백준 14658번 - 하늘에서 별똥별이 빗발친다

백준 14658번 - 하늘에서 별똥별이 빗발친다

시간제한	메모리 제한	제출	정답	맞은 사람	정답 비율
2초	256MB	1490	467	380	32.285%

문제

"오빠! 나 얼마만큼 사랑해?"

"널 위해서라면 저기 저 하늘의 별이라도 따다 줄 수 있어. 지금 따줄까?"

"에이, 거짓말!"

"정말이야. 한 번 봐봐!"

욱제는 하늘을 발로 차버렸다. 그랬더니 정말 별이 떨어졌다. 그런데, 정말로 별이 지구로 떨어지기 시작했다. 욱제는 지구를 지키는 정의의 용사가 되기로 결심했다.

"자기야, 나 세계를 지키고 올게. 꼭 돌아올 테니 조금만 기다려줘."

지구의 파괴를 막기 위해서는 지표면에 떨어지는 별똥별의 수를 최소화해야 한다. 욱제는 커다란 네모난 L*L 크기의 트램펄린을 준비했다. 별똥별이 어디로 떨어질지는 이미 알고 있기 때문에, 욱제는 이 트램펄린으로 최대한 많은 별똥별을 우주로 튕겨낼 계획이다. 하지만 학교 예산으로 트램펄린을 구매하는 욱제는 이 긴급한 와중에도 예산 심의 통과를 기다리느라 바쁘다!

욱제를 도와 세계를 구하자. 최대한 많은 별똥별을 튕겨내도록 트램펄린을 배치했을 때, 지구에는 몇 개의 별똥별이 부딪히게 될까? (별똥별이 떨어지는 위치는 겹치지 않으며 별똥별은 트램펄린의 모서리에 부딪혀도 튕겨나간다!) 트램펄린은 비스듬하게 배치 할 수 없다.

입력

첫째 줄에 네 정수 N, M, L, K가 주어진다. (1 ≤ N, M ≤ 500,000, 1 ≤ L ≤ 100,000, 1 ≤ K ≤ 100) N은 별똥별이 떨어지는 구역의 가로길이, M은 세로길이, L은 트램펄린의 한 변의 길이, K는 별똥별의 수를 뜻한다. 이후 K개의 줄에 걸쳐 별똥별이 떨어지는 위치의 좌표 (x, y)가 주어진다. (0 ≤ x ≤ N, 0 ≤ y ≤ M)

출력

욱제가 트램펄린으로 최대한 많은 별똥별을 튕겨낼 때, 지구에 부딪히는 별똥별의 개수를 출력한다.

예제 입력1

```
12 10 4 7
2 4
7 3
3 1
5 6
4 7
12 10
8 6
```

예제 출력1

```
3
```

출처

출처

알고리즘 분류

- 브루트포스 알고리즘

접근 방법

$L * L$ 크기의 정사각형 트램펄린을 $N * M$ 범위의 모든 구간들을 이동하며 완전탐색을 할 수는 없다.

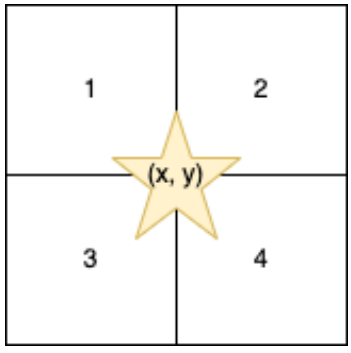
별들은 2차원 좌표계에서 점으로 표현이 되기 때문에, 하나의 트램펄린은 적어도 1개 이상의 별들을 커버할 수 있다는 것을 알 수 있다.

특정 별똥별이 최적해에 포함된다고 가정하자. 해당 별똥별의 좌표는 (x, y)라고 가정한다.

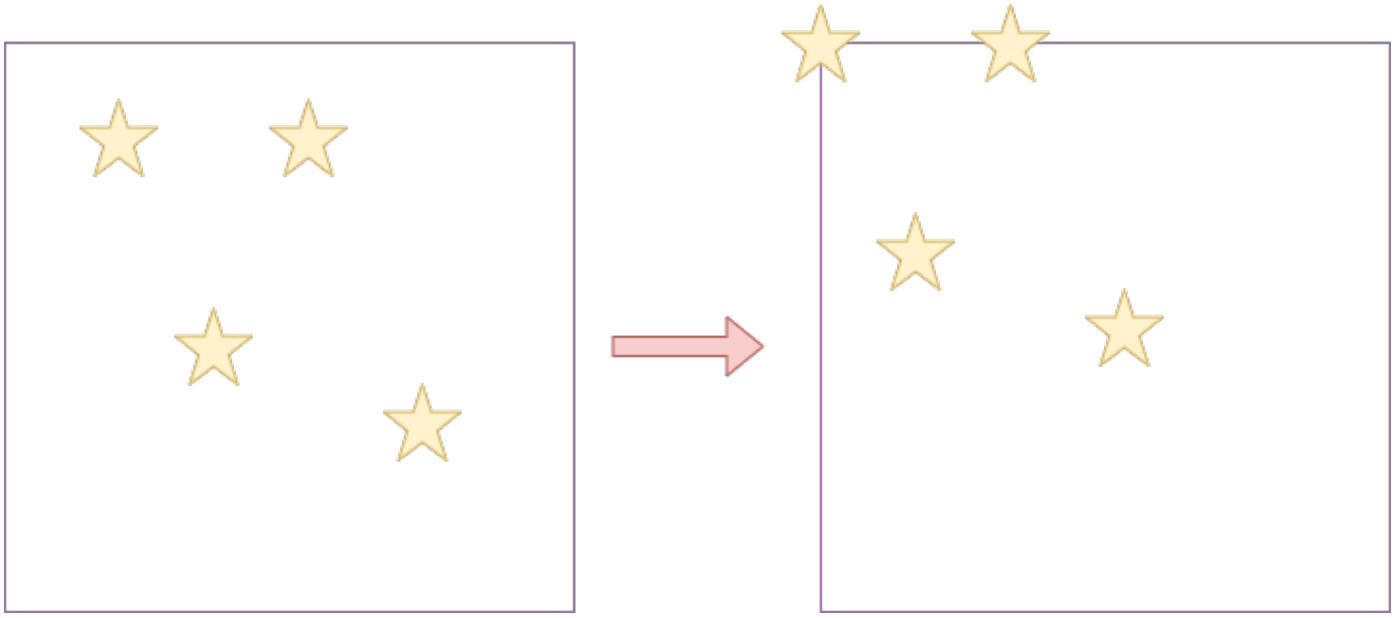
해당 별똥별이 포함되기 위해, 트램펄린을 어떻게 배치하는 것이 적절할까?

즉, 어떤 모양으로 배치를 해야 해당 별똥별을 포함하도록 트램펄린을 배치할 수 있을까?

바로 아래와 같은 모습일 것이다. 별똥별을 중점으로 하여 1, 2, 3, 4번 영역에 트램펄린을 배치하는 방법이 최선일 것이다. (이 경우, 적어도 1개의 별은 트램펄린의 빗변에 존재한다)

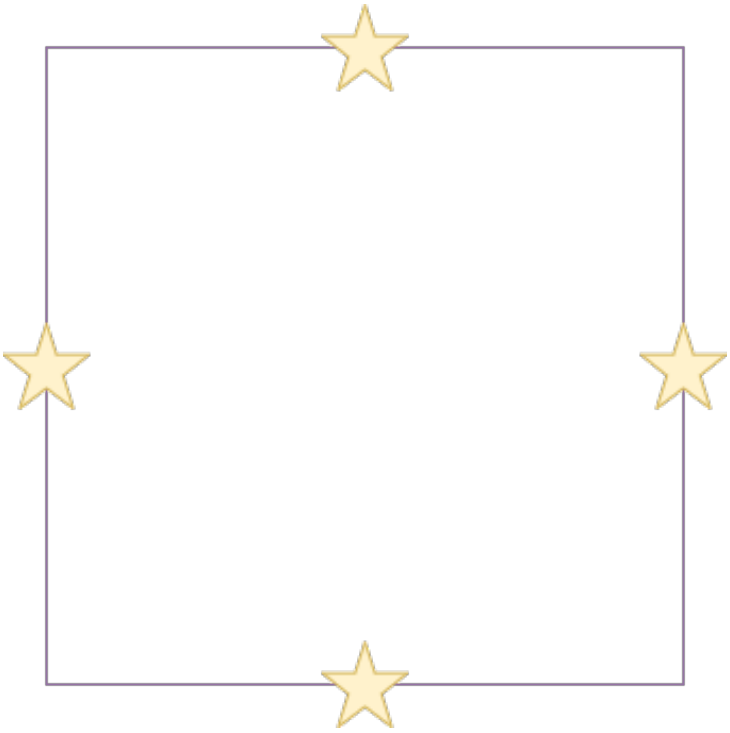


별똥별이 2개 이상 존재할 경우에, 최적해를 구성하는 모든 별들이 트램펄린 내부에 존재하는 경우가 생길 수 있다. 이 경우 또한 트램펄린을 적절하게 움직여서 빗변에 위치시키게 만들 수 있다.



그러면, 임의의 한 별을 잡고 해당 별을 중점으로 4사분면 모두를 고려하면 되지 않을까 생각이 들 수도 있다.

이러한 케이스도 존재한다. 처음 내가 접근했을 때 생각하지 못한 반례였다.



하나의 별을 기준으로 트램펄린을 배치하는 방법은 최적해를 구성하지 못한다.

이 경우, 임의의 2개의 별(동일한 2개의 별도 가능)을 기준으로 무게중심을 잡아 해당 무게중심을 기준으로 트램펄린을 배치하는 방법을 통해서 위의 반례도 해결할 수 있다.

소스코드

```
#define FASTIO cin.tie(0)->sync_with_stdio(false), cout.tie(0)
////////////////////////////////////////////////////////////////
#include <bits/stdc++.h>
```

```cpp
using namespace std;

int n, m, l, k;
vector<pair<int,int>> v;
int calc(int x, int y) {
    int res = 0;
    for (int i = 0; i < k; i++) {
        if (x <= v[i].first && v[i].first <= x + l && y <= v[i].second && v[i].second <= y + l) {
            res++;
        }
    }
    return res;
}
int main(void){
    FASTIO;
////////////////////////////////////////////////////////////////////

    cin >> n >> m >> l >> k;

    v.resize(k);
    for (auto &item : v) {
        cin >> item.first >> item.second;
    }

    int ans = 0;
    for (int i = 0; i < k; i++) {
        for (int j = 0; j < k; j++) {
            ans = max(ans, calc(v[i].first, v[j].second));
        }
    }

    cout << k - ans ;
    return 0;
}

```