Kontes Robot

Peneliti AI di University of Szeged sedang mengadakan sebuah kontes pemrograman robot. Teman Anda, Hanga, memutuskan untuk ikut serta dalam kontes tersebut. Tujuannya adalah untuk memprogram *Pulibot* yang terbaik, dalam rangka mengagumi kecerdasan ras anjing penggembala Hungaria yang terkenal, Puli.

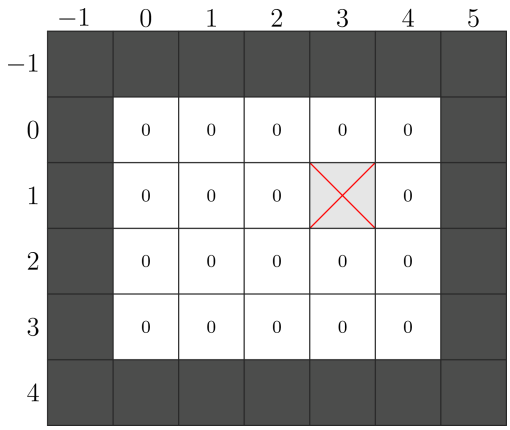
Pulibot akan diuji pada sebuah labirin yang tersusun atas petak *grid* $(H+2) \times (W+2)$. Baris-baris pada *grid* dinomori dari -1 sampai H dari utara ke selatan dan kolom-kolom pada *grid* dinomori dari -1 sampai W dari barat ke timur. Kita rujuk petak yang berada di baris r dan kolom c pada *grid* ($-1 \le r \le H$, $-1 \le c \le W$) sebagai petak (r, c).

Perhatikan petak (r, c) sedemikian sehingga $0 \le r < H$ dan $0 \le c < W$. Terdapat 4 petak yang **bersebelahan** dengan petak (r, c):

- petak $(r, c-1)$ disebut sebagai petak **barat** dari petak (r, c);
- petak $(r+1, c)$ disebut sebagai petak **selatan** dari petak (r, c);
- petak $(r, c+1)$ disebut sebagai petak **timur** dari petak (r, c);
- petak $(r-1, c)$ disebut sebagai petak **utara** dari petak (r, c).

Petak (r, c) disebut sebagai sebuah petak **pembatas** pada labirin tersebut jika $r = -1$ atau $r = H$ atau $c = -1$ atau $c = W$. Setiap petak yang bukan merupakan petak pembatas pada labirin tersebut adalah petak **penghalang** atau petak **kosong**. Sebagai tambahan, setiap petak kosong memiliki sebuah **warna**, yang dinyatakan dengan sebuah bilangan bulat nonnegatif antara 0 dan Z_{MAX}, inklusif. Awalnya, warna setiap petak kosong adalah 0.

Sebagai contoh, perhatikan sebuah labirin dengan $H = 4$ dan $W = 5$, yang mengandung satu petak penghalang $(1, 3)$:



Satu-satunya petak penghalang dilambangkan dengan sebuah tanda silang. Petak pembatas pada labirin diarsir. Angka yang tertulis pada setiap petak kosong menyatakan warnanya.

Sebuah **jalan** dengan panjang ℓ ($\ell > 0$) dari petak (r_0, c_0) ke petak (r_ℓ, c_ℓ) adalah sebuah barisan petak-petak *kosong* $(r_0, c_0), (r_1, c_1), \ldots, (r_\ell, c_\ell)$ dan untuk setiap i ($0 \leq i < \ell$) petak (r_i, c_i) dan (r_{i+1}, c_{i+1}) bersebelahan.

Perhatikan bahwa jalan dengan panjang ℓ mengandung tepat $\ell + 1$ petak.

Pada kontes tersebut, para peneliti mempersiapkan sebuah labirin sehingga terdapat setidaknya satu jalan dari petak $(0, 0)$ ke petak $(H - 1, W - 1)$. Perhatikan bahwa hal ini berarti petak $(0, 0)$ dan $(H - 1, W - 1)$ dijamin kosong.

Hanga tidak tahu petak-petak mana saja dari labirin tersebut yang kosong maupun petak penghalang.

Tugas Anda adalah untuk membantu Hanga dalam memprogram Pulibot yang mampu mencari *jalan terpendek* (yaitu, jalan dengan panjang minimum) dari petak $(0, 0)$ ke petak $(H - 1, W - 1)$ dalam sebuah labirin yang telah dibuat oleh para peneliti. Spesifikasi dari Pulibot dan aturan-aturan pada kontes tersebut dijelaskan di bawah ini.

Perhatikan bahwa bagian terakhir dari soal menjelaskan cara untuk Anda menggunakan bantuan visual untuk memvisualisasikan Pulibot.

Spesifikasi Pulibot

Definisikan **status** dari sebuah petak (r, c) untuk setiap $-1 \leq r \leq H$ dan $-1 \leq c \leq W$ sebagai sebuah bilangan bulat sehingga:

- jika petak (r, c) adalah sebuah petak pembatas maka statusnya adalah -2;
- jika petak (r, c) adalah sebuah petak penghalang maka statusnya adalah -1;
- jika petak (r, c) adalah sebuah petak kosong maka statusnya adalah warna dari petak tersebut.

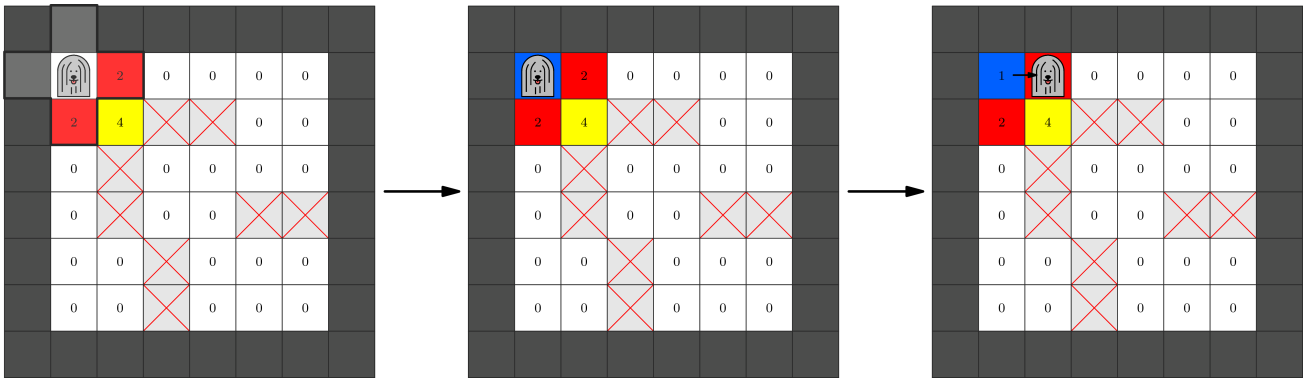
Program Pulibot dijalankan sebagai sebuah barisan langkah. Dalam setiap langkah, Pulibot mengetahui status petak-petak di dekatnya, lalu melakukan sebuah instruksi. Instruksi yang dia lakukan ditentukan berdasarkan status-status tersebut. Berikut penjelasan yang lebih rinci.

Misalkan pada awal dari langkah ini, Pulibot berada di petak (r, c), yang merupakan sebuah petak kosong. Langkah ini dilakukan sebagai berikut:

1. Pertama, Pulibot mengetahui **status *array*** saat ini, yaitu *array* $S = [S[0], S[1], S[2], S[3], S[4]]$, yang terdiri dari status petak (r, c) dan semua petak-petak yang bersebelahan:
    - $S[0]$ adalah status dari petak (r, c).
    - $S[1]$ adalah status dari petak barat.

    - $S[2]$ adalah status dari petak selatan.
    - $S[3]$ adalah status dari petak timur.
    - $S[4]$ adalah status dari petak utara.
2. Kemudian, Pulibot menentukan **instruksi** (Z, A) berdasarkan status *array* yang diketahui.
3. Terakhir, Pulibot menjalankan instruksi tersebut: dia mewarnai petak (r, c) dengan warna Z dan kemudian menjalankan instruksi A, yang merupakan salah satu dari aksi-aksi berikut:
    - *menetap* di petak (r, c);
    - *bergerak* ke salah satu dari 4 petak yang berdekatan;
    - *mengakhiri program*.

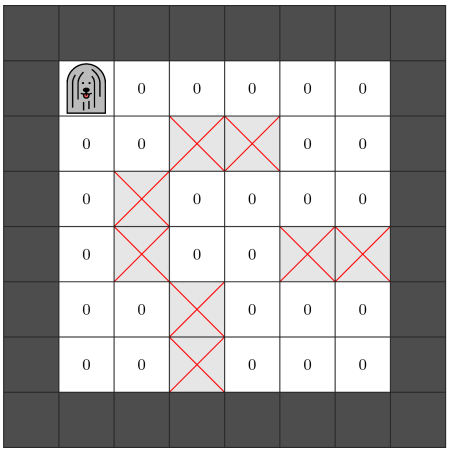
Sebagai contoh, perhatikan skenario yang ditampilkan di sebelah kiri gambar berikut. Pulibot saat ini berada di petak $(0, 0)$ dengan warna 0. Pulibot mengetahui status *array* $S = [0, -2, 2, 2, -2]$. Pulibot memiliki sebuah program yang, setelah mengetahui *array* ini, mewarnai petak sekarang dengan $Z = 1$ dan kemudian bergerak ke timur, seperti yang ditampilkan di tengah dan di sebelah kanan gambar berikut:

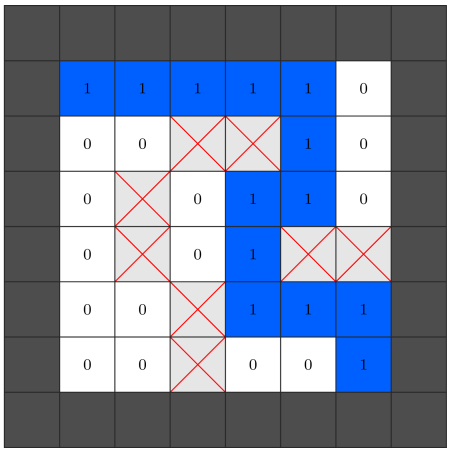


Peraturan Kontes Robot

- Pada awalnya, Pulibot ditempatkan pada petak $(0, 0)$ dan mulai menjalankan programnya.
- Pulibot dilarang untuk bergerak ke petak yang bukan merupakan petak kosong.
- Program Pulibot harus diberhentikan setelah paling banyak $500\,000$ langkah.
- Setelah program Pulibot berhenti, petak-petak kosong di labirin harus diwarnai sedemikian sehingga:
    - Terdapat jalan terpendek dari $(0, 0)$ sampai $(H - 1, W - 1)$ sehingga warna setiap petak yang termasuk dalam jalan tersebut adalah 1.
    - Warna dari setiap petak kosong lainnya adalah 0.
- Pulibot dapat menghentikan programnya di petak kosong manapun.

Sebagai contoh, gambar berikut ini menunjukkan labirin dengan $H = W = 6$. Konfigurasi awal ditampilkan di sebelah kiri dan pewarnaan yang bisa diterima untuk petak-petak kosong setelah penghentian ditampilkan di sebelah kanan:





Detail Implementasi

Anda harus mengimplementasikan prosedur berikut.

```
void program_pulibot()
```

- Prosedur ini harus membuat program Pulibot. Program ini harus bekerja dengan benar untuk semua nilai H dan W dan labirin apa pun yang memenuhi batasan soal.
- Prosedur ini dipanggil tepat sekali untuk setiap kasus uji.

Prosedur ini dapat melakukan panggilan-panggilan prosedur berikut dalam pembuatan program Pulibot:

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

- S: *array* sepanjang 5 yang mendeskripsikan sebuah status *array*.
- Z: sebuah bilangan bulat nonnegatif yang menyatakan sebuah warna.
- A: sebuah karakter yang mewakili aksi Pulibot sebagai berikut:
  - `H`: menetap;
  - `W`: bergerak ke barat;
  - `S`: bergerak ke selatan;
  - `E`: bergerak ke timur;
  - `N`: bergerak ke utara;
  - `T`: memberhentikan program.
- Memanggil prosedur ini menginstruksikan Pulibot bahwa setelah mengetahui status *array* S, dia harus menjalankan instruksi (Z, A).

Memanggil prosedur ini berkali-kali dengan status *array* S yang sama akan mendapatkan *verdict* `Output isn't correct`.

Anda tidak diharuskan untuk memanggil `set_instruction` untuk setiap kemungkinan status array S yang mungkin. Akan tetapi, jika Pulibot nantinya mengetahui status yang tidak ada

instruksinya, Anda akan mendapatkan verdict `Output isn't correct`.

Setelah `program_pulibot` selesai, *grader* akan memanggil program Pulibot dengan satu atau beberapa labirin. Panggilan-panggilan ini *tidak* dihitung terhadap batasan waktu untuk solusi Anda. *Grader tidak* bersifat adaptif, yaitu labirin-labirin sudah ditentukan sebelumnya di setiap kasus uji.

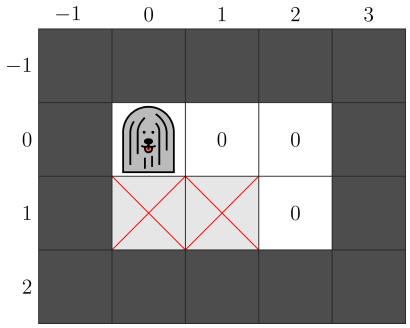
Jika Pulibot melanggar salah satu Peraturan Kontes Robot sebelum mengakhiri programnya, Anda akan mendapatkan *verdict* `Output isn't correct`.

Contoh

Prosedur `program_pulibot` dapat melakukan beberapa panggilan pada `set_instruction` sebagai berikut:

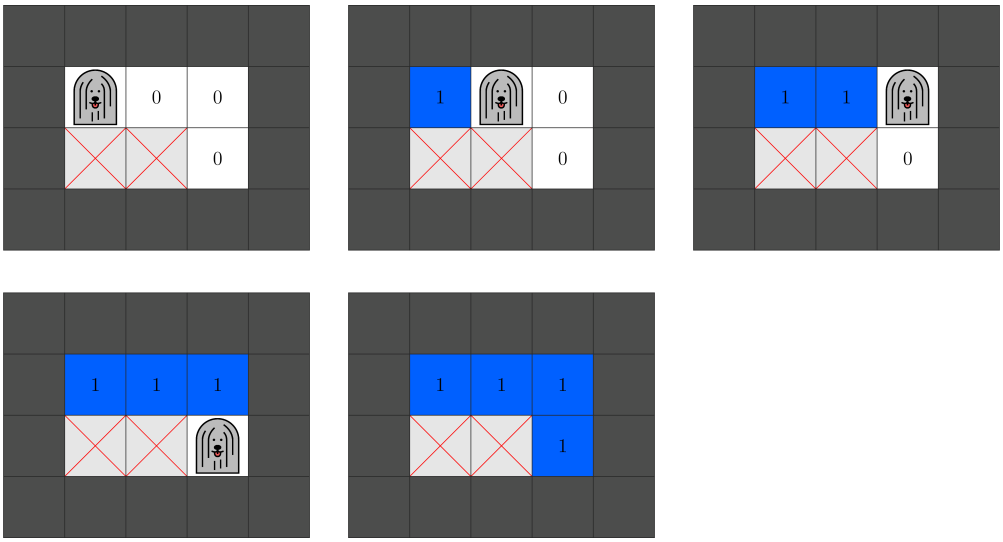
Panggilan	Instruksi untuk status *array* S
`set_instruction([0, -2, -1, 0, -2], 1, E)`	Warnai dengan 1 dan bergerak ke timur
`set_instruction([0, 1, -1, 0, -2], 1, E)`	Warnai dengan 1 dan bergerak ke timur
`set_instruction([0, 1, 0, -2, -2], 1, S)`	Warnai dengan 1 dan bergerak ke selatan
`set_instruction([0, -1, -2, -2, 1], 1, T)`	Warnai dengan 1 dan mengakhiri program

Perhatikan sebuah skenario di mana $H = 2$ dan $W = 3$, dan labirin ditampilkan pada gambar berikut.



Program Pulibot berjalan dalam empat langkah untuk labirin ini. Status *array* yang diketahui Pulibot dan instruksi yang dijalankannya sesuai dengan keempat panggilan pada `set_instruction` di atas, dengan urutan yang sama. Instruksi terakhir mengakhiri program.

Gambar berikut menunjukkan labirin sebelum masing-masing dari empat langkah dan warna setelah penghentian.



Akan tetapi, perlu diperhatikan bahwa program dengan 4 instruksi ini mungkin tidak dapat menemukan jalur terpendek di labirin lain yang valid. Oleh karena itu, jika dikumpulkan, akan mendapatkan *verdict* `Output isn't correct`.

Batasan

$Z_{MAX} = 19$. Maka Pulibot dapat menggunakan warna dari 0 sampai 19, inklusif.

Untuk setiap labirin yang diujikan kepada Pulibot:

- $2 \leq H, W \leq 15$
- Terdapat setidaknya satu buah jalan dari petak $(0, 0)$ ke petak $(H - 1, W - 1)$.

Subsoal

1. (6 poin) Tidak terdapat petak penghalang di labirin.
2. (10 poin) $H = 2$
3. (18 poin) Terdapat tepat satu jalan untuk setiap pasang petak kosong.
4. (20 poin) Jalan terpendek dari petak $(0, 0)$ ke petak $(H - 1, W - 1)$ memiliki panjang $H + W - 2$.
5. (46 poin) Tidak ada batasan tambahan.

Jika, untuk kasus uji mana pun, panggilan pada prosedur `set_instruction` atau program Pulibot selama eksekusinya tidak sesuai dengan batasan yang dijelaskan di Detail Implementasi, nilai dari solusi Anda untuk subsoal tersebut adalah 0.

Pada setiap subsoal, Anda dapat memperoleh nilai parsial dengan menghasilkan pewarnaan yang hampir benar.

Secara formal:

- Solusi dari sebuah kasus uji dikatakan **sempurna** apabila pewarnaan akhir petak kosong memenuhi Peraturan Kontes Robot.
- Solusi dari sebuah kasus uji dikatakan **parsial** apabila pewarnaan akhir petak kosong adalah sebagai berikut:
    - Terdapat jalan terpendek dari $(0, 0)$ ke $(H - 1, W - 1)$ sehingga setiap petak yang termasuk di dalam jalan tersebut adalah 1.
    - Tidak ada petak kosong lain yang diwarnai 1.
    - Beberapa petak kosong dalam *grid* diwarnai dengan warna selain 0 dan 1.

Jika solusi Anda dari sebuah kasus uji tidaklah sempurna maupun parsial, maka nilai Anda untuk kasus uji tersebut adalah 0.

Pada subsoal 1-4, nilai dari solusi sempurna adalah 100% dan nilai dari solusi parsial adalah 50% dari poin subtask tersebut.

Pada subsoal 5, nilai Anda bergantung pada banyaknya warna yang digunakan dalam program Pulibot. Lebih tepatnya, notasikan $Z^\star$ sebagai nilai maksimum dari Z dari semua panggilan yang dilakukan ke `set_instruction`. Nilai untuk kasus uji tersebut dihitung berdasarkan tabel berikut:

Kondisi	Nilai (sempurna)	Nilai (parsial)
$11 \le Z^\star \le 19$	$20 + (19 - Z^\star)$	$12 + (19 - Z^\star)$
$Z^\star = 10$	31	23
$Z^\star = 9$	34	26
$Z^\star = 8$	38	29
$Z^\star = 7$	42	32
$Z^\star \le 6$	46	36

Nilai untuk setiap subsoal adalah nilai minimum dari poin-poin yang didapatkan untuk seluruh kasus uji dalam subsoal tersebut.

Contoh *Grader*

Contoh *grader* membaca masukan dengan format berikut:

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

Di sini, m adalah *array* dengan H *array* dengan W bilangan bulat yang mendeskripsikan petak yang bukan pembatas dari labirin. $m[r][c] = 0$ jika petak $(r, c) adalah sebuah petak kosong dan$ m[r][c] = 1 $jika petak$ (r, c) adalah sebuah petak penghalang.

Contoh *grader* pertama-tama memanggil `program_pulibot()`. Jika contoh *grader* mendeteksi adanya pelanggaran protokol, contoh *grader* akan mencetak `Protocol Violation: <MSG>` dan berhenti, dengan `<MSG>` adalah salah satu dari pesan *error* berikut:

- `Invalid array`: $-2 \leq S[i] \leq Z_{MAX}$ tidak terpenuhi untuk suatu i atau panjang dari S bukanlah 5.
- `Invalid color`: $0 \leq Z \leq Z_{MAX}$ tidak terpenuhi.
- `Invalid action`: karakter A bukan merupakan salah satu dari `H`, `W`, `S`, `E`, `N` maupun `T`.
- `Same state array`: `set_instruction` dipanggil dengan *array* S yang sama setidaknya dua kali.

Sebaliknya, ketika `program_pulibot` selesai, contoh *grader* akan menjalankan program Pulibot dalam labirin yang dijelaskan oleh input.

Contoh grader menghasilkan dua keluaran.

Pertama, contoh *grader* mencetak catatan dari tindakan-tindakan Pulibot ke berkas `robot.bin` di direktori kerja. Berkas ini berfungsi sebagai masukan dari alat visualisasi yang akan dijelaskan di bagian selanjutnya.

Kedua, jika program Pulibot tidak berhasil diberhentikan, contoh *grader* akan mencetak salah satu dari pesan *error* berikut:

- `Unexpected state`: Pulibot mengetahui sebuah status *array* yang tidak dipanggil dengan `set_instruction`.
- `Invalid move`: melakukan sebuah aksi yang menggerakan Pulibot ke petak yang bukan merupakan petak kosong.
- `Too many steps`: Pulibot melakukan $500\,000$ gerakan tanpa mengakhiri programnya.

Sebaliknya, misalkan $e[r][c]$ adalah status dari petak (r, c) setelah program Pulibot berhenti. Contoh grader mencetak H baris dengan format berikut:

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

Bantuan Visual

Paket lampiran untuk soal ini memuat sebuah berkas `display.py`. Ketika dipanggil, *script* Python ini akan menampilkan aksi-aksi Pulibot di dalam labirin yang dideskripsikan pada masukan dari contoh *grader*. Berkas biner `robot.bin` akan muncul di direktori kerja.

Untuk memakai *script*, jalankan perintah berikut.

```
python3 display.py
```

Sebuah grafis antarmuka sederhana muncul. Fitur utamanya adalah sebagai berikut:

- Anda bisa mengamati status dari labirin secara penuh. Lokasi sekarang dari Pulibot disorot dengan sebuah persegi panjang.
- Anda dapat menelusuri langkah-langkah Pulibot dengan mengeklik tombol panah atau menekan tombol pintas mereka. Anda juga bisa melompat ke suatu langkah tertentu.
- Langkah mendatang dari program Pulibot ditampilkan di bagian bawah. Dia menampilkan status *array* sekarang dan instruksi yang akan dia lakukan. Setelah langkah terakhir, dia menampilkan satu atau lebih pesan *error* dari *grader*, atau `Terminated` juka program berhenti dengan sempurna.
- Untuk setiap angka yang menyatakan warna, Anda bisa memberikan sebuah warna latar visual, dan juga teks tampilan. Teks tampilah adalah sebuah *string* pendek yang akan tertulis pada setiap petak yang mempunyai warna yang sama. Anda bisa memberikan warna latar dan teks tampilan dengan salah satu dari cara berikut:
    - Atur dalam jendela dialog setelah mengeklik tombol `Colors`.
    - Mengubah isi dari berkas `colors.txt`.
- Untuk memuat ulang `robot.bin`, gunakan tombol `Reload`. Ini berguna apabila isi dari `robot.bin` telah berubah.