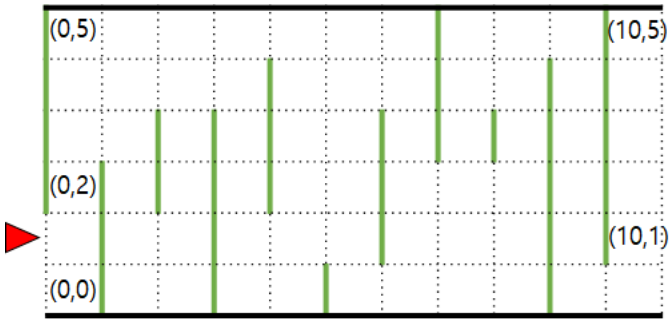
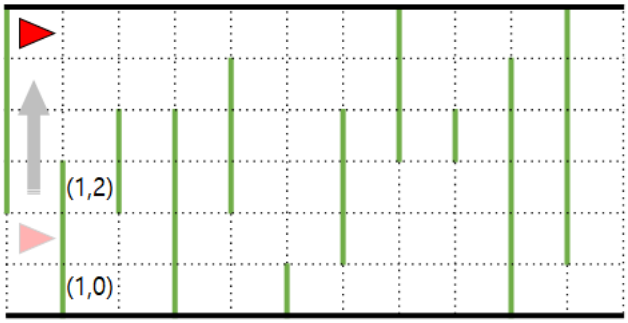
문제2 : 뚫기

가까운 미래, 싱가포르에서는 뚫기라는 게임이 유행 중이다. 게임 규칙은 간단하다. 쐐기 모양의 비행선이 $N \times M$ 크기의 터널을 왼쪽에서 오른쪽으로 통과하도록 움직이면 된다. 터널에는 비행선의 전진을 방해하기 위한 초록색의 막이 N개 존재한다. 막은 각 칸의 왼쪽 벽에 위치하며, 여러 칸에 걸쳐 연속되게 존재하는 막은 하나의 막으로 간주한다. 편의상 비행선의 전진 방향을 x축, 그에 수직된 방향을 y축으로 나타내면, 동일한 x좌표에는 하나의 막만 존재한다.

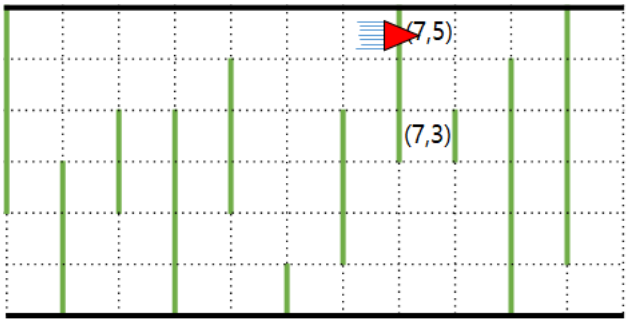
예를 들어, 아래 그림은 11×6 크기의 터널에 11개의 막이 존재하는 경우이다. 가장 왼쪽 막은 (0,2) 칸에서 (0,5) 칸까지 이어진 하나의 막이고, 가장 오른쪽 막은 (10,1) 칸에서 (10,5) 칸까지 이어진 하나의 막이다.



비행선은 각 칸에서 두 가지 움직임 중 한가지를 할 수 있다. 첫 번째 움직임은 순간이동이다. 순간이동은 현재 비행선이 위치한 칸과 동일한 x좌표의 임의의 칸으로 비행선을 이동시키는 것이다. 이 경우 이동하는 칸의 위치와는 무관하게 항상 비용 A만큼만 든다. 예를 들어, 아래 그림은 (0,1)에 위치한 비행선이 (0,5)로 순간이동하는 경우로, 비용 A가 들게 된다.



비행선의 두 번째 움직임은 전진이다. 비용은 0이다. 비행선이 전진하려는 방향에 막이 있더라도 비행선은 이를 뚫고 지나갈 수 있다. 다만 이 경우에는 비용 B가 든다. 예를 들어, 아래 그림은 (6,5)에 위치한 비행선이 (7,5)로 전진하는 경우로, 마침 (7,5)에는 막이 존재하여 비용 B가 들게 된다.



비행선이 터널을 완전히 통과하면 게임이 끝나게 되며, 게임 스코어는 터널을 통과할 때 발생한 비용의 총합으로 주어진다. 당연한 이야기지만 이 게임은 스코어가 낮을수록 좋다. 게임을 시작할 때 비행기의 y좌표는 아무 비용 없이 게이머가 선택할 수 있고, 게임이 끝날 때 비행기의 y좌표는 어디여도 상관없다.

터널의 크기와 막의 위치가 동일하더라도, 비용 A와 비용 B가 바뀌면 게임 스코어도 달라질 수 있고 최소의 게임 스코어를 얻기 위한 비행선의 움직임도 달라질 수 있다. 여러분은 주어진 터널의 크기와 막의 위치를 이용하여 비용 A와 비용 B가 바뀔 때마다 **가능한 가장 낮은 게임 스코어**를 구하기 위해서 다음 2가지 함수를 구현해야만 한다.

- `void init( int N, int M, int Y1[], int Y2[] )` ; 최초에 호출되며 단 한번 호출되는 함수이다. N과 M은 터널의 크기 $N \times M$을 나타낸다. Y1과 Y2는 막의 위치를 나타내는 크기 N인 배열로, (X, Y_1)부터 (X, Y_2)까지 하나의 막이 있다면 Y1[X]와 Y2[X]의 값은 Y_1과 Y_2이다.

- `long long minimize( int A, int B )` ; A는 비행선이 한번 순간이동하는 비용, B는 비행선이 한번 막을 뚫는 비용이다. 이를 이용하여 터널을 통과하는데 필요한 비용의 최솟값을 구하여 return 한다.

구현 세부사항

여러분은 breakthru.cpp라는 이름을 가진 하나의 파일을 제출해야만 한다. 이 파일에는 다음의 함수들이 구현되어 있어야 한다.

- `void init( int N, int M, int Y1[], int Y2[] ) ;`

- `long long minimize( int A, int B ) ;`

이 함수들은 위에서 설명한 것과 같이 동작하여야 한다. 물론, 다른 함수들을 만들어서 내부적으로 사용할 수 있다. 제출한 코드는 입출력을 수행하거나 다른 파일에 접근하여서는 안된다.

grader 예시

주어지는 grader는 다음과 같은 형식으로 입력을 읽는다.

- line 1: N M Q
  ($N \times M$: 터널의 크기, Q: 질의(Query)의 개수)
- line 2+i ($0 \le i \le N-1$): Y_1 Y_2
  (x좌표가 i인 막의 y좌표 위치가 Y_1부터 Y_2까지임)
- line 2+N+i ($0 \le i \le Q-1$): A B
  (A: 순간이동 비용, B: 막을 뚫는 비용)

주어지는 grader는 매 질의마다 여러분의 코드가 minimize() 함수에서 리턴한 값을 줄로 구분해서 출력한다.

제한 조건

- $1 \le N \le 10,000$
- $1 \le M \le 10^9$
- $1 \le Q \le 10^6$
- $0 \le Y_1 \le Y_2 \le M-1$
- $0 \le A, B \le 10^9$

서브태스크 1 [7 points]

- $Q=1$, $N \le 3,000$, $M \le 3,000$

서브태스크 2 [22 points]

- $Q \le 50$

서브태스크 3 [19 points]

- $N \le 500$

서브태스크 4 [21 points]

- $N \le 2,500$

서브태스크 5 [31 points]

- 추가 제한이 없다.

[입력 예 1]

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

[출력 예 1]

```
1
3
```

아래는 예 1에 대해 함수 호출 및 그 결과를 차례대로 보여준다.

함수호출	결과 값
`init( 3, 5, {2, 0, 1}, {4, 2, 3} )`	
`minimize( 2, 1 )`	1
`minimize( 3, 5 )`	3