Fountain Parks

В близкия парк, има n **фонтана**, номерирани с числата от 0 до $n-1$. Приемаме, че фонтаните са точки в равнината - фонтан i ($0 \leq i \leq n-1$) е точка $(x[i], y[i])$, където $x[i]$ и $y[i]$ са **четни числа**. Позициите на фонтаните са различни.

Архитектът Тимоти е нает да планира построяването на някои **пътища** и поставянето на по една **пейка** за всеки път. Път е **хоризонтална** или **вертикална** отсечка с дължина 2, чиито краища са два различни фонтана. Пътищата трябва да бъдат построени, така че да е възможно придвижването между всеки два фонтана, използвайки само пътищата. В началото няма никакви пътища в парка.

За всеки път, **точно** една пейка ще бъде поставена в парка и **разпределена** за този път. Всяка пейка трябва да бъде поставена в някоя точка (a, b), така че a и b да са **нечетни числа**. Позициите на пейките трябва да бъдат **различни**. Пейка на позиция (a, b) може да бъде разпределена за път ако и **двата** края на пътя са измежду $(a-1, b-1)$, $(a-1, b+1)$, $(a+1, b-1)$ и $(a+1, b+1)$. Например, пейка на позиция $(3, 3)$ може да бъде разпределена само за път, който е някоя от следните отсечки $(2, 2)$ – $(2, 4)$, $(2, 4)$ – $(4, 4)$, $(4, 4)$ – $(4, 2)$, $(4, 2)$ – $(2, 2)$.

Помогнете на Тимоти да определи дали е възможно да построи пътищата и да постави пейки, удовлетворявайки всички поставени условия. Ако е възможно, съобщете му едно възможно решение. Ако има повече от едно възможно решение, спазващо условията, може да съобщите което и да е.

Детайли по реализацията

Трябва да напишете следната функция:

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

- x, y: два масива с дължина n. За всяко i ($0 \leq i \leq n-1$), фонтан i е точка $(x[i], y[i])$, където $x[i]$ и $y[i]$ са четни числа.
- Ако построяването е възможно, тази функция трябва да направи точно едно извикване на функцията `build` (вижте по-надолу), за да съобщи решение, след което трябва да върне 1.
- В противен случай, тази функция трябва да върне 0 без да прави някакви извиквания на `build`.
- Тази функция се вика веднъж.

Може да викате следната функция, за да съобщите възможно решение за построяването на пътищата и поставянето на пейките:

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

- Нека m е общият брой пътища при построяването.
- u, v: два масива с дължина m, описващи пътищата, които трябва да бъдат построени. Тази пътища са номерирани с числата от 0 до $m-1$. За всяко j ($0 \leq j \leq m-1$), път j свързва фонтани $u[j]$ и $v[j]$. Всеки път трябва да бъде хоризонтална или вертикална отсечка с дължина 2. Всеки два различни пътя могат да имат най-много една обща точка (фонтан). След като пътищата са построени, трябва да е възможно придвижването между всеки два фонтана, използвайки само пътищата.
- a, b: два масива с дължина m, описващи пейките. За всяко j ($0 \leq j \leq m-1$), пейка ще бъде поставена на $(a[j], b[j])$ и разпределена за път j. Не може да има две различни пейки на една и съща позиция.

Примери

Пример 1

Нека имаме следното извикване:

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

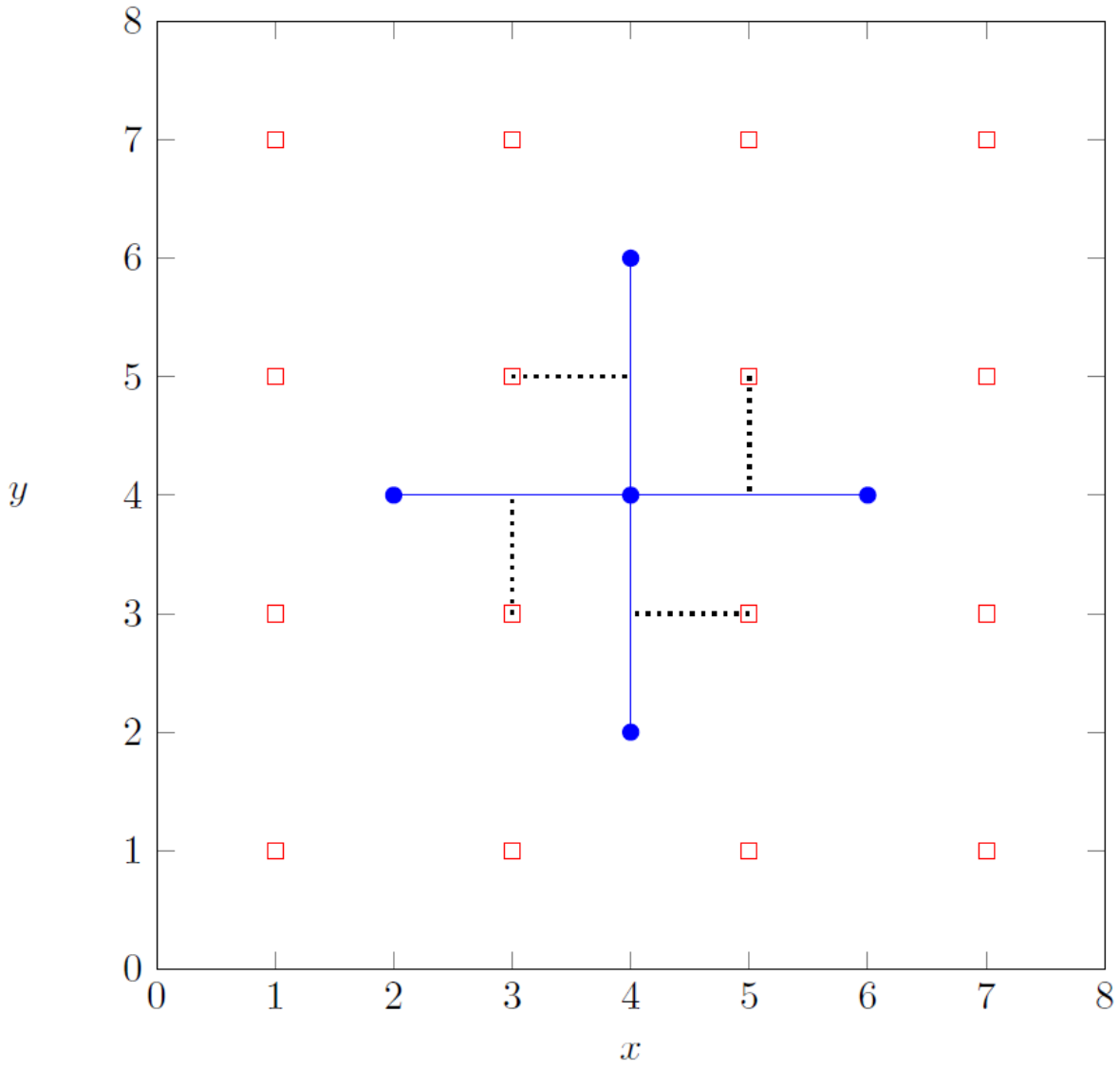
Това означава, че има 5 фонтана:

- фонтан 0 е на позиция $(4, 4)$,
- фонтан 1 е на позиция $(4, 6)$,
- фонтан 2 е на позиция $(6, 4)$,
- фонтан 3 е на позиция $(4, 2)$,
- фонтан 4 е на позиция $(2, 4)$.

Възможно е да построим следните 4 пътя, където всеки път свързва два фонтана, и да поставим съответните пейки по следния начин:

Номер на път	Номер на фонтани, които свързва пътят	Позиция на разпределената пейка
0	$0, 2$	$(5, 5)$
1	$0, 1$	$(3, 5)$
2	$3, 0$	$(5, 3)$
3	$4, 0$	$(3, 3)$

Решението съответства на следната диаграма:



За да съобщите решението, `construct_roads` трябва да направи следното извикване:

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

След което трябва да върне 1.

Обърнете внимание, че в този случай има повече от едно решение, спазващо изискванията, и всяко от тях ще се зачита за вярно. Например, извикването `build([1, 2, 3, 4], [0, 0, 0, 0], [5, 5, 3, 3], [5, 3, 3, 5])` също ще се зачете за вярно.

Пример 2

Нека имаме следното извикване:

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

Фонтан 0 е на позиция $(2, 2)$, а фонтан 1 е на позиция $(4, 6)$. Тъй като няма как да се построят пътища, спазващи условията, функцията `construct_roads` трябва да върне 0 без да

извиква функцията `build`.

Ограничения

- $1 \leq n \leq 200\,000$
- $2 \leq x[i], y[i] \leq 200\,000$ (за всяко $0 \leq i \leq n-1$)
- $x[i]$ и $y[i]$ са четни числа (за всяко $0 \leq i \leq n-1$).
- Няма два фонтана на една и съща позиция.

Подзадачи

1. (5 точки) $x[i] = 2$ (за всяко $0 \leq i \leq n-1$)
2. (10 точки) $2 \leq x[i] \leq 4$ (за всяко $0 \leq i \leq n-1$)
3. (15 точки) $2 \leq x[i] \leq 6$ (за всяко $0 \leq i \leq n-1$)
4. (20 точки) Имат най-много един начин за построяване на пътищата, така че да е възможно придвижването между всеки два фонтана, използвайки пътищата.
5. (20 точки) Не съществуват четири фонтана, които са върхове на квадрат с размери 2×2.
6. (30 точки) Няма допълнителни ограничения.

Примерен грейдър

Примерният грейдър чете от стандартния вход в следния формат:

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

Изходът на примерния грейдър е в следния формат:

- ред 1: върнатата стойност на `construct_roads`

Ако върнатата стойност на `construct_roads` е 1 и `build(u, v, a, b)` е извикана, грейдърът допълнително отпечатва:

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