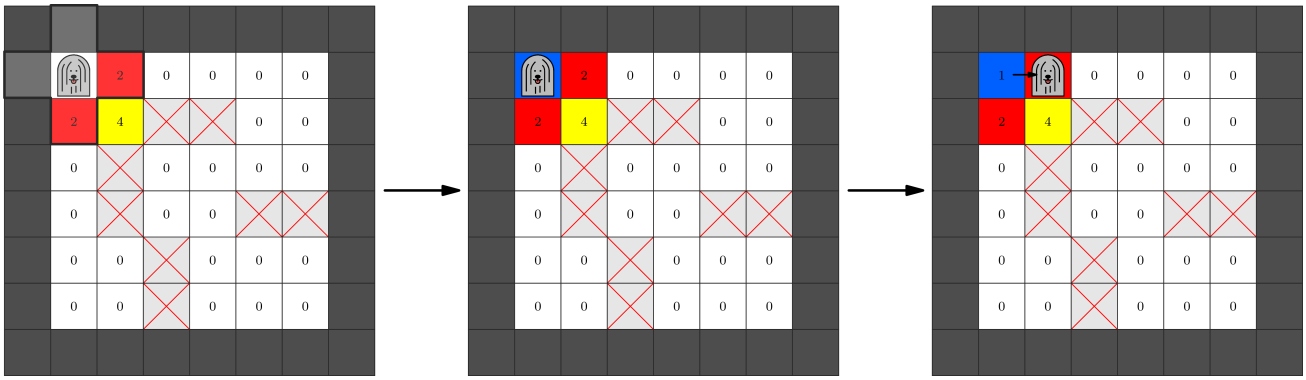
Chương trình của Pulibot được thực thi theo trình tự các bước. Trong mỗi bước, Pulibot nhận biết trạng thái của các ô lân cận và sau đó thực hiện một lệnh. Lệnh mà nó thực hiện được xác định bởi các trạng thái đã được biết. Xem một mô tả chính xác hơn dưới đây.

Giả sử ở bước hiện tại, Pulibot đang ở ô (r, c) là một ô trống. Bước này sẽ được thực hiện như sau:

1. Đầu tiên, Pulibot nhận dạng **mảng trạng thái** hiện tại, tức là mảng $S = [S[0], S[1], S[2], S[3], S[4]]$, bao gồm trạng thái của ô (r, c) và của tất cả các ô liền kề:
   - $S[0]$ là trạng thái của ô (r, c).
   - $S[1]$ là trạng thái của ô ở phía tây.
   - $S[2]$ là trạng thái của ô ở phía nam.
   - $S[3]$ là trạng thái của ô ở phía đông.
   - $S[4]$ là trạng thái của ô ở phía bắc.
2. Sau đó, Pulibot xác định **lệnh** (Z, A) tương ứng với mảng trạng thái được nhận dạng.
3. Cuối cùng, Pulibot thực hiện lệnh đó: nó đặt màu của ô (r, c) thành màu Z và sau đó nó thực hiện hành động A, là một trong những hành động sau:

- *ở lại* tại ô (r, c);
- *di chuyển* đến một trong 4 ô liền kề;
- *chấm dứt chương trình*.

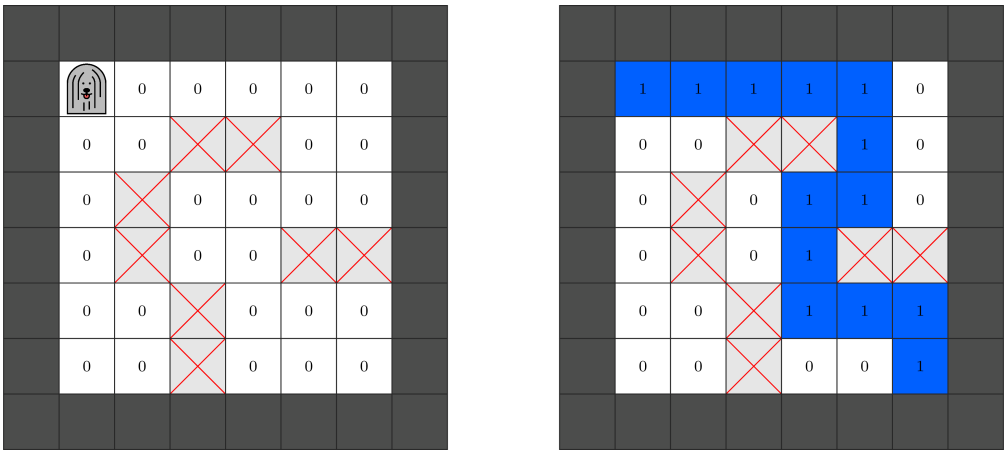
Ví dụ, xét kịch bản hiển thị bên trái của hình dưới đây. Pulibot hiện đang ở ô $(0, 0)$ với màu 0. Pulibot nhận dạng mảng trạng thái $S = [0, -2, 2, 2, -2]$. Khi nhận ra mảng này, Pulibot có thể có một chương trình thiết lập màu của ô hiện tại thành $Z = 1$ và sau đó di chuyển về phía đông, như được hiển thị ở giữa và bên phải của hình:



Các quy tắc Cuộc thi Robot

- Khi bắt đầu, Pulibot được đặt tại ô $(0, 0)$ và bắt đầu thực hiện chương trình của nó.
- Pulibot không được phép di chuyển đến ô không là ô trống.
- Chương trình của Pulibot phải kết thúc sau tối đa $500\,000$ bước.
- Sau khi kết thúc chương trình Pulibot, các ô trống trong mê cung sẽ được tô màu sao cho:
  - Tồn tại một đường đi ngắn nhất từ $(0, 0)$ đến $(H - 1, W - 1)$ mà màu của mỗi ô trong đường đi là 1.
  - Tất cả ô trống khác có màu 0.
- Pulibot có thể kết thúc chương trình của mình tại bất kì ô trống nào.

Ví dụ: hình dưới đây cho thấy một trường hợp mê cung với $H = W = 6$. Cấu hình ban đầu được hiển thị ở bên trái và một cách tô màu được chấp nhận của các ô trống sau khi kết thúc được hiển thị ở bên phải:



Chi tiết cài đặt

Bạn cần cài đặt hàm sau.

```
void program_pulibot()
```

- Hàm này cần tạo ra chương trình Pulibot. Chương trình này cần chạy đúng với tất cả các giá trị của H và W và bất kỳ mê cung nào thoả mãn các ràng buộc của bài toán.
- Hàm này được gọi đúng một lần cho mỗi test.

Hàm này có thể thực hiện các lời gọi đến hàm sau để tạo chương trình Pulibot:

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

- S: mảng có độ dài 5 mô tả một mảng trạng thái.
- Z: số nguyên không âm biểu thị một màu.
- A: ký tự đơn biểu thị một hành động của Pulibot như sau:
  - `H`: ở lại;
  - `W`: di chuyển về phía tây;
  - `S`: di chuyển về phía nam;
  - `E`: di chuyển về phía đông;
  - `N`: di chuyển về phía bắc;
  - `T`: kết thúc chương trình.
- Gọi hàm này để hướng dẫn Pulibot rằng khi nhận ra mảng trạng thái S, nó cần thực hiện lệnh (Z, A).

Gọi hàm này nhiều lần với cùng một mảng trạng thái S sẽ dẫn đến phản hồi `Output isn't correct`.

Không bắt buộc phải gọi `set_instruction` với mỗi trường hợp mảng trạng thái S. Tuy nhiên, nếu sau đó Pulibot nhận ra một mảng trạng thái mà lệnh không được đặt cho nó, bạn sẽ nhận được phản hồi `Output isn't correct`.

Sau khi `program_pulibot` hoàn thành, trình chấm gọi chương trình Pulibot đối với một hoặc nhiều trường hợp mê cung. Những lời gọi này sẽ *không* tính vào giới hạn thời gian cho lời giải của bạn. Trình chấm là *không* thích nghi, nghĩa là tập các mê cung đã được xác định trước trong mỗi test.

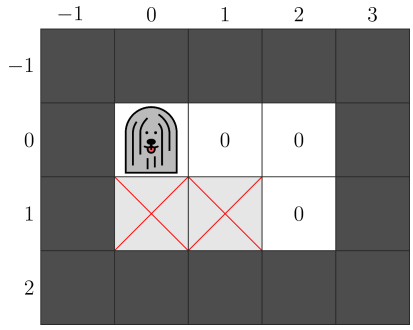
Nếu Pulibot vi phạm bất kỳ quy tắc nào của Cuộc thi Robot trước khi chấm dứt chương trình, bạn sẽ nhận được phản hồi `Output isn't correct`.

Ví dụ

Hàm `program_pulibot` có thể thực hiện các lời gọi đến `set_instruction` như sau:

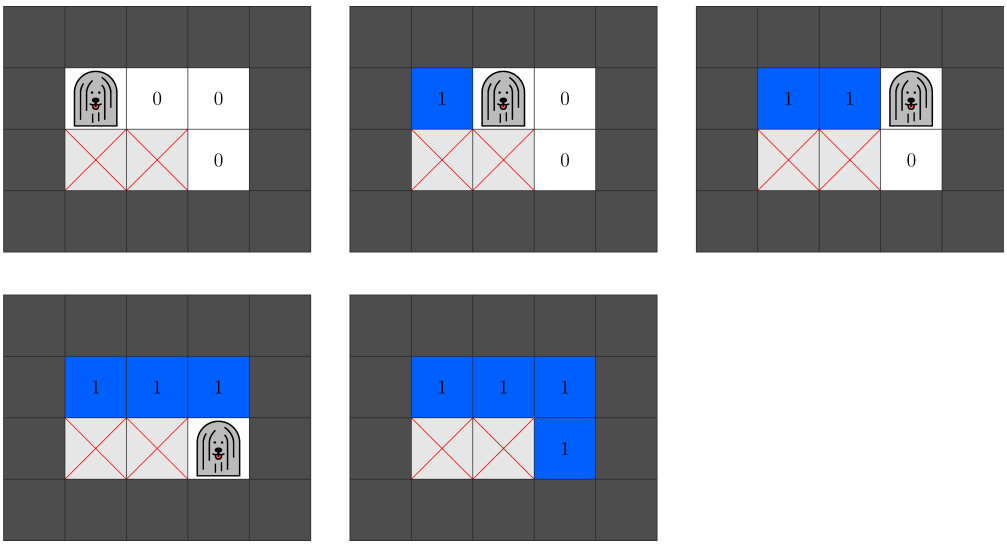
Lời gọi	Lệnh cho mảng trạng thái S
`set_instruction([0, -2, -1, 0, -2], 1, E)`	Đặt màu thành 1 và di chuyển sang phía đông
`set_instruction([0, 1, -1, 0, -2], 1, E)`	Đặt màu thành 1 và di chuyển sang phía đông
`set_instruction([0, 1, 0, -2, -2], 1, S)`	Đặt màu thành 1 và di chuyển sang phía nam
`set_instruction([0, -1, -2, -2, 1], 1, T)`	Đặt màu thành 1 và kết thúc chương trình

Xét một kịch bản với $H = 2$ và $W = 3$, và mê cung được hiển thị trong hình dưới đây.



Đối với mê cung đặc biệt này, chương trình của Pulibot chạy theo bốn bước. Mảng trạng thái Pulibot nhận dạng và các lệnh mà nó thực hiện tương ứng với đúng bốn lệnh theo thứ tự gọi tới `set_instruction` được thực hiện ở trên. Lệnh cuối cùng sẽ kết thúc chương trình.

Hình dưới đây chỉ ra mê cung trước mỗi bước trong số bốn bước và các màu cuối cùng sau khi kết thúc.



Tuy nhiên, xin lưu ý rằng chương trình của 4 lệnh này có thể không tìm thấy đường đi ngắn nhất trong các mê cung hợp lệ khác. Do đó, nếu nộp lên, chương trình sẽ nhận được phản hồi `Output isn't correct`.

Các ràng buộc

$Z_{MAX} = 19$. Do đó, Pulibot có thể sử dụng các màu từ 0 đến 19, kể cả 0 và 19.

Đối với mỗi mê cung được sử dụng để kiểm thử Pulibot:

- $2 \leq H, W \leq 15$
- Có ít nhất một đường đi từ ô $(0, 0)$ đến ô $(H - 1, W - 1)$.

Các subtask

1. (6 điểm) Không có ô chướng ngại vật trong mê cung.
2. (10 điểm) $H = 2$
3. (18 điểm) Có đúng một đường đi giữa mỗi cặp hai ô trống.
4. (20 điểm) Mỗi đường đi ngắn nhất từ ô $(0, 0)$ đến ô $(H - 1, W - 1)$ có độ dài $H + W - 2$.
5. (46 points) Không có ràng buộc nào thêm.

Nếu trong bất kỳ test nào, lệnh gọi đến hàm `set_instruction` hoặc chương trình của Pulibot trong quá trình thực thi không tuân theo các ràng buộc được mô tả trong mục Chi tiết cài đặt, thì điểm cho lời giải của bạn đối với subtask đó sẽ là 0.

Trong mỗi subtask, bạn có thể đạt được một phần điểm bằng cách đưa ra một cách gán màu gần đúng.

Cụ thể:

- Lời giải của một test là **đầy đủ** nếu cách gán màu cuối cùng của các ô trống thoả mãn các quy tắc Cuộc thi Robot.
- Lời giải của test là **một phần** nếu cách gán màu cuối cùng có dạng như sau:
  - Tồn tại một đường đi ngắn nhất từ $(0, 0)$ đến $(H - 1, W - 1)$ mà màu của mỗi ô trong đường đi là 1.
  - Không có ô trống nào khác trong lưới có màu 1.
  - Có ô trống trong lưới có màu khác 0 và 1.

Nếu lời giải của bạn cho một test không là đầy đủ hay một phần, điểm của bạn cho test tương ứng sẽ là 0.

Trong các subtask 1-4, điểm cho lời giải đầy đủ là 100% và điểm cho lời giải một phần đối với một test là 50% số điểm cho subtask của nó.

Trong subtask 5, điểm của bạn phụ thuộc vào số lượng màu được sử dụng trong chương trình của Pulibot. Một cách chính xác hơn, gọi $Z^\star$ là giá trị tối đa của Z trên tất cả các lời gọi được thực hiện

tới `set_instruction`. Điểm của test được tính theo bảng sau:

Điều kiện	Điểm (đầy đủ)	Điểm (một phần)
$11 \le Z^\star \le 19$	$20 + (19 - Z^\star)$	$12 + (19 - Z^\star)$
$Z^\star = 10$	31	23
$Z^\star = 9$	34	26
$Z^\star = 8$	38	29
$Z^\star = 7$	42	32
$Z^\star \le 6$	46	36

Điểm cho mỗi subtask là điểm nhỏ nhất trong các điểm của các test trong subtask đó.

Trình chấm mẫu

Trình chấm mẫu đọc dữ liệu vào theo định dạng sau:

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

Ở đây, m là một mảng của H mảng W số nguyên, mô tả các ô không là biên giới của mê cung. $m[r][c] = 0$ nếu ô (r, c) là một ô trống và $m[r][c] = 1$ nếu ô (r, c) là một ô chướng ngại vật.

Đầu tiên trình chấm mẫu sẽ gọi `program_pulibot()`. Nếu trình chấm mẫu phát hiện ra một vi phạm giao thức, trình chấm mẫu sẽ in ra `Protocol Violation: <MSG>` và kết thúc, trong đó `<MSG>` là một trong số các thông báo lỗi sau:

- `Invalid array`: điều kiện $-2 \le S[i] \le Z_{MAX}$ không thoả mãn với i nào đó hoặc độ dài của S không bằng 5.
- `Invalid color`: điều kiện $0 \le Z \le Z_{MAX}$ không thoả mãn.
- `Invalid action`: kí tự A không là một trong số `H`, `W`, `S`, `E`, `N` hay `T`.
- `Same state array`: `set_instruction` được gọi với cùng một mảng S ít nhất hai lần.

Mặt khác, khi `program_pulibot` hoàn thành, trình chấm mẫu sẽ thực hiện chương trình của Pulibot trong mê cung được mô tả bởi dữ liệu vào.

Trình chấm mẫu tạo ra hai kết quả ra.

Đầu tiên, trình chấm mẫu ghi nhật ký các hành động của Pulibot vào tệp `robot.bin` trong thư mục làm việc. Tệp này đóng vai trò là dữ liệu vào của công cụ trực quan được mô tả trong phần sau.

Thứ hai, nếu chương trình của Pulibot không kết thúc thành công, trình chấm mẫu sẽ in ra một trong các thông báo lỗi sau:

- `Unexpected state`: Pulibot đã nhận ra một mảng trạng thái mà `set_instruction` không được gọi cùng.
- `Invalid move`: thực hiện một hành động khiến Pulibot di chuyển đến một ô không trống.
- `Too many steps`: Pulibot đã thực hiện $500\,000$ bước mà không chấm dứt chương trình của nó.

Nếu không, gọi $e[r][c]$ là trạng thái của ô (r, c) sau khi chương trình Pulibot kết thúc. Trình chấm mẫu in ra H dòng theo định dạng sau:

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

Công cụ hiển thị

Gói đính kèm cho bài toán này chứa một tệp có tên `display.py`. Khi được gọi, chương trình Python này sẽ hiển thị các hành động của Pulibot trong mê cung được mô tả bởi dữ liệu vào của trình chấm mẫu. Để làm được điều này, tệp nhị phân `robot.bin` phải có trong thư mục làm việc.

Để gọi chương trình này, hãy thực hiện lệnh sau.

```
python3 display.py
```

Một giao diện đồ họa đơn giản sẽ hiện lên. Các tính năng chính như sau:

- Bạn có thể quan sát trạng thái của mê cung đầy đủ. Vị trí hiện tại của Pulibot được đánh dấu bởi một hình chữ nhật.
- Bạn có thể duyệt qua các bước của Pulibot bằng cách nhấp vào các nút mũi tên hoặc nhấn vào các phím nóng của chúng. Bạn cũng có thể chuyển sang một bước cụ thể.
- Bước tiếp theo trong chương trình của Pulibot được hiển thị ở phía dưới. Nó hiển thị mảng trạng thái hiện tại và hướng dẫn mà nó sẽ thực hiện. Sau bước cuối cùng, nó hiển thị một trong các thông báo lỗi của trình chấm hoặc `Terminated` nếu chương trình kết thúc thành công.
- Đối với mỗi số đại diện cho một màu, bạn có thể gán một màu nền trực quan cũng như một văn bản hiển thị. Văn bản hiển thị là một xâu ngắn hiển thị trong mỗi ô có màu đó. Bạn có thể gán các màu nền và hiển thị các văn bản theo một trong các cách sau:
  - Đặt chúng trong cửa sổ hộp thoại sau khi nhấp vào nút `Colors`.
  - Chỉnh sửa nội dung tệp `colors.txt`.
- Để tải lại `robot.bin`, hãy sử dụng nút `Reload`. Sẽ rất hữu ích nếu nội dung của `robot.bin` đã thay đổi.