문제 1. 시험

입력 파일:	`standard input`
출력 파일:	`standard output`
시간 제한:	3초
메모리 제한:	1024MB

N 명의 학생이 수학 부문과 정보 부문이 있는 시험을 쳤다. i 번째 $(1 \leq i \leq N)$ 학생은 수학에서는 S_i 점을, 정보에서는 T_i 점을 받았다. T 교수와 I 교수는 각 학생이 시험을 통과할지 말지를 점수를 기반으로 정하려고 한다.

- T 교수는 두 과목을 모두 중요하게 본다. 수학에서 A 점 이상, 정보에서 B 점 이상을 받아야 통과한 것으로 생각한다.
- I 교수는 총점만 중요하게 본다. 수학과 정보를 합쳐서 C 점 이상을 받아야 통과한 것으로 생각한다.
- 두 교수의 기준을 모두 통과한 학생만 시험을 통과할 수 있다.

당신은 기준인 A, B, C를 모른다. 하지만, Q가지의 세 정수 $(X_j,\ Y_j,\ Z_j)$ $(1 \leq j \leq Q)$ 가 주어져서 몇 명의 학생들이 $A = X_j,\ B = Y_j,\ C = Z_j$일 때 시험을 통과하는지 알고 싶다.

학생들의 수, 점수 정보와 점수 기준이 주어졌을 때, 시험을 통과하는 학생의 수를 구하여라.

입력 형식

표준 입력에서 다음과 같은 형식으로 주어진다. 모든 값은 정수이다.

$N\ Q$

$S_1\ T_1$

$\vdots$

$S_N\ T_N$

$X_1\ Y_1\ Z_1$

$\vdots$

$X_Q\ Y_Q\ Z_Q$

출력 형식

표준 출력으로 Q 개의 줄을 출력하여라. j 번째 $(1 \leq j \leq Q)$ 줄은 몇 명의 학생들이 $A = X_j,\ B = Y_j,\ C = Z_j$ 일 때 시험을 통과하는 학생 수이다.

제한

- $1 \leq N \leq 100\ 000$.
- $1 \leq Q \leq 100\ 000$
- $0 \leq S_i \leq 1\ 000\ 000\ 000$ $(1 \leq i \leq N)$.
- $0 \leq T_i \leq 1\ 000\ 000\ 000$ $(1 \leq i \leq N)$.
- $0 \leq X_j \leq 1\ 000\ 000\ 000$ $(1 \leq j \leq Q)$.
- $0 \leq Y_j \leq 1\ 000\ 000\ 000$ $(1 \leq j \leq Q)$.
- $0 \leq Z_j \leq 2\ 000\ 000\ 000$ $(1 \leq j \leq Q)$.

서브태스크 1 (2 점)

- $N \leq 3\ 000$
- $Q \leq 3\ 000$

서브태스크 2 (20 점)

- $S_i \leq 100\ 000$ $(1 \leq i \leq N)$.
- $T_i \leq 100\ 000$ $(1 \leq i \leq N)$.
- $X_j \leq 100\ 000$ $(1 \leq j \leq Q)$.
- $Y_j \leq 100\ 000$ $(1 \leq j \leq Q)$.
- $Z_j = 0$ $(1 \leq j \leq Q)$.

서브태스크 3 (21 점)

- $S_i \leq 100\ 000$ $(1 \leq i \leq N)$.
- $T_i \leq 100\ 000$ $(1 \leq i \leq N)$.
- $X_j \leq 100\ 000$ $(1 \leq j \leq Q)$.
- $Y_j \leq 100\ 000$ $(1 \leq j \leq Q)$.
- $Z_j \leq 200\ 000$ $(1 \leq j \leq Q)$.

서브태스크 4 (57 점)

추가 제한조건이 없다.

예제

standard input	standard output
5 4 35 100 70 70 45 15 80 40 20 95 20 50 120 10 10 100 60 60 80 0 100 100	2 4 1 1
10 10 41304 98327 91921 28251 85635 59191 30361 72671 28949 96958 99041 37826 10245 2726 19387 20282 60366 87723 95388 49726 52302 69501 66009 43754 45346 3158 25224 58881 18727 7298 24412 63782 24107 10583 61508 65025 29140 7278 36104 56758 2775 23126 67608 122051 56910 17272 62933 39675 15874 117117	1 3 5 8 8 3 3 3 5 6

참고 사항

첫째 예제에서

- $A = 20,\ B = 50,\ C = 120$일 때, 첫 번째와 두 번째 학생만 수학 부문에서 최소 20점, 정보 시험에서 최소 50점, 그리고 총점 120점을 넘길 수 있다. 그래서 시험을 통과하는 학생들의 수는 2이다.
- $A = 10,\ B = 10,\ C = 100$일 때, 첫 번째, 두 번째, 네 번째 그리고 다섯 번째 학생만 수학 부문에서 최소 10점, 정보 시험에서 최소 10점, 그리고 총점 100점을 넘길 수 있다. 그래서 시험을 통과하는 학생들의 수는 4이다.
- $A = 60,\ B = 60,\ C = 80$일 때, 두 번째 학생만 수학 부문에서 최소 60점, 정보 시험에서 최소 60점, 그리고 총점 80점을 넘길 수 있다. 그래서 시험을 통과하는 학생들의 수는 1이다.
- $A = 0,\ B = 100,\ C = 100$일 때, 첫 번째 학생만 수학 부문에서 최소 0점, 정보 시험에서 최소 100점, 그리고 총점 100점을 넘길 수 있다. 그래서 시험을 통과하는 학생들의 수는 1이다.

문제 2. 비버의 모임

입력 파일:	`standard input`
출력 파일:	`standard output`
시간 제한:	2초
메모리 제한:	256MB

0부터 $N-1$까지의 번호가 붙은 비버가 사는 N 개의 섬이 있다. 각 섬은 $N-1$ 개의 양방향으로 연결되는 다리로 연결되어 있다. 몇 개의 다리를 사용하면 어떠한 두 섬을 오가는 것도 가능하다. **각 섬마다, 섬에 직접 연결된 다리는 최대 18개이다.** 각 섬에는 비버가 살고 있다.

가끔, 몇몇 비버는 특정 섬에서 모임을 한다. 세 비버가 만날 때 그들은 다음과 같은 조건을 만족하는 섬에서 만난다:

- 모임을 할 때 3마리의 비버가 자신이 살고 있는 섬부터 이동할 때 사용하는 다리의 수의 합이 최소가 되는 섬 (이런 섬은 유일하게 존재한다.)

이 섬은 3마리의 비버 중 하나가 사는 섬일 수 있음에 주의하여라.

당신은 N 개의 섬이 어떤 방식으로 다리로 연결되어 있는지가 궁금해졌다. 당신은 직접 이 섬을 확인할 수 없다. 그렇기 때문에, 당신은 비버에게 명령을 내리기로 했다. 명령은 다음과 같다.

- 세 개의 섬 u, v, w ($0 \leq u \leq N-1,\ 0 \leq v \leq N-1,\ 0 \leq w \leq N-1,\ u \neq v,\ u \neq w, v \neq w$)을 지정하고 u, v, w에 사는 비버끼리 모임을 하게 한다.
- 비버가 모임을 하는 섬을 알 수 있다.

당신은 섬이 어떤 방식으로 다리로 연결되어 있는지를 적은 수의 명령으로 알고 싶다. 섬의 수와 비버와 통신하는 방법이 주어졌을 때, 섬이 연결된 방식을 알아내어라.

구현 명세

당신은 파일 하나를 제출해야 한다.

이 파일의 이름은 `meetings.cpp`이다. 파일은 다음 함수를 구현해야 한다. 또한, `meetings.h`를 include해야 한다.

- `void Solve(int N)`

  이 함수는 각 테스트 케이스마다 정확히 한 번 불린다.

  - 인자 `N`은 섬의 수 N을 나타낸다.

  당신의 프로그램은 다음 함수를 호출 할 수 있다.

  - `int Query(int u, int v, int w)`
    이 함수는 주어진 세 개의 섬에 대해서 세 비버가 만나는 섬의 번호를 반환한다.
    * 당신은 섬의 번호 u, v, w를 인자 `u`, `v`, `w`를 사용해서 나타내어야 한다. 이 번호는 $0 \leq u \leq N-1,\ 0 \leq v \leq N-1,\ 0 \leq w \leq N-1,\ u \neq v,\ u \neq w, v \neq w$을 모두 만족해야 한다. 아닌 경우에는 **오답 [1]**이 된다.
    * 당신은 이 함수를 100 000번 이상 호출해서는 안 된다. 호출 한 경우에는 **오답 [2]**이 된다.
  - `void Bridge(int u, int v)`
    이 함수는 섬이 다리로 연결되어 있는지에 대한 정보를 답할 수 있다.
    * 인자 `u`와 `v`는 섬 u와 섬 v가 다리로 서로 직접 연결되어있다는 것을 나타낸다.
    * $0 \leq u < v \leq N-1$이 아닌 경우, **오답 [3]**이 된다.
    * 섬 u와 v가 직접 다리로 연결되어있지 않으면 **오답 [4]**이 된다.

* 함수가 같은 인자 u, v를 여러 번 호출한 경우 **오답** [5]이 된다.
* N-1개의 다리가 있으므로, 함수는 정확히 N-1번 호출되어야 한다. 만약 함수 Solve가 끝날 때 이 함수의 호출 횟수가 N-1이 아니면 **오답** [6]이 된다.

참고 사항

- 당신의 프로그램은 내부에서 사용할 목적으로 함수나 전역변수를 사용할 수 있다.
- 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 테스트 하기 위한 목적의 샘플 그레이더를 받을 수 있다. 아카이브는 당신의 프로그램의 예제 소스 또한 첨부되어 있다. 샘플 그레이더는 파일 grader.cpp이다. 당신의 프로그램을 테스트하기 위해서, grader.cpp, meetings.cpp, meetings.h를 같은 디렉토리 안에 놓고, 컴파일하기 위해 다음 커맨드를 실행하여라.

- `g++ -std=gnu++14 -O2 -o grader grader.cpp meetings.cpp`

컴파일이 성공적이면, 파일 grader가 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

A_0 B_0

$\vdots$

A_{N-2} B_{N-2}

A_i와 B_i $(0 \leq i \leq N-2)$는 섬 A_i와 B_i가 다리로 직접 연결되어있다는 것을 의미한다.

출력 형식

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

- 정답으로 판단된 경우, Query함수의 호출 횟수를 "Accepted: 100"과 같은 형식으로 출력한다.
- 오답으로 판단된 경우, 오답의 종류를 "Wrong Answer [1]"과 같은 형식으로 출력한다.

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

제한

샘플 그레이더의 A_i와 B_i의 정의에 따라서

- $3 \leq M \leq 2\ 000$.
- $0 \leq A_i < B_i \leq N-1$. $(0 \leq i \leq N-2)$
- 몇 개의 다리를 사용하면 어떠한 두 섬을 오가는 것도 가능하다.
- 각 섬마다, 섬에 직접 연결된 다리는 최대 18개이다.

서브태스크 1 (7 점)

- $N \leq 7$

서브태스크 2 (10 점)

- $N \leq 50$

서브태스크 3 (12 점)

- $N \leq 300$

서브태스크 4 (71 점)

추가 제한조건이 없다.

- 서브태스크 1, 2, 3에 대해서, 서브태스크 안에 있는 모든 테스트 케이스를 맞춘 경우 점수를 준다.
- 서브태스크 4에 대해서, X를 `Query`함수의 최대 호출횟수라고 하자.
  - $40\ 000 < X \leq 100\ 000$이면 49점을 받는다.
  - $X \leq 40\ 000$이면 71점을 받는다.

예제

이 함수는 그레이더의 예제 입력과 해당하는 함수 호출을 보여준다.

예제 입력	예제 함수 호출		
	호출	호출	반환값
	`Solve(5)`		

`5`
`0 1`
`0 2`
`1 3`
`1 4`
		`Query(0, 1, 2)`	`0`

		`Query(0, 3, 4)`	`1`
		`Bridge(1, 3)`	(없음)
		`Bridge(0, 2)`	(없음)
		`Bridge(1, 4)`	(없음)
		`Bridge(0, 1)`	(없음)

문제 3. 난

입력 파일:	`standard input`
출력 파일:	`standard output`
시간 제한:	4초
메모리 제한:	256MB

JOI 카레 매점은 매우 긴 난(인도의 납작한 빵)을 판매하는 것으로 유명하다. 난에는 L 개의 맛이 있으며, 1번부터 L번까지 번호가 붙어 있다. 난 중에서 "JOI 스페셜 난"이 제일 인기가 있다. 길이가 L cm 이고, 왼쪽에서 $j-1$ cm부터 j cm까지의 부분에는 j번 ($1 \leq j \leq L$) 맛으로 되어 있다.

N명의 사람이 JOI 카레 매점에 왔다. 그들의 취향은 다른 사람과 다르다. 구체적으로, i 번째 ($1 \leq i \leq N$) 사람이 j번 ($1 \leq j \leq L$) 맛의 난을 먹었을 경우에는, 1cm당 $V_{i,j}$의 행복도를 얻을 것이다. 그들은 하나의 JOI 스페셜 난을 주문했다. 그들은 난을 다음과 같은 방법으로 나누어 가질 것이다.

1. $0 < X_1 < X_2 < \cdots < X_{N-1} < L$을 만족하는 $N-1$개의 분수 $X_1, \cdots, X_{N-1}$를 고른다.
2. N 개의 정수 $P_1, \cdots, P_N$을 고른다. 이는 $1, \cdots, N$의 순열이어야 한다.
3. 각 k ($1 \leq k \leq N-1$)에 대해서, 난을 X_k cm 지점에서 자른다. 난은 N개의 조각으로 나누어질 것이다.
4. 각 k ($1 \leq k \leq N$)에 대해서, P_k 번째 사람에게 X_{k-1} cm와 X_k cm 사이의 조각을 준다. 우리는 X_0을 0, X_N을 L이라고 생각할 것이다.

우리는 난을 공평하게 나누고 싶다. 우리는 각 사람이 혼자 JOI 스페셜 난을 모두 먹었을 때 얻는 행복도의 $1/N$ 이상을 얻었을 경우, 분배 방식이 **공평하다**고 할 것이다.

N 명의 사람의 선호가 주어졌을 때, 난을 공평하게 나누는 방법이 있는가를 출력하여라. 있는 경우, 난을 공평하게 나누는 방법에 대해 출력하여라.

입력 형식

표준 입력에서 다음과 같은 형식으로 주어진다. 모든 수는 정수이다.

N L

$V_{1,1}$ $V_{1,2}$ $\cdots$ $V_{1,L}$

$\vdots$

$V_{N,1}$ $V_{N,2}$ $\cdots$ $V_{N,L}$

출력 형식

난을 공평하게 나누는 방법이 없다면, `-1`을 첫째 줄에 출력하여라. 공평하게 나눌 수 있다면, 나누는 방법을 나타내는 $N-1$ 개의 분수 $X_1, \cdots, X_{N-1}$과 N 개의 정수 $P_1, \cdots, P_N$을 다음 형식으로 출력하여라.

A_1 B_1

A_2 B_2

$\vdots$

A_{N-1} B_{N-1}

P_1 P_2 $\cdots$ P_N

A_i, B_i는 $X_i = \dfrac{A_i}{B_i}$ ($1 \leq i \leq N$)를 만족하는 정수 쌍이다. 이 정수는 출력 제한을 따라야 한다.

제한

입력 제한

- $1 \leq N \leq 2000$.
- $0 \leq L \leq 2000$.
- $1 \leq V_{i,j} \leq 100\ 000$ $(1 \leq i \leq N,\ 1 \leq j \leq L)$.

출력 제한

난을 공평한 방식으로 나눈 방법이 존재한다면, 출력은 다음 제한을 따라야 한다.

- $1 \leq B_i \leq 1\ 000\ 000\ 000$. $(1 \leq i \leq N)$
- $0 \leq \dfrac{A_1}{B_1} < \dfrac{A_2}{B_2} \cdots < \dfrac{A_{N-1}}{B_{N-1}} < L$.
- $P_1, \cdots, P_N$은 $1, \cdots, N$의 순열이다.
- 분배에서, i 번째 사람이 가지는 행복도의 양은 $\dfrac{V_{i,1} + V_{i,2} + \cdots + V_{i,L}}{N}$ 이상이어야 한다.

A_i와 B_i는 서로소일 필요는 없다. 공평한 분배가 존재하는 경우 $1 \leq B_i \leq 1\ 000\ 000\ 000$을 만족하는 출력이 존재함을 증명할 수 있다.

서브태스크 1 (5 점)

- $N = 2$

서브태스크 2 (24 점)

- $N \leq 6$
- $V_{i,j} \leq 10$ $(1 \leq i \leq N,\ 1 \leq j \leq L)$

서브태스크 3 (71 점)

추가 제한조건이 없다.

예제

standard input	standard output
2 5 2 7 1 8 2 3 1 4 1 5	14 5 2 1

이 예제에서, 모든 난을 먹었을 때 첫 번째 사람은 $2 + 7 + 1 + 8 + 2 = 20$의 행복도를 가지고 두 번째 사람은 $3 + 1 + 4 + 1 + 5 = 14$의 행복도를 가진다. 즉, 첫 번째 사람이 $\dfrac{20}{2} = 10$ 이상의 행복도를 가지고 둘째 사람이 $\dfrac{14}{2} = 7$ 이상의 행복도를 가지면 분배는 공평하다.

난을 $\dfrac{14}{5}$cm 에서 나누면, 첫 번째 사람은 $1 \times \dfrac{1}{5} + 8 + 2 = \dfrac{51}{5}$의 행복도를 얻고, 두 번째 사람은 $3 + 1 + 4 \times \dfrac{4}{5} = \dfrac{36}{5}$의 행복도를 얻는다. 그러므로 이것은 공평한 분배이다.

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

이 예제에서는 맛이 하나 뿐이다. 난을 크기가 같은 7개의 부분으로 자르면 $P_1, \cdots, P_N$과 관계 없이 분배가 공정하다.

standard input	standard output
5 3 2 3 1 1 1 1 2 2 1 1 2 2 1 2 1	15 28 35 28 50 28 70 28 3 1 5 2 4

A_i와 B_i가 서로소 일 필요는 없다. $(1 \leq i \leq N)$

문제 4. 두 안테나

입력 파일:	`standard input`
출력 파일:	`standard output`
시간 제한:	3초
메모리 제한:	512MB

1번부터 N번까지의 번호가 붙어있는 N 개의 안테나가 일렬로 놓여 있다. 각 안테나는 다른 연속된 안테나와 1km 떨어져 있다. i번 ($1 \leq i \leq N$) 안테나의 높이는 H_i이다. i번 안테나는 자신으로부터 A_i km 이상 B_i km 이하 떨어져 있는 안테나에만 정보를 보낼 수 있다. 만약 x번 안테나와 y번 안테나가 ($1 \leq x < y \leq N$) 서로 정보를 주고받을 수 있다면, 이 둘은 통신할 수 있고, 통신 비용은 $|H_x - H_y|$이다.

JOI 공화국의 K 수상은 시민들로부터 연결상태에 관한 불만 Q개를 들었다. 조사 결과 j 번째 ($1 \leq j \leq Q$) 불만은, $L_j,\ L_j + 1, \cdots, R_j$번 안테나 중 무언가가 이상이 있는 것으로 밝혀졌다. 당신은 이 안테나들 중 서로 통신할 수 있는 안테나 쌍이 있는지, 만약 있다면 그중 가장 통신 비용이 많이 드는 쌍의 통신 비용은 얼마인지 알아보는 일을 맡았다.

안테나의 정보와 불만의 정보가 주어졌을 때, $L_j,\ L_j + 1, \cdots, R_j$번 안테나 중 서로 통신할 수 있는 쌍이 있는지, 있다면 통신 비용의 최댓값은 얼마인지를 알려주는 프로그램을 작성하여라.

입력 형식

표준 입력에서 다음과 같은 형식으로 주어진다. 모든 값은 정수이다.

N

$H_1\ A_1\ B_1$

$\vdots$

$H_N\ A_N\ B_N$

Q

$L_1\ R_1$

$\vdots$

$L_Q\ R_Q$

출력 형식

표준 출력으로 Q 개의 줄을 출력하여라. j 번째 ($1 \leq j \leq Q$) 줄은 $L_j,\ L_j + 1, \cdots, R_j$번 안테나 중 서로 통신할 수 있는 쌍이 없으면 `-1`, 있다면 통신 비용의 최댓값이어야 한다.

제한

- $2 \leq N \leq 200\ 000$.
- $0 \leq H_i \leq 1\ 000\ 000\ 000$ ($1 \leq i \leq N$).
- $1 \leq A_i \leq B_i \leq N - 1$ ($1 \leq i \leq N$).
- $1 \leq Q \leq 200\ 000$.
- $1 \leq L_j < R_j \leq N$ ($1 \leq j \leq Q$).

서브태스크 1 (2 점)

- $N \leq 300$
- $Q \leq 300$

서브태스크 2 (11 점)

- $N \leq 2\ 000$

서브태스크 3 (22 점)

- $Q = 1$
- $L_1 = 1$
- $R_1 = N$

서브태스크 4 (65 점)

추가 제한조건이 없다.

예제

standard input	standard output
5 10 2 4 1 1 1 2 1 3 1 1 1 100 1 1 5 1 2 2 3 1 3 1 4 1 5	-1 1 8 8 99

1번 안테나와 2번 안테나는 서로 통신할 수 없으므로, 첫 번째 불만에 대한 답은 -1이다.

두 번째, 세 번째, 네 번째, 다섯 번째 불만에 대한 최대 통신 비용을 가진 안테나 쌍은 각각 (2, 3), (1, 3), (1, 3), (4, 5)이다.

standard input	standard output
20 260055884 2 15 737689751 5 5 575359903 1 15 341907415 14 14 162026576 9 19 55126745 10 19 95712405 11 14 416027186 8 13 370819848 11 14 629309664 4 13 822713895 5 15 390716905 13 17 577166133 8 19 195931195 10 17 377030463 14 17 968486685 11 19 963040581 4 10 566835557 1 12 586336111 6 16 385865831 8 9 1 1 20	806460109

이 입력은 서브태스크 3의 조건을 만족한다.

문제 5. 두 요리

입력 파일:	`standard input`
출력 파일:	`standard output`
시간 제한:	5초
메모리 제한:	1024MB

요리사 비타로는 요리 대회에 참가했다. 이 대회에서 참가자는 IOI 덮밥와 JOI 카레를 요리해야 한다.

IOI 덮밥를 요리하는 방법은 N 단계로 이루어져 있다. i 번째 $(1 \leq i \leq N)$ 단계는 정확히 A_i 분이 걸린다. 요리를 시작할 때, 그는 첫 번째 단계만 실행할 수 있다. i 번째 $(2 \leq i \leq N)$ 단계를 실행하려면, $(i-1)$ 번째 단계를 끝마쳐야 한다.

JOI 카레를 요리하는 방법은 M단계로 이루어져 있다. j 번째 $(1 \leq j \leq M)$ 단계는 정확히 B_j 분이 걸린다. 요리를 시작할 때, 그는 첫 번째 단계만 실행할 수 있다. j 번째 $(2 \leq j \leq M)$ 단계를 실행하려면, $(j-1)$ 번째 단계를 끝마쳐야 한다.

각 단계를 요리할 때는 집중해야 하므로, 요리의 한 단계를 시작하면 그 단계를 끝날 때까지 다른 단계를 실행할 수 없다. 한 단계가 끝난 이후에는 다른 요리의 단계를 시작해도 상관없다. 대회가 시작하면 두 요리가 끝나기까지의 쉬는 시간은 없다.

이 대회에서는 각 참가자는 **예술 점수**를 다음 기준에 따라 받는다.

- IOI 덮밥를 만드는 i 번째 $(1 \leq i \leq N)$ 단계를 처음부터 S_i 분 안에 끝냈을 경우 P_i 점을 얻는다. P_i는 음수일 수도 있다.
- JOI 카레를 만드는 j 번째 $(1 \leq j \leq M)$ 단계를 처음부터 T_j 분 안에 끝냈을 경우 Q_j 점을 얻는다. Q_j 는 음수일 수도 있다.

비타로는 예술 점수를 최대화하고 싶다.

요리 단계의 수, 각 단계에 걸리는 시간과 예술 점수의 정보가 주어졌을 때, 비타로가 얻을 수 있는 예술 점수의 최댓값을 구하여라.

입력 형식

표준 입력에서 다음과 같은 형식으로 주어진다. 모든 값은 정수이다.

$N\ M$

$A_1\ S_1\ P_1$

$\vdots$

$A_N\ S_N\ P_N$

$B_1\ T_1\ Q_1$

$\vdots$

$B_M\ T_M\ Q_M$

출력 형식

표준 출력으로 한 개의 줄을 출력하여라. 이는 비타로가 얻을 수 있는 예술 점수의 최댓값이다.

제한

- $1 \leq N \leq 1\ 000\ 000$.
- $1 \leq M \leq 1\ 000\ 000$.
- $1 \leq A_i \leq 1\ 000\ 000\ 000$ $(1 \leq i \leq N)$.

- $1 \leq B_j \leq 1\ 000\ 000\ 000$ $(1 \leq j \leq M)$.
- $1 \leq S_i \leq 2\ 000\ 000\ 000\ 000\ 000 = 2 \times 10^{15}$ $(1 \leq i \leq N)$.
- $1 \leq T_j \leq 2\ 000\ 000\ 000\ 000\ 000 = 2 \times 10^{15}$ $(1 \leq j \leq M)$.
- $-1\ 000\ 000\ 000 \leq P_i \leq 1\ 000\ 000\ 000$ $(1 \leq i \leq N)$.
- $-1\ 000\ 000\ 000 \leq Q_j \leq 1\ 000\ 000\ 000$ $(1 \leq j \leq M)$.

서브태스크 1 (5 점)

- $N \leq 200\ 000$
- $M \leq 200\ 000$
- $S_1 = \cdots = S_N$
- $T_1 = \cdots = T_N$

서브태스크 2 (3 점)

- $N \leq 12$
- $M \leq 12$
- $P_i = 1$ $(1 \leq i \leq N)$
- $Q_j = 1$ $(1 \leq j \leq M)$

서브태스크 3 (7 점)

- $N \leq 2\ 000$
- $M \leq 2\ 000$
- $P_i = 1$ $(1 \leq i \leq N)$
- $Q_j = 1$ $(1 \leq j \leq M)$

서브태스크 4 (39 점)

- $N \leq 200\ 000$
- $M \leq 200\ 000$
- $P_i = 1$ $(1 \leq i \leq N)$
- $Q_j = 1$ $(1 \leq j \leq M)$

서브태스크 5 (11 점)

- $N \leq 200\ 000$
- $M \leq 200\ 000$

- $1 \leq P_i$ $(1 \leq i \leq N)$
- $1 \leq Q_j$ $(1 \leq j \leq M)$

서브태스크 6 (9 점)

- $1 \leq P_i$ $(1 \leq i \leq N)$
- $1 \leq Q_j$ $(1 \leq j \leq M)$

서브태스크 7 (17 점)

- $N \leq 200\ 000$
- $M \leq 200\ 000$

서브태스크 8 (9 점)

추가 제한조건이 없다.

예제

standard input	standard output
4 3 2 1 1 3 8 1 2 13 1 1 13 1 3 6 1 2 11 1 2 15 1	6

이 입력은 서브태스크 2의 조건을 만족한다.

이 입력에서 비타로는 다음과 같은 방식으로 요리 할 수 있다.

1. JOI 카레의 첫 번째 단계를 요리한다. 그는 대회가 시작한 후 3분 후에 단계를 끝낸다. 6분 안이므로, 1점을 얻는다.
2. IOI 덮밥의 첫 번째 단계를 요리한다. 그는 대회가 시작한 후 5분 후에 단계를 끝낸다. 1분 밖이므로, 아무 점수도 얻지 못한다.
3. IOI 덮밥의 두 번째 단계를 요리한다. 그는 대회가 시작한 후 8분 후에 단계를 끝낸다. 8분 안이므로, 1점을 얻는다.
4. JOI 카레의 두 번째 단계를 요리한다. 그는 대회가 시작한 후 10분 후에 단계를 끝낸다. 11분 안이므로, 1점을 얻는다.
5. IOI 덮밥의 세 번째 단계를 요리한다. 그는 대회가 시작한 후 12분 후에 단계를 끝낸다. 13분 안이므로, 1점을 얻는다.
6. IOI 덮밥의 네 번째 단계를 요리한다. 그는 대회가 시작한 후 13분 후에 단계를 끝낸다. 13분 안이므로, 1점을 얻는다.
7. JOI 카레의 세 번째 단계를 요리한다. 그는 대회가 시작한 후 15분 후에 단계를 끝낸다. 15분 안이므로, 1점을 얻는다.

총 예술점수는 6점이다. 6점보다 더 많은 점수를 얻을 수는 없으므로 6을 출력해야 한다.

standard input	standard output
5 7 16 73 16 17 73 10 20 73 1 14 73 16 18 73 10 3 73 2 10 73 7 16 73 19 12 73 4 15 73 15 20 73 14 15 73 8	63

이 입력은 서브태스크 1의 조건을 만족한다.

standard input	standard output
9 11 86 565 58 41 469 -95 73 679 28 91 585 -78 17 513 -63 48 878 -66 66 901 59 72 983 -70 68 1432 11 42 386 -87 36 895 57 100 164 10 96 812 -6 23 961 -66 54 193 51 37 709 82 62 148 -36 28 853 22 15 44 53 77 660 -19	99

문제 6. 두 운송수단

시간 제한: 1.5초
메모리 제한: 256MB

JOI 나라에는 0번부터 $N-1$번까지 번호가 붙은 도시 N 개가 있다. JOI 나라에는 A 개의 전철 노선이 있고 0번부터 $A-1$번까지의 번호가 붙어 있다. i번 $(0 \leq i \leq A-1)$ 전철 노선은 U_i번 도시와 V_i번 도시를 양방향으로 잇는다. 이 노선의 운임은 C_i이다. 서로 다른 전철 노선은 서로 다른 쌍의 도시를 잇는다. 또한, B개의 버스 노선이 있고 0번부터 $B-1$번까지의 번호가 붙어 있다. j번 $(0 \leq j \leq B-1)$ 버스 노선은 S_j번 도시와 T_j번 도시를 양방향으로 잇는다. 이 노선의 운임은 D_j이다. 서로 다른 버스 노선은 서로 다른 쌍의 버스를 잇는다. 하지만, 전철 노선과 버스 노선은 같은 쌍의 도시를 이을 수도 있다. 전철 혹은 버스를 사용하면 어떠한 쌍의 도시도 오갈 수 있다.

Azer는 0번 도시로부터 각 도시까지 가기 위한 최소 운임을 알고 싶다. Azer는 전철 노선만 알고 있기 때문에, 그는 버스 노선만 아는 Baijan과 협력해야 한다.

그들은 서로 문자 0 혹은 1을 주고받으면서 통신한다. 보내는 총 문자의 개수는 58 000개 이하여야 한다.

전철 노선의 정보가 주어진 Azer과, 버스 노선의 정보가 주어진 Baijan 사이에서 서로 통신하고 Azer가 0번 도시부터 각 도시까지 가기 위한 최소 운임을 출력하여라.

구현 명세

당신은 파일 두 개를 제출해야 한다.

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

- `void InitA(int N, int A, std::vector<int> U, std::vector<int> V, std::vector<int> C)`

  이 함수는 프로그램 시작 시에 정확히 한 번 불린다.

  - 인자 `N`은 도시의 수 N을 나타낸다.
  - 인자 `A`는 전철 노선의 수 A를 나타낸다.
  - 인자 `U`, `V`는 길이 A의 배열이다. `U[i]`와 `V[i]`는 전철 노선이 연결하는 두 도시 U_i번과 V_i번을 나타낸다. $(0 \leq i \leq A-1)$
  - 인자 `C`는 길이 A의 배열이다. `C[i]`는 i번 전철노선의 운임 C_i를 나타낸다. $(0 \leq i \leq A-1)$

- `void ReceiveA(bool x)`

  이 함수는 Baijan으로 부터 문자를 받았을 때마다 호출된다.

  - 인자 `x`는 Baijan으로부터 받은 문자를 나타낸다. `true`이면 1, `false`이면 0을 받은 것이다.

- `std::vector<int> Answer()`

  이 함수는 보내진 모든 문자를 받았을 때 실행된다. 이 함수는 0번 도시로부터 각 도시간의 최단거리를 담은 배열 `Z`를 반환해야 한다.

  - 배열 `Z`는 길이가 N이어야 한다. 길이가 N이 아닌 경우에, 프로그램은 **오답 [1]**이 된다. `Z[k]` $(0 \leq k \leq N-1)$은 0번 도시 부터 k번 도시까지 가는데 필요한 운임의 최솟값이어야 한다. 특히, `Z[0]` $= 0$ 이어야 함에 유의하여라.

이 프로그램은 다음 함수를 호출 할 수 있다.

- `void sendA(bool y)`

  Baijan에게 문자를 보내려면 이 함수를 사용해야 한다.

  - 인자 `y`는 Baijan에게 보내는 문자를 나타낸다. `true`이면 1, `false`이면 0을 보낸 것이다.

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

- `void InitB(int N, int B, std::vector<int> S, std::vector<int> T, std::vector<int> D)`

  이 함수는 프로그램 시작 시에 정확히 한 번 불린다.

  - 인자 `N`은 도시의 수 N을 나타낸다.
  - 인자 `B`는 버스 노선의 수 B를 나타낸다.
  - 인자 `S`, `T`는 길이 B의 배열이다. `S[j]`와 `T[j]`는 전철 노선이 연결하는 두 도시 S_j번과 T_j번을 나타낸다. $(0 \leq j \leq B-1)$
  - 인자 `D`는 길이 B의 배열이다. `D[j]`는 i번 버스노선의 운임 D_j를 나타낸다. $(0 \leq j \leq B-1)$

- `void ReceiveB(bool y)`

  이 함수는 Azer로 부터 문자를 받았을 때마다 호출된다.

  - 인자 `y`는 Azer로부터 받은 문자를 나타낸다. `true`이면 1, `false`이면 0을 받은 것이다.

이 프로그램은 다음 함수를 호출 할 수 있다.

- `void sendB(bool x)`

  Azer에게 문자를 보내려면 이 함수를 사용해야 한다.

  - 인자 `x`는 Azer에게 보내는 문자를 나타낸다. `true`이면 1, `false`이면 0을 보낸 것이다.

당신은 프로그램이 다음과 같은 방법으로 실행된다는 것을 가정해도 좋다. 각 테스트 케이스마다 Azer가 보낸 문자들을 담는 큐 Q_Y와 Baijan이 보낸 문자들을 담는 큐 Q_X, 두 개의 큐가 준비된다. 처음에, `InitA`와 `InitB`가 실행되고, 보낸 문자들은 각각 큐에 push된다.

- Q_X나 Q_Y가 비어있지 않으면, 문자 하나가 비어있지 않은 큐로부터 pop되고, 해당하는 `ReceiveA`혹은 `ReceiveB`가 호출된다. 단, Q_X와 Q_Y가 모두 비어있지 않을 경우에, `ReceiveA`와 `ReceiveB`중 어느 쪽이 호출되는지는 결정되어있지 않다.
- `ReceiveA`의 호출 도중에 `SendA`가 호출된 경우, 보내진 문자는 Q_Y에 push된다.
- `ReceiveB`의 호출 도중에 `SendB`가 호출된 경우, 보내진 문자는 Q_X에 push된다.
- 두 큐가 모두 빈 경우에는, `Answer`이 호출되고 프로그램이 종료된다.

Azer와 Baijan이 보낸 문자의 총 개수는 58 000보다 작거나 같아야 한다. 만약 더 큰 경우에는, **오답 [2]**가 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. 채점될 때는, Azer과 Baijan에 해당하는 두 프로세스로 나누어서 실행될 것이다. Azer의 프로세스와 Baijan의 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 테스트 하기 위해서, `grader.cpp`, `Azer.cpp`, `Baijan.cpp`, `Azer.h`, `Baijan.h`를 같은 디렉토리 안에 놓고, 컴파일하기 위해 다음 커맨드를 실행하여라.

- `g++ -std=gnu++14 -O2 -o grader grader.cpp Azer.cpp Baijan.cpp`

컴파일이 성공적이면, 파일 `grader`가 생성된다.

실제 그레이더와 샘플 그레이더는 다름에 주의하여라. 샘플 그레이더는 하나의 프로세스에서 실행 되며, 입력을 표준 입력으로 부터 받고, 출력을 표준 출력에 출력한다.

입력 형식

샘플 그레이더는 표준 입력에서 다음과 같은 형식으로 입력받는다.

N A B

U_0 V_0 C_0

$\vdots$

U_{A-1} V_{A-1} C_{A-1}

S_0 T_0 D_0

$\vdots$

S_{A-1} T_{A-1} D_{A-1}

출력 형식

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

- 오답[1] 혹은 오답 [2]로 판단된 경우, 오답의 종류를 "`Wrong Answer [1]`"과 같은 형식으로 표준 에러에 출력한다. 표준 출력에는 아무것도 출력되지 않는다.
- 아닌 경우, 보낸 문자의 총 개수 L을 "`Accepted: L`"과 같은 형식으로 표준 에러에 출력한다. 또한, 표준 출력에 답 `Z`를 다음과 같은 형식으로 출력한다:

  ```
  Z[0]
  ⋮
  Z[N - 1]
  ```

  샘플 그레이더는 `Z`가 올바른지 검사하지 않는다.

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

제한

- $3 \leq N \leq 2\ 000$.
- $0 \leq A \leq 500\ 000$.
- $0 \leq B \leq 500\ 000$.
- $0 \leq U_i \leq N - 1$. $(0 \leq i \leq A - 1)$
- $0 \leq V_i \leq N - 1$. $(0 \leq i \leq A - 1)$

- $U_i \neq V_i$. $(0 \leq i \leq A-1)$
- $(U_{i_1},\ V_{i_1}) \neq (U_{i_2},\ V_{i_2})$이고, $(U_{i_1},\ V_{i_1}) \neq (V_{i_2},\ U_{i_2})$이다. $(0 \leq i_1 < i_2 \leq A-1)$
- $0 \leq S_j \leq N-1$. $(0 \leq j \leq B-1)$
- $0 \leq T_j \leq N-1$. $(0 \leq j \leq B-1)$
- $S_j \neq T_j$. $(0 \leq j \leq B-1)$
- $(S_{j_1},\ T_{j_1}) \neq (S_{j_2},\ T_{j_2})$이고, $(S_{j_1},\ T_{j_1}) \neq (T_{j_2},\ S_{j_2})$이다. $(0 \leq j_1 < j_2 \leq B-1)$
- 전철 혹은 버스를 사용하면 어떠한 쌍의 도시도 오갈 수 있다.
- $1 \leq C_i \leq 500$. $(0 \leq i \leq A-1)$
- $1 \leq D_j \leq 500$. $(0 \leq j \leq B-1)$

서브태스크 1 (6 점)

- $A = 0$

서브태스크 2 (8 점)

- $B \leq 1\ 000$

서브태스크 3 (8 점)

- $A + B = N - 1$

서브태스크 4 (38 점)

- $N \leq 900$

서브태스크 5 (14 점)

- $N \leq 1\ 100$

서브태스크 6 (10 점)

- $N \leq 1\ 400$

서브태스크 7 (16 점)

추가 제한조건이 없다.

예제

이 함수는 그레이더의 예제 입력과 해당하는 함수 호출을 보여준다.

예제 입력	예제 함수 호출		
	호출	호출	반환값
4 3 4 0 1 6 2 1 4 2 0 10 1 2 3 3 1 1 3 2 3 3 0 7	`InitA(4, 3, {0,2,2}, {1,1,0}, {6,4,10})`		
		`SendA(true)`	
		`SendA(false)`	
	`InitB(4, 4, {1,3,3,3}, {2,1,2,0}, {3,1,3,7}`		
	`ReceiveB(true)`		
		`SendB(true)`	
	`ReceiveA(true)`		
	`ReceiveB(false)`		
	`Answer()`		{0,6,9,7}

문제 7. 특별관광도시

입력 파일:	`standard input`
출력 파일:	`standard output`
시간 제한:	2초
메모리 제한:	512MB

JOI 나라에는 N 개의 도시가 있다. 이 도시들은 1번부터 N번까지 번호가 붙어있다. 이 도시에는 $N-1$ 개의 도로가 있고, 1번부터 $N-1$번까지의 번호가 붙어있다. i번 $(1 \leq i \leq N-1)$ 도로는 노선이 두 개가 있다. 한 노선은 A_i번 도시에서 B_i번 도시로 향하는 노선이고, 다른 노선은 B_i번 도시에서 A_i번 도시로 향하는 노선이다. 즉, 모든 도로는 양방향이다. 어떤 두 도시 간에도 몇 개의 도로를 사용해서 이동하는 것이 가능하다.

처음에 모든 노선은 정비되어있지 않다. 각 도로의 각 노선에 대해 우리는 노선을 정비하는 비용을 알고 있다. i번 $(1 \leq i \leq N-1)$ 도로의 A_i번 도시에서 B_i번 도시로 향하는 노선을 정비하는 비용은 C_i이고, B_i번 도시에서 A_i번 도시로 향하는 노선을 정비하는 비용은 D_i이다.

JOI 나라의 장관인 K 이사장은 몇몇 도시를 골라 그 도시를 **특별관광도시**로 만들 것이다. x번 $(1 \leq x \leq N)$을 특별관광도시로 만들 때, 각 도로 $i(1 \leq i \leq N-1)$에 대해 다음 일이 일어날 것이다.

> A_i번과 B_i번 도시 중에서 x번 도시에 가까운 도시는 a번 도시이고, 먼 도시는 b번 도시라고 하자. 여기서 가까운 도시라고 함은 x번 도시에 가기 위해 사용해야 하는 도로의 수가 더 적은 도시를 말한다. 이때, b번 도시에서 a번 도시로 향하는 노선이 정비되지 않은 상태라면 정비된다.

특별관광도시를 만들기 위해 노선을 정비하는 비용은 세금으로 충당되지만, 특별관광도시가 만들어진 이후에 남은 도로를 정비하는 비용은 K 이사장의 개인 자금에서 나간다.

K 이사장이 계획한 Q개의 계획이 있다. j 번째 $(1 \leq j \leq Q)$ 계획에서 그는 특별관광도시가 없고 모든 노선이 정비되지 않은 상태에서 시작해서 정확히 E_j개의 도시를 특별관광도시로 만들것이다. 하지만 어떤 도시들이 특별관광도시가 될지는 계획되지 않았다. 그는 개인 자금에서 나가는 도로 정비 비용을 최소로 하고 싶다.

JOI 나라의 도시 수, 도로의 정보와 계획의 정보가 주어졌을 때, 각 계획에 대해 K 이사장의 개인 자금에서 나가는 도로 정비 비용을 최소로 하는 프로그램을 작성하여라.

입력 형식

표준 입력에서 다음과 같은 형식으로 주어진다. 모든 값은 정수이다.

N

$A_1\ B_1\ C_1\ D_1$

$\vdots$

$A_{N-1}\ B_{N-1}\ C_{N-1}\ D_{N-1}$

Q

E_1

$\vdots$

E_Q

출력 형식

표준 출력으로 Q개의 줄을 출력하여라. j 번째 $(1 \leq j \leq Q)$ 줄은 j 번째 계획에서 이사장의 개인 자금에서 나가는 도로 정비 비용의 최솟값이어야 한다.

제한

- $2 \leq N \leq 200\ 000$.

- $1 \leq A_i \leq N$ $(1 \leq i \leq N)$.
- $1 \leq B_i \leq N$ $(1 \leq i \leq N)$.
- $A_i \neq B_i$ $(1 \leq i \leq N)$.
- 어떤 두 도시 간에도 몇 개의 도로를 사용해서 이동하는 것이 가능하다.
- $0 \leq C_i \leq 1\ 000\ 000\ 000$ $(1 \leq i \leq N)$.
- $0 \leq D_i \leq 1\ 000\ 000\ 000$ $(1 \leq i \leq N)$.
- $1 \leq A_i \leq B_i \leq N - 1$ $(1 \leq i \leq N)$.
- $1 \leq Q \leq N$.
- $1 \leq E_j \leq N$ $(1 \leq j \leq Q)$.

서브태스크 1 (16 점)

- $N \leq 16$

서브태스크 2 (7 점)

- $Q = 1$
- $E_1 = 1$

서브태스크 3 (9 점)

- $Q = 1$
- $E_1 = 2$

서브태스크 4 (17 점)

- $N \leq 2000$

서브태스크 5 (17 점)

- $Q = 1$

서브태스크 6 (44 점)

추가 제한조건이 없다.

예제

standard input	standard output
4 1 2 1 2 1 3 3 4 1 4 5 6 2 1 2	9 1

첫 번째 계획은 정확히 하나의 도시를 특별관광도시로 만드는 것이다. 만약 1번 도시를 특별관광도시로 지정한다면, 1번 도로의 2번 도시에서 1번 도시로 향하는 노선, 2번 도로의 3번 도시에서 1번 도시로 향하는 노선, 3번 도로의 1번 도시에서 3번 도시로 향하는 노선이 정비될 것이다. 이때 정비되지 않고 남는 노선은 1번 도로의 1번 도시에서 2번 도시로 향하는 노선, 2번 도로의 1번 도시에서 3번 도시로 향하는 노선, 3번 도로의 1번 도시에서 4번 도시로 향하는 노선이다. 이 노선들을 정비하는 비용은 1+3+5=9이다. 이보다 더 낮은 비용으로 특별관광도시를 지정하는 방법은 없다. 그래서 답은 9이다.

두 번째 계획은 정확히 두 개의 도시를 특별관광도시로 만드는 것이다. 만약 3번 도시와 4번 도시를 특별관광도시로 지정했다면, 1번 도로의 1번 도시에서 2번 도시로 향하는 노선만 정비되지 않았을 것이다. 이 노선을 정비하는 데에 드는 비용은 1이다. 이보다 더 낮은 비용으로 두 특별관광도시를 지정하는 방법은 없다. 그래서 답은 1이다.

standard input	standard output
5 1 3 13 6 5 1 17 8 5 2 6 10 1 4 16 11 1 1	36

이 입력은 서브태스크 2의 조건을 만족한다.

standard input	standard output
6 1 6 6 12 6 2 5 16 1 4 13 4 5 1 19 3 3 1 9 13 1 2	14

이 입력은 서브태스크 3의 조건을 만족한다.

문제 8. 램프

입력 파일:	`standard input`
출력 파일:	`standard output`
시간 제한:	1초
메모리 제한:	256MB

긴 복도에 N 개의 램프가 일렬로 나열되어 있다. 램프는 왼쪽부터 차례로 1번부터 N번까지의 번호가 붙어있다. 각 램프는 off또는 on중 하나의 상태이다.

램프의 상태를 바꾸는 특별한 장치가 있어서, 한 번의 작업으로 다음 셋 중 한 가지 동작을 할 수 있다.

- $1 \leq p \leq q \leq N$을 만족하는 정수 p와 q를 골라서 $p, p+1, \cdots, q$를 off 상태로 만든다.
- $1 \leq p \leq q \leq N$을 만족하는 정수 p와 q를 골라서 $p, p+1, \cdots, q$를 on 상태로 만든다.
- $1 \leq p \leq q \leq N$을 만족하는 정수 p와 q를 골라서 $p, p+1, \cdots, q$의 상태를 바꾼다. (off를 on으로, on을 off로)

처음에 램프의 상태는 길이 N의 문자열 A로 표현된다. A의 i 번째 $(1 \leq i \leq N)$ 문자가 `0`이면 i 번째 램프가 off 상태이고, `1`이면 on 상태이다. 우리는 만들고 싶은 상태가 길이 N의 문자열 B로 표현되어 있고 작업의 수를 최소한으로 하여 만들고 싶다. B의 i 번째 $(1 \leq i \leq N)$ 문자가 `0`이면 i 번째 램프가 off 상태이고, `1`이면 on 상태이다.

램프의 수, 현재 상태와 만들고 싶은 상태가 주어졌을 때, 만들고 싶은 상태로 바꾸는 데에 드는 연산의 수의 최솟값을 출력하여라.

입력 형식

표준 입력에서 다음과 같은 형식으로 주어진다.

N

A

B

출력 형식

표준 출력으로 한 개의 줄을 출력하여라. 이는 원하는 상태를 만들기 위한 연산의 수의 최솟값이다.

제한

- $1 \leq N \leq 1\ 000\ 000$.
- A와 B는 길이 N의 문자열이다.
- A와 B를 이루는 문자들은 `0` 혹은 `1`이다.

서브태스크 1 (6 점)

- $N \leq 18$

서브태스크 2 (41 점)

- $N \leq 2000$

서브태스크 3 (4 점)

- A를 이루는 각 문자는 0이다.

서브태스크 4 (49 점)

추가 제한조건이 없다.

예제

standard input	standard output
8 11011100 01101001	4

이 입력에서 우리는 원하는 상태를 다음과 같은 방법으로 네 번의 작업으로 만들 수 있다.

1. 1, 2, 3, 4번 램프의 상태를 바꾼다. 램프의 상태는 00101100이 된다.
2. 2번 램프를 on 상태로 만든다. 램프의 상태는 01101100이 된다.
3. 6, 7, 8번 램프의 상태를 바꾼다. 램프의 상태는 01101011이 된다.
4. 6, 7번 램프를 off 상태로 만든다. 램프의 상태는 01101001이 된다.

네 번보다 더 적은 작업으로 원하는 상태를 만들 수 있는 방법은 없으므로, 4를 출력한다.

standard input	standard output
13 1010010010100 0000111001011	3
18 001100010010000110 110110001000100101	5

문제 9. 시간을 달리는 비타로

입력 파일:	`standard input`
출력 파일:	`standard output`
시간 제한:	3초
메모리 제한:	512MB

비버랜드에는 N 개의 도시가 있다. 이 도시들은 1번부터 N번까지 번호가 붙어있다. 도시와 도시를 잇는 도로는 $N-1$ 개가 있고, i 번째 $(1 \leq i \leq N-1)$ 도로는 i번 도시와 $i+1$번 도시를 양방향으로 잇는다. 또한, 비버랜드의 하루는 1 000 000 000개의 단위시간으로 분열되어 있고, 이 단위시간을 *쵸*라고 부른다. 하루가 시작하고 나서 x 쵸가 지난 시간을 시각 x라 부른다. 한 도로를 통과하는 데에는 1쵸가 걸리고, i 번째 도로는 시각 L_i와 시각 R_i 사이에만 통과할 수 있다. 구체적으로, i 번째 도로를 통과하기 위해서 우리는 i번 도시나 $i+1$번 도시를 $L_i \leq x \leq R_i - 1$ 을 만족하는 시각 x에 떠나야 하고, 다른 도시에 시각 $x+1$에 도착해야 한다.

비타로는 비버랜드에 사는 평범한 비버다. 아니, 비버였다고 하는 것이 옳은 것일까. 지각을 자주 한 비타로는 이를 개선하려고 한 결과로 시간을 거슬러 올라가는 것이 가능해졌다. 이 능력을 한 번 사용하면 1쵸 뒤로 갈 수 있다. 하지만 어제로 갈 수는 없다. 만약 그가 능력을 시각 0과 시각 1 사이에 사용했다면, 그는 시각 0으로 돌아갈 것이다. 그는 이 기술을 도시에 있을 때 사용할 수 있다. 비타로의 위치는 능력을 사용해도 변하지 않는다.

비타로는 기술을 사용하면 피곤해진다. 최소한의 기술을 사용하여 이동하는 방법을 찾기 위한 비타로는 Q 개의 사고실험을 진행했다. 사고 실험의 j 번째 단계에서 그는 다음 중 한 행동을 한다:

- P_j 번째 도로를 여행할 수 있는 시각을 바꾼다. 바뀐 이후에 시각 S_j와 시각 E_j 사이에만 P_j 번째 도로를 통과할 수 있다.
- 그가 A_j번 도시, 시각 B_j에 있다고 할 때, C_j번 도시, 시각 D_j로 이동하기 위해 사용해야 하는 능력의 수의 최솟값을 구한다.

그는 사고실험의 결과를 궁금해한다.

비버랜드의 도시의 수, 도로의 정보, 사고실험의 방법이 주어졌을 때, 사고 실험의 결과를 계산하는 프로그램을 작성하여라.

입력 형식

표준 입력에서 다음과 같은 형식으로 주어진다. 모든 값은 정수이다.

N Q

L_1 R_1

$\vdots$

L_{N-1} R_{N-1}

(Query 1)

$\vdots$

(Query Q)

여기서, (Query j)는 공백으로 구분된 4개나 5개의 정수로 이루어져 있다. T_j가 첫 번째 정수라고 하자. 그러면,

- $T_j = 1$인 경우, (Query j)는 4개의 정수 T_j, P_j, S_j, E_j로 이루어져 있다. 이것은, 사고 실험의 j 번째 단계에서, P_j 번째 도로를 지날 수 있는 시간이 시각 S_j와 시각 E_j 사이로 바뀐다는 것을 의미한다.
- $T_j = 2$인 경우, (Query j)는 5개의 정수 T_j, A_j, B_j, C_j, D_j로 이루어져 있다. 이는, j 번째 사고 실험에서, 당신의 프로그램이 비타로가 A_j번 도시, 시각 B_j에 있다고 할 때, C_j번 도시, 시각 D_j로 이동하기 위해 사용해야 하는 능력의 수의 최솟값을 구해야 한다는 것을 의미한다.

출력 형식

$T_j = 2$인 각 단계에 대해서, 사용해야 하는 능력의 수의 최솟값을 한 줄에 하나씩 차례로 출력하여라.

제한

- $1 \leq N \leq 300\ 000$.
- $1 \leq Q \leq 300\ 000$.
- $0 \leq L_i < R_i \leq 999\ 999\ 999$ $(1 \leq i \leq N-1)$.
- $1 \leq T_j \leq 2$ $(1 \leq j \leq Q)$.
- $1 \leq P_j \leq N-1$ $(1 \leq j \leq Q, T_j = 1)$.
- $1 \leq S_j \leq E_j \leq 999\ 999\ 999$ $(1 \leq j \leq Q, T_j = 1)$.
- $1 \leq A_j \leq N$ $(1 \leq j \leq Q, T_j = 2)$.
- $1 \leq B_j \leq 999\ 999\ 999$ $(1 \leq j \leq Q, T_j = 2)$.
- $1 \leq C_j \leq N$ $(1 \leq j \leq Q, T_j = 2)$.
- $1 \leq D_j \leq 999\ 999\ 999$ $(1 \leq j \leq Q, T_j = 2)$.

서브태스크 1 (4 점)

- $N \leq 1000$
- $Q \leq 1000$

서브태스크 2 (30 점)

- $T_j = 2$ $(1 \leq j \leq Q)$.

서브태스크 3 (66 점)

추가 제한조건이 없다.

예제

standard input	standard output
3 3 0 5 0 5 2 1 3 3 3 1 2 0 1 2 1 3 3 3	2 4

사고 실험의 첫 번째 단계에서, 비타로는 1번 도시에서 2번 도시로 1초만에 이동하고, 2번 도시에서 3번 도시로 1초만에 이동하여 3번 도시, 시각 5에 위치해 있다. 능력을 두 번 사용하면, 그는 3번 도시, 시각 3에 위치할 수 있다.

사고 실험의 두 번째 단계에서, 2번 도시를 통과할 수 있는 시간이 시각 0부터 시각 1까지로 바뀐다.

사고 실험의 세 번째 단계에서, 1번 도시에서 2번 도시로 1초만에 이동하여, 2번 도시, 시각 4에 위치해 있다. 여기서 능력을 네 번 사용하면, 3번 도시로 1초만에 이동하여 2초를 기다리면 3번 도시, 시각 3에 위치할 수 있다.

standard input	standard output
5 5 3 5 4 8 2 6 5 10 2 5 3 1 10 2 2 6 5 6 1 3 4 6 2 3 3 4 3 2 4 5 1 5	4 3 2 3
7 7 112103440 659752416 86280800 902409187 104535475 965602300 198700180 945132880 137957976 501365807 257419446 565237610 2 4 646977260 7 915994878 2 1 221570340 6 606208433 2 7 948545948 4 604273995 2 7 247791098 5 944822313 2 7 250362511 2 50167280 2 3 364109400 4 555412865 2 7 33882587 7 186961394	145611455 0 447180143 0 207252171 0 0
7 7 535825574 705426142 964175291 996597835 481817391 649559926 4519006 410772613 74521477 274584126 256535565 899389890 1 6 511428966 602601933 1 1 69986642 201421232 2 3 636443425 4 625975977 1 6 235225515 405336399 2 3 866680458 3 701821857 1 6 180606048 900533151 1 6 612564160 720179605	10467449 164858601

문제 10. 케이크 3

입력 파일:	`standard input`
출력 파일:	`standard output`
시간 제한:	4초
메모리 제한:	256MB

오늘은 IOI양의 생일이다. 이날을 위해 JOI군은 생일 케이크를 예약했다. 원형 케이크 하나를 통째로 예약할 생각이었지만 착오가 있어서 N 조각의 케이크를 예약해 버렸다. 각 조각에는 1번부터 N번까지 번호가 붙어 있고, i 번 $(1 \leq i \leq N)$ 조각의 가치는 V_i이며 색의 짙음은 C_i이다.

JOI군은 서로 다른 M개의 케이크를 골라 원하는 순서대로 배열해 합쳐서 원형 케이크를 만들기로 결심했다. 케이크 조각들이 k_1번, $\cdots$, k_M번 조각의 순서로 나열되어 있을 때, 이 케이크의 아름다움은

$$\sum_{j=1}^{M} V_{k_j} - \sum_{j=1}^{M} |C_{k_j} - C_{k_{j+1}}|$$

으로 정의된다. (단, $k_{M+1} = k_1$이다.) 즉, 아름다움은 사용된 케이크 조각의 가치의 합으로 부터 인접한 두 케이크의 색의 짙음에 차의 절댓값의 합계로 정의된다. JOI군은 되도록 원형 케이크의 아름다움을 최대로 하고 싶다.

케이크 조각의 개수, 각 케이크 조각의 가치와 색의 짙음, 원형 케이크를 만들기 위해 필요한 조각의 개수가 주어졌을 때, JOI군이 만들 수 있는 원형 케이크의 아름다움의 최댓값을 구하는 프로그램을 작성하여라.

입력 형식

표준 입력에서 다음과 같은 형식으로 주어진다. 모든 값은 정수이다.

N M

V_1 C_1

$\vdots$

V_N C_N

출력 형식

표준 출력으로 한 개의 줄에 하나의 수를 출력하여라. 이는 JOI군이 만들 수 있는 원형 케이크의 아름다움의 최댓값이다.

제한

- $3 \leq N \leq 200\ 000$.
- $3 \leq M \leq N$.
- $1 \leq V_i \leq 1\ 000\ 000\ 000$ $(1 \leq i \leq N)$.
- $1 \leq C_i \leq 1\ 000\ 000\ 000$ $(1 \leq i \leq N)$.

서브태스크 1 (5 점)

- $N \leq 100$.

서브태스크 2 (19 점)

- $N \leq 2\ 000$.

서브태스크 3 (76 점)

추가 제한조건이 없다.

예제

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

JOI군이 1번, 3번, 2번 조각을 골라서 순서대로 붙이면, 조각의 가치의 합은 $2 + 6 + 4 = 12$이고, 케이크의 짙음의 차이 합은 $|1 - 4| + |4 - 2| + |2 - 1| = 6$이다. 그래서 원형 케이크의 아름다움은 $12 - 6 = 6$이다.

또한, 2번, 3번, 4번 조각을 골라서 순서대로 붙여서 아름다움이 6인 원형 케이크를 만들 수도 있다.

더 아름다움이 큰 원형 케이크를 만들 수는 없기 때문에, 6을 출력해야 한다.

standard input	standard output
8 4 112103441 501365808 659752417 137957977 86280801 257419447 902409188 565237611 965602301 689654312 104535476 646977261 945132881 114821749 198700181 915994879	2323231661

문제 11. 합병

입력 파일:	`standard input`
출력 파일:	`standard output`
시간 제한:	3초
메모리 제한:	256MB

JOI 합중국에는 N개의 도시가 있어서, 1번부터 N번까지의 번호가 붙어있다. 또한, JOI 합중국에는 $N-1$개의 국도가 있다. i 번째 $(1 \leq i \leq N-1)$ 국도는 A_i번 도시와 B_i번 도시를 양방향으로 잇는다. 어떤 두 도시에 대해서도, 국도 몇 개를 이용하면 서로 오가는 것이 가능하다.

현재 JOI 합중국은 1번부터 K번까지의 번호가 붙어있는 K개의 주로 나뉘어 있다. j번 $(1 \leq j \leq N)$도시는 S_j번 주에 속한다. 모든 주에는 적어도 하나의 도시가 속해 있다.

JOI 합중국의 대통령인 K 이사장은 이 나라가 분열하지 않을까 걱정이 되었다. 다음 조건을 모두 만족하도록 모든 도시를 2개의 그룹 X, Y로 나누는 것이 가능할 때, JOI 합중국은 **분열 가능**한 상태라고 말한다.

- 모든 도시는 그룹 X 혹은 그룹 Y에 속한다.
- 그룹 X에는 적어도 하나의 도시가 속해 있다.
- 그룹 Y에는 적어도 하나의 도시가 속해 있다.
- 모든 주에 대해서, 그 주에 있는 모든 도시는 모두 같은 그룹에 속해 있다.
- 그룹 X에 속한 어떤 두 도시에 대해서도, 그룹 X에 속한 도시만을 경유해서 서로 오가는 것이 가능하다.
- 그룹 Y에 속한 어떤 두 도시에 대해서도, 그룹 X에 속한 도시만을 경유해서 서로 오가는 것이 가능하다.

K 이사장은 JOI 합중국이 분열 가능하지 않은 상태를 만들기 위해 주를 합병하려고 한다. 한 번의 합병은 두 개의 주를 골라서 합치는 것을 말한다. 새로운 주는 기존의 두 주에 속해 있던 도시가 속해있다. K 이사장은 주를 최소한의 횟수만큼 합병하여 JOI 합중국을 분열 가능하지 않은 상태로 만들고 싶어 한다.

도시와 국도의 위치, 현재 어떤 도시가 어떤 주에 속해있는가에 대한 상태가 주어졌을 때, JOI 합중국이 분열 가능하지 않은 상태로 만들기 위한 합병의 최소 횟수를 구하는 프로그램을 작성하여라.

입력 형식

표준 입력에서 다음과 같은 형식으로 주어진다. 모든 값은 정수이다.

N K

A_1 B_1

$\vdots$

A_{N-1} B_{N-1}

S_1

$\vdots$

S_N

출력 형식

JOI 합중국을 분열 가능하지 않은 상태로 만들기 위한 합병의 최소횟수를 표준 출력의 첫째 줄에 출력하여라.

제한

- $1 \leq N \leq 500\ 000$.
- $1 \leq K \leq N$.

- $1 \leq A_i \leq N$ $(1 \leq i \leq N-1)$.
- $1 \leq B_i \leq N$ $(1 \leq i \leq N-1)$.
- 어떤 두 도시에 대해서도, 국도 몇 개를 이용하면 서로 오가는 것이 가능하다.
- $1 \leq S_j \leq K$ $(1 \leq j \leq N)$
- 모든 k $(1 \leq k \leq N)$에 대해, $S_j = k$를 만족하는 j $(1 \leq j \leq N)$가 존재한다.

서브태스크 1 (10 점)

- $N \leq 100$.
- $K \leq 7$.

서브태스크 2 (24 점)

- $N \leq 3\ 000$.

서브태스크 3 (14 점)

- $N \leq 100\ 000$.
- $K \leq 50$.

서브태스크 4 (22 점)

- $N \leq 100\ 000$.
- 처음 상황에서, 모든 주에 대해, 그 주에 속한 어떤 두 도시에 대해서도, 국도 100개 이하를 이용하면 서로 오가는 것이 가능하다.

서브태스크 5 (30 점)

추가 제한조건이 없다.

예제

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

이 예제에서는, 처음의 상태는 분열 가능하다. 예를 들면, 1번, 2번, 3번, 4번 도시를 그룹 X라 하고, 5번 도시를 그룹 Y라고 하면 된다.

3번 도시와 4번 도시를 합병할 경우 분열 가능하지 않은 상태가 된다. 그러므로 답은 1이다.

standard input	standard output
5 4 1 2 2 3 3 4 4 5 1 2 3 4 1	0

이 예제에서는, 처음의 상태는 분열 가능하지 않다. 그러므로 답은 0이다.

standard input	standard output
2 2 1 2 1 2	1

문제 12. 광물

시간 제한: 1 second
메모리 제한: 256 megabytes

JOI교수의 연구실에서는 N 종류의 광물을 연구하고 있다. 연구실에는 각 종류의 광물이 두 조각씩 있다. 조각은 총 $2N$ 개가 존재하고, 각 조각에는 1번부터 $2N$번까지의 번호가 붙어있다.

하지만 어느 날, 조수 비타로는 $2N$ 개의 광물을 포함한 박스를 떨어뜨려 어떤 조각과 어떤 조각이 같은지 모르게 되어 버렸다.

연구실은 0 개 이상 $2N$ 개 이하의 조각을 넣으면, 광물이 흡수하는 빛의 파장을 측정하는 것으로, 기계 안에 몇가지 종류의 광물이 들어있는가를 판단하는 장치가 있다. 비타로는 이 장치를 사용해서 $2N$ 개의 조각에서 같은 종류의 광물 짝 N 쌍을 알고싶어 한다. 비타로는 처음에 기계에 광물을 넣지 않은 상태로부터 시작해서 다음 동작 중 하나를 반복한다.

- 기계에 새로운 조각을 하나 집어넣어서, 기계 안에 몇 가지 종류의 광물이 있는지를 알아낸다.
- 기계에서 새로운 조각을 하나 빼서, 기계 안에 몇 가지 종류의 광물이 있는지를 알아낸다.

시간이 너무 오래 걸리면 JOI교수에게 들켜버리기 때문에, 비타로는 기계를 최대 1 000 000번만 사용할 수 있다.

광물의 종류가 주어졌을 때, 기계를 사용하여 모든 쌍의 광물을 알아내는 프로그램을 작성하여라.

구현 명세

당신은 파일 하나를 제출해야 한다.

이 파일의 이름은 `minerals.cpp`이다. 파일은 다음 함수를 구현해야 한다. 또한, `minerals.h`를 include해야 한다.

- `void Solve(int N)`

  이 함수는 각 테스트 케이스마다 정확히 한 번 불린다.

  - 인자 `N`은 광물의 수 N을 의미한다.

  당신의 프로그램은 다음 함수를 호출 할 수 있다.

  - `int Query(int x)`

    이 함수는 지정된 조각의 번호에 대해서 이 조각이 이미 기계 안에 들어 있으면 기계에서 빼고, 그렇지 않으면 이 조각을 기계에 넣는다.

    * 당신은 조각의 번호 x를 인자 `x`를 사용해서 나타내어야 한다. 이 번호는 $1 \leq x \leq 2N$을 만족해야 한다. 아닌 경우에는 **오답** [**1**]이 된다.
    * 당신은 이 함수를 1 000 000번 이상 호출해서는 안된다. 호출 한 경우에는 **오답** [**2**]이 된다.

  - `void Answer(int a, int b)`

    이 함수를 사용하여, 같은 종류의 광물의 쌍을 답할 수 있다.

    * 인자 `a`와 `b`는 a 번째 광물과 b 번째 광물이 같은 종류라는 것을 의미한다. 이 수는 $1 \leq a \leq 2N$과 $1 \leq b \leq 2N$을 만족해야 한다. 이것을 만족하지 않을 경우 **오답** [**3**]이 된다. 만약 a와 b로 주어진 수가 2번 이상 나타날 경우 **오답** [**4**]이 된다. 서로 다른 종류의 광물을 지정할 경우 **오답** [**5**]이 된다.

    함수 Answer는 정확히 N 번 호출될 필요가 있다. 함수 `solve`의 실행 종료 시에 함수 `Answer`의 호출 횟수가 N 번이 아닐 경우 **오답** [**6**]이 된다.

참고 사항

- 당신의 프로그램은 내부에서 사용할 목적으로 함수나 전역변수를 사용할 수 있다.
- 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`, `minerals.cpp`, `minerals.h`를 같은 디렉토리 안에 놓고, 컴파일하기 위해 다음 커맨드를 실행하여라.

- `g++ -std=gnu++14 -O2 -o grader grader.cpp minerals.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

X_1 Y_1

$\ldots$

X_N Y_N

X_i와 Y_i $(0 \leq i \leq N-2)$는 X_i번 조각과 Y_i번 조각이 같은 종류의 광물임을 의미한다.

출력 형식

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

- 정답으로 판단된 경우, `Query` 함수의 호출 횟수를 "`Accepted: 100`"과 같은 형식으로 출력한다.
- 오답으로 판단된 경우, 오답의 종류를 "`Wrong Answer [1]`"과 같은 형식으로 출력한다.

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

제한

모든 입력 데이터는 다음의 조건을 만족한다. 샘플 그레이더의 X_i와 Y_i의 정의에 따라서

- $1 \leq N \leq 43\,000$.
- $1 \leq X_i \leq 2N$ $(1 \leq i \leq N)$.
- $1 \leq Y_i \leq 2N$ $(1 \leq i \leq N)$.
- $X_i \neq X_j$ $(1 \leq i < j \leq N)$.
- $Y_i \neq Y_j$ $(1 \leq i < j \leq N)$.
- $X_i \neq Y_j$ $(1 \leq i \leq N, 1 \leq j \leq N)$.

서브태스크 1 (6 점)

- $N \leq 100$.

서브태스크 2 (25 점)

- $N \leq 15\ 000$.
- $1 \leq X_i \leq N$ $(1 \leq i \leq N)$.
- $N + 1 \leq Y_i \leq 2N$ $(1 \leq i \leq N)$.

서브태스크 3 (9 점)

- $N \leq 15\ 000$

서브태스크 4 (30 점)

- $N \leq 38\ 000$

서브태스크 5 (5 점)

- $N \leq 39\ 000$

서브태스크 6 (5 점)

- $N \leq 40\ 000$

서브태스크 7 (5 점)

- $N \leq 41\ 000$

서브태스크 8 (5 점)

- $N \leq 42\ 000$

서브태스크 9 (10 점)

추가 제한조건이 없다.

예제

이 함수는 그레이더의 예제 입력과 해당하는 함수 호출을 보여준다.

예제 입력	예제 함수 호출		
	호출	호출	반환값
4 1 2 2 6 3 4 7 8	`Solve(4)`		
		`Query(1)`	1
		`Query(2)`	2
		`Query(5)`	2
		`Query(2)`	1
		`Answer(3, 4)`	(없음)
		`Answer(5, 1)`	(없음)
		`Answer(8, 7)`	(없음)
		`Answer(2, 6)`	(없음)