Fountain Parks

Într-un parc din apropiere sunt n **fântâni**, numerotate de la 0 la $n-1$. Modelăm fântânile ca puncte în planul bidimensional. Mai exact, fântâna i ($0 \leq i \leq n-1$) este punctul $(x[i], y[i])$ unde $x[i]$ și $y[i]$ sunt **numere întregi pare**. Locațiile fântânilor sunt toate distincte.

Arhitectul Timothy a fost angajat să proiecteze construirea unor străzi și a **câte o bancă** per stradă. O stradă este un segment de dreaptă **orizontal** sau **vertical** de lungime 2, al cărei capete sunt două fântâni distincte. Străzile trebuie să fie construite astfel încât să se poată călători între oricare două fântâni mergând numai pe străzi. Inițial nu este nicio stradă în parc.

Pentru fiecare stradă, **exact** o bancă trebuie construită și **atribuită** (adică îndreptată spre) respectivei străzi. Fiecare bancă trebuie poziționată la un punct (a, b) astfel încât a și b să fie **numere întregi impare**. Locațiile băncilor trebuie să fie toate **distincte**. O bancă poziționată la (a, b) poate fi atribuită unei străzi numai dacă **ambele** capete ale străzii sunt printre punctele $(a-1, b-1)$, $(a-1, b+1)$, $(a+1, b-1)$ și $(a+1, b+1)$. Spre exemplu, banca la poziția $(3, 3)$ poate fi atribuită numai străzilor reprezentate de unul dintre următoarele patru segmente $(2, 2)$ – $(2, 4)$, $(2, 4)$ – $(4, 4)$, $(4, 4)$ – $(4, 2)$, $(4, 2)$ – $(2, 2)$.

Ajutați-l pe Timothy să determine dacă este posibil să construiască străzi, să poziționeze și să atribuie bănci, care să satisfacă toate condițiile date mai sus, și dacă da, furnizați-i o soluție fezabilă. Dacă există mai multe soluții fezabile care satisfac toate condițiile puteți furniza oricare dintre ele.

Detalii de Implementare

Trebuie să implementați următoarea procedură:

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

- x, y: două tablouri unidimensionale de lungime n fiecare. Pentru fiecare i ($0 \leq i \leq n-1$), fântâna i este un punct $(x[i], y[i])$, unde $x[i]$ și $y[i]$ sunt numere întregi pare.
- Dacă este posibilă o construcție, procedura trebuie să apeleze exact o singură dată funcția `build` (vezi mai jos) pentru a raporta o soluție, după care procedura trebuie să returneze 1.
- Altfel, procedura trebuie să returneze 0 fără a apela funcția `build`.
- Această procedură este apelată exact o dată.

Implementarea voastră poate să apeleze următoarea funcție pentru a furniza o construcție fezabilă a străzilor și pozițiilor băncilor:

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

- Fie m numărul total de străzi ale construcției.
- u, v: două tablouri unidimensionale de lungime m fiecare, reprezentând străzile care vor fi construite. Aceste străzi sunt etichetate de la 0 la $m-1$. Pentru fiecare j ($0 \leq j \leq m-1$), strada j conectează fântânile $u[j]$ și $v[j]$. Fiecare stradă trebuie să fie un segment de dreaptă, vertical sau orizontal, de lungime 2. Oricare două străzi pot avea cel mult un punct comun (o fântână). Odată ce au fost construite străzile, trebuie să se poată călători între oricare două fântâni mergând numai pe străzi.
- a, b: două tablouri unidimensionale de lungime m fiecare, reprezentând băncile. Pentru fiecare j ($0 \leq j \leq m-1$), o bancă este localizată la $(a[j], b[j])$, și este asociată străzii j. Două bănci distincte nu pot avea aceeași poziție.

Example

Exemplul 1

Să considerăm următorul apel:

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

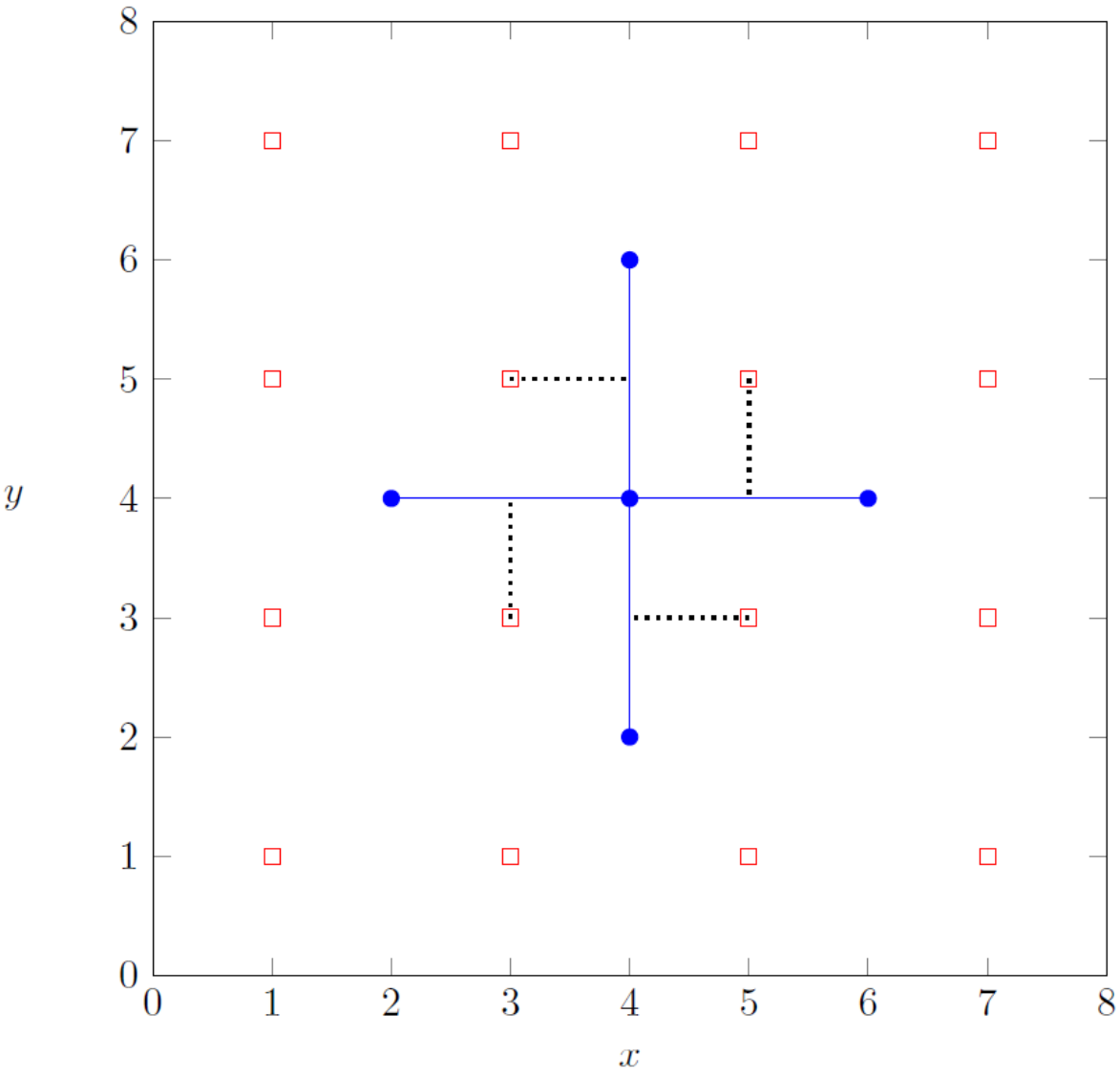
Înseamnă că avem 5 fântâni:

- fântâna 0 este situată la $(4, 4)$,
- fântâna 1 este situată la $(4, 6)$,
- fântâna 2 este situată la $(6, 4)$,
- fântâna 3 este situată la $(4, 2)$,
- fântâna 4 este situată la $(2, 4)$.

Este posibil să construim următoarele 4 străzi, unde fiecare stradă conectează două fântâni, și să poziționăm băncile corespunzătoare:

Eticheta străzii	Etichetele fântânilor conectate de stradă	Poziția băncii asociate
0	$0, 2$	$(5, 5)$
1	$0, 1$	$(3, 5)$
2	$3, 0$	$(5, 3)$
3	$4, 0$	$(3, 3)$

Soluția corespunde următoarei figuri :



Pentru a raporta soluția, `construct_roads` trebuie să facă următorul apel:

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

Apoi acesta ar trebui să returneze 1.

Remarcați că în acest caz, există mai multe soluții care respectă cerințele, oricare dintre ele fiind considerată corectă. De exemplu, este de asemenea corect să se apeleze `build([1, 2, 3, 4], [0, 0, 0, 0], [5, 5, 3, 3], [5, 3, 3, 5])` și apoi să se returneze 1.

Exemplul 2

Să considerăm următorul apel:

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

Fântâna 0 este poziționată la $(2, 2)$ și fântâna 1 este poziționată la $(4, 6)$. Cum nu este posibil să construim străzi care să satisfacă cerințele , `construct_roads` trebuie să returneze 0 fără a face

niciun apel `build`.

Restricții

- $1 \leq n \leq 200\,000$
- $2 \leq x[i], y[i] \leq 200\,000$ (pentru oricare $0 \leq i \leq n-1$)
- $x[i]$ și $y[i]$ sunt numere întregi pare (pentru orice $0 \leq i \leq n-1$).
- Nu există două fântâni la aceeași poziție.

Subtask-uri

1. (5 puncte) $x[i] = 2$ (pentru oricare $0 \leq i \leq n-1$)
2. (10 puncte) $2 \leq x[i] \leq 4$ (pentru oricare $0 \leq i \leq n-1$)
3. (15 puncte) $2 \leq x[i] \leq 6$ (pentru oricare $0 \leq i \leq n-1$)
4. (20 puncte) Există cel mult o modalitate de a construi străzi, astfel încât să se poată călători între oricare două fântâni mergând numai pe străzi.
5. (20 puncte) Nu există patru fântâni care să fie vârfurile unui pătrat 2×2 .
6. (30 puncte) Fără restricții suplimentare.

Exemplul de Grader

Grader-ul citește datele de intrare în următorul format:

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

Grader-ul afișează datele de ieșire în următorul format :

- linia 1: valoare returnată de `construct_roads`

Dacă valoarea returnată de `construct_roads` este 1 și se apelează `build(u, v, a, b)` , grader-ul afișează suplimentar:

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