Fıskiyeli Parklar

Yakın bir parkta, 0'dan $n-1$'e numaralandırılmış n tane **fıskiye** vardır. Fıskiyeleri iki boyutlu bir düzlemde noktalar olarak gösteriyoruz. Tam olarak, i numaralı fıskiye ($0 \leq i \leq n-1$), $x[i]$ ve $y[i]$ değerlerinin **çift tamsayılar** olduğu bir $(x[i], y[i])$ noktasıdır. Bütün fıskiyelerin konumları birbirlerinden farklıdır.

Mimar Timothy parka bazı **yollar** ve her bir yola da birer **bank** yerleştirmek üzere işe alındı. Bir yol, 2 uzunluğunda, uç noktaları iki farklı fıskiye olan, **yatay** ya da **dikey** bir doğru parçasıdır. Yollar öyle bir şekilde inşa edilmelidir ki herhangi bir fıskiyeden diğerine inşa edilen yollar üzerinden gitmek mümkün olmalıdır. Başlangıçta parkta hiç yol bulunmamaktadır.

Her bir yol için, **tam olarak** bir bank parka yerleştirilmeli ve o yola **atanmalıdır** (yani o yola bakmalıdır). Her bir bank (a, b) gibi bir noktaya konmalı ve a, b değerleri **tek tamsayılar** olmalıdır. Bankların hepsi birbirlerinden **farklı** pozisyonlarda olmalıdır. (a, b) noktasındaki bir bank ancak bir yolun **her iki** uç noktası $(a-1, b-1)$, $(a-1, b+1)$, $(a+1, b-1)$ ya da $(a+1, b+1)$ değerleri arasında ise o yola atanabilir. Örnek vermek gerekirse, $(3, 3)$ noktasındaki bir bank sadece şu dört doğru parçasından birisi olan bir yola atanabilir: $(2, 2)$ – $(2, 4)$, $(2, 4)$ – $(4, 4)$, $(4, 4)$ – $(4, 2)$, $(4, 2)$ – $(2, 2)$.

Timothy'ye yolları inşa etmenin ve yukarıda belirtilen kısıtlarda bankları yerleştirmenin mümkün olup olmadığını bulmasında yardımcı olunuz. Yolları inşa etme ve bank yerşleştirme mümkünse de, olası bir çözümü gösteriniz. Bütün şartları sağlayan birden fazla çözüm mümkünse içlerinden herhangi birisini verebilirsiniz.

Implementasyon Detayları

Aşağıdaki fonksiyonu implement etmelisiniz:

```
int construct_roads(int[] x, int[] y)
```

- x, y: her biri n uzunluğunda iki array. Her bir i ($0 \leq i \leq n-1$) için, i numaralı fıskiye $(x[i], y[i])$ koordinatında bir noktadır ve $x[i]$, $y[i]$ çift tamsayılardır.
- Eğer inşaat mümkün ise, bu fonksiyon `build` fonksiyonunu tam olarak bir kez çalıştırarak çözümü belirtmelidir (detaylar aşağıda), sonra da 1 dönmelidir.
- Değilse, bu fonksiyon `build` fonksiyonunu hiç çağırmadan direkt 0 dönmelidir.
- Bu fonksiyon bir kez çağrılacaktır.

Implementasyonunuz aşağıdaki fonksiyonu çağırarak yolların yapımı ve bankaların yerleştirilmesi için olası bir çözümü belirtebilir:

```
void build(int[] u, int[] v, int[] a, int[] b)
```

- İnşa edilen yolların sayısı m olsun.
- u, v: her biri m uzunluğunda iki dizi, inşa edilecek yolları gösterirler. Yollar 0'dan $m-1$'e numaralandırılmıştır. Her bir j ($0 \leq j \leq m-1$) için, j numaralı yol $u[j]$ ve $v[j]$ numaralı fıskiyeleri birbirine bağlar. Her bir yol 2 uzunluğunda yatay ya da dikey bir yol olmalıdır. Herhangi iki farklı yol en fazla bir ortak noktaya sahip olabilir (o da bir fıskiye). Yollar inşa edildikten sonra, herhangi bir fıskiyeden diğerine yollar aracılığı ile gitmek mümkün olmalıdır.
- a, b: her biri m uzunluğunda iki array, bankları gösterirler. Her bir j ($0 \leq j \leq m-1$) için, $(a[j], b[j])$ noktasına konmuş bir bank j numaralı yola atanmıştır. Farklı iki bank aynı noktada olamaz.

Örnekler

Örnek 1

Aşağıdaki çağrıyı gözönüne alınız:

```
construct_roads([4, 4, 6, 4, 2], [4, 6, 4, 2, 4])
```

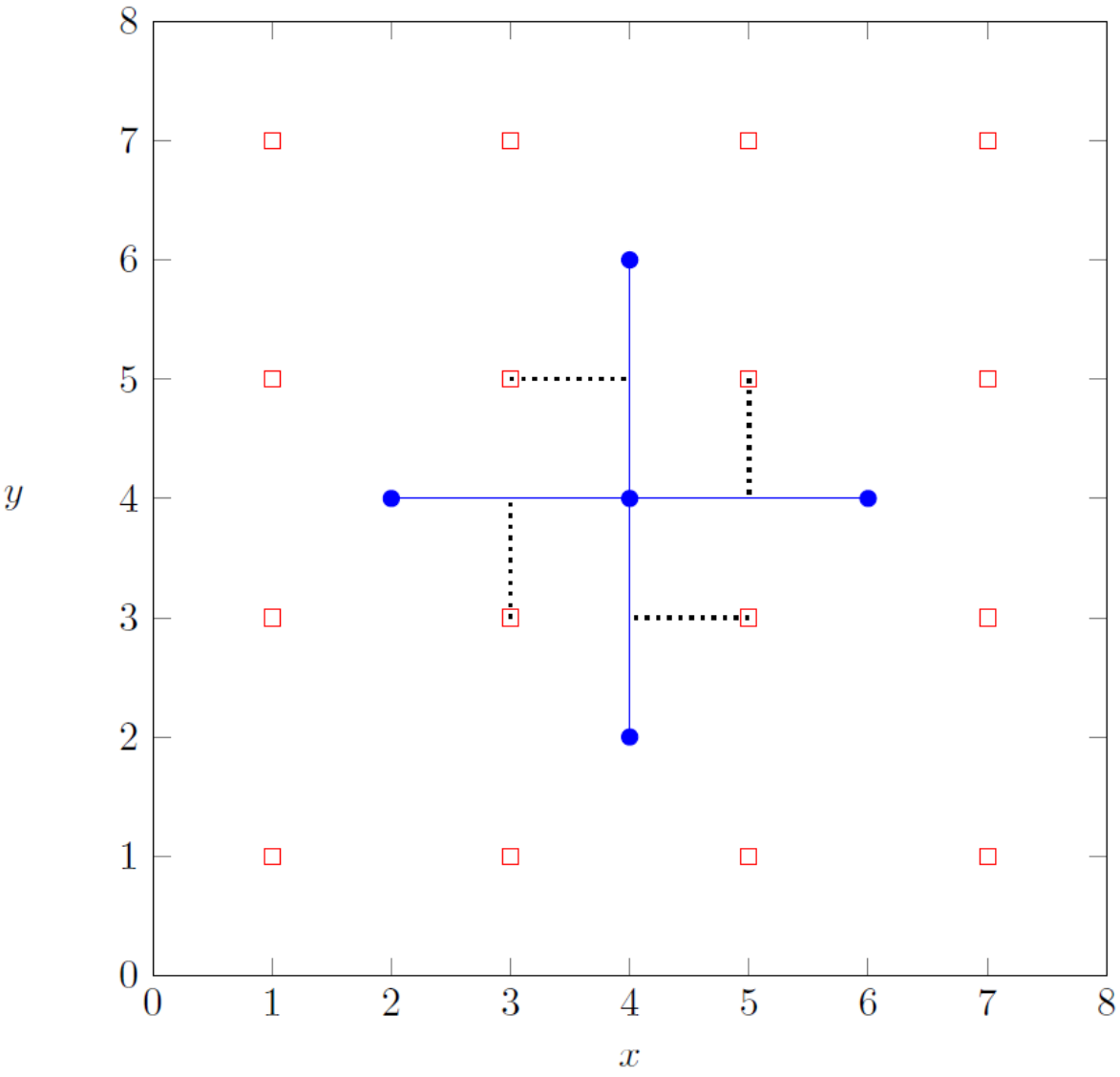
5 tane fıskiye vardır:

- fıskiye 0, $(4, 4)$ noktasındadır,
- fıskiye 1, $(4, 6)$ noktasındadır,
- fıskiye 2, $(6, 4)$ noktasındadır,
- fıskiye 3, $(4, 2)$ noktasındadır,
- fıskiye 4, $(2, 4)$ noktasındadır.

Her yol iki fıskiyeyi birbirine bağlayacak şekilde ve belirtilen bankları yerleştirerek, aşağıdaki 4 yolu inşa etmek mümkündür:

Yol numarası	Yolun bağladığı fıskiyelerin numaraları	Atanan bankın pozisyonu
0	$0, 2$	$(5, 5)$
1	$0, 1$	$(3, 5)$
2	$3, 0$	$(5, 3)$
3	$4, 0$	$(3, 3)$

Bu çözüm aşağıdaki şekilde gösterilmiştir:



Bu çözümü çıktı olarak vermek için, `construct_roads` aşağıdaki fonksiyonu çağırmalıdır:

- `build([0, 0, 3, 4], [2, 1, 0, 0], [5, 3, 5, 3], [5, 5, 3, 3])`

Sonra da 1 dönmelidir.

Bu örnek için, doğru çözüm olarak kabul edilebilecek başka olası çözümler de vardır. Mesela, `build([1, 2, 3, 4], [0, 0, 0, 0], [5, 5, 3, 3], [5, 3, 3, 5])` çağrısını yapıp sonra 1 dönmek de mümkündür.

Örnek 2

Aşağıdaki çağrıyı gözönüne alınız:

```
construct_roads([2, 4], [2, 6])
```

0 numaralı fıskiye $(2, 2)$ ve 1 numaralı fıskiye de $(4, 6)$ noktasındadır. Kısıtları sağlayacak şekilde yolları inşa etmek mümkün olmadığından, `construct_roads`, herhangi bir `build` çağrısı yapmadan

0 dönerek sonlanmalıdır.

Kısıtlar

- $1 \leq n \leq 200\,000$
- $2 \leq x[i], y[i] \leq 200\,000$ (her bir $0 \leq i \leq n-1$ için)
- $x[i]$ ve $y[i]$ çift tamsayılardır (her bir $0 \leq i \leq n-1$ için).
- Herhangi iki fıskiye aynı pozisyonda değildir.

Altgörevler

1. (5 puan) $x[i] = 2$ (her bir $0 \leq i \leq n-1$ için)
2. (10 puan) $2 \leq x[i] \leq 4$ (her bir $0 \leq i \leq n-1$ için)
3. (15 puan) $2 \leq x[i] \leq 6$ (her bir $0 \leq i \leq n-1$ için)
4. (20 puan) Herhangi bir fıskiyeden diğerine yollarla gidilebilecek şekilde, en fazla bir tane çözüm bulunmaktadır.
5. (20 puan) 2×2'lik bir karenin köşeleri olacak şekilde dört tane fıskiye bulunmamaktadır.
6. (30 puan) Ek kısıt bulunmamaktadır.

Örnek Grader

Örnek grader girdiyi aşağıdaki formatta okur

- satır 1 : n
- satır $2+i$ ($0 \leq i \leq n-1$): $x[i]$ $y[i]$

Örnek grader'ın çıktısı aşağıdaki formattadır:

- satır 1: `construct_roads`'dan dönen değer

Eğer `construct_roads`'tan dönen değer 1 ise ve `build(u, v, a, b)` çağrılmışsa, grader ek olarak aşağıdakileri de basar:

- satır 2: m
- satır $3+j$ ($0 \leq j \leq m-1$): $u[j]$ $v[j]$ $a[j]$ $b[j]$