Fountain Parks

En un parque cercano, hay n **fuentes** , etiquetadas desde 0 a $n-1$. Modelamos las fuentes como puntos en un plano bidimensional. 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 ubicaciones de las fuentes son todas distintas.

Timothy, el arquitecto, ha sido contratado para planificar la construcción de algunas **carreteras** y la colocación de un **banca** por carretera. Una carretera es un segmento de línea **horizontal** o **vertical** de longitud 2, cuyos extremos son dos fuentes distintas. Los caminos deben construirse de manera que se pueda viajar entre dos fuentes cualquiera moviéndose a lo largo de los caminos. Inicialmente, no hay carreteras en el parque.

Para cada camino, **exactamente** una banca debe colocarse en el parque y **asignarse a** (es decir, frente) ese camino. Cada banca debe colocarse en algún punto (a, b) de manera que a y b sean **enteros impares** . Las ubicaciones de las bancas deben ser **distintas** . Una banca en (a, b) sólo se puede asignar a una carretera si **ambos** extremos de la carretera 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 se puede asignar a una carretera, que es uno de los cuatro segmentos de línea $(2, 2)$ - $(2, 4)$, $(2, 4)$ - $(4, 4)$, $(4, 4)$ - $(4, 2)$, $(4, 2)$ - $(2, 2)$.

Ayude a Timothy a determinar si es posible construir caminos y coloque y asigne bancas que satisfagan todas las condiciones dadas anteriormente y, de ser así, bríndele una solución factible. Si hay varias soluciones factibles que satisfacen todas las condiciones, puede informar sobre cualquiera de ellas.

Detalles de Implementación

Debes implementar el siguiente procedimiento:

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

- x, y: son 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 $x[i]$ y $y[i]$ son enteros pares.
- Si la construccion es posible, este procedimiento debe hacer exactamente una llamada a `build` (mira más adelante) para reportar la solucion, seguidamente debe retornar 1.
- En otro caso, el procedimiento debe 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 factible de carreteras y una colocación de bancas:

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

- u, v: son dos arreglos de longitud m, representando el camino a ser construido. Estos caminos son etiquetados desde 0 hasta $m-1$. Por cada j ($0 \leq j \leq m-1$), el camino j conecta las fuentes $u[j]$ y $v[j]$. Cada camino debe ser horizontal o vertical de tamaño 2. Dos caminos distintos pueden tener como máximo un punto en común (una fuente). Una vez que se construyen las carreteras, debería ser posible viajar entre dos fuentes cualquiera moviéndose a lo largo de las carreteras.
- a, b: son dos arreglos de longitud m, representando las bancas. Por cada j ($0 \leq j \leq m-1$), una banca es ubicada en $(a[j], b[j])$, y asignado al camino j. Dos bancas distintas no pueden ser ubicadas en el mismo lugar. Diferentes bancas no pueden ser asignadas a un mismo camino.

Ejemplos

Ejemplo 1

Considera la siguiente llamada:

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

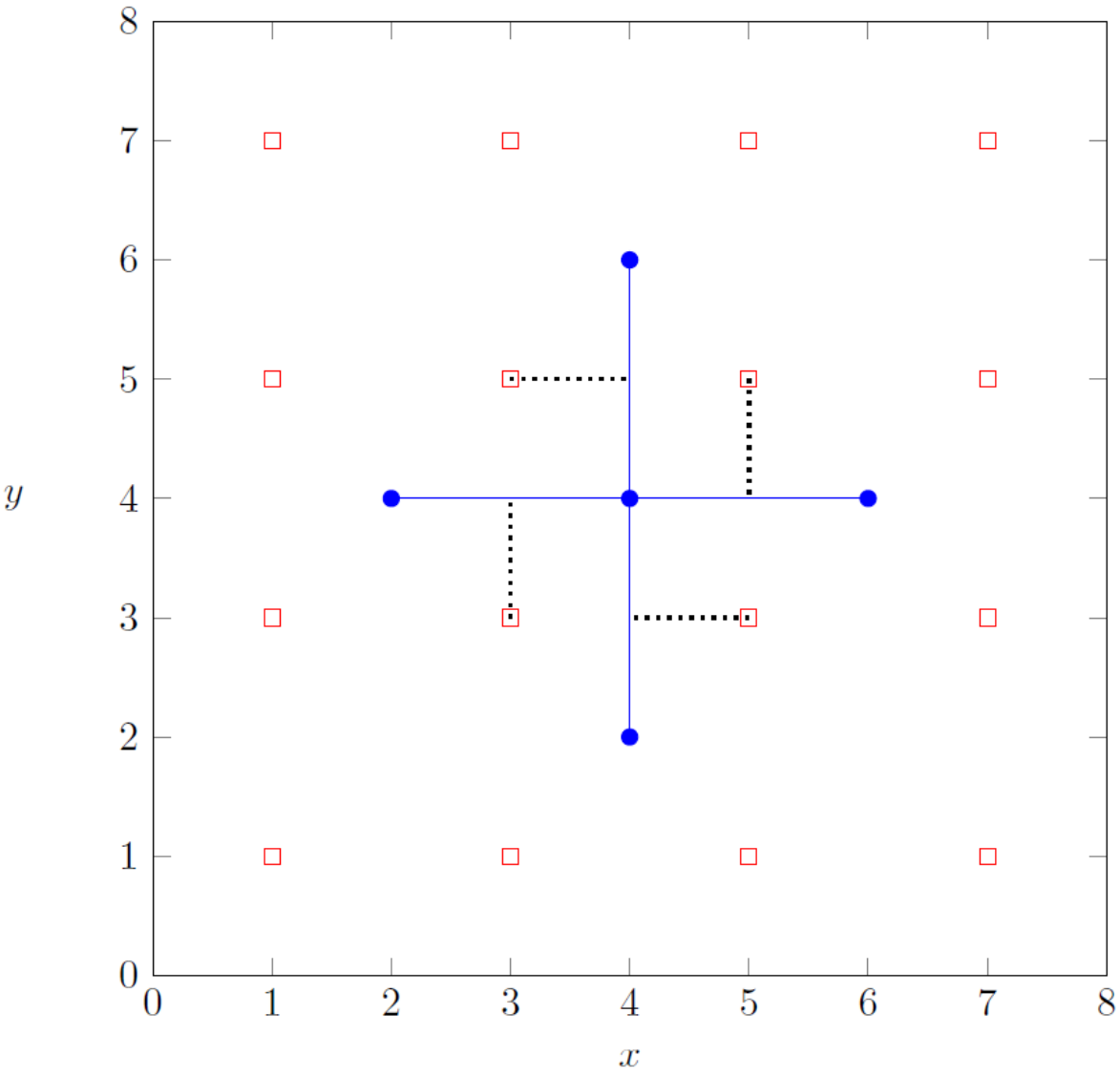
Esto significa que son 5 fuentes:

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

Es posible construir los siguientes 4 caminos, donde cada camino conecta dos fuentes, y ubica la banca correspondiente:

Etiqueta del camino	Etiquetas de las fuentes que conectan el camino	Ubicación asignada a la banca
0	$0, 2$	$(5, 5)$
1	$0, 1$	$(3, 5)$
2	$3, 0$	$(5, 3)$
3	$4, 0$	$(3, 3)$

Este es el correspondiente diagrama de la solución:



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

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

luego debería retornar 1.

Tenga en cuenta que en este caso, existen múltiples soluciones que satisfacen los requisitos, todas las cuales se considerarían 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 entonces 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 $(4, 6)$. Dado que no hay forma de construir caminos que satisfagan los requisitos, `construct_roads` debería 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 $0 \leq i \leq n-1$).
- Nunca dos fuentes tienen 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) Hay como máximo una forma de construir las carreteras, de modo que se pueda viajar entre dos fuentes cualquiera moviéndose a lo largo de las carreteras.
5. (20 puntos) No existen cuatro fuentes que formen las esquinas de un 2×2 cuadrado.
6. (30 puntos) 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 `build(u, v, a, b)` es llamado, el evaluador adicionalmente imprime:

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