2017. CZECH

1. POLICE

2s

Một mạng lưới các đường cao tốc một chiều được xây dựng để nối các thành phố trong vương quốc. Nhà vua muốn chọn một thành phố để làm thủ đô. Để đảm bảo an ninh, thủ đô phải được lựa chọn ở thành phố nào mà từ đó có thể đi đến bất kỳ một thành phố nào khác bằng một tuyến đường cao tốc (gồm nhiều con đường cao tốc). Có thể di chuyển từ thành phố về thủ đô bằng các con đường bộ (không nhất thiết phải đi bằng đường cao tốc). Nhà vua muốn biết mình có bao nhiêu thành phố có thể được lựa chọn làm thủ đô.

INPUT

Dòng đầu tiên ghi số thành phố N và số con đường cao tốc M. Sau đó là M dòng, dòng thứ i ghi 2 số A_i và B_i, thể hiện có con đường cao tốc một chiều đi từ A_i tới B_i. Không có 2 con đường cùng chiều nối giữa 1 cặp 2 thành phố.

GIỚI HẠN

$1 \le A_i, B_i \le N$. $1 \le N, M \le 10^6$.

30% tổng số test có $N \le 10^3$ và $M \le 3.10^3$.

OUTPUT

Dòng đầu tiên in ra số lượng thành phố có thể được chọn làm thủ đô. Dòng thứ 2 in ra số thứ tự các thành phố này, được sắp xếp theo thứ tự tăng dần.

Sample Input	Sample Output
5 6 1 3 1 4 4 2 2 1 2 5 5 4	4 1 2 4 5
3 2 1 3 2 3	0

2. PYRAMID

2s

Một nhà khảo cổ tìm thấy N số kỳ diệu được viết trên một bức tường. Số nào chia hết cho ít nhất 1 trong các số này cũng được coi là số kỳ diệu. Có M con số khác viết trên một bức tường khác nói rằng, số kỳ diệu thứ Q_i ($1 \le i \le M$) có đặc tính ma thuật. Và bạn phải chỉ ra những con số này.

INPUT

Dòng đầu ghi 2 số N và M. Dòng thứ hai ghi N số $A_1, A_2, \ldots, A_N$. Sau đó là M dòng, mỗi dòng ghi một số Q_i.

GIỚI HẠN

$1 \le N \le 15$, $1 \le M \le 50$. Với mọi $1 \le i \le N$ thì $2 \le A_i \le 10^{18}$.

Tích $A_1 \times A_2 \times \ldots \times A_N \le 10^{18}$

Với mọi $1 \le i \le M$ thì $2 \le Q_i \le 10^{18}$

Mọi kết quả nhỏ hơn 10^{18}.

Với 10% số test có $Q_i \le 10^6$

30% số test có $N = 2$

OUTPUT

In ra M dòng, dòng thứ i in ra số kỳ diệu thứ Q_i.

Sample Input	Sample Output
5 5 2 5 7 10 11 1 2 3 10 20	2 4 5 14 28
2 1 70 100 5	210

3. RACE

2s

Người ta tổ chức một cuộc đua robot trong mê cung. Mê cung có dạng lưới ô vuông kích thước n hàng m cột. Mỗi ô hoặc là trống, hoặc là chứa chướng ngại vật. Mỗi robot lần lượt có 1 ô xuất phát và 1 ô đích được lựa chọn ngẫu nhiên. Trong mỗi bước, robot chỉ được đi sang phải 1 ô hoặc đi xuống dưới 1 ô. Ban giám khảo muốn biết liệu robot có thể đi đến đích được hay không.

INPUT

Dòng đầu tiên ghi 3 số n, m và q là số hàng, số cột và số cặp tọa độ. Sau đó là n dòng, mỗi dòng ghi m ký tự biểu diễn các ô của lưới, ký tự . là ô trống, ký tự # là ô chứa chướng ngại vật. Sau đó là q dòng. Mỗi dòng ghi 4 số nguyên r_{1i}, c_{1i}, r_{2i}, c_{2i} là số hàng, số cột của ô xuất phát và số hàng, số cột của ô đích.

GIỚI HẠN

$1 \le n, m \le 1000$ và $1 \le q \le 10^6$.

Với mọi $1 \le i \le q$, $1 \le r_{1i}, r_{2i} \le n$, $1 \le c_{1i}, c_{2i} \le m$. Các ô (r_{1i}, c_{1i}) và (r_{2i}, c_{2i}) là các ô trống.

20% tổng số test có $1 \le q \le 300$.

OUTPUT

In ra q dòng. Dòng thứ i in ra YES nếu robot có thể đi được, in ra NO nếu không.

Sample Input	Sample Output
3 4 5 .\#.. .\#\#. 1 1 3 4 1 3 3 4 1 1 1 1 1 1 2 4 2 1 2 4	YES YES YES NO NO

Corporate life (after hostile takover)

task: `corpo`	input file: `stdin`	output file: `stdout`
points: `100`	time limit: `500 ms`	memory limit: `1 GB`

JanuszPol is a Polish company which has a long tradition to its name. Recently, though, it fell into dire financial situation, and was eventually taken over by a foreign competitor. The new board decided to completely rebuild the company organization. Until now, it was a typical tree structure:

- There was exactly one executive director, who had no superiors,
- Every other employee had exactly one superior, and there were no cyclic relations.

For every employee x, their *subordinate* is every employee y, who is under x in the tree (there is a sequence of direct superiors from y to x).

After the takeover, the company will employ the same people, and it will also be organized as a tree, but every employee will receive a diffrent position, so the shape of the tree may change completely. The executive director is, however, guaranteed to keep the position. Now everyone is afraid to give orders to anyone else – any moment now, a subordinate may become the superior...

Task

Given the description of the tree before and after the takover, for every employee x determine the number of the people who were subordinates of x and will remain the subordinates after the takeover.

Input

The first line of the input contains an integer n ($2 \leq n \leq 200\,000$): the number of employees. The employees are numbered from 1 to n, with the person number 1 being the executive director. The second line contains the company structure before the takeover: there are $n-1$ numbers $a_2, a_3, \ldots, a_n$ with a_i being the number of i's superior. The third line also contains $n-1$ numbers $b_2, b_3, \ldots, b_n$, where b_i is the superior of i after the takeover. You may assume that both descriptions define a proper tree rooted at 1.

Output

Output a single line containing n numbers – i-th of them should be the number of people who are i's subordinates in both trees at once.

Subtasks

Subtask	Points	Maximal n
1	30	2 000
2	70	200 000

Samples

input

```
5
1 1 3 3
1 2 3 1
```

output

```
4 0 1 0 0
```

Posters on the wall

task: `posters`	input file: `stdin`	output file: `stdout`
points: `100`	time limit: `2000 ms`	memory limit: `1 GB`

Deep in the top-secret base of Keepers of the Sacred Postulates, there's The Wall with many rectangular posters depicting deep knowledge derived from their most valuable postulates. They are quite rare, thus they are placed so that they don't overlap.

Occasionally, a batch of new posters worthy of the honor of being placed on The Wall arrives. The Keepers then have to decide, where to place the new posters. It's a complicated process and you are to help with one of its steps.

The Keepers are currently picking candidate positions for the posters. You are to quickly compute the total area of currently hanging posters that would be shadowed by placing a new poster at a given position.

Task

You are given a description of n disjoint gray rectangles in a plane. You are to answer q queries of the form: What is the total grey area in a given rectangle? Note that this doesn't affect the plane.

The answer has to be produced online.

Input

The first line contains five numbers, r, c, n, q, m, ($1 \leq r, c < m \leq 10^9 + 9$, $0 \leq n, q \leq 50\,000$), the height and width of the wall, number of posters on the wall, number of queries and a modulus for computing the queries (we will explain that below).

Each of the next n lines contains four numbers, x_1, y_1, x_2, y_2 ($0 \leq x_1, x_2 \leq r$, $0 \leq y_1, y_2 \leq c$), the coordinates of two opposite corners of the rectangle.

The last q lines contain five numbers $x'_1, y'_1, x'_2, y'_2, v$, each between 0 and $m-1$ (inclusive). You can compute the real coordinates x_1, y_1, x_2, y_2 using the formula below.

Let's denote by l the answer to the last query (with $l = 0$ for the first query). Then

$$x_i = (x'_i + l \cdot v) \pmod m$$

$$y_i = (y'_i + l \cdot v) \pmod m$$

The decoded coordinates x_1, y_1, x_2, y_2 satisfy following conditions: $0 \leq x_1, x_2 \leq r$, $0 \leq y_1, y_2 \leq c$.

Output

For each query output one line containing a single number: Answer to the query.

Subtasks

There are several subtasks. In offline subtasks, the value v will be zero for each query.

subtask	points	maximum r	maximum c	maximum n and q	online
1	10	500	500	500	no
2	10	5000	5000	5000	no
3	40	300 000	300 000	50 000	no
4	10	10^9	200 000	50 000	no
5	10	10^9	10^9	50 000	no
6	10	100 002	100 002	50 000	yes
7	10	$10^9 + 8$	$10^9 + 8$	50 000	yes

Samples

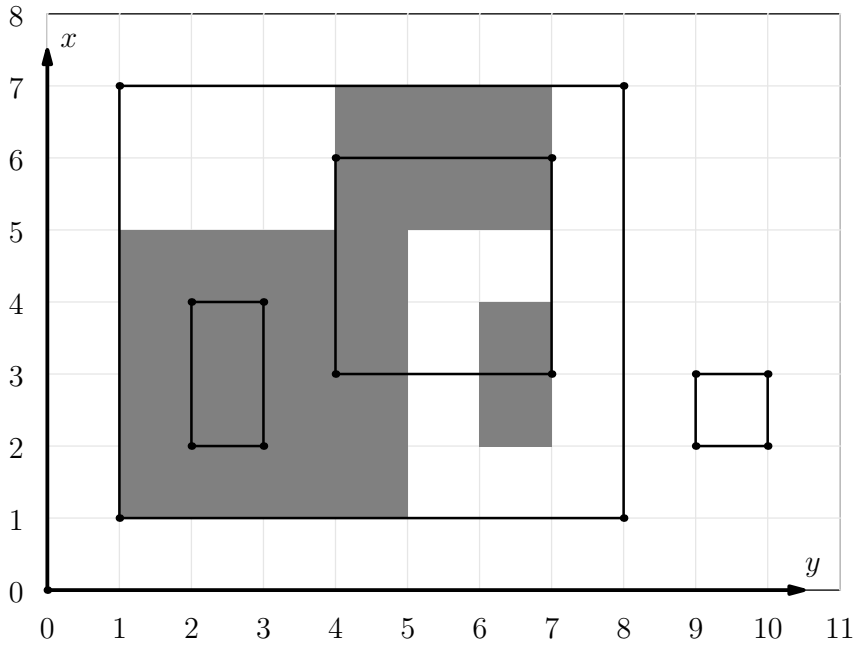
input

```
8 11 3 4 13
1 1 5 5
7 7 5 4
4 6 2 7
1 1 7 8 0
2 2 4 3 0
3 4 6 7 0
2 9 3 10 0
```

output

```
24
2
6
0
```

You can view the whole plane below.



input

```
8 11 3 4 13
1 1 5 5
7 7 5 4
4 6 2 7
1 1 7 8 4
6 6 8 7 2
2 3 5 6 7
11 5 12 6 5
```

output

```
24
2
6
0
```

This is the same input as above, using online queries.

The Battle for Wesnoth

task: `wesnoth`	input file: `stdin`	output file: `stdout`
points: `100`	time limit: `100 ms`	memory limit: `1 GB`

Do you know The Battle for Wesnoth? It is a splendid turn-based strategy game with fantasy theme. There are elves, orcs, undead, dwarves, and even drakes roaring around! As for this problem, you must know that every unit has some number of hitpoints, which is reduced during combats, and you must also understand how the combat proceeds. We will be interested only in the simplest case – a common attack at a defenseless unit.

Such an attack is described by three integers:

- d – the damage done by a successful blow;
- b – the number of blows;
- p – the probability (in percentage) that a blow is successful.

In the game, d and b are properties of the attacker, whereas p is usually given by the terrain occupied by the defender, but for our purposes, we assume that it is a property of the attacker, too (as with magical attacks).

As an example, consider an attack with $d = 6$, $b = 2$, and $p = 60$. There are three possible outcomes:

- By 16 %, the attacker misses both times and does no damage at all.
- By 48 %, the attacker hits with one of the blows and misses with the other one, doing the damage of 6 hitpoints.
- By 36 %, the attacker hits both times, doing the total damage of 12 hitpoints.

When the number of hitpoints of the defending unit becomes non-positive, the unit dies.

David is currently playing an add-on where a special unit, the Elvish Princess, has a very special magical attack: Instead of d and b, she is described by an integer m, and when she attacks, the player may choose d and b to be arbitrary positive integers as long as $d \cdot b \leq m$.

Every now and then, David needs his Elvish Princess to kill an ugly undead skeleton or a stinking orcish warrior, so he would like to know which choice of d and b gives the highest probability of killing the enemy.

Task

For every unit David needs to kill, find the best choice of d and b.

Input

The first line of the input contains an integer m ($1 \leq m \leq 10^6$) and an integer p ($1 \leq p \leq 99$) – the parameters of the attacker. The second line contains an integer n ($1 \leq n \leq 10^5$) – the number of units to be killed. The third line contains n integers $h_1, h_2, \ldots, h_n$ ($1 \leq h_i \leq 10^6$), where h_i is the number of hitpoints of the i-th unit to be killed.

Output

For each unit to be killed, output two integers: the d and b which give the maximum probability of killing the unit. Your answer will be accepted if the probability given by your output differs from the optimum by no more than 10^{-6}.

Subtasks

Subtask	Constraints	Points
1	$m \leq 10$	10
2	$m \leq 100$	10
3	$m \leq 1\,000$	20
4	$n = 1$, $m \leq 100\,000$	20
5	$m \leq 100\,000$	25
6	no additional constraints	15

Remark

The computation can be done with far enough precision using doubles. Of course, you must avoid overflow and underflow as well as doing unsafe operations like adding numbers of very different magnitudes (e.g. for $a = 0.5$ and $b = 10^{-100}$, $a + b$ would be evaluated equal to a).

Sample

input	output
10 60 3 5 6 7	5 2 2 5 7 1

The corresponding probabilities are 84 %, 68.256 %, and 60 %.

Brackets

task: `brackets`	input file: `stdin`	output file: `stdout`
points: `100`	time limit: `200 ms`	memory limit: `1 GB`

Task

A bracket symbol is one of the following: `()[]{}<>`. A correct *bracket expression* is any string consisting of bracket symbols, such that:

- Every left bracket has a matching right bracket of the same kind, and every right bracket is matched;
- No two pairs of matching brackets cross – for every two such pairs, they are either disjoint or one is contained inside the other.

For example, `([])<>` is a correct bracket expression, whereas `<{>}` is not, as the curly brackets and angle brackets cross each other.

You are given a graph of n vertices in which every (directed) edge is labeled with one of the bracket symbols. A path in this graph is *valid*, if its edges form a correct bracket expression. For some two vertices s and t, determine the length of a shortest valid path between s and t. We allow the path to pass multiple times through any vertex.

Input

On the first line of input there are four integers n, m, s, t ($1 \leq n \leq 200$, $0 \leq m \leq 2000$, $1 \leq s, t \leq n$) – the number of vertices, edges, starting and ending vertex, respectively. Each of the following m lines contains two integers x, y and a bracket symbol b ($1 \leq x, y \leq n$), which describe one graph edge. Note that there may be loops and multiple edges.

Output

Output a single line containing a single integer – the length of the shortest valid path between s and t. If there is no such path, output -1. You may assume that if a path exists, its length does not exceed 10^{18}.

Subtasks

Subtask	Points	Description
1	16	$n \leq 10, m \leq 50$
2	16	$n \leq 20, m \leq 100$
3	16	$n \leq 50$
4	16	$n \leq 100$
5	10	$s = 1$, $t = n$, $a < b$ for every edge (a, b)
6	26	no additional constraints

Samples

input

```
4 4 1 4
1 2 (
2 2 [
2 3 ]
3 4 )
```

output

```
4
```

input

```
5 4 1 5
1 2 <
2 3 {
3 4 >
4 5 }
```

output

```
-1
```

GCD-sum

task: `gcds`	input file: `stdin`	output file: `stdout`
points: `100`	time limit: `2000 ms`	memory limit: `1 GB`

Task

A multi-set (i.e. a set with possible repetitions) of n integers is given. We split the set into k disjoint groups, for every group we compute the greatest common divisor of its elements, and sum all the subsets' GCDs.

For every $k = 1, 2, \ldots, n$, determine the maximal sum which can be obtained this way.

Input

In the first line of input there is a single integer n ($1 \leq n \leq 500\,000$) – the cardinality of the set. In the second line, there are n positive integers, not exceeding 10^{12} – the given sequence.

Output

Output n line scontaining one integer each – the best sum of GCDs when partitioning into 1, 2, ..., n subsets.

Subtasks

Subtask	Constraints	Points
1	$n \leq 7$	5
2	$n \leq 15$	5
3	$n \leq 100$, $a_i \leq 500$	8
4	$n \leq 2000$, $a_i \leq 2000$, a_i are distinct	8
5	$n \leq 2000$	14
6	a_i are distinct	25
7	no additional constraints	35

Samples

input

```
4
10 9 10 3
```

output

```
1
13
23
32
```

For $k = 2$, the best partition is $(10, 10)$ and $(9, 3)$, giving the sum of $10 + 3 = 13$. For $k = 3$, the best partition is (10), (10) and $(9, 3)$ with the sum of 23.

input

```
8
15 25 29 30 43 44 45 55
```

output

```
1
56
101
145
188
221
256
286
```

Power plants

task: `power`	input file: `stdin`	output file: `stdout`
points: `100`	time limit: `3000 ms`	memory limit: `1 GB`

In distant future, humanity found a large, warm and lush planet to colonize, and is building a lot of power plants to support the new colony. The stunning technology allows the builders to harness the vast energy of the Universe. However, there is a small fact they overlooked: building two power plants too close to each other yields a considerable risk of a chain reaction, which would in turn lead to a spectacular explosion. This, obviously, should be avoided.

Fortunately, there are two types of energy that can be used in plants, called *light* and *dark* energy, which do not interact with each other. If some of the power plants work on light, and some on dark energy, the distance between identical plants may be larger than before, which would make the whole undertaking less hazardous.

Task

Given the locations of n plants (the planet is big enough so that we can treat them as points on a plane), assign to every plant either light or dark energy so that the Euclidean distance between two same-type plants is maximal possible. To avoid real numbers output, as the answer, the square of this maximal distance, as well as the partition of the plants into "light" and "dark" ones.

Input

In the first line of input there is a single integer N ($3 \leq N \leq 100\,000$) – the number of power plants. Of the following N lines, the i-th one contains two integers x_i, y_i ($0 \leq x_i, y_i \leq 10^9$) – the coordinates of the i-th plant. All points are distinct.

Output

In the first line of output, print a single integer – the square of the maximal distance which can be achieved. The second line should contain the number of plants working on light energy, the third line – a space-separated list of these plants' numbers. The fourth and fifth line should describe the dark energy plants, in the same format. If there are multiple valid solutions, output any one of them.

Subtasks

Subtask	Points	Maximal n
1	10	100
2	25	2000
3	65	100 000

Samples

input

```
4
0 3
0 0
3 0
3 3
```

output

```
18
2
1 3
2
2 4
```

Counterspells

task: `counterspells`	input file: `stdin`	output file: `stdout`
points: `100`	time limit: `1000 ms`	memory limit: `1 GB`

Magic: The Gathering card game has an interesting game mechanic of casting and countering spells. We are not going to explain it here, as it's quite complex and not necessary to solve this task. However, if you are a MtG player, you may see how this task is related to the spell countering mechanic.

Task

For each rooted tree there is a unique way of coloring its vertices by two colors (black and white) satisfying the following constraint:

- A vertex is white if and only if it has a black son.

Uniqueness of this coloring can be easily proven by induction. We will call a tree *well-colored* if it's colored in this way.

We start with a rooted tree consisting of one black vertex (the root) and do the following operation n-times:

- *add*(v) – Add a new black vertex to the tree as a son of vertex v. Then invert colors of some (possibly zero, possibly all) vertices in the tree so that the resulting tree is well-colored.

For each operation, we want to know how many vertices are inverted during this operation.

Input

The root of the tree is numbered 0, other vertices are numbered $1, 2, \ldots, n$ in the order they are added to the tree.

The first line of the input contains a single integer n ($1 \leq n \leq 200000$) – number of vertex additions.

n lines follow, i-th of them containing number v_i – ID of father of vertex added in i-th operation. It's guaranteed that vertex v_i already exists before i-th operation, that is $v_i < i$.

Output

For each operation output one line containing number of vertices whose colors will be inverted during this operation.

Subtasks

subtask	points	maximum N	additional constraint
1	20	1000	
2	20	10000	
3	20	100000	
4	20	200000	Depth of the final tree is at most 100
5	20	200000	

Samples

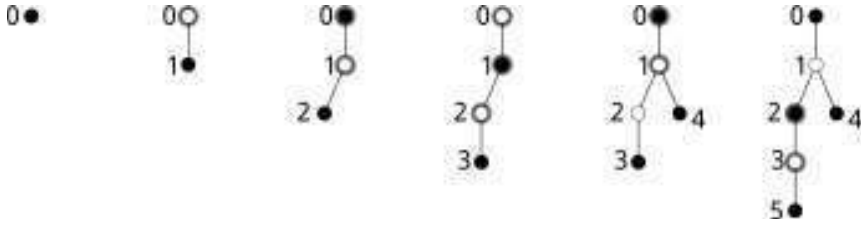
input

```
5
0
1
2
1
3
```

output

```
1
2
3
2
2
```

The situation after each operation looks like this (vertices inverted during previous operation are highlighted):


0
0
1
0
1
2
0
1
2
3
0
1
2
3
4
0
1
2
3
4
5

Skiing

task: `skiing`	input file: `stdin`	output file: `stdout`
points: `100`	time limit: `1000 ms`	memory limit: `1 GB`

After an unsuccessful attempt to qualify for IOI, Kleofáš decided to become a slalom skiing champion. Tomorrow is a very important day in Kleofáš's life: he will compete in his very first skiing contest!

During the contest, Kleofáš will have to get from a starting point to a finishing point, while passing through n gates. In order to be as fast as possible, Kleofáš wants to use the shortest possible trajectory.

Task

A skiing course can be described as a starting point S, a finishing point F and n gates. Each gate is a line segment parallel to the x axis (i. e. horizontal). No two gates are at the same y coordinate (altitude). The starting point is above each gate i. e. its y coordinate is higher than y coordinate of any gate. The finishing point is below each gate and below the starting point.

Find the shortest polygonal chain starting at point S, finishing at point F and intersecting all gates, in order **from top to bottom**. We say that a polygonal chain intersects a line segment if they have at least one common point (this point **can** be an endpoint of the line segment).

Input

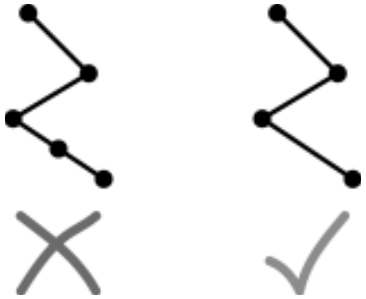
First line of the input contains one integer n ($0 \leq n \leq 10^6$) – number of gates. Second line contains four integers x_S, y_S, x_F, y_F: coordinates of points $S = (x_S, y_S)$ and $F = (x_F, y_F)$ respectively.

n lines follow, i-th of them contains three integers x_{1i}, x_{2i}, y_i, meaning that i-th gate is a segment from (x_{1i}, y_i) to (x_{2i}, y_i). For each i, $x_{1i} < x_{2i}$ holds.

All coordinates are between -10^9 and 10^9, inclusive. Gates are ordered from top to bottom, i. e. $y_S > y_1 > y_2 > \ldots > y_n > y_F$.

Output

It can be proven that there always exists a unique shortest polygonal chain and all its vertices have integer coordinates. Output this chain without any redundant vertices (i. e. only output vertices where the chain changes its direction).



On the first line of the output print a single integer k – number of vertices in the optimal chain. Then print k more lines, i-th of them containing two space-separated integers x_i, y_i – coordinates of the i-th vertex of the chain. Vertices must be named from the start of the chain to the end, thus $x_1 = x_S, y_1 = y_S, x_k = x_F, y_k = y_F$ and $y_1 > y_2 > \ldots > y_k$ must hold.

Subtasks

subtask	points	maximum n
1	20	200
2	30	2000
3	50	1000000

Samples

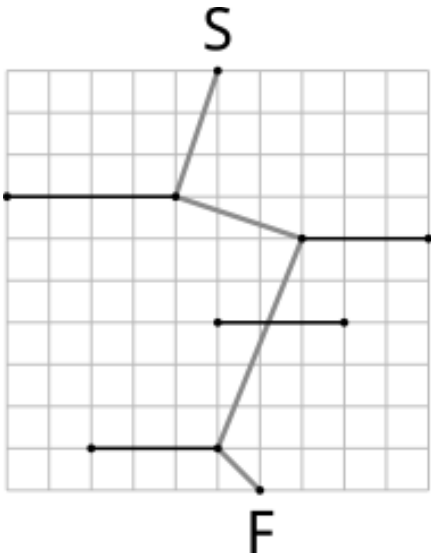
input

```
4
5 10 6 0
0 4 7
7 10 6
5 8 4
2 5 1
```

output

```
5
5 10
4 7
7 6
5 1
6 0
```

The situation looks like this:



Skiing

task: `skiing`	input file: `stdin`	output file: `stdout`
points: `100`	time limit: `1000 ms`	memory limit: `1 GB`

After an unsuccessful attempt to qualify for IOI, Kleofáš decided to become a slalom skiing champion. Tomorrow is a very important day in Kleofáš's life: he will compete in his very first skiing contest!

During the contest, Kleofáš will have to get from a starting point to a finishing point, while passing through n gates. In order to be as fast as possible, Kleofáš wants to use the shortest possible trajectory.

Task

A skiing course can be described as a starting point S, a finishing point F and n gates. Each gate is a line segment parallel to the x axis (i. e. horizontal). No two gates are at the same y coordinate (altitude). The starting point is above each gate i. e. its y coordinate is higher than y coordinate of any gate. The finishing point is below each gate and below the starting point.

Find the shortest polygonal chain starting at point S, finishing at point F and intersecting all gates, in order **from top to bottom**. We say that a polygonal chain intersects a line segment if they have at least one common point (this point **can** be an endpoint of the line segment).

Input

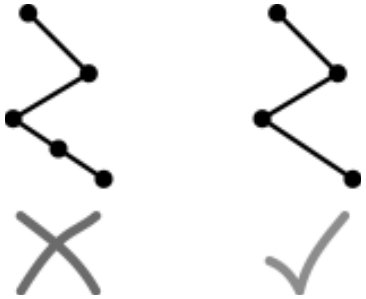
First line of the input contains one integer n ($0 \leq n \leq 10^6$) – number of gates. Second line contains four integers x_S, y_S, x_F, y_F: coordinates of points $S = (x_S, y_S)$ and $F = (x_F, y_F)$ respectively.

n lines follow, i-th of them contains three integers x_{1i}, x_{2i}, y_i, meaning that i-th gate is a segment from (x_{1i}, y_i) to (x_{2i}, y_i). For each i, $x_{1i} < x_{2i}$ holds.

All coordinates are between -10^9 and 10^9, inclusive. Gates are ordered from top to bottom, i. e. $y_S > y_1 > y_2 > \ldots > y_n > y_F$.

Output

It can be proven that there always exists a unique shortest polygonal chain and all its vertices have integer coordinates. Output this chain without any redundant vertices (i. e. only output vertices where the chain changes its direction).



On the first line of the output print a single integer k – number of vertices in the optimal chain. Then print k more lines, i-th of them containing two space-separated integers x_i, y_i – coordinates of the i-th vertex of the chain. Vertices must be named from the start of the chain to the end, thus $x_1 = x_S, y_1 = y_S, x_k = x_F, y_k = y_F$ and $y_1 > y_2 > \ldots > y_k$ must hold.

Subtasks

subtask	points	maximum n
1	20	200
2	30	2000
3	50	1000000

Samples

input

```
4
5 10 6 0
0 4 7
7 10 6
5 8 4
2 5 1
```

output

```
5
5 10
4 7
7 6
5 1
6 0
```

The situation looks like this: