Robot Yarışması

Szeged Üniversitesi'ndeki yapay zeka araştırmacıları bir robot programlama yarışması düzenliyor. Arkadaşınız Hanga yarışmaya katılmaya karar verdi. Amaç, ünlü Macar çoban köpeği Puli'nin muhteşem zekasından esinlenerek nihai *Pulibot*'u programlamaktır.

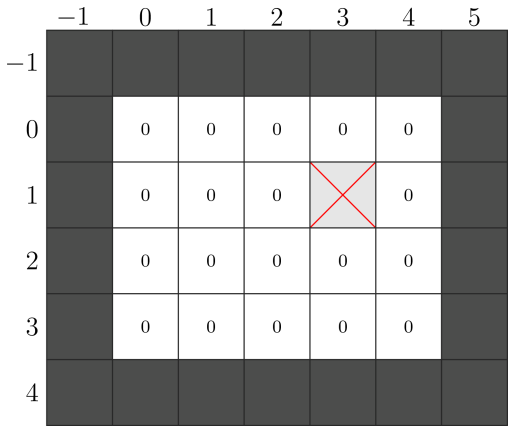
Pulibot, $(H+2) \times (W+2)$ hücre ızgarasından oluşan bir labirent üzerinde test edilecektir. Izgaranın satırları kuzeyden güneye -1'den H'ye, sütunları ise batıdan doğuya -1'den W'ye kadar numaralandırılmıştır. Izgaranın r satırında ve c sütununda bulunan hücreyi ($-1 \leq r \leq H$, $-1 \leq c \leq W$) (r, c) hücresi olarak adlandırıyoruz.

$0 \leq r < H$ ve $0 \leq c < W$ şartlarını sağlayan bir (r, c) hücresini ele alalım. Bu hücreye **komşu** 4 adet hücre vardır:

- $(r, c-1)$ hücresine (r, c) hücresinin **batısındaki** hücre denir;
- $(r+1, c)$ hücresine (r, c) hücresinin **güneyindeki** hücre denir;
- $(r, c+1)$ hücresi, (r, c) hücresinin **doğusundaki** hücre denir;
- $(r-1, c)$ hücresine (r, c) hücresinin **kuzeyindeki** hücre denir.

$r = -1$, $r = H$, $c = -1$, ve $c = W$ şartlarından en az bir tanesini sağlayan bir (r, c) hücresine labirentin **sınır** hücresi denir. Labirentin sınır hücresi olmayan her hücre ya bir **engel** hücresi ya da **boş** bir hücredir. Ek olarak, her boş hücrenin, 0 ile Z_{MAX} (dahil) arasında negatif olmayan bir tam sayıyla temsil edilen bir **rengi** vardır. Başlangıçta her boş hücrenin rengi 0'dır.

Örnek olarak $H = 4$ ve $W = 5$ olan, tek bir $(1, 3)$ engel hücresi içeren bir labirent aşağıdaki resimde verilmiştir:



Resimde tek engel hücresi çarpı işaretiyle gösterilmiştir. Labirentin sınır hücreleri gölgelidir. Her boş hücreye yazılan sayı o hücrenin rengini temsil eder.

(r_0, c_0) hücresinden (r_ℓ, c_ℓ) hücresine kadar ℓ ($\ell > 0$) uzunluğunda bir **patika** $(r_0, c_0), (r_1, c_1), \ldots, (r_\ell, c_\ell)$ olarak isimlendirdiğimiz birbirinden farklı *boş* hücreler serisidir ve burada her i ($0 \leq i < \ell$) için (r_i, c_i) ve (r_{i+1}, c_{i+1}) hücreleri komşudur.

ℓ uzunluğundaki bir yolun tam olarak $\ell + 1$ hücre içerdiğini unutmayın.

Yarışmada araştırmacılar, $(0, 0)$ hücresinden $(H - 1, W - 1)$ hücresine en az bir patikanın bulunduğu bir labirent kurdular. Bunun, $(0, 0)$ ve $(H - 1, W - 1)$ hücrelerinin boş olduğunun garanti edildiği anlamına geldiğini unutmayın.

Hanga, labirentteki hangi hücrelerin boş, hangi hücrelerin engel olduğunu bilmiyor.

Göreviniz Hanga'nın Pulibot'u bilinmeyen bir labirentte $(0, 0)$ hücresinden $(H - 1, W - 1)$ hücresine *en kısa patikayı* (yani minimum uzunluktaki patika) bulabilecek şekilde programlamasına yardımcı olmaktır. Pulibot'un özellikleri ve yarışma kuralları aşağıda açıklanmıştır.

Bu soru metninin son bölümünde Pulibot'u görselleştirmek için kullanabileceğiniz bir görüntüleme aracı açıklanmaktadır.

Pulibot'un Özellikler

$-1 \leq r \leq H$ ve $-1 \leq c \leq W$ şartlarını sağlayan bir (r, c) hücresinin **durumunu** aşağıda belirtilen şekilde bir tam sayı olarak tanımlayın.

- eğer (r, c) hücresi bir sınır hücresi ise, durumu -2 olur;
- eğer (r, c) hücresi bir engel hücresi ise, durumu -1 olur;
- (r, c) hücresi boş bir hücreyse durumu hücrenin rengidir.

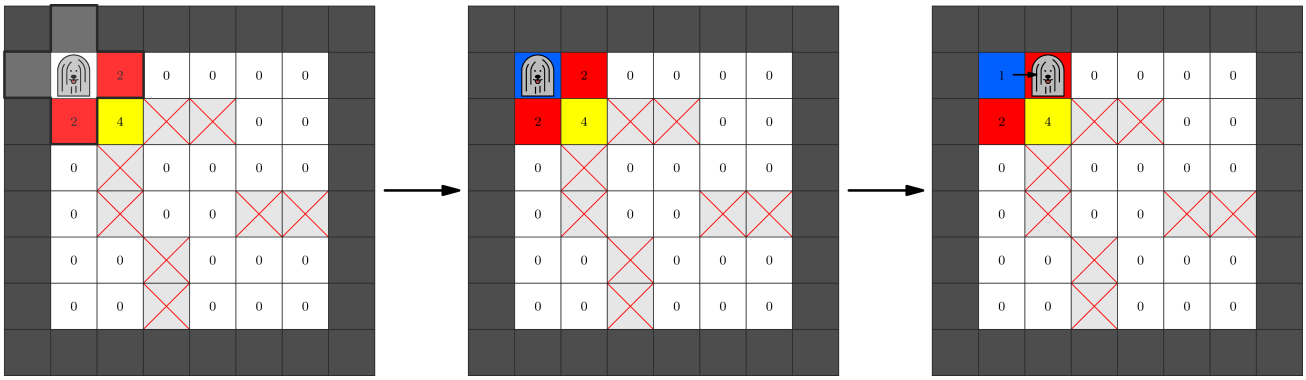
Pulibot'un programı bir dizi adımla yürütülür. Pulibot her adımda komşu hücrelerin durumlarını tanır ve ardından bir komut gerçekleştiririr. Gerçekleştirdiği komut tanınan durumlar tarafından belirlenir. Daha açık bir açıklama aşağıdadır.

Geçerli adımın başında Pulibot'un boş bir hücre olan (r, c) hücresinde olduğunu varsayalım. Adım şu şekilde gerçekleştirilir:

1. İlk olarak, Pulibot mevcut **durum dizisini**, yani (r, c) hücresinin ve tüm komşu hücrelerin durumundan oluşan $S = [S[0], S[1], S[2], S[3], S[4]]$ dizisini tanır.
    - $S[0]$, (r, c) hücresinin durumudur.
    - $S[1]$ (r, c)'nin batısındaki hücrenin durumudur.
    - $S[2]$ (r, c)'nin güneyindeki hücrenin durumudur.
    - $S[3]$ (r, c)'nin doğusundaki hücrenin durumudur.
    - $S[4]$ (r, c)'nin kuzeyindeki hücrenin durumudur.
2. Daha sonra Pulibot, tanınan durum dizisine karşılık gelen (Z, A) **komutunu** belirler.
3. Son olarak Pulibot (Z, A) komutunu şu şekilde yerine getirir: (r, c) hücresinin rengini Z rengine ayarlar ve ardından aşağıdaki eylemlerden biri olan A eylemini gerçekleştirir:

- (r, c) *hücresinde kal*;
- 4 *komşu hücreden birine git*;
- *programı sonlandır*.

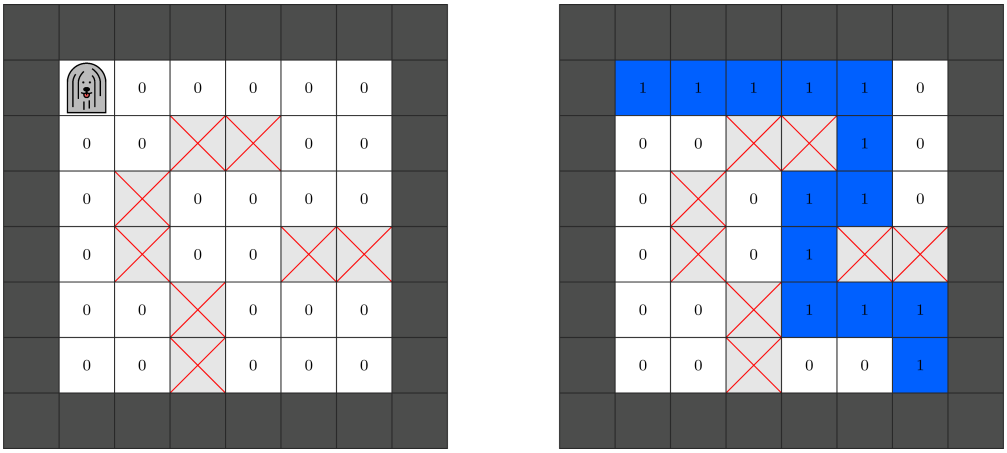
Örneğin, aşağıdaki şeklin solunda gösterilen senaryoyu düşünün. Pulibot şu anda 0 rengiyle $(0, 0)$ hücresindedir. Pulibot, $S = [0, -2, 2, 2, -2]$ durum dizisini tanır. Pulibot, bu diziyi tanıdığında mevcut hücrenin rengini $Z = 1$ olarak ayarlayan ve ardından doğuya hareket eden bir programa sahip olabilir. Şeklin ortasında ve sağında bu senaryo gösterilmektedir.



Robot Yarışması Kuralları

- Başlangıçta Pulibot $(0, 0)$ hücresine yerleştirilir ve program çalıştırılmaya başlanır.
- Pulibot'un boş olmayan bir hücreye gitmesine izin verilmez.
- Pulibot'un programı en fazla $500\,000$ adımdan sonra sonlandırılmalıdır.
- Pulibot programının sonlandırılmasından sonra labirentteki boş hücreler şu şekilde renklendirilmelidir:
  - $(0, 0)$ ile $(H - 1, W - 1)$ arasında en kısa patikalardan birinin üzerindeki bütün hücrelerin rengi 1'dir.
  - Diğer tüm boş hücrelerin rengi 0'dır.
- Pulibot herhangi bir boş hücrede programını sonlandırabilir.

Örneğin, aşağıdaki şekilde $H = W = 6$ için olası bir labirent gösterilmektedir. Başlangıç konfigürasyonu solda, programın çalışması bitince boş hücreler için kabul edilebilir bir renklendirme sağda gösterilmiştir.



Kodlama Detayları

Aşağıdaki prosedürü kodlamalısınız.

```
void program_pulibot()
```

- Bu prosedür Pulibot'un programını oluşturmalıdır. Bu program, H ve W'ın tüm değerleri ve görev kısıtlamalarını karşılayan tüm labirentler için doğru şekilde çalışmalıdır.
- Bu prosedür her test durumu için tam olarak bir kez çağrılır.

Bu prosedür, Pulibot programını oluşturmak için aşağıdaki prosedüre çağrı yapabilir:

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

- S: bir durum dizisini tanımlayan 5 uzunluğundaki dizi.
- Z: bir rengi temsil eden negatif olmayan bir tam sayı.
- A: Pulibot'un bir eylemini aşağıdaki açıklamalara uygun şekilde temsil eden tek bir karakter:
  - 'H': kal;
  - 'W': batıya doğru ilerle;
  - 'S': güneye doğru ilerle;
  - 'E': doğuya doğru ilerle;
  - 'N': kuzeye doğru ilerle;
  - 'T': programı sonlandır.
- Bu prosedürün çağrılması Pulibot'a S durum dizisini tanıdıktan sonra (Z, A) komutunu gerçekleştirmesi talimatını verir.

Bu prosedürü aynı S durum dizisiyle birden fazla kez çağırmak, `Output isn't correct` cevabıyla sonuçlanacaktır.

Her olası durum dizisi S ile `set_instruction` prosedürünü çağırmak zorunlu değildir. Ancak Pulibot daha sonra komutun ayarlanmadığı bir durumla karşılaşırsa `Output isn't correct` cevabıyla karşılaşırsınız.

`program_pulibot` prosedürünün çalışması tamamlandıktan sonra, değerlendirici Pulibot'un programını bir veya daha fazla labirent üzerinden çağırır. Bu çağrılar, çözümünüzün zaman sınırına dahil *edilmez*. Değerlendirici adaptif *değildir*, her test durumunda kullanılacak labirentler önceden tanımlanmıştır.

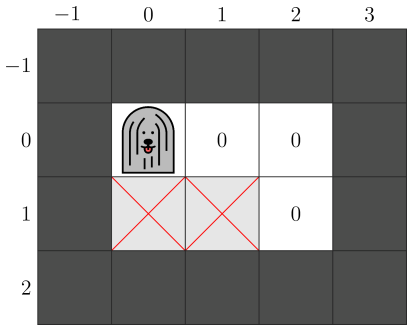
Pulibot, programını sonlandırmadan önce Robot Yarışması Kurallarından herhangi birini ihlal ederse, `Output isn't correct` cevabıyla karşılaşırsınız.

Örnek

`program_pulibot` prosedürü `set_instruction` prosedürünü aşağıdaki gibi çağırabilir:

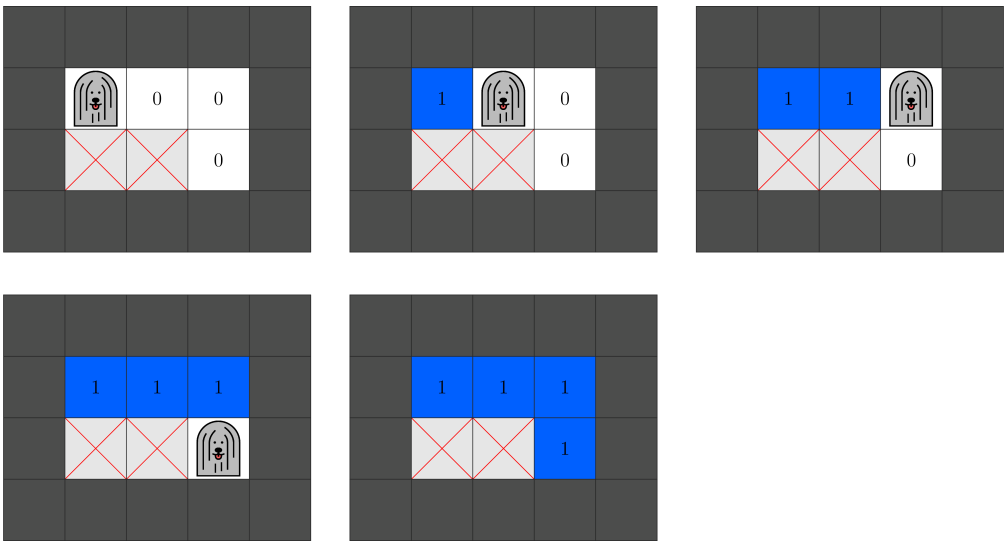
Çağrı	S durumu dizisine karşılık gelen komut
`set_instruction([0, -2, -1, 0, -2], 1, E)`	Rengi 1 olarak ayarla ve doğuya git
`set_instruction([0, 1, -1, 0, -2], 1, E)`	Rengi 1 olarak ayarla ve doğuya git
`set_instruction([0, 1, 0, -2, -2], 1, S)`	Rengi 1 olarak ayarla ve güneye git
`set_instruction([0, -1, -2, -2, 1], 1, T)`	Rengi 1 olarak ayarla ve çalışmayı sonlandır

$H = 2$ ve $W = 3$ için labirentin aşağıdaki resimdeki gibi olduğu bir senaryo düşünün.



Bu labirent için Pulibot'un programı dört adımda çalışıyor. Pulibot'un tanıdığı durum dizileri ve gerçekleştirdiği komutlar yukarıda verilen dört `set_instruction` çağrısına aynı sırayla karşılık gelmektedir. Bu talimatların sonuncusu programı sonlandırır.

Aşağıdaki şekil labirentin dört adımın her birinden önceki halini ve sonlandırıldıktan sonraki halini göstermektedir.



Ancak, bu 4 komuttan oluşan bu programın diğer geçerli labirentlerde en kısa patikayı bulamayabileceğine dikkat ediniz. Bu nedenle, gönderilirse `Output isn't correct` cevabını

alacaktır.

Kısıtlar

$Z_{MAX} = 19$. Dolayısıyla Pulibot 0'dan 19'a kadar (19 dahil) olan renkleri kullanabilir.

Pulibot'u test etmek için kullanılan her labirent için:

- $2 \leq H, W \leq 15$
- $(0, 0)$ hücresinden $(H - 1, W - 1)$ hücresine giden en az bir patika vardır.

Altgörevler

1. (6 puan) Labirentte herhangi bir engel hücresi yoktur.
2. (10 puan) $H = 2$
3. (18 puan) Her boş hücre ikilisi arasında tek bir patika vardır.
4. (20 puan) $(0, 0)$ hücresinden $(H - 1, W - 1)$ hücresine giden her en kısa patikanın uzunluğu $H + W - 2$'dır.
5. (46 puan) Başkaca kısıt yoktur.

Test senaryolarının herhangi birinde, `set_instruction' prosedürüne veya Pulibot programına yapılan çağrıların yürütülmesi sırasında Kodlama Detayları'nda açıklanan kısıtlamalara uyulmuyorsa, o alt görev için çözümünüzün puanı 0 olacaktır.

Her alt görevde neredeyse doğru bir renklendirme üreterek kısmi puan elde edebilirsiniz.

Daha açık bir ifadeyle:

- Boş hücrelerin son renklendirmesi Robot Yarışması Kurallarını karşılıyorsa test senaryosunun çözümü **tam**dır.
- Eğer son renklendirme aşağıdaki gibi ise, test senaryosunun çözümü **kısmi** demektir:
  - $(0, 0)$ ile $(H - 1, W - 1)$ arasındaki en kısa patikalardan birinin üzerindeki her hücrenin rengi 1'dir.
  - Izgarada 1 renginde başka boş hücre yoktur.
  - Izgaradaki bazı boş hücreler 0 ve 1 dışındaki bir renge sahiptir.

Bir test senaryosuna ilişkin çözümünüz ne tam ne de kısmi ise, bu test senaryosuna karşılık gelen puanınız 0 olacaktır.

1-4 arası alt görevlerde çözümünüz kısmi ise puanların %50'sini alacaksınız.

5 numaralı alt görevde puanınız Pulibot programında kullanılan renk sayısına bağlıdır. `set_instruction` prosedürüne yapılan tüm çağrılar üzerinden Z'nin maksimum değerine $Z^\star$ diyelim. Test senaryosunun puanı aşağıdaki tabloya göre hesaplanır:

Koşul	Skor (tam)	Skor (kısmi)
$11 \leq Z^\star \leq 19$	$20 + (19 - Z^\star)$	$12 + (19 - Z^\star)$
$Z^\star = 10$	31	23
$Z^\star = 9$	34	26
$Z^\star = 8$	38	29
$Z^\star = 7$	42	32
$Z^\star \leq 6$	46	36

The score for each subtask is the minimum of the points for the test cases in the subtask.

Her alt görevin puanı bu alt görevdeki test senaryolarının minimum puanına eşittir.

Örnek Değerlendirici

Örnek değerlendirici girdiyi aşağıdaki formatta okumaktadır:

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

Burada m, labirentin sınır hücresi olmayan hücrelerini tanımlayan, her biri W adet tam sayıdan oluşan H adet diziden oluşan bir dizidir. (r, c) hücresi boş bir hücre ise $m[r][c] = 0$'dır ve (r, c) hücresi bir engel hücresi ise $m[r][c] = 1$'dir.

Örnek değerlendirici ilk olarak `program_pulibot()` prosedürünü çağırır. Örnek değerlendirici bir protokol ihlali tespit ederse, `Protocol Violation: <MSG>` yazdırır. Burada `<MSG>` aşağıdaki hata mesajlarından biridir:

- `Invalid array`: $-2 \leq S[i] \leq Z_{MAX}$ şartı bir i için sağlanmamaktadır ya da S dizisinin uzunluğu 5 değildir.
- `Invalid color`: $0 \leq Z \leq Z_{MAX}$ şartı sağlanmamaktadır.
- `Invalid action`: A karakteri şunlardan biri değildir: `H`, `W`, `S`, `E`, `N` veya `T`.
- `Same state array`: `set_instruction` prosedürü aynı S dizisiyle en az iki kere çağrılmıştır.

Aksi takdirde, `program_pulibot` tamamlandığında, örnek değerlendirici Pulibot'un programını girdide tanımlanan labirent üzerinde çalıştırır.

Örnek değerlendirici iki çıktı üretir.

İlk olarak örnek değerlendirici, Pulibot'un eylemlerinin bir logunu çalışma dizinindeki `robot.bin` dosyasına yazar. Bu dosya, bir sonraki bölümde açıklanan görselleştirme aracı tarafından girdi olarak kullanılır.

İkincisi, Pulibot'un programı başarılı bir şekilde sonlandırılmazsa örnek değerlendirici aşağıdaki hata mesajlarından birini yazdırır:

- `Unexpected state`: Pulibot `set_instruction` prosedürünün çağrılmadığı bir durum dizisiyle karşılaştı.
- `Invalid move`: Pulibot'un boş olmayan bir hücreye gitmesiyle sonuçlanan bir eylem gerçekleştirildi.
- `Too many steps`: Pulibot, programını sonlandırmadan $500\,000$ adım gerçekleştirdi.

Aksi takdirde $e[r][c]$, Pulibot'un programı sona erdikten sonraki (r, c) hücresinin durumu olsun. Örnek değerlendirici H satırlarını aşağıdaki formatta yazdırır:

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

Görselleştirme Aracı

Bu göreve ilişkin eklenti paketi `display.py` adlı bir dosya içerir. Bu Python scripti çağrıldığında, Pulibot'un eylemlerini örnek değerlendiricinin girdisiyle tanımlanan labirentte gösterir. Bunun için çalışma dizininde `robot.bin` isimli ikili (binary) dosyasının bulunması gerekir. Scripti çalıştırmak için aşağıdaki komutu çalıştırın.

```
python3 display.py
```

Basit bir grafik arayüz ortaya çıkar. Ana özellikleri aşağıdaki gibidir:

- Bütün labirentin durumunu gözlemleyebilirsiniz. Pulibot'un mevcut konumu bir dikdörtgenle vurgulanmıştır.
- Ok düğmelerine tıklayarak veya kısayol (hotkey) tuşlarına basarak Pulibot'un adımlarına göz atabilirsiniz. Ayrıca belirli bir adıma da atlayabilirsiniz.
- Pulibot'un programında bir sonraki adım altta gösterilmektedir. Mevcut durum dizisini ve gerçekleştireceği komutu göstermektedir. Son adımdan sonra, değerlendiricinin hata mesajlarından birini veya program başarıyla sonlandırılırsa `Terminated` mesajını gösterir.
- Bir rengi temsil eden her sayıya, görsel bir arka plan renginin yanı sıra bir ekran metni de atayabilirsiniz. Görüntü metni, aynı renkteki her hücreye yazılacak kısa bir dizgidir (string). Aşağıdaki yollardan biriyle arka plan renklerini atayabilir ve metinleri görüntüleyebilirsiniz:
  - `Colors` düğmesine tıkladıktan sonra bunları bir iletişim penceresinde ayarlayın.
  - `colors.txt` dosyasının içeriğini düzenleyin.
- `robot.bin` dosyasını yeniden yüklemek için `Reload` düğmesini kullanın. Bunu yapmak `robot.bin` dosyasının içeriği değiştiyse faydalıdır.