# Parque de Fuentes

En un parque cercano, existen $n$ **fuentes**, numeradas de $0$ a $n-1$. Modelamos las fuentes como puntos en un espacio de dos dimensiones. Específicamente, 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 todas las fuentes son distintas.

Timothy el arquitecto fue contratado para planear la construcción de algunos **caminos** y la construcción de una **banca** en cada camino. Un camino es un segmento de línea **horizontal** o **vertical** de longitud $2$, donde sus puntos finales son dos fuentes distintas. Los caminos deben ser construidos de tal forma que es posible viajar entre cualesquiera dos fuentes usando los caminos. Inicialmente no hay caminos en el parque.

Para cada camino es necesario colocar **exactamente** una banca y asignarla (i.e alinearla) al camino. Cada banca debe ser colocada en algún punto $(a, b)$ de tal forma que $a$ y $b$ son **enteros impares**. Las posiciones de todas las bancas deben ser **distintas**. Una banca en la posición $(a, b)$ sólo puede ser asignada a un camino si **ambos** puntos finales del camino se encuentran entre $(a-1, b-1)$, $(a-1, b+1)$, $(a+1, b-1)$ y $(a+1, b+1)$. Por ejemplo, una banca en la posición $(3, 3)$ sólo puede ser asignada a un camino tal que sea alguno de los siguiente 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 asignar bancas de tal forma que cumpla todas las condiciones anteriores. En caso de que sea posible, debes encontrar una solución factible. Si existe más de una solución factible, puedes reportar cualquiera de ellas.

## Detalles de implementación

Debes implementar el siguiente procedimiento:

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

- $x, y$: dos arreglos de longitud $n$. Para cada $i$ ($0 \leq i \leq n-1$), la fuenta $i$ es un punto $(x[i], y[i])$, tal que $x[i]$ y $y[i]$ son enteros pares.
- Si la construcción es posible, este procedimiento debe hacer exactamente una llamada a `build` (ver más abajo) para reportar una solución, después, debe regresar $1$.
- De lo contrario, el procedimiento debe regresar $0$ sin hacer ninguna llamada a `build`.
- Este procedimiento será llamado sólo una vez.

Tu implementación puede llamar el siguiente procedimiento para reportar una planeación factible de caminos y 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 longitud $m$, que representan los caminos a ser construidos. Estos caminos están numerados 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 $2$. Cualesquiera dos caminos distintos pueden tener a lo más un punto en común (una fuente). Cuando los caminos sean constuidos, debe ser posible viajar entre cualesquiera dos fuentes usando los caminos.
- $a, b$: dos arreglos de longitud $m$, que representan las bancas. Para cada $j$ ($0 \leq j \leq m-1$), una banca es colocada en $(a[j], b[j])$, y asignada al camino $j$. Dos bancas distintas no pueden tener la misma posición.

# Ejemplos

## Ejemplo 1

Considera la siguiente llamada:

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

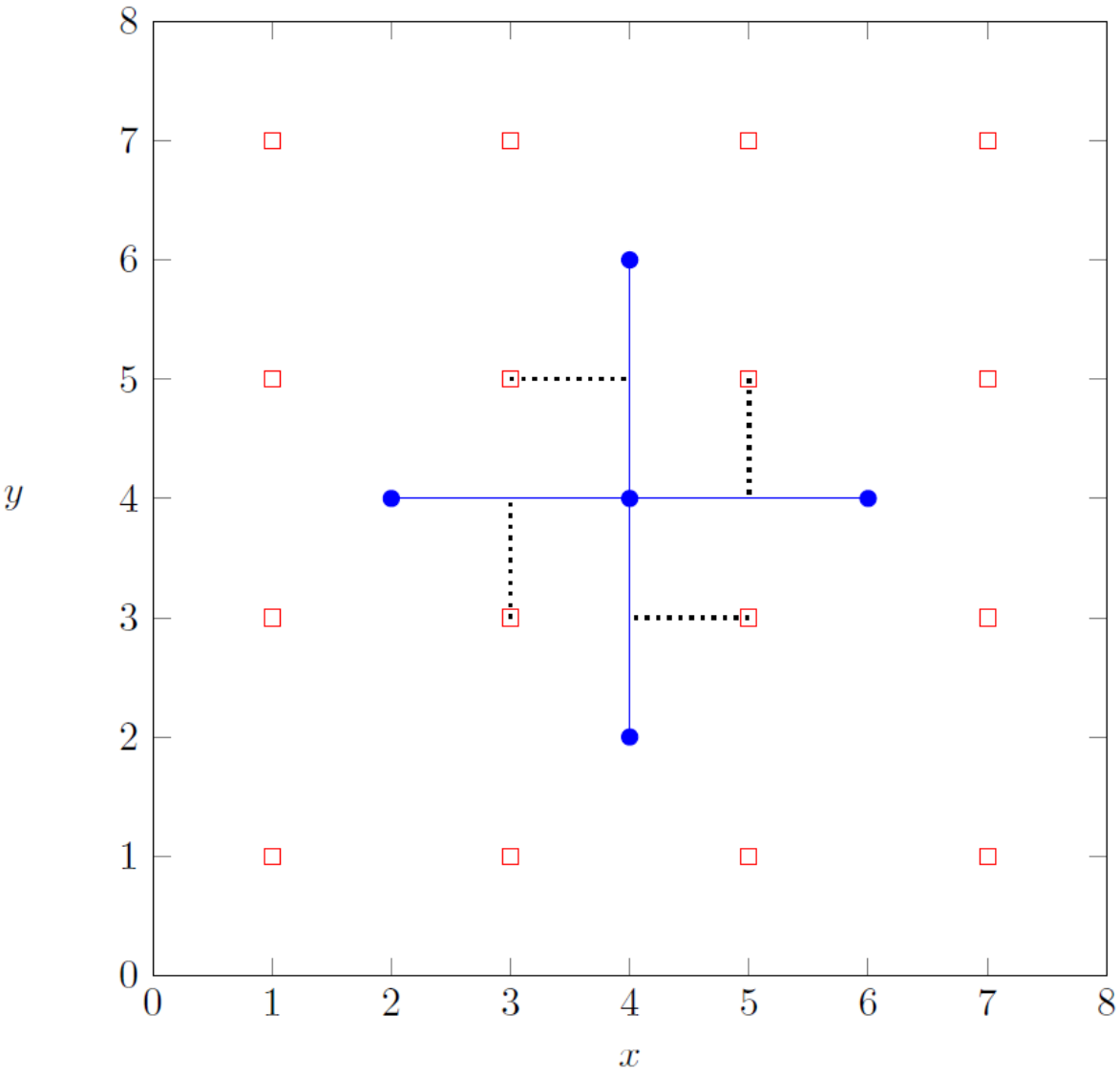
Esto signfica que hay $5$ fuentes:

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

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

| Índice del camino | Índices de las fuentes que conecta el camino | Posición de la banca asignada |
|---|---|---|
| $0$ | $0, 2$ | $(5, 5)$ |
| $1$ | $0, 1$ | $(3, 5)$ |
| $2$ | $3, 0$ | $(5, 3)$ |
| $3$ | $4, 0$ | $(3, 3)$ |

La solución corresponde al siguiente diagrama:




Para reportar 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])`

Después, debe regresar $1$.

Nota que en esta caso existen distintas soluciones que satisfacen los requerimientos, donde todas serán consideras 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 después regresar $1$.

## Ejemplo 2

Considera la siguiente llamada:

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

La fuenta $0$ está posicionada en $(2, 2)$ y la fuenta $1$ está posicionada en $(4, 6)$. Por lo tanto no hay forma de constuir los caminos de tal forma quue satisfagan los requerimientos,

`construct_roads` debe regresar $0$ 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 positivos (para todo $0 \leq i \leq n-1$).
- No hay dos fuentes con la misma posición.

## 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) A lo más hay una forma de construir los caminos de tal forma que es posible moverse entre cualesquiera dos fuentes usando los caminos.
5. (20 puntos) No existen 4 fuentes que formen las esquinas de un cuadrado de tamaño $2 \times 2$ .
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 que regresa `construct_roads`

Si el valor que regresa `construct_roads` es $1$ y `build(u, v, a, b)` fue llamado, entonces el evaluador de ejemplo imprime:

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