Sphinx'in Bilmecesi

Büyük Sphinx'in size bir bilmecesi var. N düğümden oluşan bir çizge veriliyor. Düğümler 0 ile $N-1$ arasında numaralandırılmıştır. Çizgede M adet kenar vardır. Her kenar, bir çift farklı düğümü birbirine bağlar ve iki yönlüdür. İki düğüme eğer bir kenarla birbirlerine bağlılarsa **komşu** denir. 0 dan $M-1$ e kadar her j kenarı (sınırlar dahil), $X[j]$ ve $Y[j]$ düğümlerini birbirine bağlar. Herhangi bir düğüm çiftini birbirine bağlayan en fazla bir kenar vardır.

($k \geq 0$ için) $v_0, v_1, \ldots, v_k$ düğümlerinden oluşan bir seri **yol** olarak adlandırılır eğer her iki ardışık düğüm v_l ve v_{l+1} (her l için öyle ki $0 \leq l < k$) komşu ise. $v_0, v_1, \ldots, v_k$ yolunun v_0 ve v_k düğümlerini **birleştirdiğini** söyleriz. Size verilen çizgede her bir düğüm çiftini birleştiren bir yol bulunmaktadır.

$N+1$ adet renk vardır ve bunlar 0 dan N ye kadar numaralandırılmıştır. N rengi özeldir ve **Sphinx'in rengi** olarak adlandırılır. Her düğüme bir renk atanır. Spesifik olarak, düğüm i ($0 \leq i < N$) $C[i]$ rengine sahiptir. Birden fazla düğüm aynı renge sahip olabilir, ve herhangi bir düğüme atanmamış renkler de olabilir. Hiçbir düğüm Sphinx'in rengine sahip değildir, yani $0 \leq C[i] < N$ ($0 \leq i < N$).

($k \geq 0$ için) $v_0, v_1, \ldots, v_k$ yolu **tek renkli** olarak adlandırılır eğer tüm düğümleri aynı renkte ise. Yani $C[v_l] = C[v_{l+1}]$ (her l için öyle ki $0 \leq l < k$). Ek olarak, p ve q düğümleri ($0 \leq p < N$, $0 \leq q < N$) aynı **tek renkli bileşende** bulunmaktadır ancak ve ancak bu düğümler tek renkli bir yolla birbirlerine bağlılarsa.

Düğümleri ve kenarları biliyorsunuz, ancak düğümlerin hangi renge sahip olduğunu bilmiyorsunuz. Düğümlerin renklerini **yeniden renklendirme deneyleri** yaparak bulmak istiyorsunuz.

Bir yeniden renklendirme deneyinde, istediğiniz kadar düğümü yeniden renklendirebilirsiniz. Özellikle, bir yeniden renklendirme deneyi gerçekleştirmek için ilk önce N boyutunda bir E dizisi seçersiniz, burada her i ($0 \leq i < N$) için, $E[i]$ -1 ile N arasındadır **sınırlar dahil**. Daha sonra, her bir düğüm i nin rengi $S[i]$ olur; burada $S[i]$ nin değeri şudur:

- $C[i]$, yani i nin orijinal rengi, eğer $E[i] = -1$ ise veya
- $E[i]$, aksi takdirde.

Bu, yeniden renklendirmenizde Sphinx'in rengini kullanabileceğiniz anlamına gelir.

Son olarak Büyük Sphinx, her bir i düğümün rengini $S[i]$ ($0 \leq i < N$) olarak ayarladıktan sonra çizgedeki tek renkli bileşenlerin sayısını duyurur. Yeni renklendirme yalnızca bu özel yeniden renklendirme deneyi için uygulanır. Böylece **deney bittikten sonra tüm düğümlerin renkleri orijinal renklerine geri dönüyor**.

Göreviniz, çizgedeki düğümlerin renklerini en fazla $2\,750$ yeniden renklendirme deneyi yaparak belirlemektir. Ayrıca eğer her komşu düğüm çiftinin aynı renkte olup olmadıklarını doğru bir şekilde belirlerseniz kısmi bir puan da alabilirsiniz.

Kodlama Detayları

Aşağıdaki prosedürü kodlamalısınız.

```
std::vector<int> find_colours(int N,
    std::vector<int> X, std::vector<int> Y)
```

- N : çizgedeki düğüm sayısı.
- X , Y : kenarları tanımlayan M uzunluğunda diziler.
- Bu prosedür, N uzunluğunda bir G dizisi dönmelidir. Bu dizi çizgedeki düğümlerin renklerini temsil eder.
- Bu prosedür her test durumu için tam olarak bir kez çağrılır.

Yukarıdaki prosedür yeniden renklendirme deneyleri yapmak için aşağıdaki prosedüre çağrılar yapabilir:

```
int perform_experiment(std::vector<int> E)
```

- E : düğümlerin nasıl yeniden renklendirileceğini belirten N uzunluğunda bir dizi.
- Bu prosedür, E ye göre düğümleri yeniden renklendirdikten sonra tek renkli bileşenlerin sayısını döner.
- Bu prosedür en fazla $2\,750$ kez çağrılabilir.

Değerlendirici **uyarlanabilir (adaptif) değildir**, yani, `find_colours` çağrısı yapılmadan önce düğümlerin renkleri sabitlenir.

Kısıtlar

- $2 \leq N \leq 250$
- $N - 1 \leq M \leq \frac{N \cdot (N-1)}{2}$
- $0 \leq X[j] < Y[j] < N$ her bir j için öyle ki $0 \leq j < M$.
- $X[j] \neq X[k]$ veya $Y[j] \neq Y[k]$ her bir j ve k için öyle ki $0 \leq j < k < M$.
- Her bir düğüm çifti en az bir yol ile birleştirilmiştir.
- $0 \leq C[i] < N$ her bir i için öyle ki $0 \leq i < N$.

Altgörevler

Altgörev	Puan	Ek Kısıtlar
1	3	$N = 2$
2	7	$N \leq 50$
3	33	Çizge bir yoldur: $M = N - 1$ ve j ve $j + 1$ düğümleri komşudur ($0 \leq j < M$).
4	21	Çizge tamdır (complete): $M = \frac{N \cdot (N-1)}{2}$ ve herhangi iki düğüm komşudur.
5	36	Ek kısıt yoktur.

Her alt görevde, eğer programınız her komşu düğüm çiftinin aynı renkte olup olmadıklarını doğru bir şekilde belirlerse kısmi bir puan alabilirsiniz.

Daha açık olarak, eğer tüm test durumlarında, `find_colours` tarafından dönen G dizisi C dizisiyle tam olarak aynı ise (yani $G[i] = C[i]$ $0 \leq i < N$ olacak şekilde tüm i için)
ilgili alt görevin tüm puanını alırsınız. Aksi takdirde, eğer aşağıdaki koşullar bir altgörevin tüm test durumlarında geçerliyse ilgili alt görevin 50% puanını alırsınız:

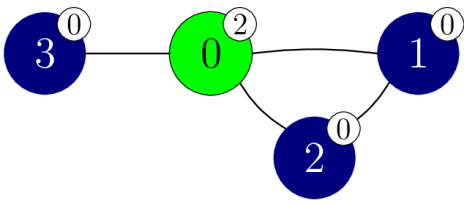
- $0 \leq G[i] < N$ her i için öyle ki $0 \leq i < N$;
- $0 \leq j < M$ olacak şekilde her j için:
  - $G[X[j]] = G[Y[j]]$ ancak ve ancak $C[X[j]] = C[Y[j]]$ ise.

Örnek

Aşağıdaki çağrıyı göz önüne alın.

```
find_colours(4, [0, 1, 0, 0], [1, 2, 2, 3])
```

Bu örnek için şunu varsayalım: (gizli) düğümlerin renkleri şu şekilde verilir: $C = [2, 0, 0, 0]$. Bu senaryo aşağıdaki şekilde gösterilmiştir. Renkler ayrıca her bir düğüme iliştirilmiş beyaz etiketlerdeki sayılarla da temsil edilir.



Prosedür `perform_experiment` i aşağıdaki gibi çağırabilir.

```
perform_experiment([-1, -1, -1, -1])
```

Bu çağrıda tüm düğümler orijinal renklerini koruduğu için hiçbir düğüm yeniden renklendirilmez.

1 ve 2 düğümlerini ele alalım. İkisinin de rengi 0 dır ve $1, 2$ yolu tek renkli bir yoldur. Sonuç olarak, 1 ve 2 düğümleri aynı tek renkli bileşendedir.

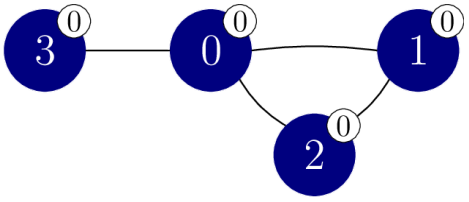
1 ve 3 düğümlerini ele alalım. Her ikisinin de rengi 0 olmasına rağmen, bu düğümleri birbirine bağlayan tek renkli yol olmadığı için bunlar farklı tek renkli bileşendedir.

Genel olarak, düğümleri $\{0\}$, $\{1, 2\}$, ve $\{3\}$ olan 3 tek renkli bileşen vardır. Bu nedenle bu çağrı 3 değerini döner.

Şimdi prosedür `perform_experiment` aşağıdaki gibi çağırabilir.

```
perform_experiment([0, -1, -1, -1])
```

Bu çağrıda, yalnızca 0 düğümü 0 rengine yeniden renklendirilir. Bu da aşağıdaki şekilde görülen renklenmeyle sonuçlanır.

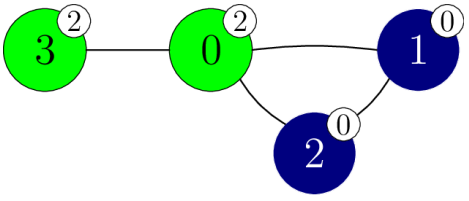


Bu çağrı, tüm düğümlerin aynı tek renkli bileşene ait olması nedeniyle 1 değerini döner. Artık 1 , 2 ve 3 düğümlerinin 0 rengine sahip olduğu sonucunu çıkarabiliriz.

Prosedür daha sonra `perform_experiment` i aşağıdaki gibi çağırabilir.

```
perform_experiment([-1, -1, -1, 2])
```

Bu çağrıda, 3 düğümü 2 rengine yeniden renklendirilir. Bu da aşağıdaki şekilde görülen renklenmeyle sonuçlanır.



Bu çağrı 2 döner, çünkü düğümleri sırasıyla $\{0, 3\}$ ve $\{1, 2\}$ olan 2 adet tek renkli bileşen vardır. 0 düğümünün 2 rengine sahip olduğu sonucunu çıkarabiliriz.

`find_colours` prosedürü daha sonra $[2, 0, 0, 0]$ dizisini döner. $C = [2, 0, 0, 0]$ olduğundan tam puan verilir.

Ayrıca, puanın 50% sinin verileceği birden fazla dönme değeri olduğunu unutmayın, örneğin $[1, 2, 2, 2]$ veya $[1, 2, 2, 3]$.

Örnek Değerlendirici

Girdi formatı:

```
N  M
C[0]  C[1] ... C[N-1]
X[0]  Y[0]
X[1]  Y[1]
...
X[M-1]  Y[M-1]
```

Çıktı formatı:

```
L  Q
G[0]  G[1] ... G[L-1]
```

Burada, L `find_colours` tarafından dönen G dizisinin uzunluğudur, ve Q `perform_experiment` için yapılan çağrıların sayısıdır.