

## Görev Navigasyon (Task Navigation)

 3 saniye  512 MB

$N \leq 1000$  düğüm ve  $M$  kenara sahip bir **bağlı yönsüz basit kaktüs çizgesi**<sup>1</sup> vardır. Bu çizgenin düğümleri renkli olup (0 ile 1499 arasında negatif olmayan tamsayılarla gösterilir) başlangıçta tüm düğümlerin rengi 0'dır. Bir **deterministik hafızasız robot**<sup>2</sup>, düğümden düğüme hareket ederek çizgeyi keşfeder. Robot, tüm düğümleri en az bir kez ziyaret etmeli ve ardından bu işlemi sonlandırmalıdır.

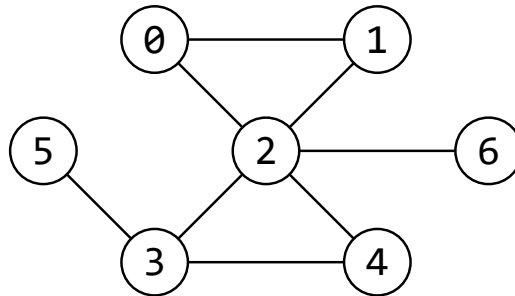
Robot, çizgedeki herhangi bir düğümden başlayabilir. Her adımda, mevcut düğümün rengini ve tüm komşu düğümlerin renklerini **mevcut düğüm için sırası sabitlenmiş** şekilde görür (yani, bir düğümü tekrar ziyaret ettiğinde, renkleri öncekinden farklı olsa bile, robot aynı komşu düğüm serisini görür). Robot, aşağıdaki iki eylemden birini gerçekleştirir:

1. Sonlandırmaya karar verir.
2. Mevcut düğüm için yeni (veya muhtemelen aynı) bir renk ve hangi komşu düğüme geçileceğini seçer. Komşu düğüm, 0 ile  $D-1$  arasında bir indeksle tanımlanır; burada  $D$ , komşu düğümlerin sayısıdır.

İkinci durumda, mevcut düğüm yeniden renklendirilir (veya muhtemelen aynı renkte kalır) ve robot seçilen komşu düğüme hareket eder. Bu, robot sonlanana veya iterasyon limiti dolana kadar tekrarlanır. Robot, tüm düğümleri ziyaret edip  $L = 3000$  adımlık iterasyon sınırı içinde sonlandırırsa kazanır (aksi takdirde kaybeder).

Bu tür kaktüs çizgelerinde problemi çözebilecek bir robot stratejisi tasarlamalısınız. Ayrıca, çözümünüzde kullandığınız farklı renk sayısını en aza indirmeye çalışmalısınız. Burada renk 0 her zaman kullanılmış olarak sayılır.

<sup>1</sup>Bağlı yönsüz basit kaktüs çizgesi, her kenarın en fazla bir basit döngüye ait olduğu (basit döngü, her düğümü en fazla bir kez içeren bir döngüdür) bağlı yönsüz basit bir çizgedir (her düğüm diğer tüm düğümlerden ulaşılabilir; kenarlar çift yönlüdür; kendi kendine döngü veya çoklu kenar içermez). Aşağıdaki resim bir örnektir.



<sup>2</sup>Bir robot, eylemleri yalnızca mevcut girdilerine bağlıysa (yani bir adımdan diğer adıma veri depolamıyorsa) ve aynı girdiler verildiğinde her zaman aynı eylemi seçiyorsa, deterministik ve hafızasızdır.



## İmplementasyon detayları

Robotun stratejisi aşağıdaki fonksiyon ile kodlanmalıdır:

```
std::pair<int, int> navigate(int currColor, std::vector<int> adjColors)
```

Parametre olarak mevcut düğümün rengini ve tüm komşu düğümlerin renklerini (sırayla) alır. Bu fonksiyon, ilk elemanı mevcut düğümün yeni rengi, ikinci elemanı ise robotun hareket etmesi gereken düğümün komşuluk indeksi olan bir çift dönmelidir. Bunun yerine robotun sonlandırması gerekiyorsa, fonksiyon  $(-1, -1)$  çiftini dönmelidir.

Bu fonksiyon, robotun eylemlerini seçmek için tekrar tekrar çağrılacaktır. Deterministik olduğu için, `navigate` bazı parametrelerle zaten çağrılmışsa, aynı parametrelerle bir daha asla çağrılmayacaktır; bunun yerine önceki dönen değer yeniden kullanılacaktır. Ek olarak, her test  $T \leq 5$  alt test içerebilir (farklı çizgeler ve/veya başlangıç konumları) ve bunlar eşzamanlı olarak çalıştırılabilir (yani, programınız farklı alt testler hakkında dönüşümlü çağrılar alabilir). Son olarak, `navigate` çağrıları programınızın **farklı çalıştırmalarında** gerçekleşebilir (ancak bazen aynı çalıştırmada da gerçekleşebilir). Programınızın toplam çalıştırma sayısı  $P = 100$ 'dür. Tüm bunlar nedeniyle, programınız farklı çağrılar arasında bilgi aktarmaya çalışmamalıdır.



## Kısıtlar

- $3 \leq N \leq 1000$
- $0 \leq \text{Color}(\text{Renk}) < 1500$
- $L = 3000$
- $T \leq 5$
- $P = 100$



## Puanlama

Aldığınız alt görevin puanlarının  $S$  kısmı,  $C$  değerine bağlıdır.  $C$ , çözümünüzün o alt görevdeki veya diğer gerekli alt görevlerdeki herhangi bir testte kullandığı farklı renklerin (renk 0 dahil) maksimum sayısıdır:

- Çözümünüz herhangi bir alt testte başarısız olursa,  $S = 0$  olur.
- Eğer  $C \leq 4$  ise  $S = 1.0$ .
- Eğer  $4 < C \leq 8$  ise  $S = 1.0 - 0.6 \frac{C-4}{4}$ .
- Eğer  $8 < C \leq 21$  ise  $S = 0.4 \frac{8}{C}$ .
- Eğer  $C > 21$  ise  $S = 0.15$ .



## Altgörevler

Altgörev	Puan	Gerekli altgörevler	$N$	Ek kısıtlar
0	0	—	$\leq 300$	Örnek.
1	6	—	$\leq 300$	Çizge bir dögüdür. <sup>1</sup>
2	7	—	$\leq 300$	Çizge bir yıldızdır. <sup>2</sup>
3	9	—	$\leq 300$	Çizge bir yoldur. <sup>3</sup>
4	16	2 – 3	$\leq 300$	Çizge bir ağaçtır. <sup>4</sup>
5	27	—	$\leq 300$	Tüm düğümlerin en fazla 3 adet komşu düğümü vardır ve robotun başladığı düğümün 1 adet komşu düğümü vardır.
6	28	0 – 5	$\leq 300$	—
7	7	0 – 6	—	—

<sup>1</sup>Bir dögü çizgesi şu kenarlara sahiptir:  $(i, (i + 1) \bmod N)$  for  $0 \leq i < N$ .

<sup>2</sup>Bir yıldız çizgesi şu kenarlara sahiptir:  $(0, i)$  for  $1 \leq i < N$ .

<sup>3</sup>Bir yol çizgesi şu kenarlara sahiptir:  $(i, i + 1)$  for  $0 \leq i < N - 1$ .

<sup>4</sup>Ağaç, dögüsüz bir çizgedir.



## Örnek

Görüntüdeki örnek çizgeyi ele alalım. Bu çizgede  $N = 7$ ,  $M = 8$  ve kenarlar  $(0, 1)$ ,  $(1, 2)$ ,  $(2, 0)$ ,  $(2, 3)$ ,  $(3, 4)$ ,  $(4, 2)$ ,  $(3, 5)$  ve  $(2, 6)$  vardır. Ayrıca, düğümlerin komşuluk listelerindeki elemanların sıralamaları da önemli olduğundan, bunları bu tabloda veriyoruz:

Düğüm	Komşu düğümler
0	2, 1
1	2, 0
2	0, 3, 4, 6, 1
3	4, 5, 2
4	2, 3
5	3
6	2

Robotun 5 düğümünden başladığını varsayalım. O zaman aşağıdaki, olası (başarısız) etkileşim serisidir:



#	Renkler	Düğüm	navigate olan çağrı	Dönen değer
1	0, 0, 0, 0, 0, 0, 0	5	navigate(0, {0})	{1, 0}
2	0, 0, 0, 0, 0, 1, 0	3	navigate(0, {0, 1, 0})	{4, 2}
3	0, 0, 0, 4, 0, 1, 0	2	navigate(0, {0, 4, 0, 0, 0})	{0, 3}
4	0, 0, 0, 4, 0, 1, 0	6	<sup>1</sup> navigate(0, {0})	{1, 0}
5	0, 0, 0, 4, 0, 1, 1	2	navigate(0, {0, 4, 0, 1, 0})	{8, 0}
6	0, 0, 8, 4, 0, 1, 1	0	navigate(0, {8, 0})	{3, 0}
7	3, 0, 8, 4, 0, 1, 1	2	navigate(8, {3, 4, 0, 1, 0})	{2, 2}
8	3, 0, 2, 4, 0, 1, 1	4	navigate(0, {2, 4})	{1, 1}
9	3, 0, 2, 4, 1, 1, 1	3	navigate(4, {1, 1, 2})	{-1, -1}

Burada robot toplam 6 farklı renk kullandı: 0, 1, 2, 3, 4 and 8 (tüm düğümler 0 rengiyle başladığı için, robot 0 rengini hiç dönmemiş olsa bile 0 kullanılmış sayılırdı). Robot, sonlandırılmadan önce 9 iterasyon yaptı. Ancak, 1 düğümünü ziyaret etmeden sonlandırıldığı için başarısız oldu.

<sup>1</sup> İterasyon 4'te `navigate` çağrısının aslında gerçekleşmeyeceğini unutmayın. Bunun nedeni, bu çağrının iterasyon 1'deki çağrıya eşdeğer olmasıdır, bu nedenle grader, eski çağrıdan fonksiyonunuzun dönme değerini yeniden kullanacaktır. Ancak bu, yine de robotun bir iterasyonu olarak sayılır.



## Örnek grader

Örnek grader, programınızı birden fazla kez çalıştırmaz, bu nedenle `navigate` fonksiyonuna yapılan tüm çağrılar programınızın aynı çalıştırmasında yer alır.

Girdi formatı şu şekildedir: İlk olarak  $T$  (alt testlerin sayısı) okunur. Ardından her bir alt test için:

- satır 1: iki tamsayı -  $N$  ve  $M$ ;
- satır  $2 + i$  ( $0 \leq i < M$  için): iki tamsayı -  $A_i$  ve  $B_i$ , bunlar  $i$  kenarının bağladığı iki düğümdür ( $0 \leq A_i, B_i < N$ ).

Örnek grader, çözümünüzün kullandığı farklı renklerin sayısını ve sonlandırılmadan önce ihtiyaç duyduğu iterasyon sayısını yazdıracaktır. Alternatif olarak, çözümünüz başarısız olursa bir hata mesajı yazdıracaktır.

Varsayılan olarak, örnek grader robotun her iterasyonda gördüğü ve yaptığı ile ilgili ayrıntılı bilgiler yazar. `DEBUG` değerini `true` yerine `false` olarak değiştirerek bunu devre dışı bırakabilirsiniz.