Фонтан парктары

Жакынкы паркта 0дөн $(n-1)$ге чейин белгиленген n **фонтан** бар. Фонтандарды эки өлчөмдүү тегиздиктин чекиттери катары моделдейбиз. Тактап айтканда, i-фонтан ($0 \leq i \leq n-1$) ($x[i], y[i]$)- чекитте, $x[i], y[i]$ **жуп сандар** . Фонтандардын жайгашкан жерлери ар башка.

Архитектор Тимофей кээ бир **жолдорду** курууну жана ар бир жолго бир **отургучту** жайгаштыруу пландаштыруу үчүн жалданган. Жол - узундугу 2 болгон **горизонталдык** же **вертикалдык** кесинди, анын учтары эки башка фонтан. Жолдорду каалаган эки фонтандын ортосунда жолдор менен жылып жүрө тургандай кылып куруу керек. Башында, паркта эч кандай жол жок.

Ар бир жол үчүн **так** бир отургучту паркка жайгаштырып, ал отургуч ошол жолго **катталат** (б.а. багытталат). Ар бир отургуч кандайдыр бир (a, b)-чекитке жайгаштырылышы керек, анткени a жана b **так сандар** болот. Бардык отургучтардын чекиттери ар түрдүү болуш керек. Эгерде жолдун **экөө** учу $(a-1, b-1), (a-1, b+1), (a+1, b-1)$ жана $(a+1, b+1)$ ичинде болсо гана, анда (a, b)-отургуч ал жолго каттала алат.

Мисалы, $(3, 3)$-отургуч $(2, 2) - (2, 4), (2, 4) - (4, 4), (4, 4) - (4, 2), (4, 2) - (2, 2)$ төрт кесиндинин бири болгон жолго гана катталса болот.

Тимофейге жолдорду курууга болорун аныктоого жана жогоруда келтирилген бардык шарттарды канааттандырган отургучтарды жайгаштырууга жана дайындоого жардам бериңиз, эгерде андай болсо, ага мүмкүн болгон чечимди сунуштаңыз. Эгерде бардык шарттарды канааттандырган бир нече мүмкүн болгон чечим болсо, анда алардын каалаганын билдире аласыз.

Implementation Details

You should implement the following procedure:

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

- x, y: two arrays of length n. For each i ($0 \leq i \leq n-1$), fountain i is a point $(x[i], y[i])$, where $x[i]$ and $y[i]$ are even integers.
- If a construction is possible, this procedure should make exactly one call to `build` (see below) to report a solution, following which it should return 1.
- Otherwise, the procedure should return 0 without making any calls to `build`.
- This procedure is called exactly once.

Your implementation can call the following procedure to provide a feasible construction of roads and a placement of benches:

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

- u, v: two arrays of length m, representing the roads to be constructed. These roads are labeled from 0 to $m-1$. For each j ($0 \leq j \leq m-1$), road j connects fountains $u[j]$ and $v[j]$. Each road must be a horizontal or vertical line segment of length 2. Any two distinct roads can have at most one point in common (a fountain). Once the roads are constructed, it should be possible to travel between any two fountains by moving along roads.
- a, b: two arrays of length m, representing the benches. For each j ($0 \leq j \leq m-1$), a bench is placed at $(a[j], b[j])$, and is assigned to road j. No two distinct benches can have the same location. Different benches cannot be assigned to the same road.

Examples

Example 1

Consider the following call:

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

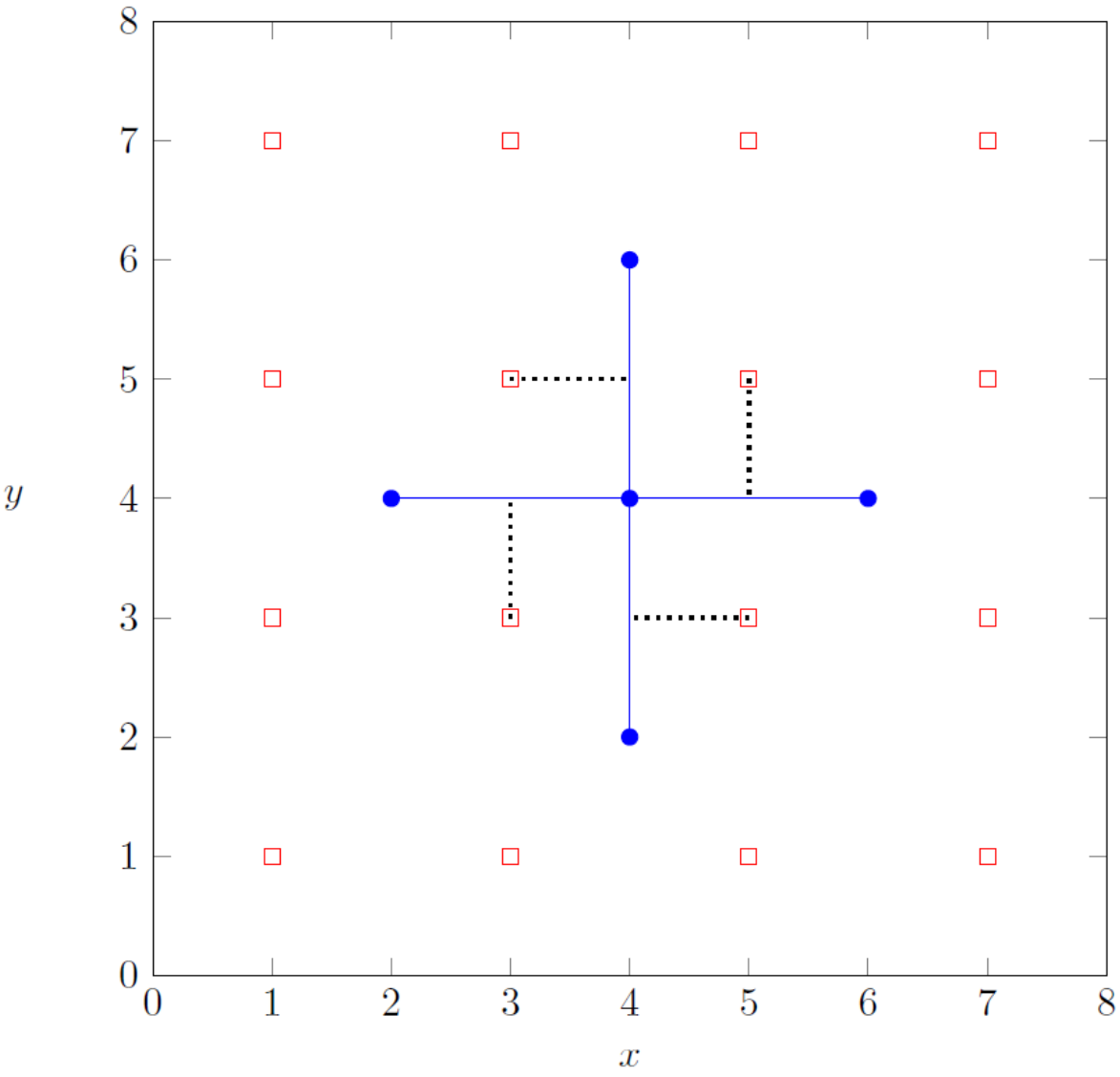
This means that there are 5 fountains:

- fountain 0 is located at $(4, 4)$,
- fountain 1 is located at $(4, 6)$,
- fountain 2 is located at $(6, 4)$,
- fountain 3 is located at $(4, 2)$,
- fountain 4 is located at $(2, 4)$.

It is possible to construct the following 4 roads, where each road connects two fountains, and place the corresponding benches:

Road label	Labels of the fountains the road connects	Location of the assigned bench
0	$0, 2$	$(5, 5)$
1	$0, 1$	$(3, 5)$
2	$3, 0$	$(5, 3)$
3	$4, 0$	$(3, 3)$

This solution corresponds to the following diagram:



To report this solution, `construct_roads` should make the following call:

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

It should then return 1.

Note that in this case, there are multiple solutions that satisfy the requirements, all of which would be considered correct. For example, it is also correct to call `build([1, 2, 3, 4], [0, 0, 0, 0], [5, 5, 3, 3], [5, 3, 3, 5])` and then return 1.

Example 2

Consider the following call:

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

Fountain 0 is located at $(2, 2)$ and fountain 1 is located at $(4, 6)$. Since there is no way to construct roads that satisfy the requirements, `construct_roads` should return 0 without making any

call to `build`.

Constraints

- $1 \leq n \leq 200\,000$
- $2 \leq x[i], y[i] \leq 200\,000$ (for all $0 \leq i \leq n-1$)
- $x[i]$ and $y[i]$ are even integers (for all $0 \leq i \leq n-1$).
- No two fountains have the same location.

Subtasks

1. (5 points) $x[i] = 2$ (for all $0 \leq i \leq n-1$)
2. (10 points) $2 \leq x[i] \leq 4$ (for all $0 \leq i \leq n-1$)
3. (15 points) $2 \leq x[i] \leq 6$ (for all $0 \leq i \leq n-1$)
4. (20 points) There is at most one way of constructing the roads, such that one can travel between any two fountains by moving along roads.
5. (20 points) There do not exist four fountains that form the corners of a 2×2 square.
6. (30 points) No additional constraints.

Sample Grader

The sample grader reads the input in the following format:

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

The output of the sample grader is in the following format:

- line 1: the return value of `construct_roads`

If the return value of `construct_roads` is 1 and `build(u, v, a, b)` is called, the grader then additionally prints:

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