Fountain Parks

In een nabijgelegen park zijn n **fonteinen**, gelabeled van 0 tot en met $n-1$. We bekijken de fonteinen als punten in een twee-dimensionaal vlak. Fontein i ($0 \leq i \leq n-1$) is een punt $(x[i], y[i])$ waarbij $x[i]$ en $y[i]$ **even integers** zijn. De locaties van de fonteinen zijn allemaal verschillend.

Timothy is aangenomen als architect om de bouw van een aantal **wegen** en het plaatsen van één **bank** per weg te plannen. Een weg is een **horizontaal** of **verticaal** lijnstuk met een lengte 2, waarvan de eindpunten twee verschillende fonteinen zijn. De wegen moeten zodanig aangelegd worden dat je tussen iedere twee fonteinen kunt reizen over de wegen. Aanvankelijk zijn er geen wegen in het park.

Voor iedere weg moet er **precies** één bank worden geplaatst in het park en **toegewezen worden** aan (dat wil zeggen, gericht naar) die weg. Iedere bank moet worden geplaatst op een punt (a, b) zodanig dat a en b **oneven integers** zijn. De locaties van de banken moeten allemaal **anders zijn**. Een bank op (a, b) kan alleen toegewezen worden aan een weg als **beide** eindpunten van de weg komen uit $(a-1, b-1)$, $(a-1, b+1)$, $(a+1, b-1)$ en $(a+1, b+1)$. Bijvoorbeeld, de bank op $(3, 3)$ kan alleen toegewezen worden aan de weg die een van de volgende vier lijnstukken is $(2, 2)$ – $(2, 4)$, $(2, 4)$ – $(4, 4)$, $(4, 4)$ – $(4, 2)$, $(4, 2)$ – $(2, 2)$.

Help Timothy vast te stellen of het mogelijk is om wegen aan te leggen en banken te plaatsen en toe te wijzen, waarbij aan alle bovengenoemde voorwaarden wordt voldaan. Als dat zo is, geef hem dan een mogelijke oplossing. Als er meerdere mogelijke oplossingen zijn die voldoen aan alle voorwaarden, mag je een willekeurige geven.

Implementatiedetails

Implementeer de volgende functie:

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

- x, y: twee arrays van lengte n. Voor iedere i ($0 \leq i \leq n-1$), is fontein i een punt $(x[i], y[i])$, waarbij $x[i]$ en $y[i]$ even integers zijn.
- Als er een constructie mogelijk is, moet de functie een oplossing melden door precies één keer `build` aan te roepen (zie hieronder) en vervolgens 1 teruggeven.
- Anders moet de functie 0 teruggeven zonder `build` aan te roepen.
- Deze functie wordt precies één keer aangeroepen.

Jouw implementatie kan de volgende functie aanroepen om een werkbare constructie van wegen en banken te melden:

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

- Laat m het aantal wegen in de constructie zijn.
- u, v: twee arrays van lengte m, die de wegen voorstellen. Deze wegen hebben een label van 0 tot en met $m-1$. Voor iedere j ($0 \leq j \leq m-1$), verbindt weg j fonteinen $u[j]$ en $v[j]$. Iedere weg moet een horizontaal of verticaal lijnstuk zijn van lengte 2. Twee verschillende wegen hebben hoogstens één gemeenschappelijk punt (een fontein). Als de wegen zijn aangelegd, moet het mogelijk zijn om iedere fontein te bereiken vanuit iedere andere fontein door over de wegen te reizen.
- a, b: twee arrays van lengte m die de banken voorstellen. Voor iedere j ($0 \leq j \leq m-1$) wordt een bank op $(a[j], b[j])$ geplaatst en toegewezen aan weg j. Twee verschillende banken kunnen niet dezelfde locatie hebben.

Voorbeelden

Voorbeeld 1

Neem de volgende aanroep:

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

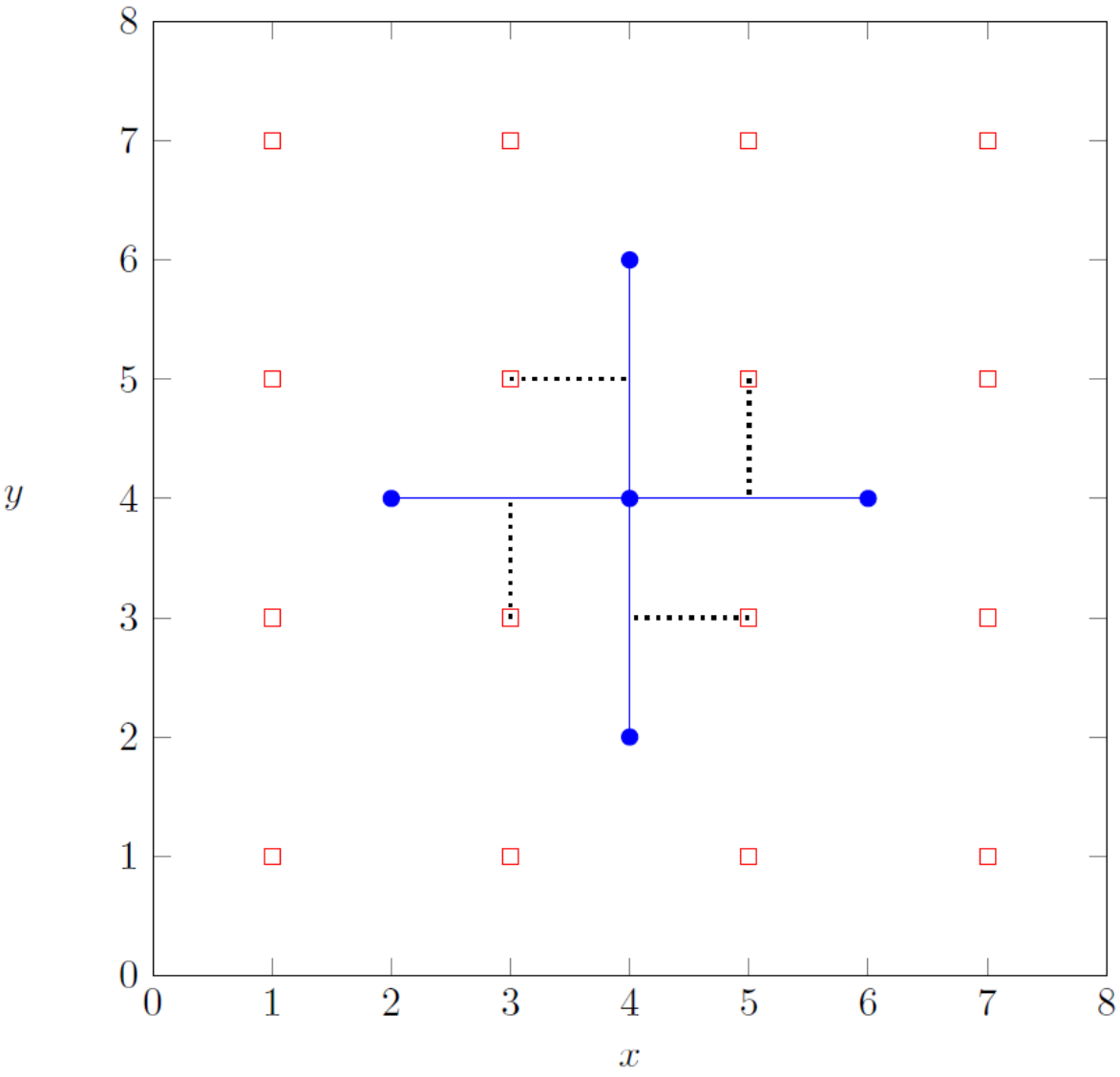
Dit betekent dat er 5 fonteinen zijn:

- fontein 0 ligt op $(4, 4)$,
- fontein 1 ligt op $(4, 6)$,
- fontein 2 ligt op $(6, 4)$,
- fontein 3 ligt op $(4, 2)$,
- fontein 4 ligt op $(2, 4)$.

Het is mogelijk om de volgende 4 wegen aan te leggen, waar iedere weg twee fonteinen verbindt, en bij deze wegen de volgende banken te plaatsen:

Weg label	Labels van de fonteinen die de weg verbindt	Locatie van de toegewezen bank
0	$0, 2$	$(5, 5)$
1	$0, 1$	$(3, 5)$
2	$3, 0$	$(5, 3)$
3	$4, 0$	$(3, 3)$

Deze oplossing komt overeen met het volgende diagram:



Om deze oplossing te melden, moet `construct_roads` de volgende aanroep maken:

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

Het moet vervolgens 1 teruggeven.

Merk op dat er in dit geval meerdere oplossingen zijn die voldoen aan alle voorwaarden, die allemaal correct zouden zijn.

Het is bijvoorbeeld ook correct om `build([1, 2, 3, 4], [0, 0, 0, 0], [5, 5, 3, 3], [5, 3, 3, 5])` aan te roepen en vervolgens 1 terug te geven.

Voorbeeld 2

Neem de volgende aanroep:

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

Fontein 0 ligt op $(2, 2)$ en fontein 1 ligt op $(4, 6)$. Omdat er geen manier is om wegen aan te leggen die voldoen aan de voorwaarden, moet `construct_roads` 0 teruggeven zonder `build` aan te roepen.

Randvoorwaarden

- $1 \leq n \leq 200\,000$
- $2 \leq x[i], y[i] \leq 200\,000$ (voor alle $0 \leq i \leq n - 1$)
- $x[i]$ en $y[i]$ zijn even integers (voor alle $0 \leq i \leq n - 1$).
- Geen twee fonteinen hebben dezelfde locatie.

Subtaken

1. (5 punten) $x[i] = 2$ (voor alle $0 \leq i \leq n - 1$)
2. (10 punten) $2 \leq x[i] \leq 4$ (voor alle $0 \leq i \leq n - 1$)
3. (15 punten) $2 \leq x[i] \leq 6$ (voor alle $0 \leq i \leq n - 1$)
4. (20 punten) Er is hoogstens één manier om de wegen aan te leggen, zodat men kan reizen tussen twee willekeurige fonteinen door te reizen over de wegen.
5. (20 punten) Er zijn geen vier fonteinen die de hoeken vormen van een 2×2 vierkant.
6. (30 punten) Geen aanvullende voorwaarden.

Voorbeeldgrader

De voorbeeldgrader leest de invoer in het volgende formaat:

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

De voorbeeldgrader schrijft naar de uitvoer in het volgende formaat:

- regel 1: de waarde teruggekregen van `construct_roads`

Als de waarde teruggekregen van `construct_roads` 1 is en `build(u, v, a, b)` is aangeroepen, dan schrijft de grader ook naar de uitvoer:

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