

## 문제 1. 별자리 3

입력 파일: standard input  
출력 파일: standard output  
시간 제한: 2 seconds  
메모리 제한: 512 megabytes

JOI군은 야경 사진을 찍었다. 이 사진은 가로  $N$ 개, 세로  $N$ 개의 픽셀로 되어있는 사진이다. 왼쪽에서  $x$  번째 열, 아래에서  $y$  번째 행 ( $1 \leq x \leq N, 1 \leq y \leq N$ ) 에 있는 픽셀을 픽셀  $(x, y)$  라고 부른다.

이 사진의 각 픽셀은 빌딩, 밤하늘 혹은 별이다. 색은 각각 하얀색, 검은색, 노란색이다. 각  $i$ 에 ( $1 \leq i \leq N$ ) 대해  $i$ 번째 열에 있는 픽셀 중 아래에서  $A_i$ 행까지는 하얀색 픽셀이다. 별이 찍힌 노란색 픽셀은  $M$ 개가 있고, 그 중  $j$  번째 ( $1 \leq j \leq M$ ) 픽셀은 픽셀  $(X_j, Y_j)$ 이다. 이 이외의 픽셀은 모두 밤하늘을 찍은 검은색 픽셀이다. 이 사진의 어느 직사각형 영역이 다음 두 조건을 만족한다면 **별자리를 찍은 것**이 된다.

- 직사각형 영역 내에 하얀색 픽셀이 존재하지 않는다.
- 직사각형 영역 내에 노란색 픽셀이 두 개 이상 존재한다.

JOI군은 별자리를 보는 것이 지쳤다. 몇몇 노란색 픽셀을 검은색으로 칠하는 것으로 어떤 직사각형 영역도 별자리를 찍은 것이 되지 않도록 하고 싶다. 하지만 너무 많은 노란색 픽셀을 없애 버린 경우 사진의 **부자연스러움**이 올라간다. 구체적으로는  $j$  번째 ( $1 \leq j \leq M$ ) 노란색 픽셀을 검은색으로 만들면 사진의 부자연스러움이  $C_j$  증가한다. 처음 사진의 부자연스러움은 0이다.

사진의 정보와 각 노란색 픽셀을 없앴을 때 증가하는 부자연스러움이 주어졌을 때, 어떤 직사각형 영역도 별자리를 찍은 것이 되지 않도록 하면서 노란색 픽셀을 지운 사진의 부자연스러움의 최솟값을 구하는 프로그램을 작성하여라.

### 입력 형식

표준 입력에서 다음과 같은 형식으로 주어진다. 모든 값은 정수이다.

$N$   
 $A_1 \cdots A_N$   
 $M$   
 $X_1 Y_1 C_1$   
 $\vdots$   
 $X_M Y_M C_M$

### 출력 형식

어떤 직사각형 영역도 별자리를 찍은 것이 되지 않도록 하면서 노란색 픽셀을 지울 때, 사진의 부자연스러움의 최솟값을 표준 출력으로 첫째 줄에 출력하여라.

### 제한

- $1 \leq N \leq 200\,000$ .
- $1 \leq A_i \leq N$  ( $1 \leq i \leq N$ ).
- $1 \leq M \leq 200\,000$ .
- $1 \leq X_j \leq N$  ( $1 \leq j \leq M$ ).
- $1 \leq Y_j \leq N$  ( $1 \leq j \leq M$ ).
- $1 \leq C_j \leq 1\,000\,000\,000$  ( $1 \leq j \leq M$ ).

- $A_{X_j} < Y_j$  ( $1 \leq j \leq M$ ).
- $(X_j, Y_j) \neq (X_k, Y_k)$  ( $1 \leq j < k \leq M$ ).

### 서브태스크 1 (14 점)

- $N \leq 300$ .
- $M \leq 300$ .

### 서브태스크 2 (21 점)

- $N \leq 2\,000$ .
- $M \leq 2\,000$ .

### 서브태스크 3 (65 점)

추가 제한조건이 없다.

### 예제

standard input	standard output
5 1 3 4 2 3 3 1 5 3 4 3 2 2 4 2	2

이 입력 예제에서, 픽셀 (1,5)을 왼쪽 아래의 정점, 픽셀 (2,4)를 오른쪽 아래의 정점으로 하는 직사각형 영역은 별자리를 찍은 것이다. 세 번째 노란색 픽셀을 검은색으로 만들면 사진의 부자연스러움은 2 증가하고 어떤 직사각형 영역도 별자리를 찍은 것이 되지 않는다. 이것이 최솟값이므로 2를 출력한다.

이 입력 예제의 사진은 (그림 1)에 대응된다.

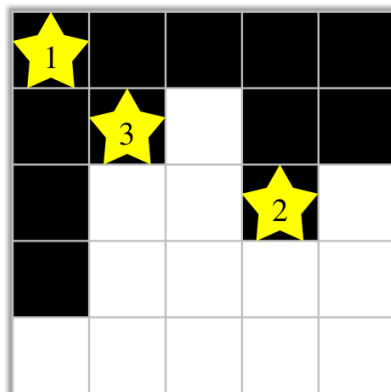


그림 1

standard input	standard output
7 5 6 2 3 6 7 6 5 7 7 5 3 3 7 3 7 10 1 7 6 4 7 8	16

이 입력 예제에서, 세 번째와 네 번째 노란색 픽셀을 검은색으로 만들면 된다.

standard input	standard output
8 6 8 5 7 3 4 2 1 10 8 2 9 6 6 7 8 3 18 5 8 17 8 5 3 5 5 3 5 4 8 1 8 13 1 7 5 7 4 13	44

## 문제 2. 수확

입력 파일: standard input  
출력 파일: standard output  
시간 제한: 3 seconds  
메모리 제한: 512 megabytes

IOI 농장은 사과를 기르는 거대한 농장이다. 이 농장은 큰 원 모양의 호수 둘레에 위치한 것으로 유명하다. IOI 농장에는 1번부터  $N$ 번까지 번호가 붙어있는  $N$ 명의 직원이 있다. 또한, IOI 농장에는  $M$ 개의 사과나무가 심겨 있고, 1번부터  $M$ 번까지 번호가 붙어 있다. 호수의 둘레는  $L$  미터이다.

처음에,  $i$ 번 ( $1 \leq i \leq N$ ) 직원은 호수의 가장 북쪽에 위치한 점으로부터 시계방향으로  $A_i$  미터 떨어진 곳에서 기다리고 있다.  $A_i$ 는 ( $1 \leq i \leq N$ ) 서로 다르다.  $j$ 번 ( $1 \leq j \leq M$ ) 사과나무는 호수의 가장 북쪽에 위치한 점으로부터 시계방향으로  $B_j$  미터 떨어진 곳에 심겨 있다.  $B_j$ 는 ( $1 \leq j \leq M$ ) 모두 다르다. 게다가, 어떤 사과나무도 직원이 있는 곳에 심겨 있지 않다.

IOI 농장에 심어진 사과를 품종개량을 하기 위한 결과, 하나의 나무에는 한 번에 하나의 사과만 열린다. 그리고 사과가 수확되었을 때 새로운 사과는 정확히  $C$ 초 이후에 생긴다. 시각 0 초에 모든 사과나무에는 사과가 열려 있고 모든 직원이 시계방향으로 걷기 시작한다. 모든 직원의 속도는 초당 1 미터이다. 만약 직원이 사과나무에 도착했을 때 사과나무에 사과가 열려 있으면 직원은 그 사과를 항상 수확할 것이다(만약 사과가 열리는 시간과 동시에 직원이 사과나무에 도착한다면 직원은 그 사과도 항상 수확한다). 우리는 사과를 수확하는 데 걸리는 시간을 무시할 것이다.

K 이사장은 IOI 농장의 주식 지분을 가지고 있다. 당신이 IOI 농장의 관리인이기 때문에 K 이사장은 직원의 효율성에 대해서 보고하기를 바란다. 정확히는 K 이사장은 다음  $Q$ 개의 값을 알고 싶다.

각  $k$ 에 ( $1 \leq k \leq Q$ ) 대해,  $V_k$ 번 직원이 시각  $T_k$  초까지 수확한 사과의 개수 (시각  $T_k$ 에 수확한 사과가 존재한다면, 이도 포함한다.)

직원의 수, 사과나무의 수, 호수의 둘레, 새로운 사과가 열리는 데 필요한 시간, 직원의 위치, 사과나무의 위치와  $Q$ 개의 쿼리에 대한 정보가 주어졌을 때, 각 쿼리에 대해 수확한 사과의 개수를 출력하여라.

### 입력 형식

표준 입력에서 다음과 같은 형식으로 주어진다. 모든 수는 정수이다.

$N \ M \ L \ C$

$A_1 \cdots A_N$

$B_1 \cdots B_M$

$Q$

$V_1 \ T_1$

$\cdots$

$V_Q \ T_Q$

### 출력 형식

표준 출력에  $Q$ 개의 줄을 출력하여라.  $k$  번째 ( $1 \leq k \leq Q$ ) 줄은,  $k$  번째 쿼리에 대한 답이다.

### 제한

- $1 \leq N \leq 200\ 000$ .
- $1 \leq M \leq 200\ 000$ .
- $N + M \leq L \leq 1\ 000\ 000\ 000$ .

- $1 \leq C \leq 1\,000\,000\,000$ .
- $0 \leq A_i < L$  ( $1 \leq i \leq N$ ).
- $A_i < A_{i+1}$  ( $1 \leq i \leq N-1$ ).
- $0 \leq B_j < L$  ( $1 \leq j \leq M$ ).
- $B_j < B_{j+1}$  ( $1 \leq j \leq M-1$ ).
- $A_i \neq B_j$  ( $1 \leq i \leq N, 1 \leq j \leq M$ ).
- $1 \leq Q \leq 200\,000$ .
- $1 \leq V_k \leq N$  ( $1 \leq k \leq Q$ ).
- $1 \leq T_k \leq 1\,000\,000\,000\,000\,000\,000 = 10^{18}$  ( $1 \leq k \leq Q$ ).

### 서브태스크 1 (5 점)

- $N \leq 3\,000$ .
- $M \leq 3\,000$ .
- $Q \leq 3\,000$ .

### 서브태스크 2 (20 점)

- $T_k \geq 1\,000\,000\,000\,000\,000 = 10^{15}$  ( $1 \leq k \leq Q$ ).

### 서브태스크 3 (75 점)

추가 제한조건이 없다.

### 예제

standard input	standard output
3 2 7 3	2
1 4 6	1
0 5	1
3	
1 7	
2 3	
3 8	

- 시각 1 초에, 2번 직원은 2번 사과나무에서, 3번 직원은 1번 사과나무에서 사과를 수확한다.
- 시각 3 초에, 2번 직원은 1번 사과나무에 도착한다. 그때 사과가 열리지 않았으므로, 직원은 사과를 수확하지 않는다.
- 시각 4 초에, 1번 직원은 2번 사과나무에서 사과를 수확한다.
- 시각 6 초에, 1번 직원은 1번 사과나무에서 사과를 수확한다. 3번 직원은 2번 사과나무에 도착하지만, 그 때 사과가 열리지 않았으므로, 직원은 사과를 수확하지 않는다.
- 시각 8 초에, 2번 직원은 2번 사과나무에서 사과를 수확한다. 3번 직원은 1번 사과나무에 도착하지만, 그 때 사과가 열리지 않았으므로, 직원은 사과를 수확하지 않는다.

시각 7 초까지 1번 직원이 수확한 사과와 개수의 개수가 2개이므로, 첫째 줄에는 2를 출력한다.

standard input	standard output
5 3 20 6 0 4 8 12 16 2 11 14 9 4 1932 2 93787 1 89 5 98124798 1 2684 1 137598 3 2 3 8375 4 237	146 7035 7 7359360 202 10320 0 628 18
8 15 217 33608 0 12 71 96 111 128 152 206 4 34 42 67 76 81 85 104 110 117 122 148 166 170 212 14 2 223544052420046341 3 86357593875941375 4 892813012303440034 1 517156961659770735 7 415536186438473633 6 322175014520330760 7 557706040951533058 6 640041274241532527 5 286263974600593111 8 349405886653104871 1 987277313830536091 5 989137777159975413 2 50689028127994215 7 445686748471896881	33230868503053 3 5 1 123542793648997 8 165811220737767 8 7 1 1 7 7535161012043 132506837660717

## 문제 3. 길고양이

시간 제한: 2 seconds  
메모리 제한: 512 megabytes

Anthony는 JOI 도시에 사는 개미이다. JOI 도시에는 0번 부터  $N - 1$ 번까지 번호가 붙어있는  $N$ 개의 마을이 있다. Anthony는 0번 마을에 살고 있다. 또한, 0번 부터  $M - 1$ 번까지 번호가 붙어 있는  $M$ 개의 도로가 있다.  $i$ 번 ( $0 \leq i \leq M - 1$ ) 도로는 마을  $U_i$ 와  $V_i$ 를 양방향으로 잇고 있다. 여러 개의 도로가 같은 쌍의 마을을 연결하는 경우는 없다. 어떤 마을에서도 다른 모든 마을까지 몇 개의 도로를 거치면 이동하는 것이 가능하다.

Anthony의 친구인 고양이 Catherine이 JOI 도시에 놀러 올 예정이다. Catherine은 도로의 정보를 모르기 때문에 길을 잃는 일이 잦았다. Anthony는 도로에 표식을 표시하기로 생각했다. 표식에는 0번부터  $A - 1$ 번까지 번호가 붙어 있는  $A$  종류의 표식이 있다.

이제, Catherine은 JOI 도시의 한 마을에 도착했다. Catherine은 0번 마을을 제외한 마을에 있을 경우,

(존재한다면) 직전에 지났던 도로를 제외하고, 지금 있는 마을에 인접한 도로 중 어떤 표식이 몇 개 있는가

를 셀 수 있고, 다음의 어떤 도로로 가는지 고르는 것이 가능하다. 직전에 지났던 도로를 제외하고는, 도로를 표식의 종류로만 구분할 수 있다. 도로를 올바르게 고르는 것으로, 0번 마을까지 시간을 너무 많이 소비하지 않고 가고 싶다. 구체적으로는, 0번 마을까지 가는데 사용하는 도로의 최소 개수가  $d$  개인 경우, 0번 마을까지 최대  $d + B$  개만 사용하여 도착하고 싶다.

도로의 정보가 주어졌을 때, 도로에 표식을 표시하는 Anthony의 전략을 구현한 프로그램과 도로의 정보를 받아서 길을 걷는 Catherine의 전략을 구현한 프로그램을 작성하여라.

### 구현 명세

당신은 파일 두 개를 제출해야 한다.

첫째 파일의 이름은 `Anthony.cpp`이다. 이 파일은 Anthony의 일을 나타내고, 다음 함수를 구현해야 한다. 또한, `Anthony.h`를 include해야 한다.

```
• std::vector<int> Mark(int N, int M, int A, int B,  
                      std::vector<int> U, std::vector<int> V)
```

이 함수는 최초에 정확히 한 번 불린다.

- 인자  $N$ 은 마을의 수  $N$ 을 나타낸다.
- 인자  $M$ 은 도로의 수  $A$ 를 나타낸다.
- 인자  $A$ 는 표식의 가짓수  $A$ 를 나타낸다.
- 인자  $B$ 는 길을 걷는 횟수의 여유  $B$ 를 나타낸다.
- 인자  $U, V$ 는 길이  $M$ 의 배열이다.  $U[i]$ 와  $V[i]$ 는  $i$ 번 도로가 연결하는 두 마을의 번호  $U_i, V_i$ 를 의미한다. ( $0 \leq i \leq M - 1$ )
- 반환값  $x$ 는 길이  $M$ 의 배열이어야 한다. 길이가  $M$ 이 아닐 경우 **오답 [1]**이 된다.  $x[i]$ 는 ( $0 \leq i \leq M - 1$ ) 도로  $i$ 에 붙어있는 표식의 번호를 의미한다.  $0 \leq x[i] \leq A - 1$ 을 만족해야 한다.  $0 \leq x[i] \leq A - 1$ 이 아닐 경우 **오답 [2]**이 된다.

둘째 파일의 이름은 `Catherine.cpp`이다. 이 파일은 Catherine의 일을 나타내고, 다음 함수를 구현해야 한다. 또한, `Catherine.h`를 include해야 한다.

```
• void Init(int A, int B)
```

이 함수는 최초에 정확히 한 번 불린다.

- 인자  $A$ 는 표식의 가짓수  $A$ 를 나타낸다.

– 인자  $B$ 는 길을 걷는 횟수의 여유  $B$ 를 나타낸다.

• `int Move(std::vector<int> y)`

이 함수는 Catherine이 0번 마을 이외의 마을을 방문할 때마다 불린다.

- 인자  $y$ 는 배열이고, (존재한다면) 직전에 지났던 도로를 제외하고, 지금 있는 마을에 인접한 도로 중  $j$  번째 ( $0 \leq j \leq A-1$ ) 표식이  $y[j]$ 개 있다는 것을 의미한다.
- 반환값  $z$ 는  $-1 \leq z \leq A-1$ 을 만족해야 한다.  $-1 \leq z \leq A-1$ 을 만족하지 않을 경우 **오답 [3]**이 된다.  $z = -1$ 인 경우 직전에 왔던 도로로 돌아간다는 것을 의미하고,  $0 \leq z \leq A-1$ 인 경우 직전에 지났던 도로 이외의  $z$ 번 표식이 붙어있는 도로를 고른다는 것을 의미한다. 함수 `Move`가 처음으로 호출 될 때,  $z = -1$ 인 경우 **오답 [4]**이 된다.  $0 \leq z \leq A-1$  이고  $y[z] = 0$  인 경우 **오답 [5]**이 된다.

Catherine은 어떤 마을에서 직전에 지났던 도로 이외의 길을 고르는 경우 도로는 `Move`의 반환값에 해당하는 표식이 있는 도로 중 하나를 따라간다. 이 선택이 무작위가 아닐 수 있음에 유의하여라.

Catherine이 0번 마을에  $d + B$ 개의 도로를 사용한 이후 (즉, 함수 `Move`가  $d + B$  번 호출 된 이후)에 도착하지 못할 경우, **오답 [6]**이 된다.

## 참고 사항

- 당신의 프로그램은 내부에서 사용할 목적으로 함수나 전역변수를 사용할 수 있다. 제출된 파일들은 같이 컴파일되어 하나의 실행 파일이 된다. 모든 글로벌 변수나 함수는 충돌을 피하기 위해 이름이 없는 namespace에 구현되어야 한다. 채점 될 때는, Anthony와 Catherine에 해당하는 두 프로세스로 나누어서 실행될 것이다. Anthony의 프로세스와 Catherine의 프로세스는 전역변수를 공유할 수 없다.
- 당신의 프로그램은 표준 입출력을 사용해서는 안 된다. 당신의 프로그램은 어떠한 방법으로도 다른 파일에 접근해서는 안 된다. 단, 당신의 프로그램은 디버그 목적으로 표준 에러출력에 출력할 수 있다.

당신은 대회 홈페이지의 아카이브에서 프로그램을 테스트하기 위한 목적의 샘플 그레이더를 받을 수 있다. 아카이브는 당신의 프로그램의 예제 소스 또한 첨부되어 있다. 샘플 그레이더는 파일 `grader.cpp`이다. 당신의 프로그램을 테스트하기 위해서, `grader.cpp`, `Anthony.cpp`, `Catherine.cpp`, `Anthony.h`, `Catherine.h`를 같은 디렉토리 안에 놓고, 컴파일하기 위해 다음 커맨드를 실행하여라.

• `g++ -std=gnu++14 -O2 -o grader grader.cpp Anthony.cpp Catherine.cpp`

컴파일이 성공적이면, 파일 `grader`가 생성된다.

실제 그레이더와 샘플 그레이더는 다름에 주의하여라. 샘플 그레이더는 하나의 프로세스에서 실행되며, 입력을 표준 입력으로부터 받고, 출력을 표준 출력에 출력한다.

## 입력 형식

샘플 그레이더는 표준 입력에서 다음과 같은 형식으로 입력받는다.

$N \ M \ A \ B \ S$

$U_0 \ V_0$

$\vdots$

$U_{M-1} \ V_{M-1}$

여기서  $S$ 는 Catherine이 처음으로 도착하는 말의 번호이다.

## 출력 형식

프로그램이 정상적으로 종료되었다면, 샘플 그레이더는 다음과 같은 정보를 표준 출력 및 표준 에러에 출력한다. (따옴표는 출력하지 않는다.)



- 오답 [1], [2], [3], [4] 혹은 [5]로 판단 된 경우, 오답의 종류를 “Wrong Answer [1]”과 같은 형식으로 출력한다.
- Catherine이 0번 마을에  $N + B$ 번 이동 이내에 도착하지 못한 경우, “Wrong Answer; Number of moves > N+B”를 출력한다.
- 아닌 경우, 이동 횟수(Move의 호출 횟수)를 “Number of moves = 4”와 같은 형식으로 출력한다. 샘플 그레이더는 오답 [6] 인지 아닌지를 검사하지 않는다.

프로그램이 다양한 오답의 종류에 속해 있으면, 샘플 그레이더는 그중 하나만 출력할 것이다.

샘플 그레이더에서, Catherine이 자기가 직전에 지났던 도로 이외의 도로를 선택 할 때, 선택한 도로는 Move의 반환값에 해당하는 표식이 있는 도로 중 하나가 동일한 확률로 선택되며, 이는 주어진 시드에 따른 유사난수생성기에 의해 결정된다. 다른 시드로 프로그램을 실행하고 싶은 경우, 그레이더를 다음과 같은 형식으로 실행하여라:

```
./grader 2020
```

여기서, 첫 번째 인자는 유사난수생성기의 시드로 사용되는 정수이다.

## 제한

- $2 \leq N \leq 20\,000$ .
- $1 \leq M \leq 20\,000$ .
- $1 \leq S \leq N - 1$ .
- $0 \leq U_i < V_i \leq N - 1$  ( $0 \leq i \leq M - 1$ ).
- $(U_i, V_i) \neq (U_j, V_j)$  ( $0 \leq i < j \leq M - 1$ ).
- 어떤 마을에서도 다른 모든 마을까지 몇 개의 도로를 거치면 이동하는 것이 가능하다.

## 서브태스크 1 (2 점)

- $A = 4$ .
- $B = 0$ .
- $M = N - 1$ .

## 서브태스크 2 (2 점)

- $A = 4$ .
- $B = 0$ .

## 서브태스크 3 (2 점)

- $A = 3$ .
- $B = 0$ .
- $M = N - 1$ .

## 서브태스크 4 (9 점)

- $A = 3$ .
- $B = 0$ .

## 서브태스크 5 (5 점)

- $A = 2$ .
- $B = 2N$ .
- $M = N - 1$ .
- $6 \leq N \leq 500$ .

## 서브태스크 6 (71 점)

- $A = 2$ .
- $B = 12$ .
- $M = N - 1$ .

## 서브태스크 5 (9 점)

- $A = 2$ .
- $B = 6$ .
- $M = N - 1$ .

## 예제

이 함수는 그레이더의 예제 입력과 해당하는 함수 호출을 보여준다.

예제 입력
7 6 2 6 1
0 2
0 4
1 2
1 3
1 5
4 6

Anthony		Catherine	
호출	반환값	호출	반환값
Mark(7,6,2,6,[0,0,1,1,1,4],[2,4,2,3,5,6])	[1,0,0,1,0,1]		
		Init(2, 6)	
		Move([2, 1])	0
		Move([0, 0])	-1
		Move([1, 1])	0
		Move([0, 1])	1

이 예제 입력에서, Catherine은 1, 5, 1, 2, 0 순으로 마을을 방문한다.  $d = 2$ 이고, Catherine은 4번 움직였다.  
이 입력은 서브태스크 7의 조건을 만족한다.

대회 홈페이지의 아카이브에서 받을 수 있는 파일 중, sample-02.txt는 서브태스크 4의 조건을 만족한다.