Brunnenpark

In einem nahegelegenen Park gibt es n **Brunnen**, nummeriert von 0 bis $n-1$. Wir modellieren die Brunnen als Punkte im zweidimensionalen Raum. Der Brunnen i ($0 \leq i \leq n-1$) entspricht dem Punkt $(x[i], y[i])$, wobei $x[i]$ und $y[i]$ **gerade Zahlen** sind. Alle Brunnen befinden sich auf unterschiedlichen Punkten.

Architekt Tim wurde beauftragt, einige **Wege** zu bauen und zu jedem Weg eine zugehörige **Bank** zu errichten. Ein Weg entspricht einer **horizontalen** oder **vertikalen** Strecke der Länge 2, welche zwei verschiedene Brunnen verbindet. Über die gebauten Wege soll es möglich sein, von jedem Brunnen zu jedem anderen Brunnen zu gelangen. Zu Beginn hat der Park keine Wege.

Zu jedem Weg muss **genau** eine Bank im Park platziert und diesem Weg **zugewiesen** (auf ihn ausgerichtet) werden. Jede Bank soll auf einem Punkt (a, b) positioniert werden, wobei a und b **ungerade Zahlen** sind. Die Positionen der Bänke müssen alle **verschieden** sein. Eine Bank an Position (a, b) kann nur einem Weg zugewiesen werden, dessen **beide** Endpunkte einer der Punkte $(a-1, b-1)$, $(a-1, b+1)$, $(a+1, b-1)$ oder $(a+1, b+1)$ sind. Zum Beispiel kann die Bank auf $(3, 3)$ nur einem der folgenden Wege zugewiesen werden: $(2, 2)$ – $(2, 4)$, $(2, 4)$ – $(4, 4)$, $(4, 4)$ – $(4, 2)$, $(4, 2)$ – $(2, 2)$.

Hilf Tim zu bestimmen, ob es möglich ist, für gegebene Brunnen die Wege und Bänke so zu planen, dass alle obigen Bedingungen erfüllt sind. Falls ja, finde für ihn eine solche Planung, die wir auch als "Lösung" bezeichnen. Sollte es mehrere mögliche Lösungen geben, kannst du eine beliebige davon ausgeben.

Implementierung

Implementiere die folgende Funktion:

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

- x, y: Zwei Arrays der Länge n. Für jedes i ($0 \leq i \leq n-1$) ist Brunnen i auf Punkt $(x[i], y[i])$, und $x[i]$ und $y[i]$ sind gerade Zahlen.
- Falls es eine Lösung gibt, soll die Funktion genau einmal `build` (siehe unten) aufrufen, um die Lösung einzureichen, und daraufhin 1 zurückgeben.
- Ansonsten soll die Funktion 0 zurückgeben, ohne vorher `build` aufzurufen.
- Die Funktion wird genau einmal aufgerufen.

Deine Implementierung kann die folgende Funktion aufrufen, um eine Lösung, also eine mögliche Platzierung der Wege und Bänke einzureichen:

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

- Sei m die Anzahl Wege in der Lösung.
- u, v: Zwei Arrays der Länge m, welche die Wege beschreiben. Die Wege sind nummeriert von 0 bis $m-1$. Für jedes j ($0 \leq j \leq m-1$) verbindet Weg j die Brunnen $u[j]$ und $v[j]$. Jeder Weg muss eine horizontale oder vertikale Strecke der Länge 2 sein. Zwei verschidene Wege dürfen höchstens einen gemeinsamen Punkt besitzen (nämlich einen Brunnen). Über die so beschriebenen Wege sollte es möglich sein, von jedem Brunnen aus zu jedem anderen Brunnen zu gelangen.
- a, b: Zwei Arrays der Länge m, welche die Bänke beschreiben. Für jedes j ($0 \leq j \leq m-1$), wird eine Bank auf $(a[j], b[j])$ platziert, und dem Weg j zugewiesen. Keine zwei verschiedenen Bänke dürfen die gleiche Position haben.

Beispiele

Beispiel 1

Betrachte den folgenden Aufruf:

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

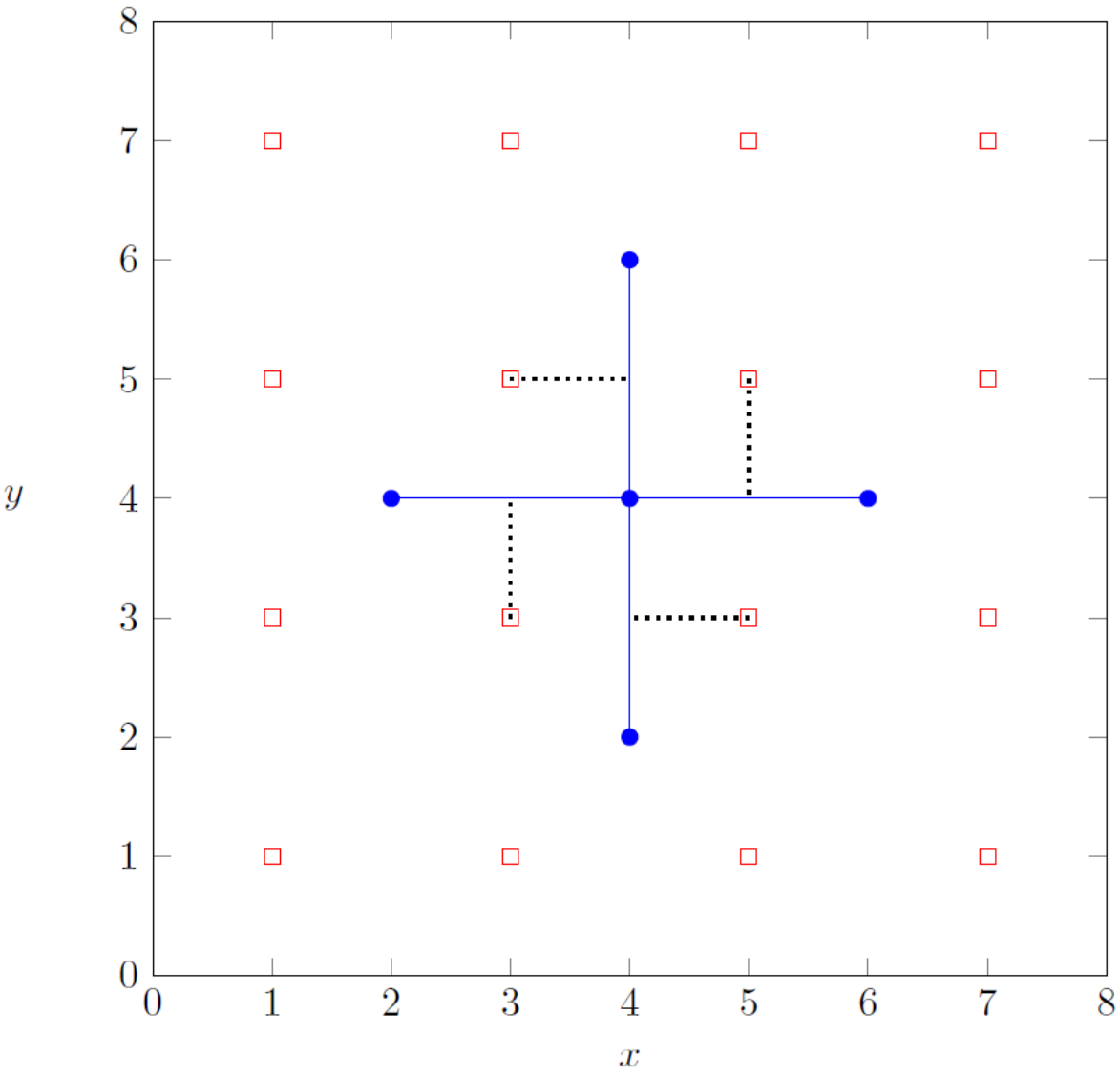
Dies bedeutet, dass es 5 Brunnen gibt:

- Brunnen 0 befindet sich auf Position $(4, 4)$,
- Brunnen 1 befindet sich auf Position $(4, 6)$,
- Brunnen 2 befindet sich auf Position $(6, 4)$,
- Brunnen 3 befindet sich auf Position $(4, 2)$,
- Brunnen 4 befindet sich auf Position $(2, 4)$.

Es gibt eine Lösung mit den folgenden 4 Wegen, die jeweils zwei Brunnen verbinden, und mit entsprechend platzierten Bänken:

Nummer des Wegs	Nummern der zwei Brunnen auf den Endpunkten	Position der zugewiesenen Bank
0	$0, 2$	$(5, 5)$
1	$0, 1$	$(3, 5)$
2	$3, 0$	$(5, 3)$
3	$4, 0$	$(3, 3)$

Diese Lösung entspricht folgendem Bild:



Um diese Lösung einzureichen, soll `construct_roads` die Funktion `build` wie folgt aufrufen:

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

Anschließend soll 1 zurückgegeben werden.

Beachte, dass es in diesem Beispiel mehrere mögliche Lösungen gibt und alle davon als richtig bewertet werden. Zum Beispiel ist es ebenfalls korrekt, `build([1, 2, 3, 4], [0, 0, 0, 0], [5, 5, 3, 3], [5, 3, 3, 5])` aufzurufen und danach 1 zurückzugeben.

Beispiel 2

Betrachte den folgenden Aufruf:

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

Brunnen 0 befindet sich an Position $(2, 2)$ und Brunnen 1 an Position $(4, 6)$. Da es nicht möglich ist, die Wege so zu bauen, dass die Bedingungen erfüllt sind, soll `construct_roads` sofort 0

zurückgeben, ohne vorher `build` aufzurufen.

Beschränkungen

- $1 \leq n \leq 200\,000$
- $2 \leq x[i], y[i] \leq 200\,000$ (für alle $0 \leq i \leq n-1$)
- $x[i]$ und $y[i]$ sind gerade Zahlen (für alle $0 \leq i \leq n-1$).
- Keine zwei Brunnen haben die gleiche Position.

Subtasks

1. (5 Punkte) $x[i] = 2$ (für alle $0 \leq i \leq n-1$)
2. (10 Punkte) $2 \leq x[i] \leq 4$ (für alle $0 \leq i \leq n-1$)
3. (15 Punkte) $2 \leq x[i] \leq 6$ (für alle $0 \leq i \leq n-1$)
4. (20 Punkte) Es gibt höchstens eine Möglichkeit, die Wege so zu planen, dass alle Brunnen verbunden sind.
5. (20 Punkte) Keine vier Brunnen befinden sich auf den Eckpunkten eines 2×2-Quadrats.
6. (30 Punkte) Keine weiteren Beschränkungen.

Beispiel-Grader

Der Beispiel-Grader liest die Eingabe in folgendem Format:

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

Die Ausgabe des Beispiel-Graders ist im folgenden Format:

- Zeile 1: der Rückgabewert von `construct_roads`

Falls der Rückgabewert von `construct_roads` den Wert 1 hat und `build(u, v, a, b)` aufgerufen wurde, gibt der Grader zusätzlich noch aus:

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