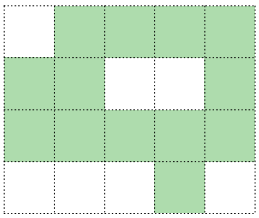
ㄹ. 로카히아 유적

시간 제한 : 3 초, 메모리 제한 : 512 MB

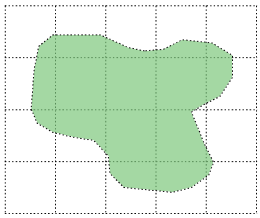
아르키미아 평야에는 한때 유구한 역사를 자랑했던 로카히아 왕국이 자리 잡았다. 역사학자 성진이는 로카히아 왕국의 유적을 분석하여 로카히아 왕국의 국경을 가늠해보려고 한다.

아르키미아 평야는 가로 M 킬로미터, 세로 N 킬로미터 크기의 직사각형 모양이며, 로카히아 왕국은 오직 아르키미아 평야 안에서만 흥망성쇠를 겪어왔다. 먼 옛날 아르키미아에서 태동한 고대 카타로티타 문명의 영향으로, 로카히아 왕국은 아르키미아 평야를 1km × 1km 단위로 측량하여 N × M 개의 블록으로 쪼갰다. 북쪽으로부터 R번째에, 서쪽으로부터 C번째에 있는 블록은 편의상 (R, C)로 표기하자.

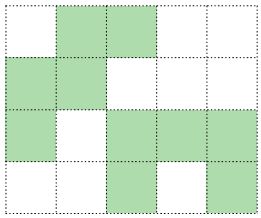
로카히아 왕국의 영토는 N × M 개의 블록 중 몇 개를 선택한 형태로 되어있다. 어떠한 시대에서도 영토 안의 임의의 두 지역 사이를 로카히아 왕국의 국경선이나 외부 지역을 지나지 않고 드나들 수 있다. (1)과 (2), (3)은 로카히아 왕국의 영토로 가능한 경우와 가능하지 않은 경우를 나타내고 있다.



(1) 가능한 영토 형태



(2) 불가능한 영토 형태



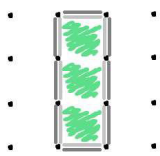
(3) 불가능한 영토 형태

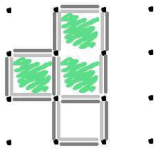
로카히아 왕국의 영토는 철저히 성벽으로 둘러싸여 있다. 로카히아의 성벽은 그 자체로도 훌륭한 예술적 가치를 지니고 있으며 그곳에 그려진 벽화도 역사적으로 의미 있다. 성진이는 아르키미아 평야에 W개의 로카히아 성벽이 남아있는 것을 확인했다.

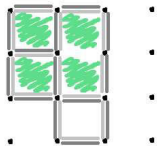
로카히아 왕국은 성벽을 단순히 국경선이라는 의미뿐만 아니라, 로카히아 문화의 정수로 보았기 때문에 왕실에서 세심하게 관리하고 있다. 로카히아 왕국에서는 성벽을 아래와 같은 규칙을 만족하도록 쌓았다.

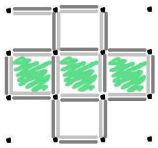
1. 모든 성벽은 안과 밖이 확실해야 한다. 성벽의 안쪽은 실제 로카히아 영토와 닿아있어야 하며, 성벽의 바깥쪽은 로카히아 외부 지역과 닿아있어야 한다.
2. 로카히아 영토와 그렇지 않은 지역이 맞닿아있다면, 반드시 그 사이에 성벽을 세워야 한다.
3. 왕실에서는 로카히아의 과거 성벽도 중요하게 여기므로, 영토 안에 있는 성벽을 딱히 철거하지 않는다. (다만, 외세 침입 등 여러 요인으로 인하여 영토 내부 혹은 외부 성벽이 전소될 수 있다.)
4. 왕국에서 국가 천도나 영토 확장 등의 사건이 일어나 영토가 바뀌는 경우, 이미 있던 성벽을 활용한다. 만약 성벽의 안팎이 달라서 사용할 수 없다면, 영토를 조정한다.

예를 들어 로카히아가 고대에서부터 근현대까지 (4), (5), (6), (7)과 같은 국경을 이룩했다면, 동시에 (4), (5), (6), (7)과 같이 성벽이 세워진다.

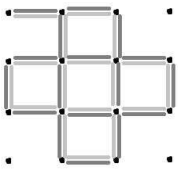

(4) 고대

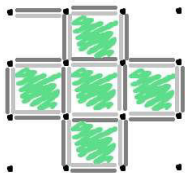

(5) 중세


(6) 근대


(7) 근현대

성진이가 (8)과 같이 남아있는 W개의 성벽을 가지고 영토를 추측했을 때 고대, 중세, 근현대 시대의 영토는 유추할 수 있으며, 어쩌면 (9)와 같은 형태의 영토도 생각할 수 있다. 근대 시대에 쌓아놓은 성벽 중 하나가 전소되었기 때문에 성진이가 근대 시대의 영토를 생각하긴 힘들다.


(8) 남은 성벽


(9) 다른 영토

성진이는 현재 아르키미아 평야에서 비슷한 문화를 가진 두 지역이 실제로 한 시대에 동시에 로카히아 왕국의 땅이었는지 알아보고 있다. 성진이는 오직 성벽의 분포만을 이용하여 판단하므로 (2, 3)과 (3, 2)는 실제 역사서에 같은 영토인 적이 없지만 성진이는 두 지역이 동시에 로카히아 왕국에 속해있었다고 간주한다. 아르키미아 평야가 매우 넓기 때문에 성진이가 일일이 손으로 처리하기에는 시간이 매우 오래 걸린다. 성진이를 도와 두 지역이 주어지면 한 시대에 동시에 로카히아의 영토였는지 빠르게 구하는 프로그램을 짜주자.

한편, 성진이의 친한 동생 도운이는 단지 역사서에 언급된 '로카히아 왕국은 전성기에 아르키미아 평야의 절반을 넘는(정확히 절반은 제외) 영토를 갖고 있었다.' 라는 문구의 진위를 알고 싶어한다. 아무리 친한 동생이라도 발굴중인 고대 유적을 직접 보여주는 것은 규정 위반이므로 성진이는 도운이가 '(R1, C1)과 (R2, C2)가 동시에 로카히아 왕국의 영토인 적이 있었어?'와 같은 방식으로만 질문하길 원한다. 도운이는 언급한 방식으로 성진이에게 질문하여 로카히아 왕국이 아르키미아 평야의 절반을 넘는 영토를 가진 적이 있었는지 알아보려고 한다. 이것 역시 실제로 로카히아 왕국이 넓은 영토를 차지한 적이 없어도 남아있는 성벽을 가지고 넓은 국경선을 만들 수 있으면 '그렇다'라고 간주한다.

도운이가 너무 많이 물어보면 성진이가 뭐라 그러기 때문에, 도운이는 최대한 적게 질문하여 로카히아 왕국의 규모를 유추하려고 한다. 도운이를 도와 성진이에게 가능한 적은 질문을 통해 로카히아 왕국의 규모를 유추하는 프로그램을 짜주자.

요구사항

당신은 성진이가 해야 할 일을 하는 프로그램 (sungjin.c / sungjin.cpp) 과 도운이가 해야 할 일을 하는 프로그램 (dowoon.c / dowoon.cpp) 을 작성해야 한다.

당신은 sungjin.c / sungjin.cpp 에서 다음과 같은 함수를 작성해야 한다.

void Init(int N, int M, int W, int R[], int C[], int dir[])

- N : 아르키미아 평야의 세로 크기
- M : 아르키미아 평야의 가로 크기
- W : 로카히아 왕국의 성벽 수
- R[i], C[i] : i번째 성벽의 좌표 ($0 \le i \le W-1$)
- dir[i] : i번째 성벽이 놓인 방향. dir[i]=1이면 (R[i], C[i]) 동쪽에, 바깥이 동쪽을 향하는 성벽이 있다는 것을 의미하며, dir[i]=2이면 서쪽에, dir[i]=3이면 남쪽에, dir[i]=4이면 북쪽에 성벽이 있다는 것을 의미한다.
- 하는 일 : WereSameTerritory가 호출되기 전 필요한 전처리를 여기서 하면 된다.

int WereSameTerritory(int R1, int C1, int R2, int C2)

- R1, C1 : 첫 번째 마을의 좌표 ($1 \le R1 \le N$, $1 \le C1 \le M$)
- R2, C2 : 두 번째 마을의 좌표 ($1 \le R2 \le N$, $1 \le C2 \le M$, $(R1, C1) \ne (R2, C2)$)
- 하는 일 : 두 마을이 동시에 로카히아의 영토였을 가능성이 있다면 1을 반환하고, 그렇지 않으면 0을 반환한다.
- 유의사항 : 실제 채점 프로그램은 이 함수의 반환값이 올바른지 모두 검사한다.

당신은 dowoon.c / dowoon.cpp 에서 다음과 같은 함수를 작성해야 한다.

int Guess(int N, int M)

- N : 아르키미아 평야의 세로 크기
- M : 아르키미아 평야의 가로 크기
- 하는 일 : 로카히아 왕국이 아르키미아 평야의 절반을 **넘게** 차지했을 가능성이 있다면 1을 반환하고, 아니면 0을 반환한다.
- 유의사항 : 당신은 이 함수 안에서 Ask(R1, C1, R2, C2)를 여러 번 호출할 수 있다. Ask(R1, C1, R2, C2) 함수는 sungjin.c / sungjin.cpp에 있는 함수 WereSameTerritory(R1, C1, R2, C2)와 하는 일이 동일하다. 단, R1, C1, R2, C2의 조건 ($1 \le R2 \le N$, $1 \le C2 \le M$, $(R1, C1) \ne (R2, C2)$)을 만족하지 않는 경우 프로그램을 바로 종료하며 오답으로 간주한다.

예시 프로그램은 다음과 같다. 주석은 예제 입력과 같은 상황에서 함수의 반환값을 의미한다.

```c
#include "sungjin.h"

void Init(int N, int M, int W, int R[], int C[], int dir[]) {
	// ToDo
}

int WereSameTerritory(int R1, int C1, int R2, int C2) {
	if (R1 == 2 && R2 == 2) return 1;
	if (C1 == 2 && C2 == 2) return 1;
	return 0;
}
```

sungjin.c / sungjin.cpp

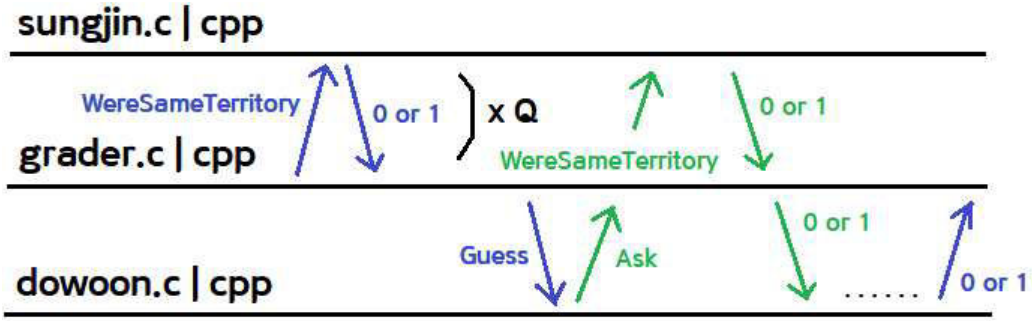
```c
#include "dowoon.h"

int Guess(int N, int M) {
	Ask(2, 1, 2, 3); // 반환값 = 1
	Ask(1, 2, 3, 2); // 반환값 = 1
	Ask(1, 1, 3, 3); // 반환값 = 0
	return 1;
}
```

dowoon.c / dowoon.cpp

샘플 인터페이스

문제 페이지에서 샘플 코드를 다운로드받을 수 있다. 만약 Visual Studio나 Eclipse, Code::Blocks와 같은 IDE 툴을 사용한다면 sungjin.cpp, sungjin.h, dowoon.cpp, dowoon.h, grader.cpp (또는 sungjin.c, sungjin.h, dowoon.c, dowoon.h, grader.c)를 한 프로젝트에 넣어서 컴파일하면 된다. 터미널에서 코드를 컴파일한다면 대회 페이지에 있는 컴파일 명령어를 이용하면 된다.

프로그램을 실행한 다음 표준입력(stdin)으로 N, M, W를 입력받고 W줄에 걸쳐 R[i], C[i], dir[i]를 입력받는다. 그런 다음 Q를 입력받고 Q줄에 걸쳐 R1, C1, R2, C2, ans를 입력받은 후 '로카히아 왕국의 가능한 최대 영토 넓이'를 입력받는다. 그러면 샘플 채점기는 정답/오답 여부와 Ask의 호출 횟수를 출력한다. 샘플 그레이더의 함수 호출 구조는 다음 그림과 같다.



답안을 제출할 때에는 sungjin.c와 dowoon.c를, 혹은 sungjin.cpp와 dowoon.cpp를 제출하면 된다.

제한

- N, M, W, Q는 정수이다.
- $2 \le N, M \le 1000$
- $0 \le W \le N(M+1) + M(N+1)$
- $0 \le Q \le 100{,}000$
- 모든 성벽은 겹쳐있지 않다. (같은 곳에 두 개 이상의 성벽이 있지 않다.)

서브태스크 1 (13점)

- 로카히아 왕국의 국경선으로 가능한 형태가 유일하며 그 국경선을 제외한 나머지 지역에는 성벽이 없다.
- Ask 함수는 최대 2MN번 호출할 수 있다.

서브태스크 2 (최대 87점)

- 추가 제한은 없다.
- 최악의 경우 Ask 함수의 호출 횟수를 C라고 했을 때, 다음과 같은 점수를 받는다.
  - $C > 2MN$ 이면 0점.
  - $MN < C \le 2MN$ 이면 $\left\lfloor 87 - 70 \times \left(\frac{C}{MN} - 1 \right) \right\rfloor$ 점 (소수점 아래 버림). 즉, $C = 1.5MN$ 이면 52점을 받는다.
  - $C \le MN$ 이면 87점.

입출력 예제

입력 (stdin)	출력 (stdout)
3 3 17 1 1 4 1 2 1 1 2 2 1 2 4 2 1 2 2 1 3 2 1 4 2 2 1 2 2 2 2 2 3 2 2 4 2 3 1 2 3 3 2 3 4 3 2 1 3 2 2 3 2 3 2 2 1 2 2 1 1 2 3 3 0 5	Sungjin does Correct! Dowoon does Correct! 3

참고

메모리 접근, 시스템 호출 등의 비정상적인 방법을 사용하면 오답 처리될 수 있음에 유의하여라.

※ 채점기가 쓰는 시간이 약 1.5초이므로 실제로 당신의 프로그램은 1.5초 이내에 돌아가야 한다.