Fontane e panchine

Nel parchetto sotto casa ci sono n **fontane**, per cui la fontana i ($0 \leq i \leq n-1$) si trova nel punto $(x[i], y[i])$, dove $x[i]$ e $y[i]$ sono **interi pari**. L'architetto Timothy ora deve costruire alcuni **sentieri** tra le fontane, mettendo una **panchina** per ogni sentiero.

Più precisamente, un sentiero è una linea **orizzontale** o **verticale** di lunghezza esattamente 2 i cui estremi sono due fontane, e devono essere costruiti di modo che sia possibile muoversi tra ogni due fontane (inizialmente non ci sono sentieri).

Ad ogni sentiero deve essere assegnata **esattamente** una panchina (affacciandola ad esso, vedi immagine dell'esempio 1). Le panchine sono posizionate in punti **distinti** (a, b) con a e b **interi dispari**, e una tale panchina può essere assegnata ad un sentiero solo se **entrambi** i suoi estremi sono nel quadrato 2×2 che racchiude il punto (a, b), e quindi sono tra $(a-1, b-1)$, $(a-1, b+1)$, $(a+1, b-1)$ e $(a+1, b+1)$. Per esempio, una panchina in $(3, 3)$ può essere assegnata ad una dei quattro sentieri $(2, 2)$ – $(2, 4)$, $(2, 4)$ – $(4, 4)$, $(4, 4)$ – $(4, 2)$, $(4, 2)$ – $(2, 2)$.

Trova un **piano ammissibile** di costruzione di sentieri e panchine (che soddisfi tutte le condizioni sopra), se questo esiste. Se ci sono più piani ammissibili, puoi riportare uno qualunque di essi.

Note di implementazione

Devi implementare la seguente funzione:

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

- x, y: array di lunghezza n, per cui $(x[i], y[i])$ ($0 \leq i \leq n-1$) è la posizione della i-esima fontana.
- Se esiste un piano ammissibile, la funzione deve fare esattamente una chiamata a `build` (vedi sotto) e poi restituire 1.
- Altrimenti, deve restituire 0 senza fare chiamate a `build`.
- Questa funzione è chiamata esattamente una volta.

Il tuo programma può chiamare la seguente funzione per riportare un piano di costruzione ammissibile di sentieri e panchine:

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

- Sia m il numero totale di sentieri nel tuo piano.
- u, v: array di lunghezza m che rappresentano i sentieri da costruire: per ogni j ($0 \leq j \leq m-1$), il sentiero j collega le fontane di indice $u[j]$ e $v[j]$. Ogni sentiero deve essere una linea orizzontale o verticale di lunghezza 2, i sentieri devono collegare tutte le fontane, e ogni due sentieri possono avere in comune al massimo un punto (una fontana).
- a, b: array di lunghezza m che rappresentano le panchine: per ogni j ($0 \leq j \leq m-1$), una panchina è posizionata in $(a[j], b[j])$ affacciandosi sul sentiero j. Non ci possono essere due panchine nello stesso punto.

Esempi

Esempio 1

Considera la seguente chiamata:

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

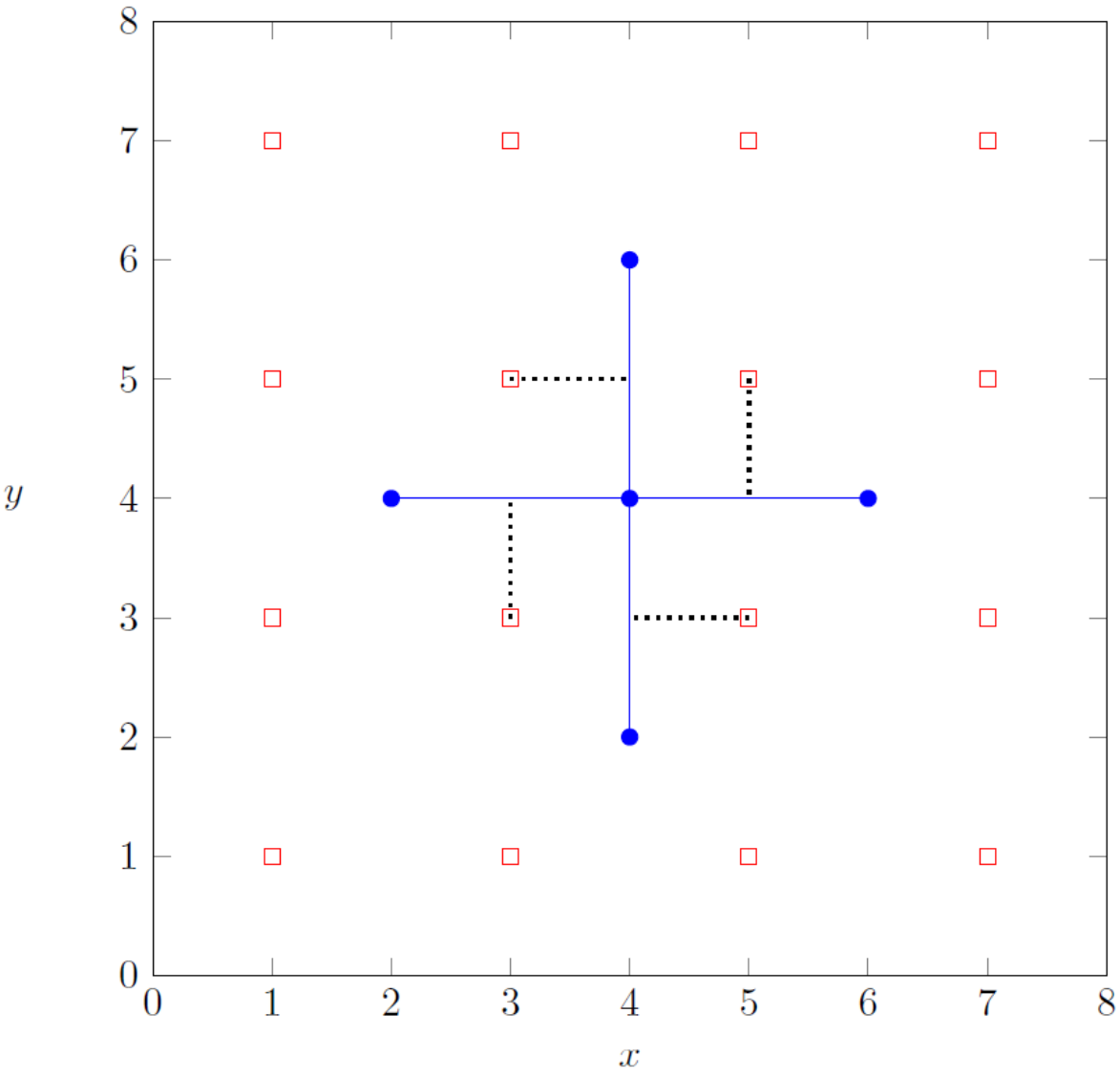
In questo caso, ci sono 5 fontane:

- la fontana 0 è nel punto $(4, 4)$,
- la fontana 1 è nel punto $(4, 6)$,
- la fontana 2 è nel punto $(6, 4)$,
- la fontana 3 è nel punto $(4, 2)$,
- la fontana 4 è nel punto $(2, 4)$.

Un piano ammissibile è costruire le seguenti 4 sentieri e panchine:

Strada	Fontane `(u,v)`	Panchina `(a,b)`
0	$0, 2$	$(5, 5)$
1	$0, 1$	$(3, 5)$
2	$3, 0$	$(5, 3)$
3	$4, 0$	$(3, 3)$

Questa soluzione può essere visualizzata come:



Per riportare questa soluzione, `construct_roads` deve fare la chiamata:

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

e poi restituire 1.

Nota che in questo caso ci sono più piani ammissibili, e tutti sono considerati corretti. Per esempio, andrebbe bene anche chiamare `build([1, 2, 3, 4], [0, 0, 0, 0], [5, 5, 3, 3], [5, 3, 3, 5])` e poi restituire 1.

Esempio 2

Considera la seguente chiamata:

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

La fontana 0 è nel punto $(2, 2)$ mentre la fontana 1 è nel punto $(4, 6)$. Dato che non esistono modi ammissibili per costruire sentieri e panchine, `construct_roads` deve restituire 0 senza fare chiamate a `build`.

Assunzioni

- $1 \leq n \leq 200\,000$.
- $2 \leq x[i], y[i] \leq 200\,000$ (per ogni $0 \leq i \leq n - 1$).
- $x[i]$ e $y[i]$ sono pari (per ogni $0 \leq i \leq n - 1$).
- Non ci sono due fontane nello stesso punto.

Subtask

1. (5 punti) $x[i] = 2$ (per ogni $0 \leq i \leq n - 1$).
2. (10 punti) $2 \leq x[i] \leq 4$ (per ogni $0 \leq i \leq n - 1$).
3. (15 punti) $2 \leq x[i] \leq 6$ (per ogni $0 \leq i \leq n - 1$).
4. (20 punti) Esiste al più un modo per costruire sentieri che connettono tutte le fontane.
5. (20 punti) Non esistono quattro fontane ai vertici di un quadrato 2×2.
6. (30 punti) Nessuna limitazione aggiuntiva.

Grader di esempio

Il grader di esempio legge l'input nel seguente formato:

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

Il grader di esempio stampa l'output nel seguente formato:

- riga 1: il valore restituito da `construct_roads`

Se il valore restituito da `construct_roads` è 1 e hai chiamato `build(u, v, a, b)`, il grader stampa anche:

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