Натпревар за роботи

Истражувачите за AI на Универзитетот во Сегед одржуваат натпревар во програмирање на роботи. Вашето другарче, Никола, решило да учествува на натпреварот. Задачата е да го програмира најдобриот *Pulibot*, добивајќи инспирација од интелигенцијата на познатата унгарска раса на кучиња - Puli.

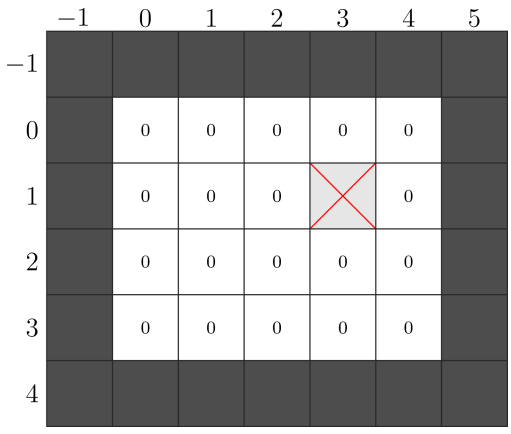
Pulibot ќе биде тестиран на лавиринт кој се состои од $(H + 2) \times (W + 2)$ правоаголна табела со полиња. Редиците се нумерирани од -1 до H почнувајќи од север према југ, а колоните се нумерирани од -1 до W почнувајќи од запад кон исток. Келијата која се наоѓа во редица r и колона c од табелата ($-1 \leq r \leq H$, $-1 \leq c \leq W$) ќе ја дефинираме како (r, c).

За келијата (r, c) така што $0 \leq r < H$ и $0 \leq c < W$ може да дефинираме 4 келии **соседни** на келијата (r, c), и тоа:

- келија $(r, c - 1)$ која ќе ја наречеме келија **западно** од келијата (r, c);
- келија $(r + 1, c)$ која ќе ја наречеме келија **јужно** од келијата (r, c);
- келија $(r, c + 1)$ која ќе ја наречеме келија **источно** од келијата (r, c);
- келија $(r - 1, c)$ која ќе ја наречеме келија **северно** од келијата (r, c).

Келијата (r, c) е наречена **гранична** келија од лавиринтот ако $r = -1$ или $r = H$ или $c = -1$ или важи $c = W$. Секоја келија која не е гранична келија во лавиринтот може да биде **пречка** келија или **празна** келија. Дополнително, секоја празна келија има **боја**, претставена како ненегативен цел број помеѓу 0 и Z_{MAX}, вклучително. На почеток, бојата на секоја празна келија е 0.

На пример, ќе разгледаме лавиринт со $H = 4$ и $W = 5$, кој содржи само една пречка келија $(1, 3)$:



На сликата пречката е прикажана со црвен икс. Граничните келии се дадени со темна боја. Во секоја празна келија е впишана вредноста на нејзината моментална боја.

Дефинираме **патека** со должина ℓ ($\ell > 0$) од келија (r_0, c_0) до келија (r_ℓ, c_ℓ) како секвенца од различни по парови *празни* келии $(r_0, c_0), (r_1, c_1), \ldots, (r_\ell, c_\ell)$ во која секвенца за секое i ($0 \leq i < \ell$) келиите (r_i, c_i) и (r_{i+1}, c_{i+1}) се соседни.

Да забележиме дека патеката со должина ℓ содржи точни $\ell + 1$ келии.

На натпреварот, истражувачите го припремаат лавиринтот така што постои барем една патека од келијата $(0, 0)$ до келијата $(H - 1, W - 1)$. Да забележиме дека ова значи дека келиите $(0, 0)$ и $(H - 1, W - 1)$ се гарантирано празни келии.

Никола не знае кои келии во лавиринтот се празни а кои се со пречки.

Вашата задача е да му помогнете на Никола да го програмира Pulibot за да може да го најде *најкраткиот пат* (тоа е патот со најмала должина) од келијата $(0, 0)$ до келијата $(H - 1, W - 1)$ во непознатиот лавиринт кој го спремиле истражувачите.

Спецификацијата на Pulibot и Правилата на натпреварот се дадени во продолжение.

Последната секција на текстот на оваа задача ја опишува алатката за визуелизација што може да ја користите за да подобро го видите движењето на Pulibot низ лавиринтот.

Спецификацијата на Pulibot

Да дефинираме **состојба** на келија (r, c) за секое $-1 \leq r \leq H$ и $-1 \leq c \leq W$ како цел број така што:

- ако келијата (r, c) е гранична келија, има состојба -2;
- ако келијата (r, c) е келија пречка, има состојба -1;
- ако келијата (r, c) е празна келија, има состојба како бојата на таа келија.

Програмата на Pulibot се извршува како секвенца од чекори. За секој чекор, Pulibot ги препознава состојбите на околните келии и тогаш ја извршува зададената инструкција. Инструкцијата која ја извршува е одредена од препознаените состојби на келиите. Попрецизно објаснување следи во продолжение.

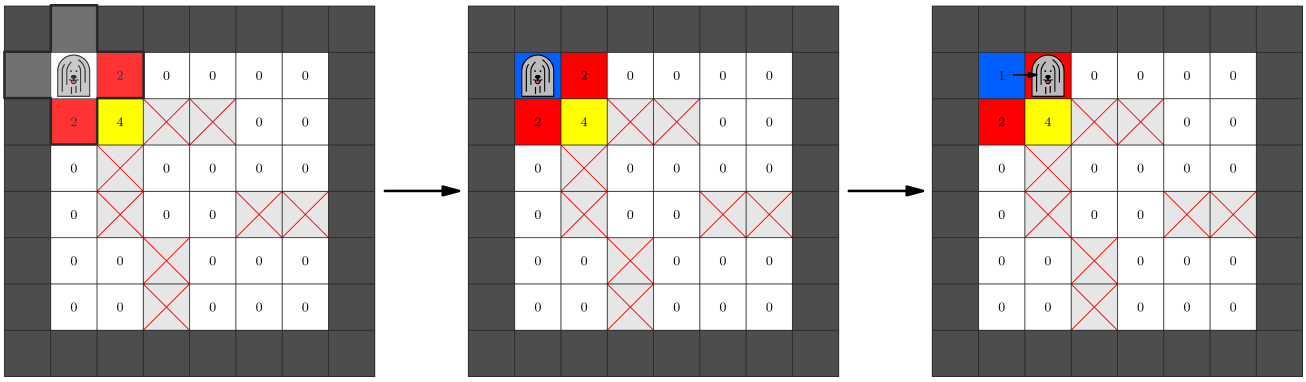
Ако претпоставиме дека на почеток на моменталниот чекор, Pulibot е во келија (r, c), која е празна келија - чекорот се извршува на следниот начин:

1. Прво, Pulibot ја препознава моменталната **низа на состојби**, која е, низата $S = [S[0], S[1], S[2], S[3], S[4]]$, која се состои од состојбите на келијата (r, c) и сите соседни келии:
    - $S[0]$ е состојбата на келијата (r, c).
    - $S[1]$ е состојбата на келијата западно.

- $S[2]$ е состојбата на келијата јужно.
- $S[3]$ е состојбата на келијата источно.
- $S[4]$ е состојбата на келијата северно.

2. Потоа, Pulibot ја одредува **инструкцијата** (Z, A) која соодветствува на препознаената низа на состојби.
3. Конечно, Pulibot ја извршува таа инструкција: ја поставува бојата на келијата (r, c) во боја Z и потоа извршува акција A, која може да биде една од следните акции:
   - *остани* во келија (r, c);
   - *придвижи се* во една од 4 соседни келии;
   - *заврши ја програмата*.

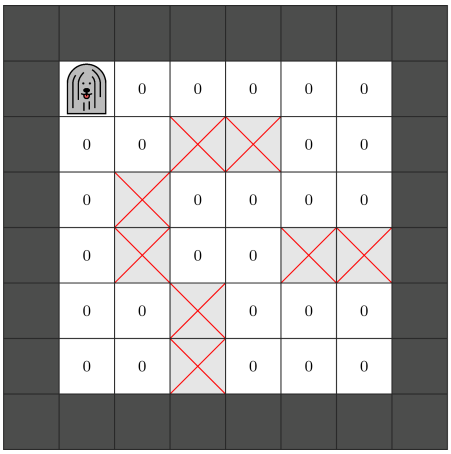
На пример, да го разгледаме сценариото дадено на сликата долу лево. Pulibot е моментално во келијата $(0, 0)$ со боја 0. Pulibot ја препознава низата на состојби $S = [0, -2, 2, 2, -2]$. Pulibot ја има програмата така што по препознавањето на низата, ја поставува бојата на моменталната келија $Z = 1$ и потоа се придвижува кон исток како што е прикажано на сликите долу:

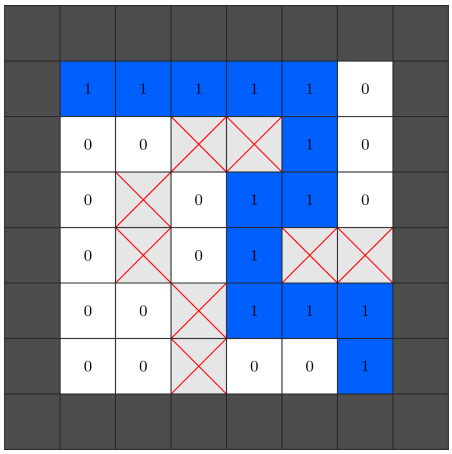


Правила на натпреварот за роботи

- На почеток, Pulibot е поставен во келијата $(0, 0)$ по што почнува со извршување на својата програма.
- Pulibot не смее да се придвижи во келија која не е празна.
- Програмата на Pulibot мора да заврши по најмногу $500\,000$ чекори.
- По завршување на програмата на Pulibot, празните келии во лавиринтот ќе се обојат на следниот начин:
  - Постои најкраток пат од $(0, 0)$ до $(H - 1, W - 1)$ за кои бојата на секоја келија вклучена во тој пат е 1.
  - Бојата на секоја друга празна келија е 0.
- Pulibot може да ја заврши својата програма во било која празна келија.

На пример, на следната слика е прикажан еден лавиринт со $H = W = 6$. Почетната состојба е дадена на левата страна, а едно прифатливо боење на празните келии е дадено десно:





Имплементациски детали

Потребно е да ја имплементирате следната процедура.

```
void program_pulibot()
```

- Оваа процедура треба да ја креира програмата за Pulibot. Оваа програма треба да работи коректно за сите вредности на H и W и за секој лавиринт кој ги исполнува ограничувањата во задачата.
- Оваа процедура ќе биде повикана точно еднаш за секој тест случај.

Оваа процедура може да прави повици кон процедурата дадена подолу за да ја креира посакуваната програма:

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

- S: низа со должина 5 која ја опишува низата на состојби.
- Z: ненегативен цел број кој ја опишува бојата.
- A: еден знак кој ја претставува акцијата на Pulibot која може да биде:
  - `H`: остани;
  - `W`: придвижи се кон запад;
  - `S`: придвижи се кон југ;
  - `E`: придвижи се кон исток;
  - `N`: придвижи се кон север;
  - `T`: заврши ја програмата.
- Повикувањето на оваа процедура му кажува на Pulibot по препознавањето на низата на состојби S да ја изврши инструкцијата (Z, A).

При повикување на процедурата повеќе пати со иста низа на состојби S ќе добиете резултат `Output isn't correct`.

Не е потребно да се повика процедурата `set_instruction` за секоја можна низа на состојби S. Меѓутоа, ако Pulibot подоцна препознае низа на состојби за која не е повиката процедурата ќе добиете резултат `Output isn't correct`.

Откако процедурата `program_pulibot` ќе заврши, оценувачот ќе ја повика програмата на Pulibot на еден или повеќе лавиринти. Овие повикувања *не се бројат* во временскиот лимит на вашето решение. Оценувачот *не е адаптивен*, односно сетот на лавиринти се преддефинирани за секој тест случај.

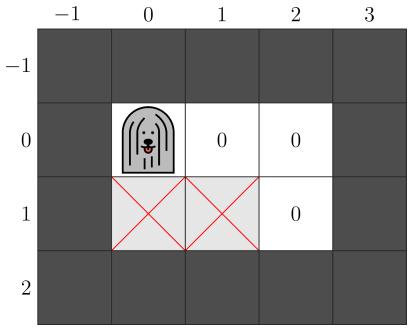
Ако Pulibot прекрши некое од Правилата на натпреварот пред да заврши неговата програма, ќе добиете резултат `Output isn't correct`.

Пример

Процедурата `program_pulibot` прави повици кон процедурата `set_instruction` на следниот начин:

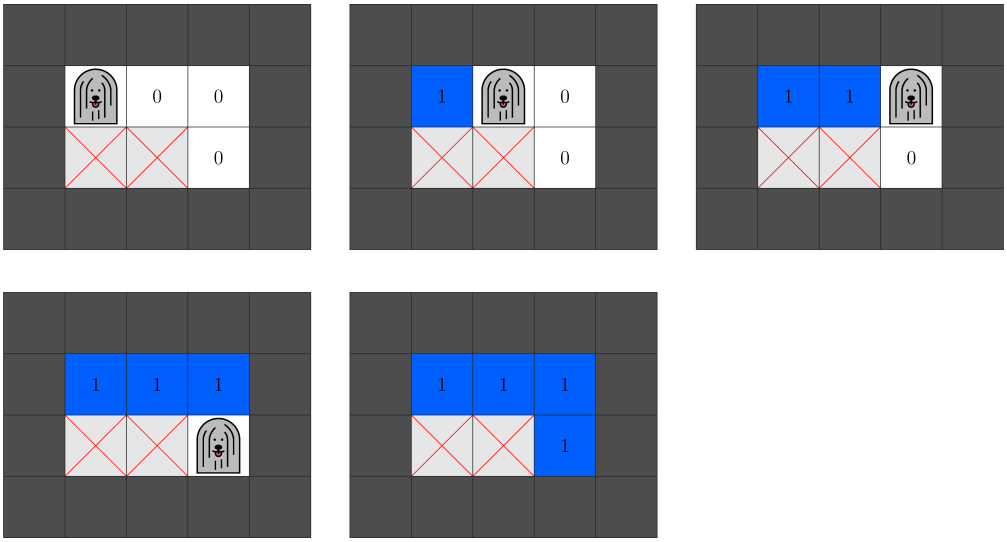
Повик	Инструкции за низа на состојби S
`set_instruction([0, -2, -1, 0, -2], 1, E)`	Постави боја 1 и придвижи се кон исток
`set_instruction([0, 1, -1, 0, -2], 1, E)`	Постави боја 1 и придвижи се кон исток
`set_instruction([0, 1, 0, -2, -2], 1, S)`	Постави боја 1 и придвижи се кон југ
`set_instruction([0, -1, -2, -2, 1], 1, T)`	Постави боја 1 и заврши ја програмата

Разгледајте го сценариото каде што $H = 2$ и $W = 3$, и лавиринтот е даден на следната слика.



За овој лавиринт, програмата на Pulibot работи во 4 чекори. Низите на состојби кои ги препознава Pulibot и соодветните инструкции кои ги извршува одговараат точно на четирите повици на процедурата `set_instruction` направени по горниот распоред. Последната инструкција ја завршува програмата.

Следната слика го покажува лавиринтот пред секој од четирите чекори и финалните бои по завршувањето на програмата.



Меѓутоа, внимавајте дека оваа програма од 4 инструкции не наоѓа најкраток пат за други валидни лавиринти. Затоа, ако се испрати како решение ќе добиете резултат `Output isn't correct`.

Ограничувања

$Z_{MAX} = 19$. Значи, Pulibot може да користи бои од 0 до 19, вклучително.

За секој лавиринт кој може да се користи за тестирање на Pulibot:

- $2 \leq H, W \leq 15$
- Постои барем еден пат од келијата $(0, 0)$ до келијата $(H - 1, W - 1)$.

Подзадачи

1. (6 поени) Нема келии пречки во лавиринтот.
2. (10 поени) $H = 2$
3. (18 поени) Има точно една патека помеѓу секој пар од празни келии.
4. (20 поени) Секој најкраток пат од келијата $(0, 0)$ до келијата $(H - 1, W - 1)$ има должина $H + W - 2$.
5. (46 поени) Без дополнителни ограничувања.

Ако, во било кој од тест случаите, повиците кон процедурата `set_instruction` или програмата на Pulibot за време на нејзиното извршување не одговара на ограничувањата опишани во Имплементациските детали, резултатот на вашето решение за таа подзадача ќе биде 0.

Во секоја подзадача, може да добиете парцијални бодови доколку се добие боење кое е скоро точно.

Формално:

- Решението за некој тест случај е **целосно** ако финалното боење на празните келии ги задоволува Правилата на натпреварот.
- Решението за некој тест случај е **парцијално** ако конечното боење на празните келии изгледа на следниот начин:
  - Постои најкраток пат од келијата $(0, 0)$ до келијата $(H - 1, W - 1)$ за кој бојата на секоја келија вклучена во патот е 1.
  - Нема друга празна келија со боја 1.
  - Некоја друга празна келија во лавиринтот има боја различна од 0 и 1.

Ако вашето решение за некој тест случај не е целосно или парцијално, вашите бодови за тој тест случај ќе бидат 0.

Во подзадачите 1-4, поените за целосно решение се 100%, а поените за секое парцијално решение на тест случај се 50% за таа подзадача.

Во подзачата 5, вашиот резултат зависи од бројот на бои кои ќе бидат искористени во програмата на Pulibot. Попрецизно, ако со $Z^\star$ е означена максималната вредност на Z за сите повици направени кон процедурата `set_instruction`, резултатот за секој тест случај се пресметува согласно следната табела :

Услов	Резултат (целосно)	Резултат (парцијално)
$11 \leq Z^\star \leq 19$	$20 + (19 - Z^\star)$	$12 + (19 - Z^\star)$
$Z^\star = 10$	31	23
$Z^\star = 9$	34	26
$Z^\star = 8$	38	29
$Z^\star = 7$	42	32
$Z^\star \leq 6$	46	36

Поените за секоја подзадача е минимумот на поени од сите тест случаи на таа подзадача.

Пример оценувач

Пример оценувачот чита податоци од влез во следниот формат:

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

Тука, m е низа на H низи во кои има W цели броеви, кои ги опишуваат негранични келии на лавиринтот.. $m[r][c] = 0$ ако келијата (r, c) е празна и $m[r][c] = 1$ ако келијата (r, c) е келија пречка.

Пример оценувачот прво ја повикува процедурата `program_pulibot()`. Доколку пример оценувачот открие грешка во протоколот за комуникација, пример оценувачот печати `Protocol Violation: <MSG>` и прекинува со работа, каде што `<MSG>` е една од следните пораки за грешки:

- `Invalid array`: $-2 \leq S[i] \leq Z_{MAX}$ не е исполнето за некое i или должината на низата S не е 5.
- `Invalid color`: $0 \leq Z \leq Z_{MAX}$ не е исполнето.
- `Invalid action`: знакот A не е еден од можните `H`, `W`, `S`, `E`, `N` или `T`.
- `Same state array`: процедурата `set_instruction` е повикана со истата низа S најмалце два пати.

Во спротивно, кога процедурата `program_pulibot` ќе заврши, пример оценувачот ја извршува добиената програма за Pulibot во лавиринтот опишан во влезот.

Пример оценувачот креира два излези.

Прво, пример оценувачот креира лог на сите акции на Pulibot во датотека `robot.bin` во работниот фолдер. Оваа датотека се користи како влезни податоци во алатката за визуелизација опишана подолу.

Второ, доколку програмата на Pulibot не заврши успешно, пример оценувачот печати една од следните пораки за грешка:

- `Unexpected state`: Pulibot препознал низа на состојби за која не е повикана процедурата `set_instruction`.
- `Invalid move`: направената акција предизвикала Pulibot да се придвижи во келија која не е празна.
- `Too many steps`: Pulibot направил повеќе од $500\,000$ чекори без да ја заврши својата програма.

Во спротивно, нека $e[r][c]$ претставува состојбата на келијата (r, c) по завршување на програмата на Pulibot. Пример оценувачот печати H линии во следниот формат:

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

Алатка за визуелизација

Во прилогот за оваа задача кој може да се преземе од CMS има датотека `display.py`. Кога ќе биде повикана, оваа Python скрипта дава визуелизација на акциите на Pulibot во лавиринтот

опишани од пример оценувачот. За да ова функционира, бинарната датотека `robot.bin` мора да е присутна во работниот фолдер.

За повикување на алатката се користи следната команда.

```
python3 display.py
```

Се појавува едноставен графички интерфејс со следните функционалности:

- Може да ја видите состојбата на лавиринтот. Моменталната локација на Pulibot е означена со квадратче.
- Може да се движите низ чекорите на Pulibot со кликање на копчињата со стрелки (или со притискање на нивните букви) Исто така, можете да одите на некој одреден чекор.
- Следниот чекор во програмата на Pulibot е даден најдолу. Таму е прикажана низата на состојби и инструкцијата која ќе ја изврши роботот. По завршување на последниот чекор, се прикажува некоја од пораките за грешка или пораката `Terminated` ако програмата успешно завршила.
- За секој број кој претставува боја, можете да дефинирате бои, текстуална легенда за боја, како и боја на позадина и боја на текст. Текстуалната легенда ќе се впише на секоја келија со иста боја. Можете да ги сетирате боите на следните начини:
  - Преку дијалог прозорецот со кликање на копчето `Colors`.
  - Едитирање на содржина на датотеката `colors.txt`.
- За да ја превчитате содржината на датотеката `robot.bin`, користете го копчето `Reload`. Ова е практично ако содржината на `robot.bin` е променета.