Fountain Parks

U obližnjem parku postoji ukupno n **fontana**, označenih brojevima od 0 do $n-1$. Modeliraćemo ove fontane pomoću tačaka u dvodimenzionalnoj ravni. To znači da će fontana i ($0 \leq i \leq n-1$) biti tačka $(x[i], y[i])$ tako da su $x[i]$ i $y[i]$ **parni brojevi**. Lokacije, tačke, fontana su sve različite međusobno.

Tim je arhitekta angažovan da napravi plan izgradnje nekoliko **puteva** kao i plan postavljanja jedne **klupe** na svakom putu. Put je ili **horizontalna** ili **vertikalna** duž dužine 2 čije su krajnje tačke dvije različite fontane. Putevi treba da budu konstruisani tako da neko može putovati između bilo koje dvije fontane koristeći napravljene puteve. Na početku, ne postoji ni jedan put u parku.

Za svaki put, **tačno** jedna klupa mora biti postavljena u parku i **dodjeljena** (drugim riječima, okrenuta prema) određenom putu. Svaka klupa mora biti postavljena na nekoj tački (a, b) tako da su i a i b **neparni brojevi**. Lokacije, tačke, na koje su klupe postavljene moraju sve biti **različite** međusobno. Jedna klupa na tački (a, b) može biti dodjeljena putu ako **obje** krajnje tačke tog puta imaju neku od ovih vrijednosti $(a-1, b-1)$, $(a-1, b+1)$, $(a+1, b-1)$ i $(a+1, b+1)$. Na primjer, klupa na poziciji $(3, 3)$ može biti dodjeljena putu koji je, u stvari, neka od ove četiri duži $(2, 2)$ – $(2, 4)$, $(2, 4)$ – $(4, 4)$, $(4, 4)$ – $(4, 2)$, $(4, 2)$ – $(2, 2)$.

Pomozite Timu da odredi da li je moguće konstruisati puteve, postaviti i dodijeliti klupe tako da su svi postavljeni uslovi zadovoljeni, i ako je to moguće, konstruišite jedno takvo moguće rješenje. Ukoliko postoji više mogućih rješenja koja zadovoljavaju sve postavljene uslove dovoljno je dati samo jedno od njih.

Implementacijski detalji

Trebate implementirati sljedeću proceduru:

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

- x, y: dva niza dužine n. Za svaki i ($0 \leq i \leq n-1$), fontana i je tačka $(x[i], y[i])$, gdje su $x[i]$ i $y[i]$ parni brojevi.
- Ukoliko je cijela konstrukcija moguća ova procedura treba da pozove tačno jednom proceduru `build` (vidjeti detalje poslije) koja će dati jedno rješenje, nakon čega treba da vrati 1.
- U suprotnom, procedura treba da vrati 0 bez i jednog poziva procedure `build`.
- Ova procedura se poziva tačno jednom.

Vaša implementacija može pozvati sljedeću proceduru kako bi efektivno konstruisali moguću mrežu puteva i odredili pozicije svih klupa:

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

- u, v: dva niza dužine m koji predstavljaju puteva koje treba konstruisati. Ovi putevi su označeni brojevima od 0 do $m-1$. Za svaki j ($0 \leq j \leq m-1$), put j povezuje fontane $u[j]$ i $v[j]$. Svaki put je ili horizontalna ili vertikalna duž dužine 2. Neka dva puta mogu imati najviše jednu zajedničku tačku (koja mora biti fontana). Nakon što su svi putevi konstruisani treba da je moguće doći od bilo koje fontane do bilo koje druge fontane.
- a, b: dva niza dužine m koji predstavljaju klupe. Za svako j ($0 \leq j \leq m-1$), klupa je postavljena na tačku $(a[j], b[j])$, i dodjeljena putu j. Nikoje dvije klupe ne mogu imati istu lokaciju. Različite klupe ne mogu biti dodjeljene istom putu.

Primjeri

Primjer 1

Razmotrimo sljedeći poziv:

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

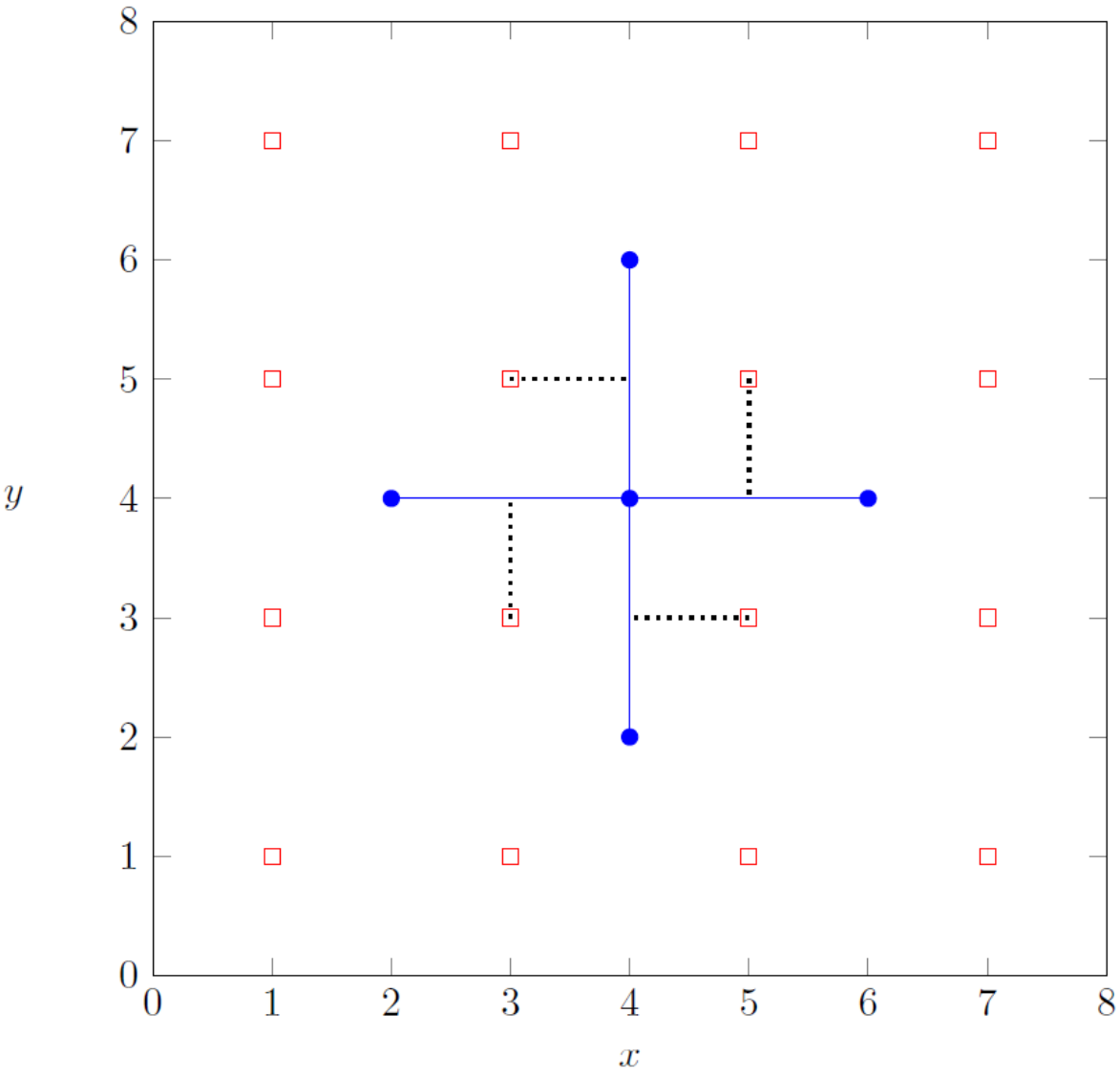
Ovdje znamo da imamo ukupno 5 fontana:

- fontana 0 se nalazi u tački $(4, 4)$,
- fontana 1 se nalazi u tački $(4, 6)$,
- fontana 2 se nalazi u tački $(6, 4)$,
- fontana 3 se nalazi u tački $(4, 2)$,
- fontana 4 se nalazi u tački $(2, 4)$.

Moguće je konstruisati sljedeća 4 puta gdje svaki put povezuje dvije fontane i moguće je postaviti odgovarajuće klupe:

Oznake puteva	Oznake fontana koje putevi povezuju	Pozicija klupa
0	$0, 2$	$(5, 5)$
1	$0, 1$	$(3, 5)$
2	$3, 0$	$(5, 3)$
3	$4, 0$	$(3, 3)$

Ovo rješenje je prikazano na slici koja slijedi:



Kako bi prijavili ovo rješenje potrebno je napraviti sljedeći poziv procedure `build` unutar procedure `construct_roads`:

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

Nakon toga vaša procedura treba vratiti 1.

Primjetimo da u ovom slučaju postoji više različitih rješenja koja zadovoljavaju sve uslove i svako takvo rješenje se smatra ispravnim. Na primjer, moguće je pozvati i `build([1, 2, 3, 4], [0, 0, 0, 0], [5, 5, 3, 3], [5, 3, 3, 5])` pa onda opet vratiti 1.

Primjer 2

Posmatrajmo sljedeći poziv:

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

Fontana 0 se nalazi u tački $(2, 2)$ a fontana 1 u tački $(4, 6)$. Kako ne postoji način da se konstruišu putevi koji zadovoljavaju sve postavljene uslove, `construct_roads` treba da vrati 0 i to bez i jednog poziva funkciju `build`.

Ograničenja

- $1 \leq n \leq 200\,000$
- $2 \leq x[i], y[i] \leq 200\,000$ (za sve $0 \leq i \leq n - 1$)
- $x[i]$ i $y[i]$ su parni brojevi (za sve $0 \leq i \leq n - 1$).
- Nikoje dvije fontane se ne nalaze na istoj lokaciji.

Podzadaci

1. (5 bodova) $x[i] = 2$ (za sve $0 \leq i \leq n - 1$)
2. (10 bodova) $2 \leq x[i] \leq 4$ (za sve $0 \leq i \leq n - 1$)
3. (15 bodova) $2 \leq x[i] \leq 6$ (za sve $0 \leq i \leq n - 1$)
4. (20 bodova) Postoji najviše jedan korektan način konstrukcije puteva, tako da je moguće doći od bilo koje fontana do bilo koje druge fontane koristeći te puteve.
5. (20 bodova) Ne postoje četiri fontane koje se nalaze na vrhovima nekog kvadrata dimenzija 2×2.
6. (30 bodova) Nema dodatnih ograničenja.

Sample Grader

Sample grader čita ulaz koji je dat u sljedećem formatu:

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

Izlaz sample grader je u sljedećem formatu:

- linija 1: vrijednost koju vraća procedura `construct_roads`

Ako je 1 vraćena vrijednost procedure `construct_roads` i ako je procedura `build(u, v, a, b)` bila pozvana, onda će grader odštampati i dodatne linije kako slijedi:

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