Szökőkutas park

Egy közeli parkban n **szökőkút** van, 0-tól $n-1$-ig sorszámozva. A szökőkutakat pontokkal modellezzük egy kétdimenziós síkban. Pontosabban, az i. szökőkút ($0 \leq i \leq n-1$) egy $(x[i], y[i])$ pont, ahol $x[i]$ és $y[i]$ **páros egész számok**. Minden szökőkút helye különböző.

Timothy, az építész dolga, hogy megtervezze néhány **út** építését, és minden úthoz egy **pad** elhelyezését. Egy út egy 2 egység hosszúságú **vízszintes** vagy **függőleges** egyenes szakasz, amelynek két végpontja két különböző szökőkút. Az utakat úgy kell megépíteni, hogy bármely két szökőkút elérhető legyen egymásból az utakon át haladva. Kezdetben nincs egyetlen út sem a parkban.

Minden úthoz **pontosan** egy padot kell tenni a parkba, és **hozzá kell rendelni** ahhoz az úthoz (az út felé kell fordítani). Minden padot egy olyan (a, b) pontba kell tenni, amelyre a és b **páratlan egész számok**. Minden padnak **különböző** helyen kell lennie. Egy (a, b) pontban lévő pad csak akkor rendelhető egy úthoz, ha az út **mindkét** végpontja az $(a-1, b-1)$, $(a-1, b+1)$, $(a+1, b-1)$ és $(a+1, b+1)$ pontok közül kerül ki. Például egy pad a $(3, 3)$ pontban csak a következő négy szakasznak megfelelő úthoz rendelhető: $(2, 2)$ – $(2, 4)$, $(2, 4)$ – $(4, 4)$, $(4, 4)$ – $(4, 2)$, $(4, 2)$ – $(2, 2)$.

Segíts Timothy-nak megállapítani, hogy lehetséges-e megépíteni az utakat, valamint elhelyezni és hozzájuk rendelni a padokat úgy, hogy az összes fenti feltétel teljesüljön! Ha igen, adj meg egy lehetséges megoldást! Ha több lehetséges megoldás is van, ami az összes feltételt teljesíti, bármelyiket megadhatod.

Megvalósítás

A következő függvényt kell elkészítened:

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

- x, y: két n elemű tömb. Minden i-re ($0 \leq i \leq n-1$) az i. szökőkút egy $(x[i], y[i])$ pont, ahol $x[i]$ és $y[i]$ páros egész számok.
- Ha van jó konstrukció, pontosan egyszer kell meghívni a `build` függvényt a megoldás megadásához, és ezt követően az 1 értékkel kell visszatérnie ennek a függvénynek.
- Egyébként, a függvény 0 értéket adjon vissza a `build` meghívása nélkül.
- Ez a függvény pontosan egyszer lesz meghívva.

A megoldásod a következő függvényt hívhatja meg, hogy megadja egy megfelelő elhelyezését az utaknak és a padoknak:

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

- Legyen m a megépítendő utak száma.
- u, v: két m elemű tömb, amelyek a megépítendő utakat adják meg. Az utak 0-tól to $m-1$-ig vannak sorszámozva. Minden j-re ($0 \leq j \leq m-1$), a j. út az $u[j]$ és $v[j]$ szökőkutakat köti össze. Minden út csak egy vízszintes vagy függőleges, 2 egység hosszúságú szakasz lehet. Bármely két útnak legfeljebb egy közös pontja lehet (ami egy szökőkút). Az utak megépítése után bármely két szökőkút elérhető kell legyen egymásból az utakon keresztül.
- a, b: két m elemű tömb, amelyek a padokat adják meg. Minden j-re ($0 \leq j \leq m-1$), az $(a[j], b[j])$ pontban van egy pad, ami a j. úthoz van rendelve. Nem lehet két különböző pad ugyanazon a helyen.

Példák

1. Példa

Tekintsük a következő hívást:

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

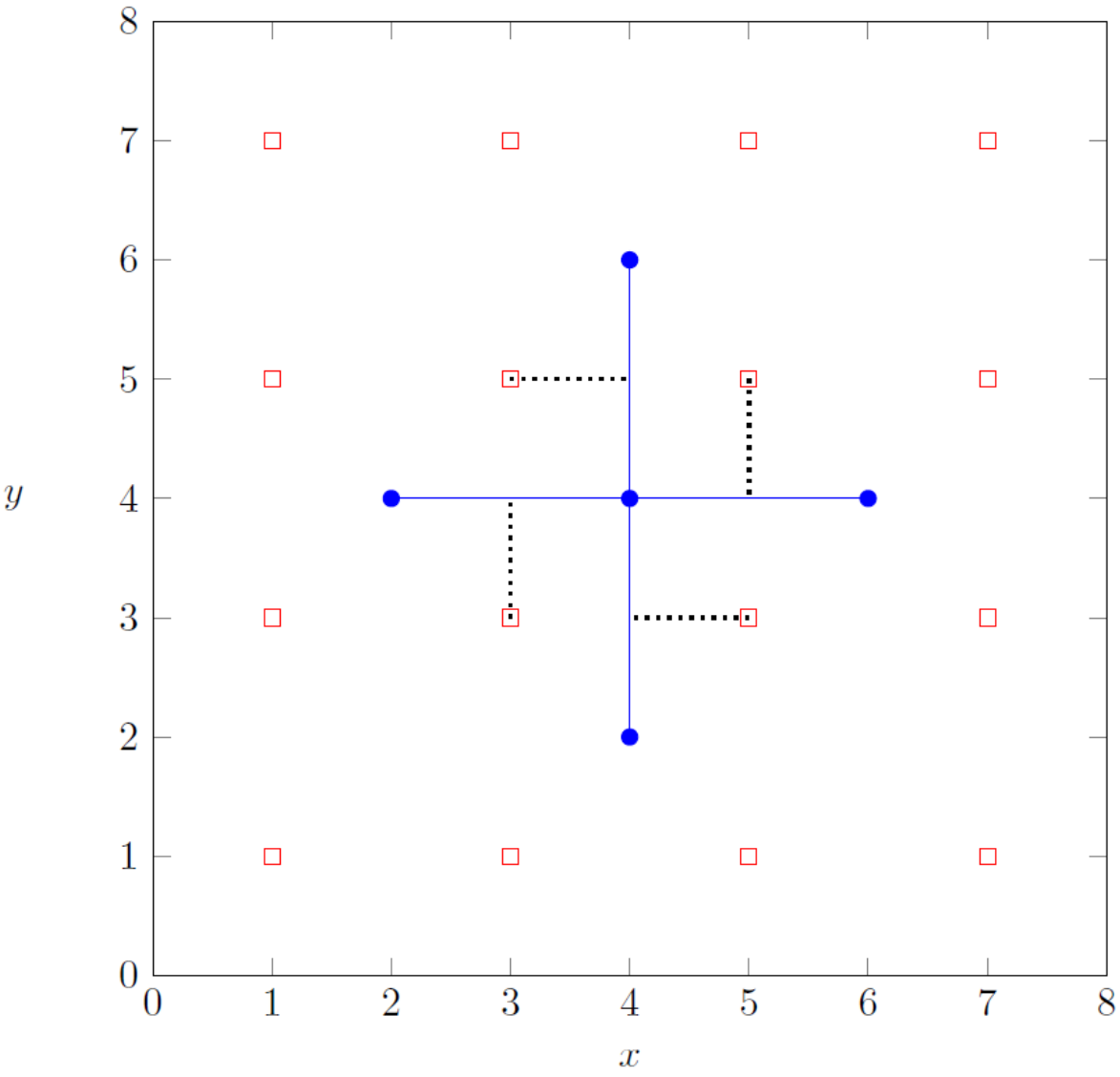
Ez azt jelenti, hogy 5 szökőkút van:

- a 0. szökőkút a $(4, 4)$ pontban van,
- az 1. szökőkút a $(4, 6)$ pontban van,
- a 2. szökőkút a $(6, 4)$ pontban van,
- a 3. szökőkút a $(4, 2)$ pontban van,
- a 4. szökőkút a $(2, 4)$ pontban van.

Megépíthető az alábbi 4 út, bármely két út két szökőkutat köt össze, és el lehet helyezni a hozzájuk tartozó padokat:

Út sorszáma	Összekötött szökőkutak sorszámai	Hozzárendelt pad helye
0	$0, 2$	$(5, 5)$
1	$0, 1$	$(3, 5)$
2	$3, 0$	$(5, 3)$
3	$4, 0$	$(3, 3)$

Ez a megoldás az alábbi ábrának felel meg:



Ennek a megoldásnak a megadásához a `construct_roads` függvényen belül a következő hívást kell végrehajtani:

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

Ezután az 1 értéket kell eredményül adni.

Megjegyezzük, hogy ebben az esetben több lehetséges megoldás is van, ezek mindegyike elfogadott. Például, szintén helyes megoldás a `build([1, 2, 3, 4], [0, 0, 0, 0], [5, 5, 3, 3], [5, 3, 3, 5])` hívás végrehajtása és utána visszatérés az 1 értékkel.

2. példa

Tekintsük a következő hívást:

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

A 0. szökőkút a $(2, 2)$ pontban van és az 1. szökőkút a $(4, 6)$ pontban van. Mivel nem lehet a feltételeknek megfelelően megépíteni az utakat, a `construct_roads` függvénynek 0 értéket kell eredményül adnia anélkül, hogy meghívná a `build` függvényt.

Korlátok

- $1 \leq n \leq 200\,000$
- $2 \leq x[i], y[i] \leq 200\,000$ ($0 \leq i \leq n - 1$)
- $x[i]$ és $y[i]$ páros egész számok ($0 \leq i \leq n - 1$).
- Nincs két szökőkút ugyanabban a pontban.

Részfeladatok

1. (5 pont) $x[i] = 2$ ($0 \leq i \leq n - 1$)
2. (10 pont) $2 \leq x[i] \leq 4$ ($0 \leq i \leq n - 1$)
3. (15 pont) $2 \leq x[i] \leq 6$ ($0 \leq i \leq n - 1$)
4. (20 pont) Legfeljebb egy lehetséges módja van az utak megépítésének úgy, hogy bármely két szökőkút az utakon keresztül elérhető legyen egymásból.
5. (20 pont) Nincs négy olyan szökőkút, amelyek egy 2×2-es négyzet csúcsaiban helyezkednek el.
6. (30 pont) Nincs további korlátozás.

Minta értékelő

A minta értékelő az alábbi formában olvassa a bemenetet:

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

A következő formában írja ki a választ:

- 1. sor: a `construct_roads` függvény visszatérési értéke.

Ha a `construct_roads` függvény 1 értékkel tér vissza és meghívta a `build(u, v, a, b)` függvényt, akkor még az alábbiakat írja ki:

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