Parques de fuentes

En un parque cercano, hay n **fuentes**, numeradas desde 0 hasta $n-1$. Modelamos las fuentes como puntos en el plano 2D. Específicamente, la fuente i ($0 \leq i \leq n-1$) es un punto $(x[i], y[i])$ donde $x[i]$ e $y[i]$ son **enteros pares**. Las ubicaciones de las fuentes son todas diferentes.

El arquitecto Timoteo fue contratado para planificar la construcción de algunos **senderos** y la colocación de un **banco** por sendero. Un sendero es un segmento de línea **horizontal** o **vertical** de longitud 2, cuyos extremos son dos fuentes distintas. Los senderos deben construirse de modo tal que sea posible viajar entre cualquier par de fuentes mediante movimientos por los senderos. En un comienzo, no hay ningún sendero en el parque.

Por cada sendero, se debe colocar en el parque **exactamente** un banco, y este banco **se debe asignar** al sendero (es decir, el banco debe estar frente al sendero). Cada banco debe colocarse en algún punto (a, b) tal que a y b sean **enteros impares**. Las ubicaciones de los bancos deben ser todas **distintas**. Un banco en (a, b) solamente puede asignarse a un sendero si **ambos** extremos del sendero están entre los puntos $(a-1, b-1)$, $(a-1, b+1)$, $(a+1, b-1)$ y $(a+1, b+1)$. Por ejemplo, el banco en $(3, 3)$ solamente puede asignarse a un sendero que sea uno de los cuatro segmentos $(2, 2)$ – $(2, 4)$, $(2, 4)$ – $(4, 4)$, $(4, 4)$ – $(4, 2)$, $(4, 2)$ – $(2, 2)$.

Ayuda a Timoteo a determinar si es posible construir los senderos, y colocar y asignar los bancos cumpliendo todas las condiciones anteriores, y de ser así, dale una posible solución. Si existen múltiples soluciones que cumplan 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 longitud n. Por 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 pares.
- Si es posible la construcción, esta función debe realizar exactamente una llamada a `build` (ver más abajo) para reportar una solución, y luego debe retornar 1.
- De lo contrario, la función debe retornar 0 sin realizar ninguna llamada a `build`.
- Esta función es llamada exactamente una vez.

Tu implementación puede llamar a la siguiente función para dar una posible construcción de senderos y una colocación de bancos:

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

- Sea m la cantidad total de senderos a construir.
- u, v: dos arreglos de longitud m, que representan los senderos a construir. Estos senderos se numeran desde 0 hasta $m-1$. Por 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 recta horizontal o vertical de longitud 2. Dos senderos diferentes pueden tener a lo sumo un punto en común (una fuente). Una vez construidos los senderos, debe ser posible viajar entre cualquier par de fuentes mediante movimientos por los senderos.
- a, b: dos arreglos de longitud m, que representan los bancos. Por cada j ($0 \leq j \leq m-1$), se coloca un banco en $(a[j], b[j])$, y se lo asigna al sendero j. Dos bancos diferentes no pueden tener la misma ubicación.

Ejemplos

Ejemplo 1

Considera la siguiente llamada:

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

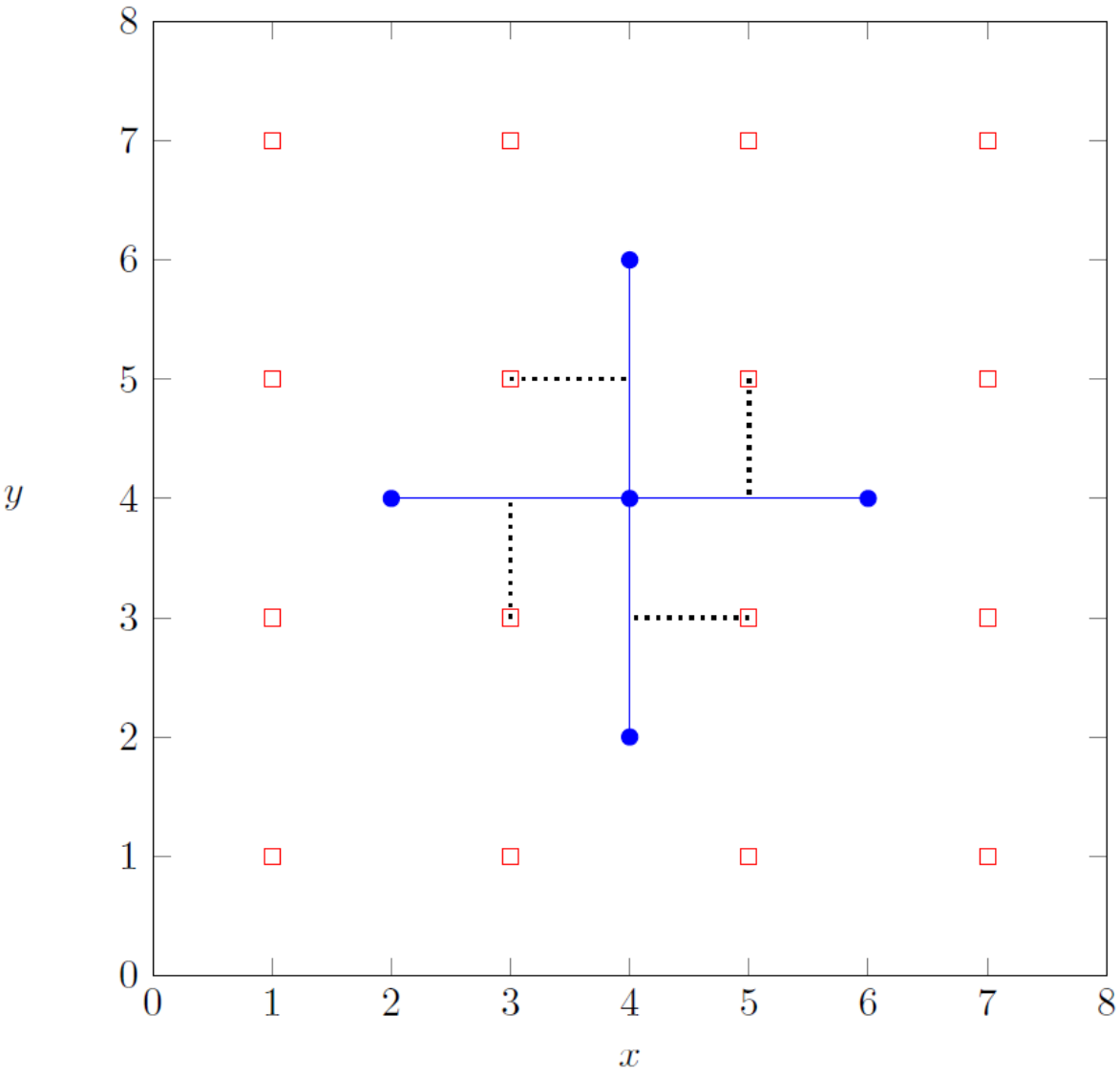
Esto significa que hay 5 fuentes:

- la fuente 0 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 los bancos correspondientes:

Número de sendero	Números de las fuentes que conecta el sendero	Ubicación del banco asignado
0	$0, 2$	$(5, 5)$
1	$0, 1$	$(3, 5)$
2	$3, 0$	$(5, 3)$
3	$4, 0$	$(3, 3)$

Esta solución corresponde al diagrama siguiente:



Para reportar esta solución, `construct_roads` debería realizar la siguiente llamada:

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

Luego debería retornar 1.

Notar que en este caso, existen múltiples soluciones que cumplen los requisitos, y todas ellas se consideran correctas. Por ejemplo, también sería correcto realizar la llamada `build([1, 2, 3, 4], [0, 0, 0, 0], [5, 5, 3, 3], [5, 3, 3, 5])` y luego retornar 1.

Ejemplo 2

Considera la siguiente llamada:

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

La fuente 0 está ubicada en $(2, 2)$ y la fuente 1 está ubicada en $(4, 6)$. Como no existe ninguna forma de construir senderos que cumplan con los requisitos, `construct_roads` debe retornar 0 sin

llamar 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 pares (para todo $0 \leq i \leq n-1$).
- No hay dos fuentes con 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 como máximo una forma de construir los senderos, de modo tal que sea posible viajar entre cualquier par de fuentes mediante movimientos por los senderos.
5. (20 puntos) No existen cuatro fuentes que formen las esquinas de un cuadrado de 2×2.
6. (30 puntos) Sin más restricciones.

Evaluador local

El evaluador local lee la entrada con el siguiente formato:

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

La salida del evaluador local tiene el siguiente formato:

- línea 1: el valor retornado por `construct_roads`

Si el valor retornado por `construct_roads` es 1 y se realizó la llamada `build(u, v, a, b)`, el evaluador además escribe:

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