Konkurs programowania robotów

Uniwersytet w Szeged organizuje konkurs programowania robotów. Twoja koleżanka Hanga postanowiła wziąć w nim udział. Celem konkursu jest zaprogramowanie *Pulibota*, nazwanego na cześć bardzo inteligentnych psów rasy Puli, które pochodzą z Węgier.

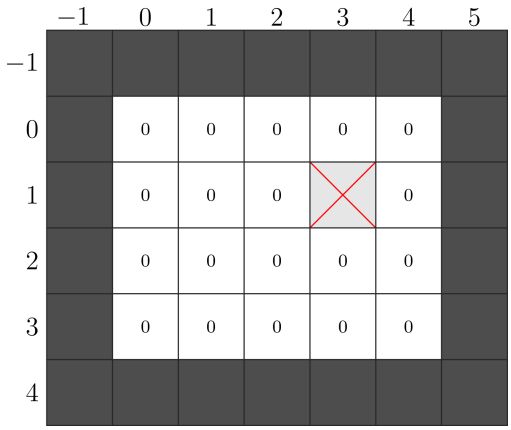
Pulibot testowany jest w labiryncie. Labirynt ma kształt prostokąta o wymiarach $(H + 2) \times (W + 2)$ i jest podzielony na kwadratowe komórki rozmiaru 1×1. Komórki ułożone są w H wierszy ponumerowanych od -1 do H (z północy na południe) oraz W kolumn ponumerowanych od -1 do W (z zachodu na wschód). Komórkę znajdującą się w wierszu r i kolumnie c ($-1 \leq r \leq H$, $-1 \leq c \leq W$) nazywamy komórką (r, c).

Rozważmy komórkę (r, c), gdzie $0 \leq r < H$ i $0 \leq c < W$. Istnieją 4 komórki **sąsiednie** do komórki (r, c).

- komórkę $(r, c - 1)$ nazywamy komórką na **zachód** od komórki (r, c);
- komórkę $(r + 1, c)$ nazywamy komórką na **południe** od komórki (r, c);
- komórkę $(r, c + 1)$ nazywamy komórką na **wschód** od komórki (r, c);
- komórkę $(r - 1, c)$ nazywamy komórką na **północ** od komórki (r, c).

Komórkę (r, c) nazywamy **brzegową** jeśli $r = -1$ lub $r = H$, lub $c = -1$, lub $c = W$. Każda komórka, która nie jest komórką brzegową, zawiera **przeszkodę** albo jest **pusta**. Każda pusta komórka ma przypisany **kolor** określony nieujemną liczbą całkowitą pomiędzy 0 a Z_{MAX} włącznie. Początkowo, każda pusta komórka ma kolor 0.

Dla przykładu, rozważmy labirynt o wymiarach $H = 4$ i $W = 5$, który zawiera dokładnie jedną przeszkodę w komórce $(1, 3)$.



Na powyższym rysunku przeszkoda została zaznaczona krzyżykiem. Komórki brzegowe są zacienione. Liczby w pustych komórkach określają ich kolory.

Ścieżką długości ℓ ($\ell > 0$) z komórki (r_0, c_0) do komórki (r_ℓ, c_ℓ) nazywamy ciąg różnych od siebie *pustych* komórek $(r_0, c_0), (r_1, c_1), \ldots, (r_\ell, c_\ell)$, w którym dla każdego i ($0 \leq i < \ell$) komórki (r_i, c_i) i (r_{i+1}, c_{i+1}) są sąsiednie.

Zauważ, że ścieżka długości ℓ składa się z dokładnie $\ell + 1$ komórek.

W trakcie konkursu organizatorzy przygotowują labirynt, w którym istnieje co najmniej jedna ścieżka z komórki $(0, 0)$ do komórki $(H - 1, W - 1)$. W szczególności oznacza to, że komórki $(0, 0)$ i $(H - 1, W - 1)$ są puste.

Hanga nie wie, które komórki labiryntu są puste, a które zawierają przeszkody.

Twoim zadaniem jest pomóc Handze zaprogramować Pulibota, tak aby potrafił on znaleźć *najkrótszą ścieżkę* (ścieżkę o jak najmniejszej długości) z komórki $(0, 0)$ do komórki $(H - 1, W - 1)$ w nieznanym labiryncie przygotowanym przez organizatorów. Specyfikacja Pulibota oraz zasady konkursu przedstawione są poniżej.

Ostatnia część treści niniejszego zadania opisuje narzędzie pozwalające wizualizować ruchy Pulibota.

Specyfikacja Pulibota

Stan komórki (r, c), dla każdego $-1 \leq r \leq H$ i $-1 \leq c \leq W$ to następująca liczba całkowita:

- jeśli komórka (r, c) jest komórką brzegową, jej stan to -2;
- jeśli komórka (r, c) zawiera przeszkodę, stanem komórki jest -1;
- jeśli komórka (r, c) jest pusta, jej stan to jej kolor;

Program Pulibota składa się z ciągu kroków. W każdym kroku Pulibot odczytuje stany pobliskich komórek i na ich podstawie wykonuje instrukcję.

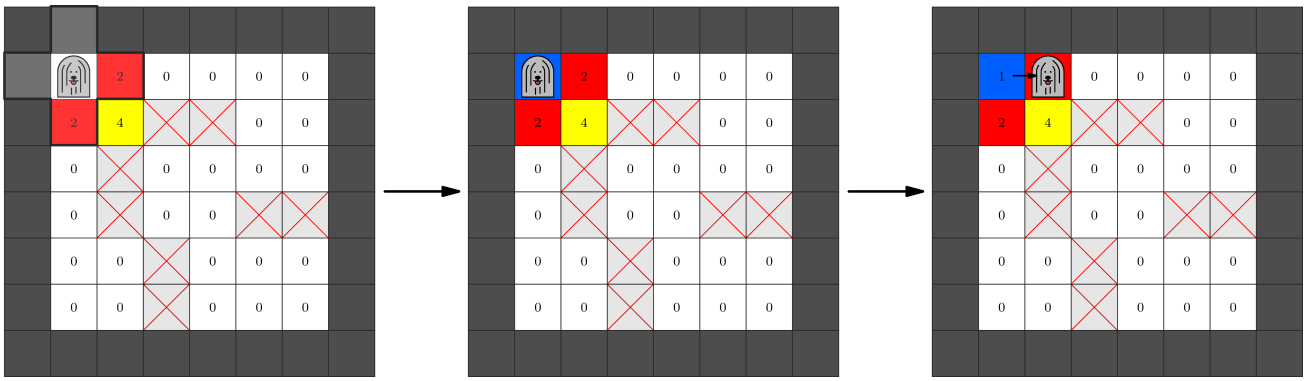
Załóżmy, że na początku aktualnego kroku Pulibot znajduje się w komórce (r, c), która jest pusta. Wykonanie kroku przebiega następująco:

1. Najpierw Pulibot konstruuje **tablicę stanów**, tj. tablicę $S = [S[0], S[1], S[2], S[3], S[4]]$, w której:
    - $S[0]$ to stan komórki (r, c).
    - $S[1]$ to stan komórki na zachód od komórki (r, c).
    - $S[2]$ to stan komórki na południe od komórki (r, c).
    - $S[3]$ to stan komórki na wschód od komórki (r, c).
    - $S[4]$ to stan komórki na północ od komórki (r, c).

2. Następnie Pulibot na podstawie **tablicy stanów** wyznacza **instrukcję** (Z, A).

3. Na koniec Pulibot wykonuje instrukcję: ustawia kolor komórki (r, c) na Z i wykonuje akcję A, która jest jedną z poniższych:
   - *zostań* w komórce (r, c);
   - *przesuń się* do jednej z 4 sąsiednich komórek;
   - *zakończ program*.

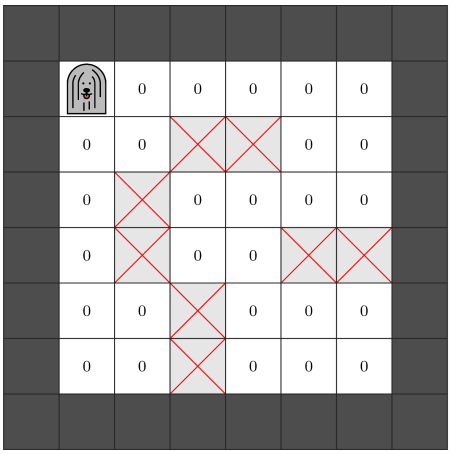
Dla przykładu rozważmy scenariusz pokazany na lewym rysunku poniżej. Pulibot znajduje się w komórce $(0, 0)$, której kolor to 0. Następnie, Pulibot konstruuje tablicę stanów $S = [0, -2, 2, 2, -2]$. Pulibot może być zaprogramowany by w przypadku takiej tablicy stanów ustawić kolor komórki, w której stoi, na $Z = 1$, a następnie przesunąć się na wschód, co ilustruje środkowy i prawy rysunek poniżej.

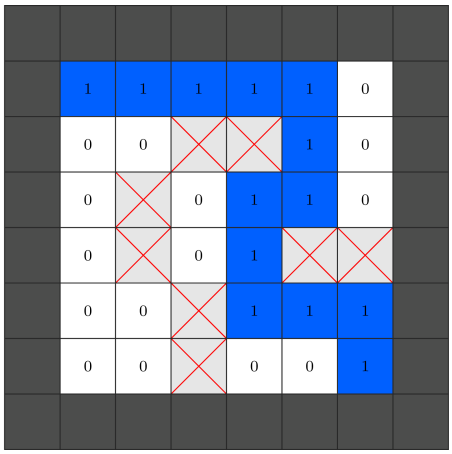


Zasady konkursu robotów

- Na początku Pulibot jest umieszczony w komórce $(0, 0)$ i rozpoczyna wykonanie programu.
- Pulibotowi nie wolno przesunąć się do komórki, która nie jest pusta.
- Program Pulibota musi zakończyć się po co najwyżej $500\,000$ krokach.
- Po zakończeniu programu Pulibota puste komórki w labiryncie powinny być pokolorowane tak, by spełnić poniższe warunki:
  - Istnieje najkrótsza ścieżka z komórki $(0, 0)$ do $(H - 1, W - 1)$, która składa się wyłącznie z komórek koloru 1.
  - Wszystkie inne puste komórki mają kolor 0.
- Pulibot może zakończyć program, będąc w dowolnej pustej komórce.

Poniższy rysunek przedstawia przykładowy labirynt ($H = W = 6$). Lewy rysunek pokazuje konfigurację początkową, a prawy ilustruje pewne akceptowalne kolorowanie komórek po zakończeniu programu.





Szczegóły implementacji

Zaimplementuj poniższą procedurę.

```
void program_pulibot()
```

- Ta procedura powinna wyprodukować program Pulibota. Program powinien działać poprawnie dla wszystkich wartości H i W oraz dowolnego labiryntu spełniających ograniczenia podane w treści zadania.
- Ta procedura jest wywołana dokładnie raz w każdym przypadku testowym.

Procedura `program_pulibot` może wywoływać poniższą procedurę, aby wyprodukować program Pulibota.

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

- S: tablica długości 5 opisująca tablicę stanów.
- Z: nieujemna liczba całkowita określająca kolor.
- A: znak określający akcję Pulibota:
  - `H`: stój w miejscu;
  - `W`: przesuń się o 1 komórkę na zachód;
  - `S`: przesuń się o 1 komórkę na południe;
  - `E`: przesuń się o 1 komórkę na wschód;
  - `N`: przesuń się o 1 komórkę na północ;
  - `T`: zakończ program.
- Wywołanie tej procedury instruuje Pulibota, że w sytuacji, gdy skonstruuje on tablicę stanów S, powinien wykonać instrukcję (Z, A).

Wykonanie procedury `set_instruction` wielokrotnie z taką samą tablicą stanów S powoduje, że wynikiem wykonania Twojego rozwiązania jest werdykt `Output isn't correct`.

Nie ma potrzeby wywołania `set_instruction` dla każdej możliwej tablicy stanów S. Niemniej jednak, jeśli Pulibot skonstruuje tablicę stanów, której nie została przypisana instrukcja do wykonania, wynikiem wykonania Twojego rozwiązania będzie werdykt `Output isn't correct`.

Po wykonaniu `program_pulibot`, program sprawdzający wykonuje program Pulibota dla jednego lub więcej labiryntów. Te wykonania *nie* są uwzględniane przy obliczaniu czasu działania Twojego rozwiązania. Program sprawdzający *nie* jest adaptywny, a zatem zestaw labiryntów jest określony z góry w każdym przypadku testowym.

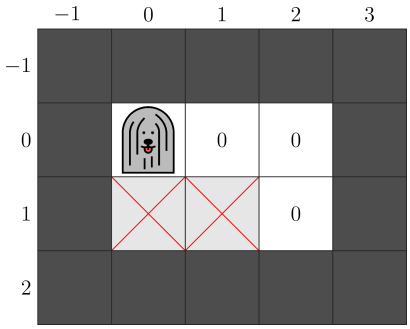
Jeśli Pulibot złamie dowolną z zasad opisanych w Zasadach konkursu robotów przed zakończeniem programu, wynikiem wykonania Twojego rozwiązania będzie werdykt `Output isn't correct`.

Przykład

Przyjmijmy, że procedura `program_pulibot` wykonuje następujące wywołania `set_instruction`:

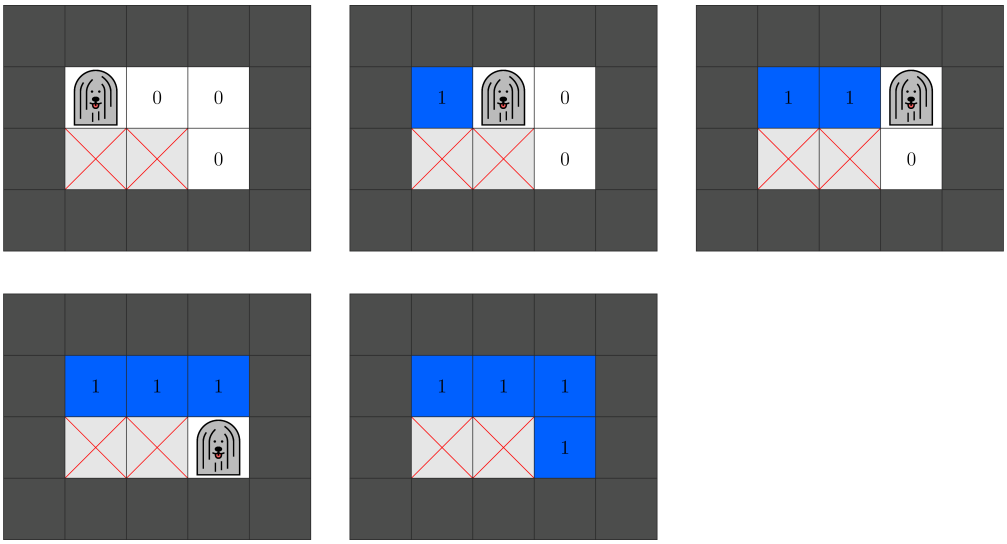
Wywołanie	Instrukcja dla podanej tablicy stanów
`set_instruction([0, -2, -1, 0, -2], 1, E)`	Ustaw kolor na 1 i przesuń się na wschód
`set_instruction([0, 1, -1, 0, -2], 1, E)`	Ustaw kolor na 1 i przesuń się na wschód
`set_instruction([0, 1, 0, -2, -2], 1, S)`	Ustaw kolor na 1 i przesuń się na południe
`set_instruction([0, -1, -2, -2, 1], 1, T)`	Ustaw kolor na 1 i zakończ program

Rozważmy scenariusz, w którym $H = 2$ i $W = 3$, a labirynt wygląda tak, jak na poniższym rysunku.



Dla tego labiryntu program Pulibota wykonuje cztery kroki. Skonstruowane tablice stanów oraz wykonane instrukcje odpowiadają czterem kolejnym wierszom w powyższej tabelce. Ostatnia z tych instrukcji kończy program.

Poniższy rysunek przedstawia labirynt przed każdym krokiem i po zakończeniu programu.



Niemniej jednak, ten program niekoniecznie znajdzie najkrótszą ścieżkę w innych poprawnych labiryntach. Jeśli go zgłosisz, wynikiem oceny Twojego zgłoszenia będzie `Output isn't correct`.

Ograniczenia

$Z_{MAX} = 19$, a zatem Pulibot może używać kolorów od 0 to 19 włącznie.

Dla każdego labiryntu użytego do testowania Pulibota:

- $2 \leq H, W \leq 15$
- Istnieje co najmniej jedna ścieżka z komórki $(0, 0)$ do komórki $(H - 1, W - 1)$.

Podzadania

1. (6 punktów) Żadna z komórek nie zawiera przeszkody.
2. (10 punktów) $H = 2$
3. (18 punktów) Istnieje dokładnie jedna ścieżka pomiędzy każdą parą pustych komórek.
4. (20 punków) Każda najkrótsza ścieżka z komórki $(0, 0)$ do komórki $(H - 1, W - 1)$ ma długość $H + W - 2$.
5. (46 punktów) Brak dodatkowych ograniczeń.

Jeśli w dowolnym przypadku testowym wywołania `set_instruction` lub program Pulibota w trakcie wykonania nie będą respektowały zasad opisanych w Szczegółach implementacji, Twoje rozwiązanie otrzyma 0 punktów.

W każdym podzadaniu możesz również otrzymać punkty, jeśli Twoje rozwiązanie wyprodukuje częściowo poprawne przypisanie kolorów do pól. Konkretnie:

- Rozwiązanie nazywamy **pełnym**, jeśli kolory po wykonaniu programu Pulibota spełniają Zasady konkursu robotów.
- Rozwiązanie nazywamy **częściowym**, jeśli kolory po wykonaniu programu Pulibota spełniają następujące warunki:

- Istnieje ścieżka z komórki $(0, 0)$ do komórki $(H - 1, W - 1)$ składająca się z komórek o kolorze 1.
- Żadna inna pusta komórka nie ma koloru 1.
- Pewna inna pusta komórka ma kolor inny niż 0 lub 1.

Jeśli Twoje rozwiązanie przypadku testowego nie jest ani pełne, ani częściowe, Twoje rozwiązanie otrzyma 0 punktów.

W podzadaniach 1-4, jeśli Twoje rozwiązanie jest częściowe, otrzyma 50% punktów przewidzianych dla danego podzadania (i oczywiście 100% punktów, jeśli jest pełne).

W podzadaniu 5, punktacja rozwiązania zależy od liczby kolorów użytej w programie Pulibota. Oznaczmy przez $Z^\star$ maksymalną wartość Z wśród wszystkich wywołań `set_instruction`. Punktacja za dany przypadek testowy jest obliczana na podstawie poniższej tabelki:

Warunek	Punkty (pełne rozwiązanie)	Punkty (częściowe rozwiązanie)
$11 \leq Z^\star \leq 19$	$20 + (19 - Z^\star)$	$12 + (19 - Z^\star)$
$Z^\star = 10$	31	23
$Z^\star = 9$	34	26
$Z^\star = 8$	38	29
$Z^\star = 7$	42	32
$Z^\star \leq 6$	46	36

Liczba punktów, którą Twoje rozwiązanie otrzymuje za dane podzadanie, to minimum z liczby punktów po wszystkich przypadkach testowych w danym podzadaniu.

Przykładowy program oceniający

Przykładowy program oceniający wczytuje wejście w następującym formacie:

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

m to tablica dwuwymiarowa liczb całkowitych, tj. H tablic długości W, opisująca komórki labiryntu, które nie są brzegowe. $m[r][c] = 0$ oznacza, że komórka (r, c) jest pusta, $m[r][c] = 1$ oznacza, że komórka (r, c) zawiera przeszkodę.

Przykładowy program oceniający najpierw wywołuje `program_pulibot()`. Jeśli zostanie wykryte naruszenie protokołu, przykładowy program oceniający wypisuje `Protocol Violation: <MSG>` i kończy swoje wywołanie, gdzie `<MSG>` to jeden z poniższych komunikatów:

- `Invalid array`: $-2 \leq S[i] \leq Z_{MAX}$ nie zachodzi dla pewnego i lub długość S jest różna od 5.
- `Invalid color`: nie zachodzi $0 \leq Z \leq Z_{MAX}$.
- `Invalid action`: znak A nie jest jednym z `H`, `W`, `S`, `E`, `N` lub `T`.
- `Same state array`: procedura `set_instruction` została wykonana co najmniej dwukrotnie dla takiej samej tablicy S.

W przeciwnym przypadku, gdy wywołanie `program_pulibot` zakończy się, przykładowy program oceniający wykonuje program Pulibota na labiryncie opisanym w wejściu.

Przykładowy program oceniający wypisuje wyjście w dwóch miejscach.

Po pierwsze, zapisuje ciąg akcji Pulibota do pliku `robot.bin` w bieżącym katalogu. Plik ten służy do wizualizacji (patrz sekcja poniżej).

Po drugie, jeśli program Pulibota nie zakończy się poprawnie, przykładowy program oceniający wypisuje jeden z poniższych komunikatów:

- `Unexpected state`: Pulibot skonstruował tablicę stanów, dla której nie wywołano `set_instruction`.
- `Invalid move`: Pulibot próbował przesunąć się na niepuste pole.
- `Too many steps`: Pulibot wykonał $500\,000$ kroków bez zakończenia programu.

W przeciwnym razie, oznaczmy przez $e[r][c]$ stan komórki (r, c) po zakończeniu programu Pulibota. Przykładowy program oceniający wypisuje H wierszy w poniższym formacie:

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

Wizualizacja

Załącznik do niniejszego zadania zawiera plik o nazwie `display.py`. Jest to skrypt Pythona, który pokazuje akcje Pulibota w labiryncie podanym uprzednio na wejściu przykładowego programu oceniającego. W tym celu, plik binarny `robot.bin` musi znajdować się w bieżącym katalogu.

Aby wykonać skrypt, uruchom poniższe polecenie:

```
python3 display.py
```

Na ekranie pojawi się okienko programu, które udostępnia poniższe funkcje.

- Możesz obserwować status labiryntu. Obecne położenie Pulibota jest zaznaczone prostokątem.
- Możesz przeglądać kroki Pulibota, naciskając przyciski ze strzałkami lub klawisze strzałek na klawiaturze. Możesz także przeskoczyć bezpośrednio do konkretnego kroku.

- Dolna część okienka opisuje następny krok w programie Pulibota. Opis składa się z aktualnej tablicy stanów oraz instrukcji, która zostanie wykonana. Po ostatnim kroku dolna część okienka pokazuje komunikat o błędzie lub komunikat `Terminated`, jeśli program zakończył się poprawnie.
- Każdej liczbie reprezentującej kolor możesz przypisać kolor użyty do wizualizacji, jak również tekst. Tekst ten zostanie wyświetlony w każdej komórce o danym kolorze. Możesz przypisać kolory i teksty na dwa sposoby:
  - Ustawiając je w oknie dialogowym po naciśnięciu przycisku `Colors`.
  - Edytując plik `colors.txt`.
- Aby wczytać ponownie plik `robot.bin`, użyj przycisku `Reload`. Przydaje się to, gdy zawartość pliku `robot.bin` ulegnie zmianie.