پارک فواره‌ها

در یک پارک در همین نزدیکی، n **فواره** با شماره‌های 0 تا $n-1$ وجود دارند. ما فواره‌ها را به صورت یک نقطه در صفحه دو بعدی مدل می‌کنیم. می‌گوییم فواره i $(0 \leq i \leq n-1)$ نقطه $(x[i], y[i])$ است که $x[i]$ و $y[i]$ **اعداد زوج** هستند. مکان تمام فواره‌ها متفاوت است.

تیموتی معمار برای برنامه‌ریزی ساخت تعدادی **جاده** و قرارگیری یک **نیمکت** به ازای هر جاده، استخدام شده است. یک جاده یک خط افقی یا عمودی به طول ۲ است، که دو سر آن دو فواره متفاوت است. جاده‌ها باید به صورتی ساخته شوند که هر فردی بتواند با حرکت کردن روی جاده‌ها، بین هر دو فواره‌ای حرکت کند. در ابتدا هیچ جاده‌ای در پارک نیست.

برای هر جاده، **دقیقاً** یک نیمکت لازم است که درون پارک قرار بگیرد و به آن جاده **اختصاص یافته** (یعنی رو به آن) باشد. هر نیمکت باید در یک نقطه‌ای مانند (a, b) قرار بگیرد که a و b **اعداد فرد** هستند. مکان همه نیمکت‌ها باید **متفاوت** باشد. یک نیمکت در (a, b) تنها می‌تواند به یک جاده اختصاص یابد اگر **هر دو** انتهای جاده یکی از نقاط $(a-1, b-1)$، $(a-1, b+1)$، $(a+1, b-1)$ و $(a+1, b+1)$ باشد. برای مثال، نیمکت واقع در $(3, 3)$ تنها می‌تواند به یک جاده اختصاص یابد که یکی از خط‌های $(2, 2)$ – $(2, 4)$، $(2, 4)$ – $(4, 4)$، $(4, 4)$ – $(4, 2)$، $(4, 2)$ – $(2, 2)$ باشد.

به تیموتی کمک کنید که مشخص کند آیا چنین جاده‌هایی را می‌توان ساخت، و نیکمت‌ها را قرار داد و اختصاص داد به صورتی که تمام شرایط داده شده در بالا را ارضا کنند، و اگر می‌شود، به او یک راه‌حل امکان پذیر ارائه دهید. اگر چندین راه‌حل امکان پذیر وجود داشت که تمام شرایط را ارضا می‌کردند، شما می‌توانید هر کدام از آن‌ها را ارائه کنید.

جزئیات پیاده‌سازی

شما باید تابع زیر را پیاده‌سازی کنید:

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

- x, y : دو آرایه به طول n . برای هر i $(0 \leq i \leq n-1)$، فواره i نقطه $(x[i], y[i])$ است که $x[i]$ و $y[i]$ اعداد زوج هستند.
- اگر یک ساخت ممکنی وجود داشته باشد، این تابع باید دقیقاً یک بار فراخوانی `build` (پایین را ببینید) را انجام دهد تا یک راه‌حل ارائه دهد، و در ادامه باید 1 برگرداند.
- در غیر این صورت، تابع باید 0 برگرداند، بدون هیچ فراخوانی از `build`.
- این تابع دقیقا یکبار فراخوانی می‌شود.

پیاده سازی شما می‌تواند تابع زیر را برای ارائه یک ساخت امکان پذیر از جاده‌ها و قرارگیری نیمکت‌ها، فراخوانی کند:

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

- فرض کنید m تعداد کل جاده‌ها در ساخت شما است.

- u, v : دو آرایه به طول m ، که جاده‌هایی که ساخته می‌شوند را نشان می‌دهند. این جاده‌ها از 0 تا $m-1$ شماره گذاری شده‌اند. برای هر j $(0 \leq j \leq m-1)$، جاده j فواره‌های $u[j]$ و $v[j]$ را به هم متصل می‌کند. هر جاده باید یک خط افقی یا عمودی به طول 2 باشد. هر دو جاده متفاوت می‌توانند حداکثر در یک نقطه مشترک باشند (یک فواره). زمانی که جاده‌ها ساخته شد، باید حرکت بین هر دو فواره‌ای توسط جاده‌ها ممکن باشد.
- a, b : دو آرایه به طول m ، نشان‌دهنده‌ی نیمکت‌ها. برای هر j $(0 \leq j \leq m-1)$، یک نیمکت در $(a[j], b[j])$ قرار گرفته است، و به جاده j اختصاص یافته است. هیچ دو نیمکت متفاوتی نمی‌توانند مکان یکسانی داشته باشند.

مثال‌ها

مثال ۱

فراخوانی زیر را در نظر بگیرید:

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

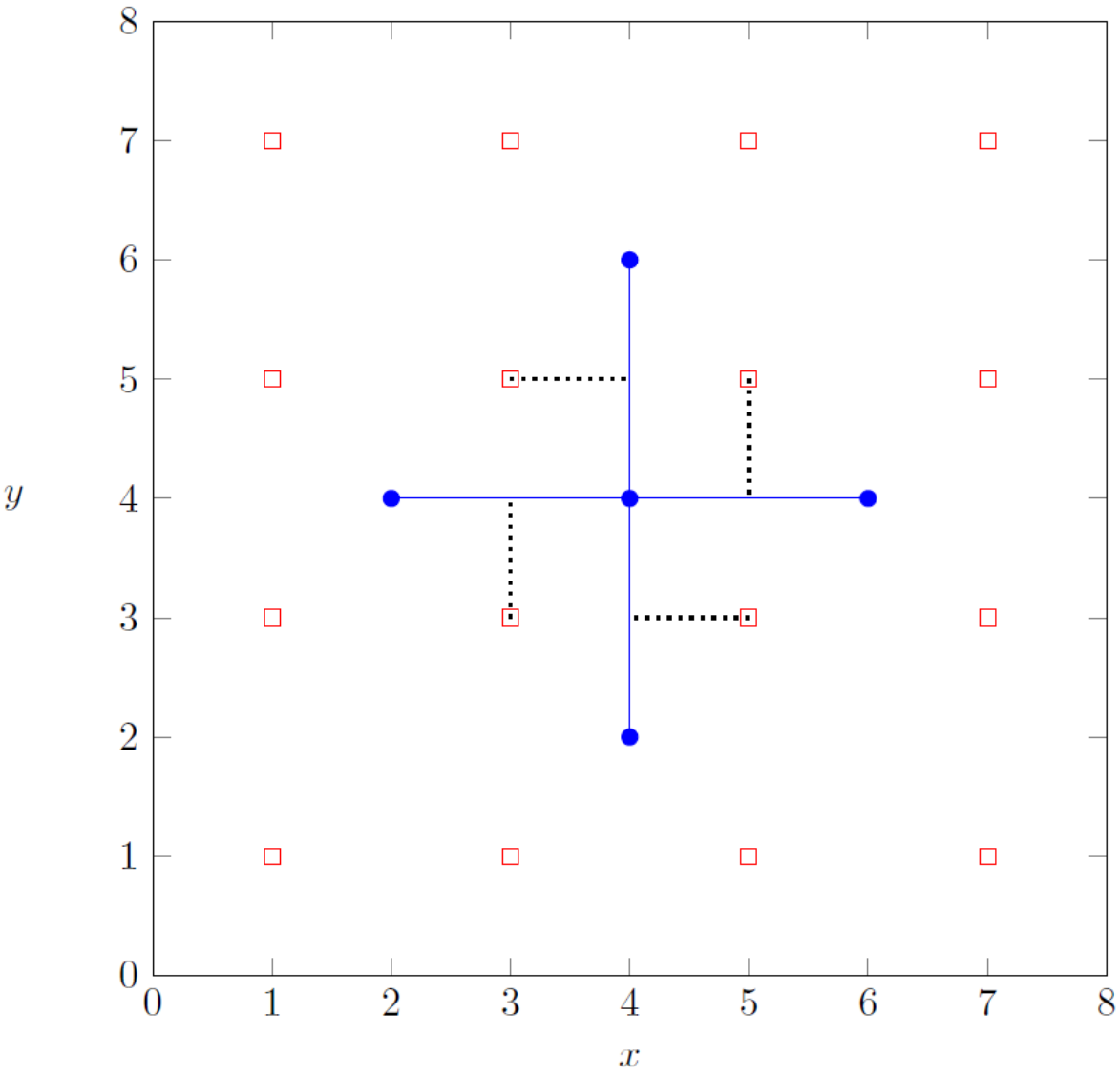
این بدان معنی است که 5 فواره وجود دارند:

- فواره 0 در $(4, 4)$ قرار گرفته است،
- فواره 1 در $(4, 6)$ قرار گرفته است،
- فواره 2 در $(6, 4)$ قرار گرفته است،
- فواره 3 در $(4, 2)$ قرار گرفته است،
- فواره 4 در $(2, 4)$ قرار گرفته است.

ساخت 4 جاده‌ی زیر امکان پذیر است، که هر جاده دو فواره را متصل می‌کند، و قرار دادن نیمکت‌های متناظر:

شماره جاده	شماره فواره‌هایی که جاده‌ها متصل می‌کنند	مکان نیمکت‌های اختصاص یافته
0	$0, 2$	$(5, 5)$
1	$0, 1$	$(3, 5)$
2	$3, 0$	$(5, 3)$
3	$4, 0$	$(3, 3)$

این جواب متناظر تصویر زیر است:



برای ارائه دادن این جواب، `construct_roads` باید این فراخوانی را انجام دهد:

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

سپس باید 1 برگرداند.

توجه کنید در این مثال، چندین راه حل که تمام الزامات را ارضا کنند وجود دارد، که همه آن‌ها درست در نظر گرفته می‌شوند. برای مثال، فراخوانی

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

و بعد آن برگرداندن 1 نیز درست است.

مثال ۲

فراخوانی زیر را در نظر بگیرید:

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

فواره 0 در $(2, 2)$ قرار گرفته است و فواره 1 در $(4, 6)$ قرار گرفته است. از آن جایی که هیچ راهی برای ساختن جاده‌هایی که الزامات را ارضا کنند وجود ندارد، `construct_roads` باید 0 برگرداند بدون هیچ فراخوانی از `build`.

محدودیت‌ها

- $1 \leq n \leq 200\,000$
- $2 \leq x[i], y[i] \leq 200\,000$ (برای هر $0 \leq i \leq n - 1$)
- $x[i]$ و $y[i]$ اعداد زوج هستند (برای هر $0 \leq i \leq n - 1$).
- هیچ دو فواره‌ای مکان یکسان ندارند.

زیرمسئله‌ها

1. (۵ نمره) $x[i] = 2$ (برای هر $0 \leq i \leq n - 1$)
2. (۱۰ نمره) $2 \leq x[i] \leq 4$ (برای هر $0 \leq i \leq n - 1$)
3. (۱۵ نمره) $2 \leq x[i] \leq 6$ (برای هر $0 \leq i \leq n - 1$)
4. (۲۰ نمره) حداکثر یک راه برای ساختن جاده‌ها وجود دارد، به صورتی که هر فردی بتواند بین هردو فواره‌ای با حرکت روی آن جاده‌ها حرکت کند.
5. (۲۰ نمره) هیچ چهار فواره‌ای وجود ندارند که یک مربع 2×2 بسازند.
6. (۳۰ نمره) بدون محدودیت اضافی.

ارزیاب نمونه

ارزیاب نمونه ورودی را در قالب زیر می‌خواند:

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

خروجی ارزیاب نمونه در قالب زیر است:

- خط 1 : مقدار برگردانده شده تابع `construct_roads`

اگر مقدار برگردانده شده تابع `construct_roads` برابر 1 باشد و `build(u, v, a, b)` فراخوانی شده باشد، آن‌گاه ارزیاب مقادیر زیر را نیز چاپ می‌کند:

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