Les fontaines du parc

Dans un parc voisin, il y a n **fontaines**, numérotées de 0 à $n-1$. Nous modélisons les fontaines comme des points dans un plan à deux dimensions où la fontaine i ($0 \leq i \leq n-1$) est un point $(x[i], y[i])$ et où $x[i]$ et $y[i]$ sont **des entiers pairs**. Les positions des fontaines sont toutes distinctes.

Timothé est l'architecte qui a été embauché pour planifier la construction de quelques **routes** et le positionnement d'un **banc** par route. Une route est un segment **horizontal** ou **vertical** de longueur 2, dont les extrémités sont deux fontaines distinctes. Les routes doivent être construites de façon à ce qu'on puisse se déplacer entre deux fontaines en se déplaçant le long des routes. Initialement, il n'y a aucune route dans le parc.

Pour chaque route, **exactement** un banc doit être positionné dans le parc. Inversement chaque banc doit être **affecté à** une route (et il sera positioné en face de la route). Chaque banc doit être positionné à un point (a, b) tel que a et b sont **des entiers impairs**. Les positions des bancs doivent être toutes **distinctes**. Un banc en (a, b) ne peut être affecté à une route que si **les deux** extrémités de la route sont parmi $(a-1, b-1)$, $(a-1, b+1)$, $(a+1, b-1)$ et $(a+1, b+1)$. Par exemple, le banc à $(3, 3)$ ne peut être affecté qu'à une route, qui est l'un des quatre segments $(2, 2)$ – $(2, 4)$, $(2, 4)$ – $(4, 4)$, $(4, 4)$ – $(4, 2)$, $(4, 2)$ – $(2, 2)$.

Aidez Timothé à déterminer s'il est possible de construire les routes, de positionner et d'affecter les bancs qui satisfont toutes les conditions énoncées ci-dessus, et si c'est le cas, lui fournir une solution réalisable. S'il existe plusieurs solutions réalisables qui satisfont à toutes les conditions, vous pouvez fournir n'importe laquelle.

Détails d'implémentation

Vous devez implémenter la fonction suivante :

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

- x, y : deux tableaux de longueur n. Pour chaque i ($0 \leq i \leq n-1$), la fontaine i est un point $(x[i], y[i])$, où $x[i]$ et $y[i]$ sont des entiers pairs.
- Si une construction est possible, cette fonction doit faire appel exactement une fois à `build` (voir plus bas) pour fournir une solution, après cela elle doit retourner 1.
- Sinon, la fonction doit se terminer 0 sans faire d'appels à `build`.
- Cette fonction est appelée exactement une fois.

Votre implémentation peut appeler la fonction suivante pour fournir une construction faisable des routes et un placement des bancs :

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

- Soit m le nombre total de routes dans la construction.
- u, v : deux tableaux de longueur m, qui représentent les routes à construire. Ces routes sont numérotés de 0 à $m-1$. Pour chaque j ($0 \leq j \leq m-1$), la route j connecte les fontaines $u[j]$ et $v[j]$. Chaque route doit être un segment horizontal ou vertical de longueur 2. Deux routes distinctes peuvent avoir au plus un point en commun (une fontaine). Une fois les routes construites, il doit être possible de se déplacer entre deux fontaines en se déplaçant le long des routes.
- a, b : deux tableaux de longueur m, qui représentent les bancs. Pour chaque j ($0 \leq j \leq m-1$), un banc est placé en $(a[j], b[j])$, et affecté à la route j. Deux bancs distincts ne peuvent pas avoir le même emplacement.

Exemples

Exemple 1

Considérons l'appel suivant :

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

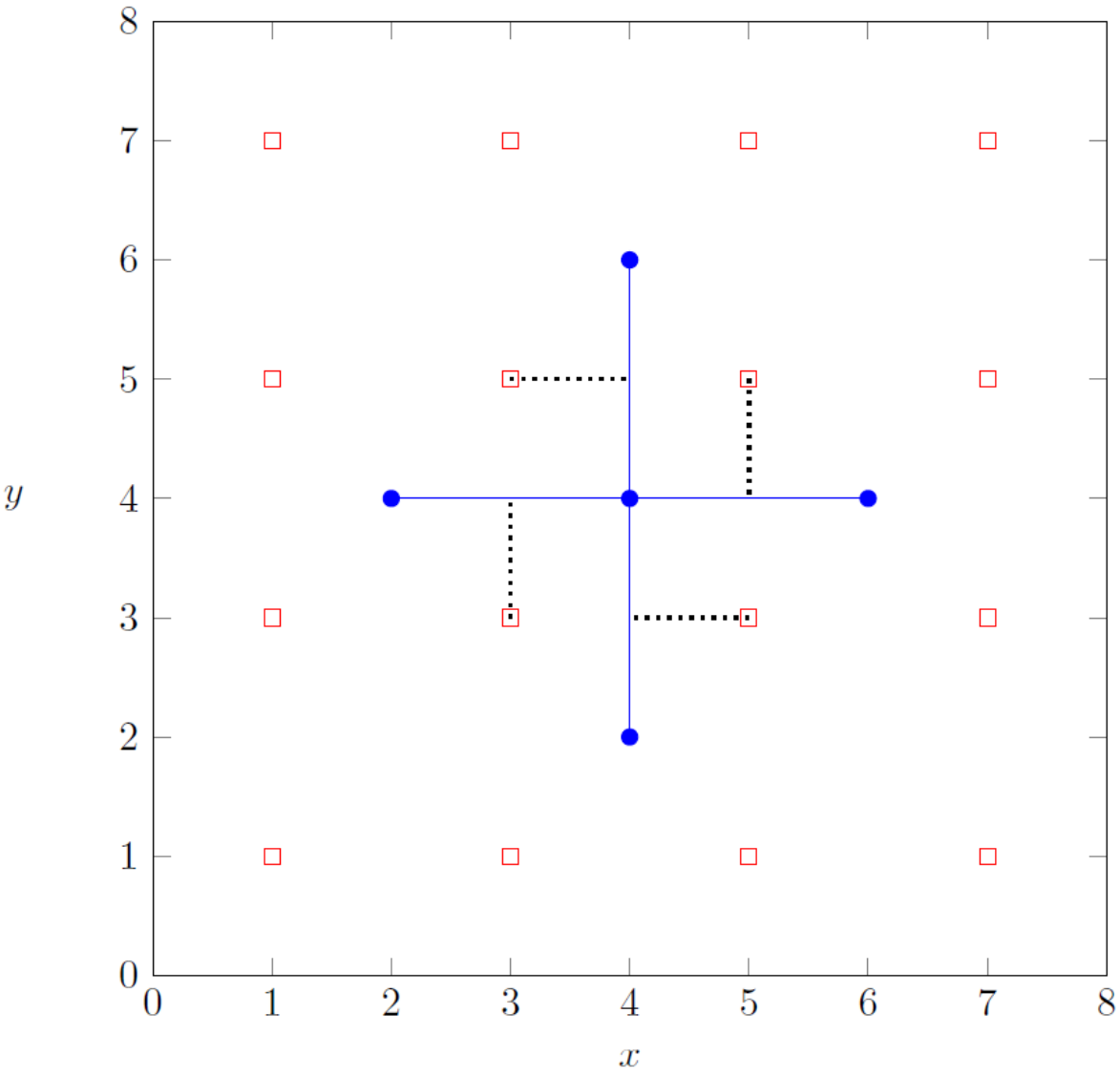
Cela veut dire qu'il existe 5 fontaines :

- La fontaine 0 se situe en $(4, 4)$,
- La fontaine 1 se situe en $(4, 6)$,
- La fontaine 2 se situe en $(6, 4)$,
- La fontaine 3 se situe en $(4, 2)$,
- La fontaine 4 se situe en $(2, 4)$.

Il est possible de construire les 4 routes suivantes, chaque route connectant deux fontaines, et de placer les bancs suivants :

Route	Fontaines connectées par la route	Position du banc affecté
0	$0, 2$	$(5, 5)$
1	$0, 1$	$(3, 5)$
2	$3, 0$	$(5, 3)$
3	$4, 0$	$(3, 3)$

Cette solution correspond au diagramme suivant :



Pour fournir cette solution, `construct_roads` doit faire l'appel suivant :

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

1 doit être renvoyé par la suite.

Notez que dans ce cas, il existe plusieurs solutions satisfaisant aux contraintes et qui seraient considérées comme correctes. Par exemple, il est également correct d'effectuer l'appel `build([1, 2, 3, 4], [0, 0, 0, 0], [5, 5, 3, 3], [5, 3, 3, 5])` et de renvoyer 1 par la suite.

Exemple 2

Considérons l'appel suivant :

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

La fontaine 0 est située en $(2, 2)$ et la fontaine 1 en $(4, 6)$. Puisqu'il n'existe aucun moyen de construire des routes satisfaisant les contraintes, `construct_roads` doit renvoyer 0 sans appeler la

fonction `build`.

Contraintes

- $1 \leq n \leq 200\,000$
- $2 \leq x[i], y[i] \leq 200\,000$ (pour chaque $0 \leq i \leq n-1$)
- $x[i]$ et $y[i]$ sont des entiers pairs (pour chaque $0 \leq i \leq n-1$).
- Les positions des fontaines sont toutes distinctes.

Sous-tâches

1. (5 points) $x[i] = 2$ (pour tout $0 \leq i \leq n-1$)
2. (10 points) $2 \leq x[i] \leq 4$ (pour tout $0 \leq i \leq n-1$)
3. (15 points) $2 \leq x[i] \leq 6$ (pour tout $0 \leq i \leq n-1$)
4. (20 points) Il existe au plus une manière de construire les routes, de façon à ce que l'on puisse aller d'une fontaine à n'importe quelle autre en se déplaçant le long des routes.
5. (20 points) Il n'existe pas quatre fontaines qui forment les coins d'un carré de taille 2×2.
6. (30 points) Pas de contraintes additionnelles.

Évaluateur d'exemple (Sample Grader)

L'évaluateur lit l'entrée au format suivant :

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

La sortie de l'évaluateur est au format suivant :

- ligne 1 : la valeur renvoyée par `construct_roads`

Si la valeur renvoyée par `construct_roads` est 1 et `build(u, v, a, b)` est appelée, l'évaluateur affiche en plus :

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