Parque de Fuentes

En un parque cercano, hay n **fuentes**, etiquetadas de 0 a $n-1$. Se modelan las fuentes como puntos en un plano bi-dimensional. Es decir, la fuente i ($0 \leq i \leq n-1$) es un punto $(x[i], y[i])$ donde $x[i]$ y $y[i]$ son **enteros pares**. Las posiciones de las fuentes son todas distintas.

Timothy el arquitecto ha sido contratado para planificar la construcción de algunos **caminos** y la posición de una **banca** por camino. Un camino es un segmento de línea **horizontal** o **vertical** de largo 2, cuyos puntos de llegada y partida son dos fuentes distintas. Los caminos deberán ser construidos de tal manera que una persona pueda viajar entre cualquier pareja de fuentes moviéndose a través de los caminos. Inicialmente, no hay caminos en el parque.

Para cada camino, **exactamente** una banca deberá ser ubicada en el parque y **asignada a** (es decir, viendo hacia) ese camino. Cada banca debe ser colocada en algún punto (a, b) tal que a y b sean **enteros impares**. Las ubicaciones de las bancas deben ser todas **distintas**. Una banca en (a, b) puede ser asignada a un camino si **ambos** extremos del camino están entre $(a-1, b-1)$, $(a-1, b+1)$, $(a+1, b-1)$ y $(a+1, b+1)$. Por ejemplo, la banca en $(3, 3)$ puede ser asignada únicamente al camino que sea formado por uno de los cuatro segmentos de línea $(2, 2)$ – $(2, 4)$, $(2, 4)$ – $(4, 4)$, $(4, 4)$ – $(4, 2)$, $(4, 2)$ – $(2, 2)$.

Ayuda a Timothy a determinar si es posible construir caminos, y ubicar y asignar bancas que satisfagan todas las condiciones dadas y, de ser así, darle una solución posible. Si existen multiples soluciones que satisfagan todas las condiciones, puedes darle cualquiera de ellas.

Detalles de Implementación

Deberás implementar la siguiente función:

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

- x, y: dos arreglos de tamaño n. Para cada i ($0 \leq i \leq n-1$), la fuente i es un punto $(x[i], y[i])$, donde $x[i]$ e $y[i]$ son enteros positivos.
- Si una construcción es posible, esta función deberá hacer exactamente una llamada a la función `build` (ver abajo) para reportar una solución y luego deberá retornar 1.
- De lo contrario, la función deberá retornar 0 sin hacer llamadas a `build`.
- Esta función es llamada exactamente una vez.

Tu implementación puede llamar la siguiente función para dar una posible construcción de caminos y posiciones de bancas:

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

- Sea m el número total de caminos en la construcción.
- u, v: dos arreglos de tamaño m, representando los caminos a ser construidos. Estos caminos son etiquetados de 0 a $m-1$. Para cada j ($0 \leq j \leq m-1$), el camino j conecta a las fuentes $u[j]$ y $v[j]$. Cada camino deberá ser un segmento de recta horizontal o vertical de largo 2. Cualquier pareja distinta de caminos deberán tener a lo sumo un punto en común (una fuente). Una vez los caminos son construidos, deberá ser posible viajar entre cualquier pareja de fuentes moviéndose a través de los caminos.
- a, b: dos arreglos de tamaño m, representando las bancas. Para cada j ($0 \leq j \leq m-1$), una banca es colocada en $(a[j], b[j])$, y es asignada al camino j. No existen dos bancas distintas con la misma ubicación.

Ejemplos

Ejemplo 1

Considera la siguiente llamada:

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

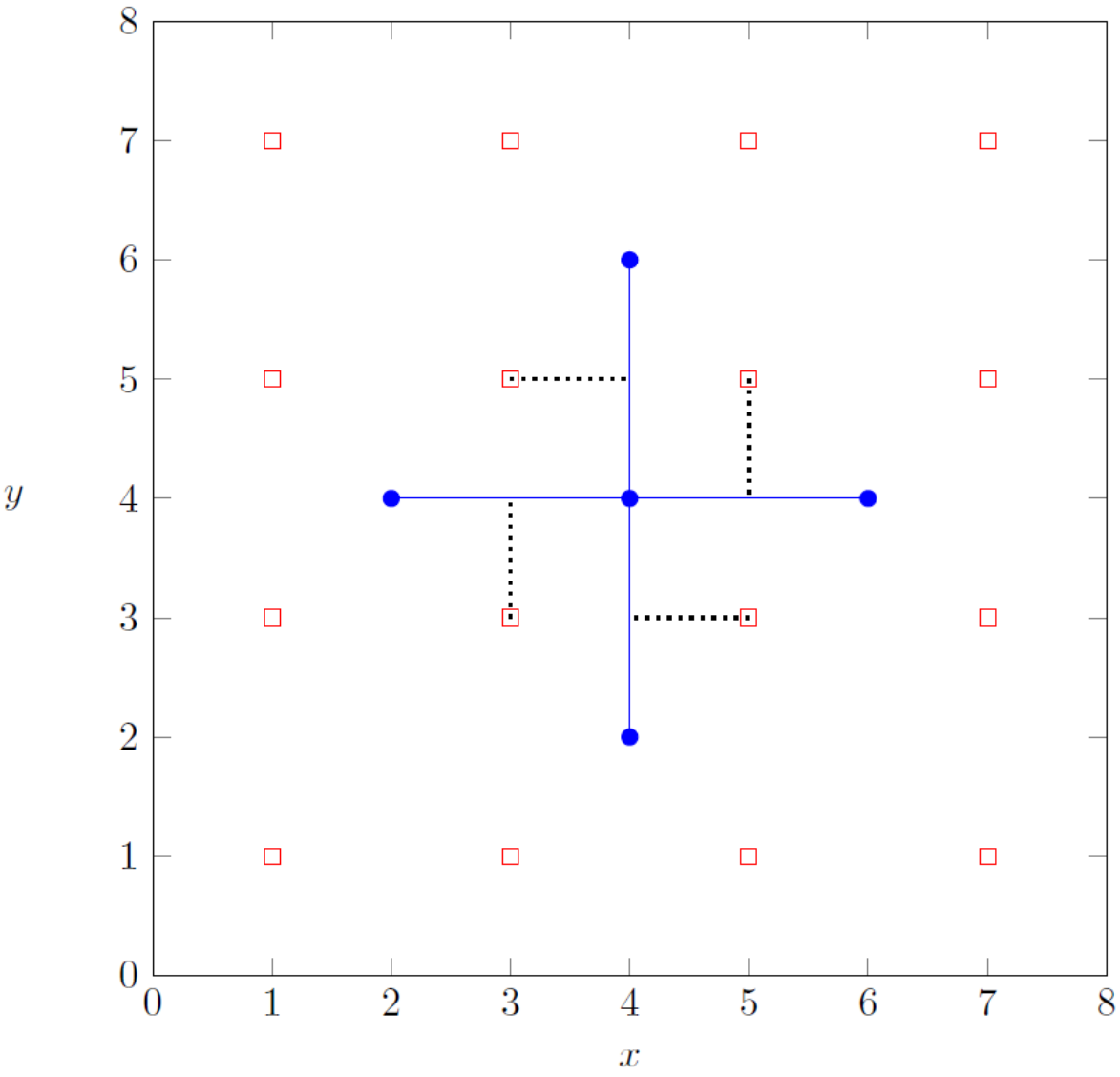
Esto significa que hay 5 fuentes:

- la fuente 0 se ubica en $(4, 4)$,
- la fuente 1 se ubica en $(4, 6)$,
- la fuente 2 se ubica en $(6, 4)$,
- la fuente 3 se ubica en $(4, 2)$,
- la fuente 4 se ubica en $(2, 4)$.

Es posible construir los siguientes 4 caminos, donde cada camino conecta dos fuentes, y colocar las bancas correspondientes:

Camino	Etiquetas de las fuentes conectadas por un camino	Ubicación de la banca asignada
0	$0, 2$	$(5, 5)$
1	$0, 1$	$(3, 5)$
2	$3, 0$	$(5, 3)$
3	$4, 0$	$(3, 3)$

Esta solución corresponde al siguiente diagrama:



Para reportar esta solución, `construct_roads` deberá hacer la siguiente llamada:

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

Y luego deberá retornar 1.

Note que en este caso, hay múltiples soluciones que satisfacen los requisitos y todas ellas podrán ser consideradas como correctas. Por ejemplo, también es correcto llamar `build([1, 2, 3, 4], [0, 0, 0, 0], [5, 5, 3, 3], [5, 3, 3, 5])` y luego retornar 1.

Ejemplo 2

Considere la llamada siguiente:

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

La fuente 0 se ubica en $(2, 2)$ y la fuente 1 se ubica en $(4, 6)$. Ya que no hay forma de construir caminos que satisfagan los requisitos, `construct_roads` deberá retornar 0 sin haber llamado a

`build`.

Restricciones

- $1 \leq n \leq 200\,000$
- $2 \leq x[i], y[i] \leq 200\,000$ (para todo $0 \leq i \leq n-1$)
- $x[i]$ e $y[i]$ son enteros positivos (para todo $0 \leq i \leq n-1$).
- No existen dos fuentes que tengan la misma ubicación.

Subtareas

1. (5 puntos) $x[i] = 2$ (para todo $0 \leq i \leq n-1$)
2. (10 puntos) $2 \leq x[i] \leq 4$ (para todo $0 \leq i \leq n-1$)
3. (15 puntos) $2 \leq x[i] \leq 6$ (para todo $0 \leq i \leq n-1$)
4. (20 puntos) Existe a lo sumo una forma de construir los caminos, de tal forma que pueda viajarse entre cualquier pareja de fuentes moviéndose por los caminos.
5. (20 puntos) No existen cuatro fuentes que formen las esquinas de un cuadrado de 2×2.
6. (30 puntos) Sin restricciones adicionales.

Calificador de Ejemplo

El calificador de ejemplo lee la entrada en el siguiente formato:

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

La salida del calificador de ejemplo está en el siguiente formato:

- línea 1: el valor de retorno de `construct_roads`

Si el valor de retorno de `construct_roads` es 1 y `build(u, v, a, b)` es llamado, el calificador imprime adicionalmente:

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