Parques de la Fuente

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

El arquitecto Timothy ha sido contratado para planificar la construcción de unas **carreteras** y la colocación de un **banco** por carretera. Un camino es un segmento de línea **horizontal** o **vertical** de longitud 2, cuyos puntos extremos son dos fuentes distintas. Los caminos deben construirse de tal manera que se pueda viajar entre dos fuentes cualesquiera moviéndose por los caminos. Inicialmente, no hay caminos en el parque.

Para cada camino, **exactamente** hay que colocar un banco en el parque y **asignado a** (es decir, frente a) ese camino. Cada banco debe colocarse en un punto (a, b) tal que a y b sean **enteros pares**. Las ubicaciones de los bancos deben ser todas **distintas**. Un banco en (a, b) sólo puede asignarse a un camino si **ambos** puntos finales del camino están entre $(a-1, b-1)$, $(a-1, b+1)$, $(a+1, b-1)$ y $(a+1, b+1)$.

Por ejemplo, el banco en $(3, 3)$ sólo puede asignarse 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 carreteras y colocar y asignar bancos que satisfagan todas las condiciones dadas anteriormente y, en caso afirmativo, proporciónele una solución factible. Si hay varias soluciones factibles que satisfacen todas las condiciones, puede informar de cualquiera de ellas.

Detalles de la implementación

Usted deberá aplicar el siguiente procedimiento:

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

- Sea m el número total de caminos en la construcción.
- x, y: dos matrices de longitud 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 pares.
- Si la construcción es posible, este procedimiento debe hacer exactamente una llamada a `construir` (ver más abajo) para informar de una solución, tras lo cual debe devolver 1.
- En caso contrario, el procedimiento debe devolver 0 sin hacer ninguna llamada a `build`.
- Este procedimiento se llama exactamente una vez.

Su implementación puede llamar al siguiente procedimiento para proporcionar una construcción factible de caminos y una colocación de bancos:

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

- u, v: dos matrices de longitud m, que representan los caminos a construir. Estos caminos están etiquetados de 0 a $m-1$. Para cada j ($0 \leq j \leq m-1$), el camino j conecta las fuentes $u[j]$ y $v[j]$. Cada camino debe ser un segmento de línea horizontal o vertical de longitud 2. Dos caminos distintos pueden tener como máximo un punto en común (una fuente). Una vez construidos los caminos, debería ser posible viajar entre dos fuentes cualesquiera moviéndose por los caminos.
- a, b: dos matrices de longitud m, que representan los bancos. Para cada j ($0 \leq j \leq m-1$), se coloca un banco en $(a[j], b[j])$, y se asigna al camino j. No puede haber dos bancos distintos con la misma ubicación. No se pueden asignar bancos distintos a la misma carretera.

Ejemplos

Ejemplo 1

Consider the following call:

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

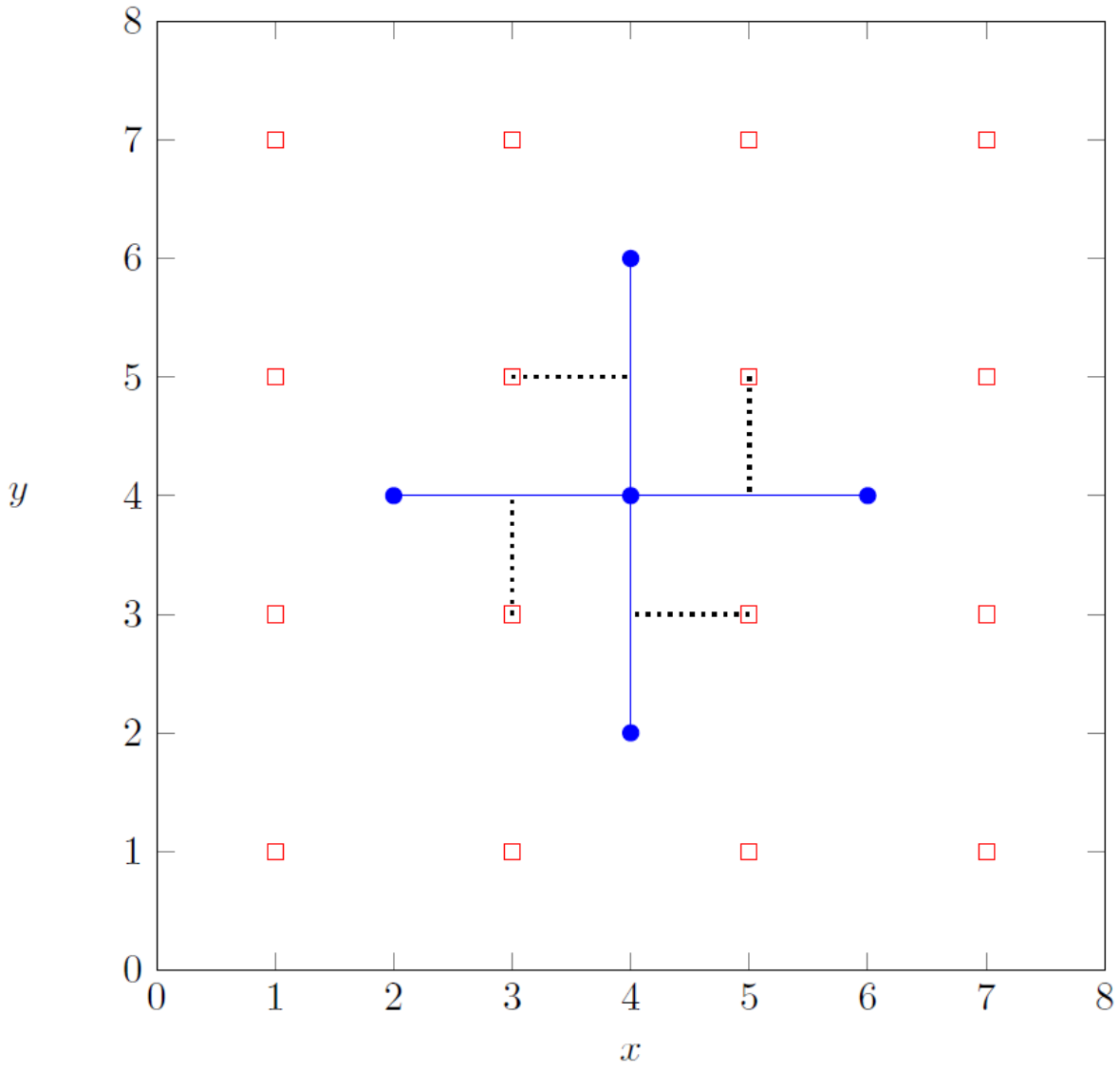
Esto significa que hay 5 fuentes:

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

Es posible construir los siguientes 4 caminos, donde cada camino conecta dos fuentes, y situar los bancos correspondientes:

Etiqueta de la carretera	Etiquetas de las fuentes que la carretera conecta	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 siguiente diagrama:



Para informar de esta solución, `construct_roads` debe hacer la siguiente llamada:

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

Entonces debería devolver 1.

Tenga en cuenta que, en este caso, hay múltiples soluciones que satisfacen los requisitos, 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 entonces retorna 1.

Ejemplo 2

Considere la siguiente llamada:

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

La fuente 0 está situada en $(2, 2)$ y la fuente 1 está situada en $(4, 6)$. Como no hay forma de construir caminos que satisfagan los requisitos, `construct_roads` debe devolver 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) $x[i] = 2$ (for all $0 \leq i \leq n-1$)
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 los caminos, de manera que se pueda viajar entre dos fuentes cualesquiera moviéndose por los caminos.
5. (20 puntos) No existen cuatro fuentes que formen las esquinas de un cuadrado 2×2.
6. (30 puntos) No additional constraints.

Ejemplo del Calificador

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 ejemplo del calificador 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 `construir(u, v, a, b)`, el calificador imprime adicionalmente:

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