Roboterwettbewerb

Die KI-Forscher der Universität Szeged veranstalten einen Roboterwettbewerb. Deine Freundin Hanga will daran teilnehmen. Bei dem Wettbewerb soll der *Pulibot* programmiert werden, zu Ehren der für ihre Klugheit berühmten ungarischen Hunderasse Puli.

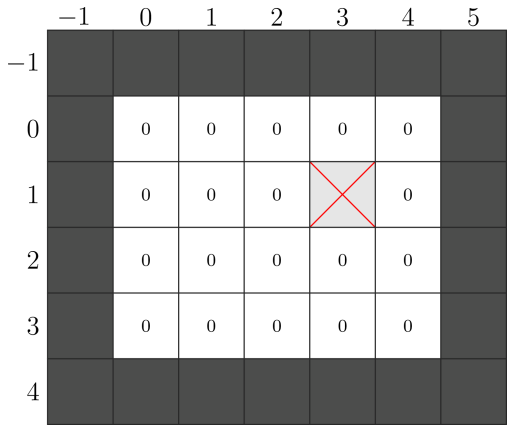
Der Pulibot wird in einem Labyrinth getestet, das aus einem Raster von $(H+2) \times (W+2)$ Feldern besteht. Die Zeilen des Rasters sind von Norden nach Süden mit -1 bis H nummeriert. Die Spalten des Rasters sind von Westen nach Osten mit -1 bis W nummeriert. Das Feld in Zeile r und Spalte c des Rasters ($-1 \leq r \leq H$, $-1 \leq c \leq W$) bezeichnen wir als Feld (r, c).

Wir betrachten ein Feld (r, c) mit $0 \leq r < H$ und $0 \leq c < W$. Das Feld (r, c) hat 4 **benachbarte** Felder:

- Das Feld $(r, c-1)$ ist **westlich** von (r, c).
- Das Feld $(r+1, c)$ ist **südlich** von (r, c).
- Das Feld $(r, c+1)$ ist **östlich** von (r, c).
- Das Feld $(r-1, c)$ ist **nördlich** von (r, c).

(r, c) ist ein **Randfeld** des Labyrinths, wenn $r = -1$ oder $r = H$ oder $c = -1$ oder $c = W$. Jedes Feld, das kein Randfeld ist, ist entweder ein **Hindernis** oder ein **leeres** Feld. Jedes leere Feld hat eine **Farbe**, die als nichtnegative ganze Zahl zwischen 0 and Z_{MAX} angegeben wird. Zu Beginn hat jedes leere Feld die Farbe 0.

Als Beispiel siehst du hier ein Labyrinth mit $H = 4$ und $W = 5$ und einem einzigen Hindernis $(1, 3)$:



Das Hindernis ist mit einem Kreuz markiert, und die Randfelder sind dunkel gefärbt. Die Zahlen in den leeren Feldern geben deren Farben an.

Ein **Pfad** der Länge ℓ ($\ell > 0$) von Feld (r_0, c_0) zu Feld (r_ℓ, c_ℓ) ist eine Folge paarweise verschiedener *leerer* Felder $(r_0, c_0), (r_1, c_1), \ldots, (r_\ell, c_\ell)$, so dass aufeinanderfolgende Felder (r_i, c_i) und (r_{i+1}, c_{i+1}) ($0 \leq i < \ell$) benachbart sind. Ein solcher Pfad besteht also aus $\ell + 1$ Feldern.

Beim Wettbewerb geben die Veranstalter ein Labyrinth vor, bei dem es innerhalb der Umrandung mindestens einen Pfad von der nordwestlichen Ecke (Feld $(0, 0)$) zur südöstlichen Ecke (Feld $(H - 1, W - 1)$) gibt. Diese beiden Felder sind also leer. Wie alle Teilnehmenden weiß Hanga nicht, welche Felder des Labyrinths leer sind und wo möglicherweise Hindernisse liegen.

Hilf Hanga, den Pulibot so zu programmieren, dass er in dem vorgegebenen Labyrinth einen **kürzesten Pfad** (also einen Pfad minimaler Länge) von Feld $(0, 0)$ zu Feld $(H - 1, W - 1)$ finden kann. Im Folgenden findest du Pulibots Spezifikation sowie die Regeln des Roboterwettbewerbs.

Beachte, dass der letzte Abschnitt dieser Aufgabenbeschreibung ein Visualisierungstool erklärt, mit dem du Pulibot visualisieren kannst.

Pulibots Spezifikation

Der **Zustand** eines Feldes (r, c), für jedes $-1 \leq r \leq H$ und $-1 \leq c \leq W$, ist wie folgt definiert:

- Falls das Feld (r, c) ein Randfeld ist, dann ist sein Zustand -2.
- Falls das Feld (r, c) ein Hindernis ist, dann ist sein Zustand -1.
- Falls das Feld (r, c) ein leeres Feld ist, dann ist sein Zustand die Farbe des Feldes.

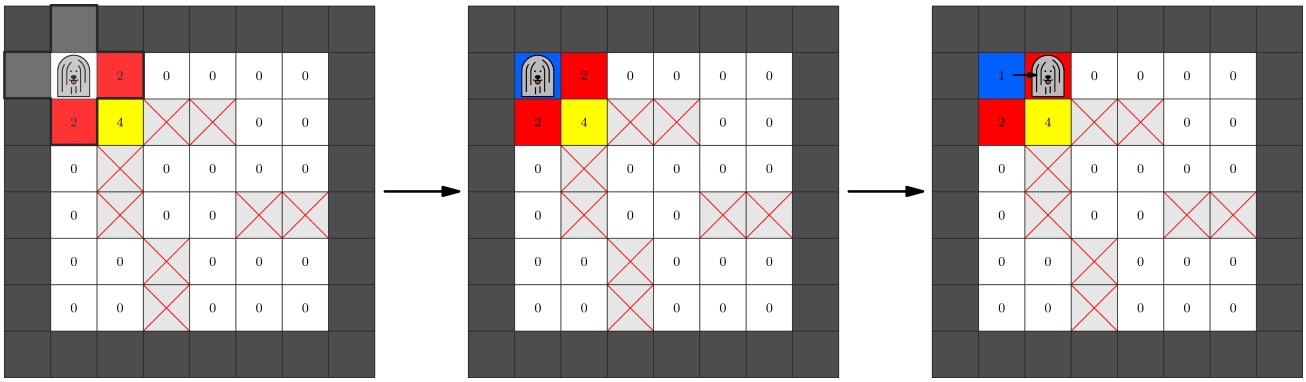
Pulibots Programm wird als Abfolge von Schritten ausgeführt. In jedem Schritt erkennt Pulibot die Zustände seines Feldes und der benachbarten Felder und führt dann eine Anweisung aus. Die auszuführende Anweisung wird durch die erkannten Zustände bestimmt. Eine genaue Beschreibung folgt.

Angenommen, zu Beginn des aktuellen Schrittes befindet sich Pulibot auf dem leeren Feld (r, c). Der Schritt wird wie folgt ausgeführt:

1. Zuerst erkennt Pulibot das aktuelle **Zustandsarray**, also das Array $S = [S[0], S[1], S[2], S[3], S[4]]$, das aus dem Zustand des Feldes (r, c) und den Zuständen aller benachbarter Felder besteht:
    - $S[0]$ ist der Zustand des Feldes (r, c).
    - $S[1]$ ist der Zustand des Feldes im Westen.
    - $S[2]$ ist der Zustand des Feldes im Süden.
    - $S[3]$ ist der Zustand des Feldes im Osten.
    - $S[4]$ ist der Zustand des Feldes im Norden.
2. Dann bestimmt Pulibot die **Anweisung** (Z, A), die dem erkannten Zustandsarray entspricht.
3. Zuletzt führt Pulibot diese Anweisung aus: Er setzt die Farbe des Feldes (r, c) auf die Farbe Z und führt dann die Aktion A aus, die eine der folgenden Aktionen ist:
    - *bleibe* in dem Feld (r, c);
    - *bewege* dich zu einem der 4 benachbarten Felder;

- *beende das Programm*.

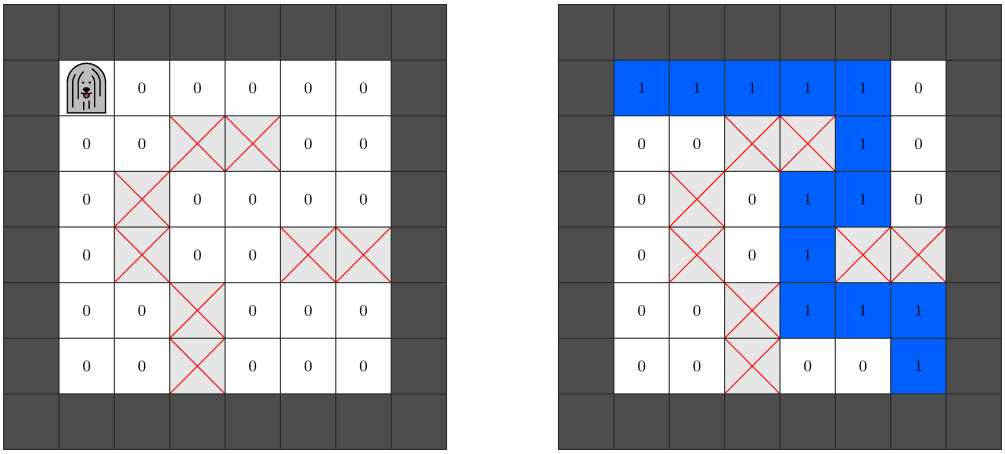
Betrachte als Beispiel das Szenario, das links in der folgenden Abbildung dargestellt ist. Pulibot befindet sich gerade in dem Feld $(0, 0)$ mit der Farbe 0. Pulibot erkennt das Zustandsarray $S = [0, -2, 2, 2, -2]$. Pulibot könnte ein Programm haben, das beim Erkennen dieses Arrays die Farbe des aktuellen Feldes auf $Z = 1$ setzt und sich dann nach Osten bewegt, wie in der Mitte und rechts in der Abbildung dargestellt:



Roboterwettbewerbsregeln

- Zu Beginn wird Pulibot auf dem Feld $(0, 0)$ platziert und beginnt, sein Programm auszuführen.
- Pulibot darf sich nur auf leere Felder bewegen.
- Pulibots Porgramm muss nach höchstens $500\,000$ Schritten beenden.
- Nachdem Pulibots Programm beendet ist, müssen die Farben der leeren Felder wie folgt sein:
  - Es existiert ein kürzester Pfad von $(0, 0)$ nach $(H - 1, W - 1)$, dessen Felder alle die Farbe 1 haben.
  - Alle anderen leeren Felder haben Farbe 0.
- Pulibot darf sein Programm an einem beliebigen leeren Feld beenden.

Als Beispiel zeigt die folgende Abbilung ein mögliches Labyrinth mit $H = W = 6$. Links ist die Ausgangskonfiguration dargestellt und rechts eine korrekte Färbung der leeren Felder nach Beendigung des Programms:



Implementierungsdetails

Implementiere die folgende Funktion:

```
void program_pulibot()
```

- Diese Funktion soll Pulibots Programm erzeugen. Dieses Programm soll für alle Werte von H und W und für jedes Labyrinth, das die Aufgabenbedingungen erfüllt, korrekt funktionieren.
- Diese Funktion wird genau einmal für jeden Testfall aufgerufen.

Die obige Funktion kann die folgende Funktion aufrufen, um Pulibots Programm zu erzeugen:

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

- S: ein Array der Länge 5, welches das Zustandsarray beschreibt.
- Z: eine nichtnegative ganze Zahl, die eine Farbe beschreibt.
- A: ein einzelnes Zeichen, das eine Aktion von Pulibot wie folgt beschreibt:
  - `H`: bleibe in dem derzeitigen Feld;
  - `W`: nach Westen bewegen;
  - `S`: nach Süden bewegen;
  - `E`: nach Osten bewegen;
  - `N`: nach Norden bewegen;
  - `T`: das Programm beenden.
- Ein Aufruf dieser Funktion weist Pulibot an, nach Erkennung des Zustandsarrays S die Anweisung (Z, A) auszuführen.

Falls deine Lösung diese Funktion mehrfach mit dem gleichen Zustandsarray S aufruft, wird deine Lösung mit `Output isn't correct` bewertet.

Es ist nicht nötig, `set_instruction` mit allen möglichen Zustandsarrays S aufzurufen. Falls Pulibot allerdings später ein Zustandsarray erkennt, für das keine Anweisung gesetzt wurde, wird deine Lösung mit `Output isn't correct` bewertet.

Nachdem `program_pulibot` terminiert, ruft der Grader Pulibots Programm für ein oder mehrere Labyrinthe auf. Diese Aufrufe zählen *nicht* zu dem Zeitlimit deiner Lösung. Der Grader ist *nicht* adaptiv, das heißt, dass alle Labyrinthe in jedem Testfall vordefiniert sind.

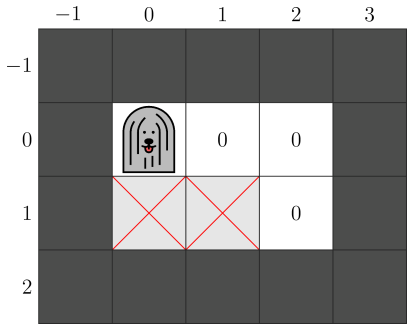
Falls Pulibot gegen eine der Roboterwettbewerbsregeln verstößt, bevor sich sein Programm beendet, wird deine Lösung mit `Output isn't correct` bewertet.

Beispiel

Die Funktion `program_pulibot` kann `set_instruction` wie folgt aufrufen:

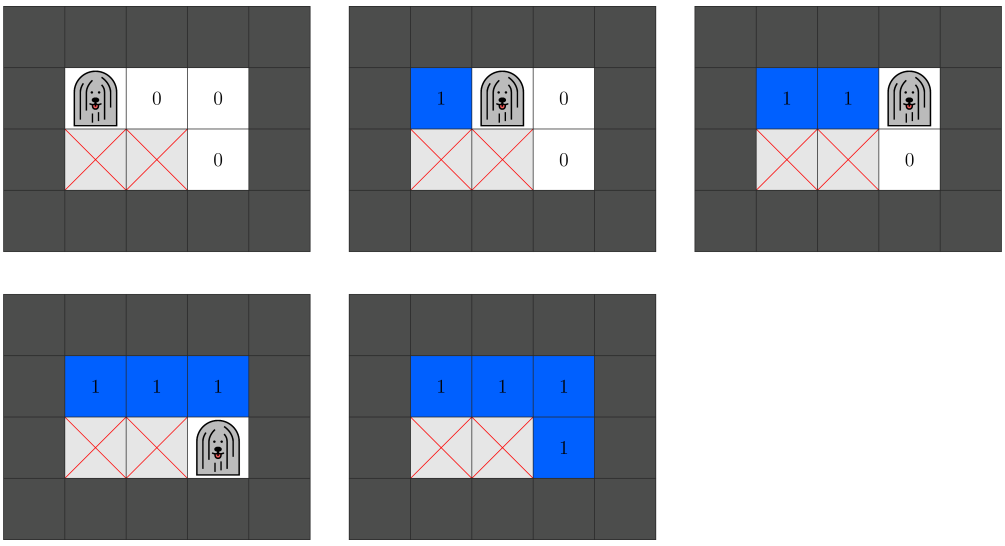
Aufruf	Anweisung für Zustandsarray S
`set_instruction([0, -2, -1, 0, -2], 1, E)`	Farbe auf 1 setzen und nach Osten bewegen
`set_instruction([0, 1, -1, 0, -2], 1, E)`	Farbe auf 1 setzen und nach Osten bewegen
`set_instruction([0, 1, 0, -2, -2], 1, S)`	Farbe auf 1 setzen und nach Süden bewegen
`set_instruction([0, -1, -2, -2, 1], 1, T)`	Farbe auf 1 setzen und das Programm beenden

Betrachte ein Szenario mit $H = 2$ und $W = 3$, bei dem das Labyrinth wie folgt aussieht:



Pulibots Programm führt für dieses Labyrinth vier Schritte aus. Die Zustandsarrays und die ausgeführten Anweisungen entsprechen genau den vier obigen Aufrufen von `set_instruction` in genau dieser Reihenfolge. Die letzte dieser Anweisungen beendet das Programm.

Das folgende Bild zeigt das Labyrinth vor jedem der vier Schritte und die finalen Farben der Felder nach Beendigung des Programms.



Beachte, dass dieses Programm mit 4 Anweisungen in anderen gültigen Labyrinthen keinen kürzesten Pfad finden könnte. Deshalb wird diese Lösung mit `Output isn't correct` bewertet,

falls sie eingesendet wird.

Einschränkungen

$Z_{MAX} = 19$. Daher kann Pulibot Farben von 0 bis 19 verwenden.

Für jedes Labyrinth, mit dem Pulibot getestet wird, gilt:

- $2 \leq H, W \leq 15$
- Es existiert mindestens ein Pfad von Feld $(0, 0)$ zu Feld $(H - 1, W - 1)$.

Teilaufgaben

1. (6 Punkte) Es gibt kein Hindernis im Labyrinth.
2. (10 Punkte) $H = 2$
3. (18 Punkte) Es existiert genau ein Pfad zwischen je zwei leeren Feldern.
4. (20 Punkte) Jeder kürzeste Pfad von Feld $(0, 0)$ zu Feld $(H - 1, W - 1)$ hat Länge $H + W - 2$.
5. (46 Punkte) Keine weiteren Einschränkungen.

Falls in irgendeinem der Testfälle die Aufrufe der Funktion `set_instruction` oder die Ausführung von Pulibots Programm nicht den Vorgaben aus den Implementierungsdetails entsprechen, wird die Punktzahl deiner Lösung für diese Teilaufgabe 0 sein.

In jeder Teilaufgabe kannst du Teilpunkte erhalten, indem du die Farben fast korrekt setzt.

Formal:

- Die Lösung eines Testfalls ist **vollständig**, falls die finalen Farben der leeren Felder die Roboterwettbewerbsregeln einhalten.
- Die Lösung eines Testfalls ist **partiell**, falls die finalen Farben wie folgt sind:
  - Es existiert ein kürzester Pfad von $(0, 0)$ nach $(H - 1, W - 1)$, dessen Felder alle die Farbe 1 haben.
  - Es gibt kein anderes leeres Feld im Raster mit Farbe 1.
  - Mindestens ein leeres Feld im Raster hat eine Farbe, die weder 0 noch 1 ist.

Falls deine Lösung eines Testfalls weder vollständig noch partiell ist, ist deine Punktzahl für den entsprechenden Testfall 0.

In den Teilaufgaben 1-4 beträgt die Punktzahl für eine vollständige Lösung eines Testfalls 100% und die Punktzahl für eine partielle Lösung eines Testfalls 50% der Punkte der entsprechenden Teilaufgabe.

In Teilaufgabe 5 hängt deine Punktzahl von der Anzahl an Farben ab, die in Pulibots Programm genutzt werden. Sei $Z^\star$ der maximale Wert von Z über alle Aufrufe von `set_instruction`. Die Punktzahl des Testfalls wird nach der folgenden Tabelle berechnet:

Bedingung	Punktzahl (vollständig)	Punktzahl (partiell)
$11 \leq Z^\star \leq 19$	$20 + (19 - Z^\star)$	$12 + (19 - Z^\star)$
$Z^\star = 10$	31	23
$Z^\star = 9$	34	26
$Z^\star = 8$	38	29
$Z^\star = 7$	42	32
$Z^\star \leq 6$	46	36

Die Punktzahl für jede Teilaufgabe ist das Minimum der Punkte für die Testfälle in der Teilaufgabe.

Beispielgrader

Der Beispielgrader liest die Eingabe im folgenden Format:

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

Hierbei ist m ein Array bestehend aus H Arrays von W ganzen Zahlen, das die nicht an den Rändern liegenden Felder des Labyrinths beschreibt. Falls das Feld (r, c) ein leeres Feld ist, dann ist $m[r][c] = 0$, und falls das Feld (r, c) ein Hindernis ist, dann ist $m[r][c] = 1$.

Der Beispielgrader ruft zuerst `program_pulibot()` auf. Falls der Beispielgrader einen fehlerhaften Aufruf von `set_instruction` erkennt, gibt er `Protocol Violation: <MSG>` aus, wobei `<MSG>` eine der folgenden Fehlermeldungen ist:

- `Invalid array`: $-2 \leq S[i] \leq Z_{MAX}$ wird für mindestens ein i nicht erfüllt oder die Länge von S ist nicht 5.
- `Invalid color`: $0 \leq Z \leq Z_{MAX}$ wird nicht erfüllt.
- `Invalid action`: Das Zeichen A ist nicht eines von `H`, `W`, `S`, `E`, `N` oder `T`.
- `Same state array`: `set_instruction` wurde mindestens zweimal mit demselben Array S aufgerufen.

Andernfalls, wenn `program_pulibot` terminiert, führt der Beispielgrader Pulibots Programm im durch die Eingabe beschriebenen Labyrinth aus.

Der Beispielgrader erzeugt zwei Ausgaben.

Zuerst schreibt der Beispielgrader ein Protokoll von Pulibots Aktionen in die Datei `robot.bin` im Arbeitsverzeichnis. Diese Datei dient als Eingabe für das Visualisierungstool, das im folgenden Abschnitt beschrieben wird.

Zweitens, falls Pulibots Programm nicht erfolgreich beendet wird, gibt der Beispielgrader eine der folgenden Fehlermeldungen aus:

- `Unexpected state`: Pulibot hat ein Zustandsarray erkannt, für das `set_instruction` nicht aufgerufen wurde.
- `Invalid move`: Die Ausführung einer Aktion führte dazu, dass sich Pulibot auf ein nicht leeres Feld bewegt hat.
- `Too many steps`: Pulibot hat $500\,000$ Schritte ausgeführt, ohne sein Programm zu beenden.

Andernfalls sei $e[r][c]$ der Zustand des Feldes (r, c) nachdem Pulibots Programm beendet wurde. Der Beispielgrader gibt H Zeilen im folgenden Format aus:

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

Visualisierungstool

Das beigefügte Archiv im Anhang zu dieser Aufgabe enthält eine Datei mit dem Namen `display.py`. Wenn diese Python-Skriptdatei ausgeführt wird, zeigt sie Pulibots Aktionen im Labyrinth an, das durch die Eingabe des Beispielgraders beschrieben wird. Dafür muss die Binärdatei `robot.bin` im Arbeitsverzeichnis vorhanden sein.

Um das Skript auszuführen, führe den folgenden Befehl aus:

```
python3 display.py
```

Eine einfache grafische Benutzeroberfläche wird angezeigt. Die Hauptfunktionen sind wie folgt:

- Du kannst den Zustand des gesamten Labyrinths beobachten. Die aktuelle Position von Pulibot wird durch ein Rechteck hervorgehoben.
- Du kannst durch die Schritte von Pulibots Programm gehen, indem du die Pfeiltasten anklickst oder die dazugehörige Taste drückst. Du kannst auch zu einem bestimmten Schritt springen.
- Der nächste Schritt in Pulibots Programm wird unten angezeigt. Es zeigt das aktuelle Zustandsarray und die auszuführende Anweisung an. Nach dem letzten Schritt wird entweder eine der Fehlermeldungen des Graders oder `Terminated` angezeigt, falls das Programm erfolgreich beendet wurde.
- Du kannst jedem Zahlwert, der eine Farbe darstellt, eine Hintergrundfarbe zuweisen als auch einen Anzeigetext. Der Anzeigetext ist eine kurze Zeichenkette, die in jedem Feld mit dieser Farbe erscheint. Du kannst Hintergrundfarben und Anzeigetexte auf eine der folgenden Arten zuweisen:
  - Setze sie in einem Dialogfenster fest, nachdem du auf die Schaltfläche `Colors` drückst.
  - Bearbeite den Inhalt der Datei `colors.txt`.
- Um `robot.bin` neu zu laden, verwende die Schaltfläche `Reload`. Das ist nützlich, wenn sich der Inhalt von `robot.bin` geändert hat.