Fuentes en el parque

En un parque cercano, hay n **fuentes**, etiquetadas de 0 a $n-1$. Las fuentes están representadas como puntos en un plano bidimensional. Es decir, la fuente i ($0 \leq i \leq n-1$) es el punto $(x[i], y[i])$ donde $x[i]$ e $y[i]$ son **enteros pares**. Se garantiza que todas las ubicaciones de las fuentes son distintas.

Timothy, el arquitecto, ha sido contratado para diseñar la construcción de algunos **senderos** y el colocado de una **banca** por sendero. Un sendero es un segmento de línea **horizontal** o **vertical** de longitud 2, cuyos extremos son dos fuentes distintas. Los senderos deben ser construidos de tal manera que se pueda viajar entre dos fuentes cualquiera moviéndose a lo largo de los senderos. Inicialmente no hay senderos en el parque.

Para cada sendero, se debe colocar **exactamente** una banca en el parque y **asignarse a** (es decir, estar al lado de) dicho sendero. Cada banca debe ser colocada en algún punto (a, b) tal que a y b sean **enteros impares**. Las bancas tienen que colocarse en ubicaciones **distintas**. Una banca en (a, b) solo puede ser asignada a un sendero si es que **ambos** extremos del sendero se encuentran entre $(a-1, b-1)$, $(a-1, b+1)$, $(a+1, b-1)$ y $(a+1, b+1)$. Por ejemplo, La banca en $(3, 3)$ solo puede ser asignada a un sendero, el cual es 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 senderos, colocar y asignar bancas que satisfagan todas las condiciones descritas anteriormente, y de ser así, proveerle de una solución factible. Si es que hay múltiples soluciones que satisfagan todas las condiciones, puedes reportar cualquiera de ellas.

Detalles de Implementación

Debes 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 $x[i]$ y $y[i]$ son enteros pares.
- Si una construcción es posible, este procedimiento deberá realizar exactamente una llamada a `build` (véase más abajo) para reportar una solución, después de ello, se deberá retornar 1.
- Caso contrario, el procedimiento deberá retornar 0 sin hacer ninguna llamada a `build`.
- Este procedimiento es llamado exactamente una vez.

Tu implementación puede llamar al siguiente procedimiento para proporcionar una construcción de senderos y colocación de bancas factible:

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

- Sea m el número total de senderos en construcción.
- u, v: dos arreglos de tamaño m, que representan los senderos a ser construidos. Estos senderos están etiquetados de 0 a $m-1$. Para cada j ($0 \leq j \leq m-1$), el sendero j conecta las fuentes $u[j]$ y $v[j]$. Cada sendero debe ser un segmento de línea de tamaño 2. Cualquier par de senderos distintos puede tener como máximo un punto en común (una fuente). Una vez que los senderos hayan sido construidos, debería de ser posible viajar entre dos fuentes cualesquiera moviéndose a lo largo de los senderos.
- a, b: dos arreglos de tamaño m, que representan a las bancas. Para cada j ($0 \leq j \leq m-1$), se coloca una banca $(a[j], b[j])$, y se asigna al sendero j. No pueden haber dos bancas distintos que tengan la misma ubicación.

Ejemplos

Ejemplo 1

Considere la siguiente llamada:

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

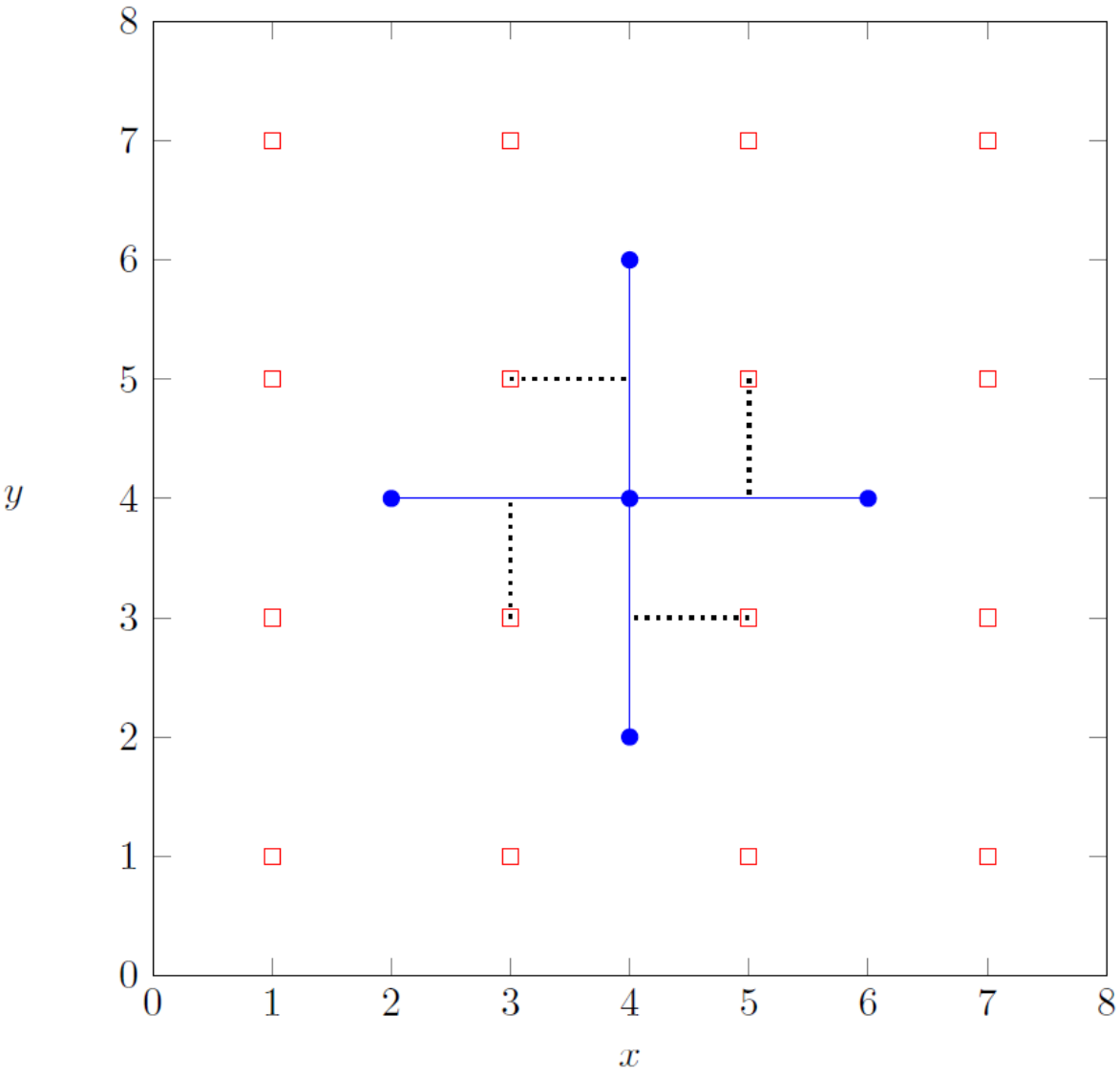
Esto significa que existen 5 fuentes:

- la fuente 0 está ubicada en $(4, 4)$,
- la fuente 1 está ubicada en $(4, 6)$,
- la fuente 2 está ubicada en $(6, 4)$,
- la fuente 3 está ubicada en $(4, 2)$,
- la fuente 4 está ubicada en $(2, 4)$.

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

Sendero	Fuentes que el sendero conecta	Ubicación asignada a la banca
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` debe realizar la siguiente llamada:

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

Y luego debe retornar 1.

tenga en cuenta que en este caso existen múltiples soluciones que satisfacen las condiciones, todas las cuales se considerarían correctas. Por ejemplo, también es correcto llamar a `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 siguiente llamada:

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

La fuente 0 está ubicada en $(2, 2)$ y la fuente 1 está ubicada en $(4, 6)$. Dado que no hay forma de construir senderos que satisfagan las condiciones, `construct_roads` debe retornar 0 sin hacer

ninguna llamada a `build`.

Restricciones

- $1 \leq n \leq 200\,000$
- $2 \leq x[i], y[i] \leq 200\,000$ (para todo $0 \leq i \leq n-1$)
- $x[i]$ y $y[i]$ son enteros pares (para todo $0 \leq i \leq n-1$).
- No hay 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 points) Hay como máximo una forma de construir los senderos, de tal forma que se pueda viajar entre dos fuentes cualquiera moviéndose por los senderos.
5. (20 points) No existen cuatro fuentes que formen las esquinas de un cuadrado de tamaño 2×2.
6. (30 points) Sin restricciones adicionales.

Evaluador de ejemplo

El evaluador 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 evaluador de ejemplo tiene el siguiente formato:

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

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

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