Problem A. Matryoshka

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

당신은 마트료시카 인형을 판매하는 가게를 열기로 했다. 당신은 N개의 마트료시카 인형을 공장에 주문했다. 여기에는 1부터 N까지의 번호가 붙어 있었다. 그 중 i번째($1 \leq i \leq N$) 마트료시카 인형은 바닥의 지름이 R_i cm, 높이가 H_i cm인 속이 빈 원기둥으로 생각 할 수 있다.

마트료시카 인형은 중첩해서 안에 넣을 수 있다. 각각의 마트료시카 인형은, 자신보다 지름과 높이가 모두 작은 마트료시카 인형을 1개까지만 안에 넣을 수 있다. 수납된 마트료시카 인형 안에 다른 인형을 넣을 수도 있다.

어느 날, 마트료시카 인형을 주문한 공장에서 연락이 왔다. 주문한 N개의 마트료시카 인형을 모두 한번에 준비 할 수 없어서, N개의 마트료시카 인형 중 **바닥의 직경이 A cm 이상, 높이 B cm 이하**의 마트료시카만 모두 사전에 보내준다고 한다.

A와 B는 갑자기 변할지도 모른다. 그래서 당신은, Q개의 쌍 (A_j, B_j) $(1 \leq j \leq Q)$ 각각에 대해, 사전에 도착하는 마트료시카 인형을 중첩해서 넣었을 때, 어떤 마트료시카 인형에도 들어가지 못하는 인형의 갯수의 최솟값을 구하기로 하였다.

각각의 마트료시카 인형의 바닥의 지름과 높이에 대한 정보와, Q개의 쌍 (A_j, B_j) $(1 \leq j \leq Q)$이 주어진다. 각각의 쌍에 대해서, 사전에 도착하는 마트료시카 인형을 중첩해서 넣을 때, 어떤 마트료시카 인형에도 들어가지 못하는 마트료시카 인형의 갯수의 최솟값을 구하는 프로그램을 작성하여라.

Input

첫째 줄에는, 정수 N, Q가 공백으로 구분되어 들어온다. 이것은, 주문한 마트료시카 인형이 N개이고, (A, B)쌍의 수가 Q개라는 것을 의미한다.

다음 N개의 줄의 i번째 줄($1 \leq i \leq N$)에는, 정수 R_i, H_i가 공백으로 구분되어 들어온다. 이것은, i번째 마트료시카 인형은, 바닥의 지름이 $R_i$cm 이고, 높이가 $H_i$cm 임을 의미한다.

다음 Q개의 줄의 j번째 줄($1 \leq j \leq Q$)에는, 정수 A_j, B_j가 공백으로 구분되어 들어온다.

Output

출력은 Q개의 줄로 한다. j번째 줄($1 \leq j \leq Q$)에는, 쌍 (A_j, B_j) 에 대해, 사전에 도착한 마트료시카 인형을 중첩해서 넣을 때, 어떤 마트료시카 인형에도 들어가지 못하는 마트료시카 인형의 갯수의 최솟값을 출력하여라.

Constraints

모든 입력데이터는 다음의 조건을 만족한다.

- $1 \leq N \leq 200\,000$
- $1 \leq Q \leq 200\,000$
- $1 \leq R_i \leq 1\,000\,000\,000$ $(1 \leq i \leq N)$
- $1 \leq H_i \leq 1\,000\,000\,000$ $(1 \leq i \leq N)$
- $1 \leq A_j \leq 1\,000\,000\,000$ $(1 \leq j \leq Q)$
- $1 \leq B_j \leq 1\,000\,000\,000$ $(1 \leq j \leq Q)$

Subtask 1 (11 points)

다음의 조건을 만족한다.

- $N \leq 10$
- $Q = 1$

Subtask 2 (15 points)

다음의 조건을 만족한다.

- $N \leq 100$
- $Q = 1$

Subtask 3 (25 points)

다음의 조건을 만족한다.

- $N \leq 2\ 000$
- $Q \leq 2\ 000$

Subtask 4 (49 points)

추가 제한조건이 없다.

Examples

standard input	standard output
7 3 9 5 3 7 10 6 5 10 2 6 10 10 4 1 10 5 3 5 3 9	0 1 2

- $(A, B) = (10, 5)$ 일 때, 바닥의 지름이 10cm 이상이고, 높이가 5cm 이하인 마트료시카 인형은 한 개도 없기 때문에 0을 출력한다.
- $(A, B) = (3, 5)$ 일 때, 바닥의 직경이 3cm 이상이고, 높이가 5cm 이하인 마트료시카 인형, 즉 1번째와 7번째 마트료시카 인형이 사전에 도착한다. 7번째 마트료시카 인형은 1번째 마트료시카 인형 안에 넣을 수 있다. 어떤 마트료시카 인형에도 들어가 있지 않은 마트료시카 인형의 최솟값은 1이다.
- $(A, B) = (3, 9)$ 일 때, 바닥의 직경이 3cm 이상이고, 높이가 9cm 이하인 마트료시카 인형, 즉 1번째, 2번째, 3번째와 7번째 마트료시카 인형이 사전에 도착한다. 이 때, 7번째 마트료시카 인형은 1번째 마트료시카 인형의 안에 들어갈 수 있고, 1번째 마트료시카 인형은 3번째 마트료시카 인형 안에 들어갈 수 있다. 어떤 마트료시카 인형에도 들어가 있지 않은 마트료시카 인형의 최솟값은 2이다.

standard input	standard output
10 8 14 19 9 16 11 2 7 18 20 16 9 5 10 9 20 6 4 17 13 8 7 14 9 3 9 13 4 19 12 4 19 16 18 10 7 14	3 1 3 5 0 2 1 3

This page is intentionally left blank

Problem B. Memory2

Time limit:	2 seconds
Memory limit:	256 megabytes

0이상 $N-1$이하의 정수가 앞면에 1개씩 쓰인 카드가 2장씩 있다. 당신과 JOI군은, 이 $2N$장의 카드를 사용해서 신경쇠약이라는 게임의 연습을 하고 있다. 게임의 연습을 시작할 때에는, 카드가 모두 뒷면인 채로 책상 위에 가로 일렬로 놓여 있다. 왼쪽부터 $i+1$번째 $(0 \leq i \leq 2N-1)$ 카드를, **카드** i라고 부른다. 카드 i에 쓰인 정수를 A_i $(0 \leq i \leq 2N-1)$라고 한다. 처음에, JOI군과 당신은, $A_i (0 \leq i \leq 2N-1)$를 알지 못한다. 당신과 JOI군은, 다음의 주고받기를 K번 까지 실행할 수 있다.

1. 당신은, $2N$장의 카드 중 2장의 카드를 정한다.
2. JOI군은, 정해진 2장의 카드를 뒤집어, 뒤에 무슨 정수가 쓰여 있는지 당신에게 보이지 않게 몰래 본다. 만약, 2장의 카드의 뒤에 쓰인 정수가 같은 경우에는, 그 수를 당신에게 알려준다. 그렇지 않은 경우에는, 표에 쓰인 정수 중 JOI군이 기억하기 쉬운 정수를 기억해서 당신에게 알려준다.

JOI군이 정수를 기억하기 쉬운 정도는, N개의 정수 $P_0, P_1, \cdots, P_{N-1}$로 표시된다. 이 정수는, 다음 두가지 조건을 만족한다

- $0 \leq P_i \leq N-1$ $(0 \leq i \leq N-1)$
- $P_i \neq P_j$ $(0 \leq i < j \leq N-1)$

JOI군에게, i가 j보다 기억하기 쉽다는 것은, $P_i < P_j$가 성립하는 것과 동치이다.

당신의 문제는, JOI군과 K번 이하의 주고받기를 실행하여, 각각의 카드에 쓰인 정수를 특정하는 것이다. 단, 당신은, JOI군이 정수를 기억하기 쉬운 정도를 나타내는 정수 $P_0, P_1, \cdots, P_{N-1}$가 어떤 수인지 알지 못한다.

JOI군과 주고받기를 실행하여, 각각의 카드에 쓰인 수를 특정하는 프로그램을 작성하여라.

당신은, 각각의 카드에 쓰인 정수를 특정하는 방법이 구현된, 1개의 프로그램을 작성해야 한다. 프로그램은 `Memory2_lib.h`를 `include`해야한다.

Interaction Protocol

프로그램에는, 다음의 함수가 구현되어야 한다.

- `void Solve(int T, int N)` 이 함수는, 각 테스트케이스에 대해 1번만 실행된다. 인자 `T`는 Subtask의 번호이고, `N`은 카드가 $2N$개 있는 것을 의미한다.

  이 함수는, `Flip`를 호출하는 것으로, 카드에 쓰인 정수를 특정하여, 그 내용을 `Answer`를 호출하는 것으로 답해야한다.

  프로그램중에 이하의 함수를 호출하는 것이 가능하다.

  - `int Flip(int I, int J)` 이 함수는, JOI군에게 카드를 정할 때 호출한다. 인자 `I`, `J`는, JOI군이 뒤집을 카드의 번호인 I, J이다.

    I와 J는 0이상 $2N-1$이하인 서로 다른 수여야 한다. 이것을 만족하지 않게 `Flip`을 호출하면 **오답** [**1**]이 된다.

    이 함수는, 정수 A_I와 정수 A_J가 같은 경우 그 수를, 아닐 경우엔 정수 A_i와 정수 A_J중 JOI군이 기억하기 쉬운 값을 반환한다.

    이 함수를 K번 초과하여 호출할 경우 **오답** [**2**]이 된다.
  - `void Answer(int I, int J, int X)` 이 함수는, 정수 X가 쓰인 카드의 번호를 특정할 때 호출한다. 인자 `I, J, X`는, 다음의 조건을 만족해야한다.
    - $0 \leq I \leq 2N-1$
    - $0 \leq J \leq 2N-1$

* $I \neq J$
* $A_I = A_J = X$

이것을 만족하지 않는 인자로 `Answer`를 호출한 경우, **오답** [3]이 된다.
인자 `X`는, 전에 호출되었던 인자 `X`와 달라야 한다. 이것을 만족하지 않은 경우 **오답** [4]가 된다.
이 함수는, 정확히 N번 호출되어야 한다. 이것을 만족하지 않은 경우 **오답** [5]가 된다.

내부 사용을 위해서 다른 함수를 구현하거나, 글로벌 변수를 선언하는것은 자유이다. 하지만, 당신의 제출은 표준 입출력이나, 다른 함수에 접근하면 안 된다.

작성한 프로그램을 테스트하기 위한 채점 프로그램 샘플이, 콘테스트 사이트에서 다운로드 받을 수 있는 아카이브 안에 있다. 이 아카이브는, 제출해야하는 파일의 샘플도 들어있다.

채점 프로그램 샘플은 1개의 파일이다. 이 파일은 `grader.c` 혹은 `grader.cpp`이다. 작성한 프로그램을 테스트 하기 위해서는, 다음의 커맨드를 실행한다.

- C의 경우 `gcc -std=c11 -O2 -o grader grader.c Memory2.c -lm`
- C++의 경우 `g++ -std=c++11 -O2 -o grader grader.cpp Memory2.cpp`

컴파일에 성공하면, `grader`라는 이름의 파일이 생성된다.

실제의 채점 프로그램은, 채점 프로그램 샘플과는 다르므로 주의한다. 채점 프로그램의 샘플은 단일 프로세스로 실행된다. 이 프로그램은, 표준입력에서 입력을 받아서, 표준출력으로 결과를 출력한다.

채점 프로그램 샘플은, 표준입력에서 다음과 같은 데이터를 읽는다.

Input

첫째 줄에는, 정수 T, N, K가 공백으로 구분되어 들어온다. 이것은, Subtask의 번호 T이고, 카드가 $2N$개 있고, JOI군과 주고받기가 K번 까지 가능하다는 것을 의미한다.

둘째 줄에는, 정수 P_0, P_1, $\cdots$, P_{N-1}이 공백으로 구분되어 들어온다. 이것은 JOI군이 카드를 기억하기 쉬운 정도를 의미한다.

셋째 줄에는, 정수 A_0, A_1, $\cdots$, A_{2N-1}이 공백으로 구분되어 들어온다. 이것은 카드에 쓰인 정수를 의미한다.

Output

프로그램의 실행이 정상적으로 종료된 경우, 채점 프로그램 샘플은 표준출력으로 다음의 정보를 첫째 줄에 출력한다. (따옴표는 출력되지 않는다.)

- 정답일 경우, `"Accepted"`를 출력한다.
- 오답일 경우, 오답의 종류를 `"Wrong Answer [2]"`처럼 출력한다.

Constraints

모든 입력데이터는 다음의 조건을 만족한다.

- $1 \leq N \leq 50$
- $1 \leq P_i \leq N-1$ $(0 \leq i \leq N-1)$
- $P_i \neq P_j (0 \leq i < j \leq N-1)$
- $0 \leq A_i \leq N-1$ $(0 \leq i \leq 2N-1)$
- 어떤 x $0 \leq x \leq N-1)$ 에 대해서도, $A_i = x$를 만족하는 i $(0 \leq i \leq 2N-1)$ 가 정확히 2개 존재한다.

Subtask 1 (10 points)

다음의 조건을 만족한다.

- $T = 1$
- $K = 10\ 000$
- $P_i = i\ (0 \leq i \leq N - 1)$

Subtask 2 (50 points)

다음의 조건을 만족한다.

- $T = 2$
- $K = 400$
- $P_i = i\ (0 \leq i \leq N - 1)$

Subtask 3 (40 points)

다음의 조건을 만족한다.

- $T = 3$
- $K = 300$

Example

다음은 채점 프로그램 샘플이 입력받은 입력 예제와, 그에 대응되는 함수 호출 예이다.

standard input	interaction	
	Function call	Return value
1 3 10000 0 1 2 1 0 2 0 1 2	`Flip(0, 2)`	`1`
	`Flip(0, 4)`	`1`
	`Flip(1, 2)`	`0`
	`Answer(0, 4, 1)`	
	`Flip(1, 3)`	`0`
	`Flip(5, 2)`	`2`
	`Flip(4, 5)`	`1`
	`Answer(1, 3, 0)`	
	`Answer(5, 2, 2)`	

이 예제의 함수 호출은, 반드시 의미 있는 호출은 아니라는 점에 주의하라.

This page is intentionally left blank

Problem C. Solitaire

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

JOI군은, 세로 3칸, 가로 N칸의 직사각형 모양의 보드와, 몇개의 말을 가지고 게임을 하고 있다. 게임의 초기상태에는, 1개 이상에 칸에 말이 놓여져 있고, 1개 이상의 칸에 말이 놓여 있지 않다. 이 게임의 목적은, 말이 놓여 있지 않은 칸에 말을 1개씩 놓아서, 보드 위에 모든 칸에 말이 위치해 있는 상태로 만드는 것이다. 단, 어떤 칸에 말을 놓으려면 다음 중 하나의 조건을 만족해야한다.

- 그 칸의 한 칸 위 칸과 한 칸 아래 칸 양쪽에 말이 놓여 있다.
- 그 칸의 한 칸 왼 칸과 한 칸 오른 칸 양쪽에 말이 놓여 있다.

JOI군은 게임의 초기상태에서 시작하여, 목적에 맞게 말을 놓는 순서가 모두 몇 가지인지가 궁금했다. 단, 수가 매우 커 질 수 있다.

당신의 과제는, JOI군 대신, 게임의 초기상태에서 목적에 맞게 칸에 말을 놓는 순서의 가짓수를 1 000 000 007로 나눈 나머지를 구하는 것이다.

게임의 초기상태가 주어질 때, 목적에 맞게 칸에 말을 놓는 순서의 가짓수를 1 000 000 007로 나눈 나머지를 구하는 프로그램을 작성하여라.

Input

첫째 줄에는, 정수 N이 들어온다. 이것은, 게임에 사용하는 보드의 크기가 세로 3칸, 가로 N칸이라는 것을 의미한다.

이후 3개의 줄 각각에는, N개의 문자로 이루어진 문자열이 들어온다. 각 문자는 `'o'`나 `'x'`이다. 이 3개의 줄 중 i번째 ($1 \leq i \leq 3$) 줄의 왼쪽에서 j번째 문자 ($1 \leq j \leq N$)에는, 보드의 위쪽에서 i번째 행, 왼쪽에서 j번째 열에 해당하는 칸의 초기상태를 의미한다. 이 문자가 `'o'`인 경우, 게임의 초기상태에 그 칸에는 말이 놓여 있는 것을 의미한다. `'x'`인 경우, 게임의 초기상태에 그 칸에는 말이 놓여 있지 않은 것을 의미한다.

Output

목적을 달성하기 위해 말을 배치하는 순서의 가짓수를 1 000 000 007로 나눈 나머지를 첫째 줄에 출력한다.

Constraints

모든 입력데이터는 다음의 조건을 만족한다.

- $1 \leq N \leq 2\ 000$

Subtask 1 (10 points)

다음의 조건을 만족한다.

- 게임에 초기상태에, 말이 위치해 있지 않은 칸의 갯수는 16개 이하이다.
- $N \leq 30$

Subtask 2 (12 points)

- 게임의 초기상태에, 말이 놓여있지 않은 각 칸에 대해, 그 칸의 상하좌우에 말이 놓여 있는 칸의 갯수가 2개 이상이다.

Subtask 3 (20 points)

다음의 조건을 만족한다.

- 게임의 초기상태에, 말이 위치해있지 않은 칸이 세로로 3개 연달아 존재하지 않는다.
- $N \leq 30$

Subtask 4 (28 points)

- $N \leq 300$을 만족한다.

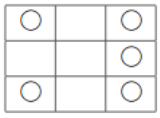
Subtask 5 (20 points)

추가 제한조건이 없다.

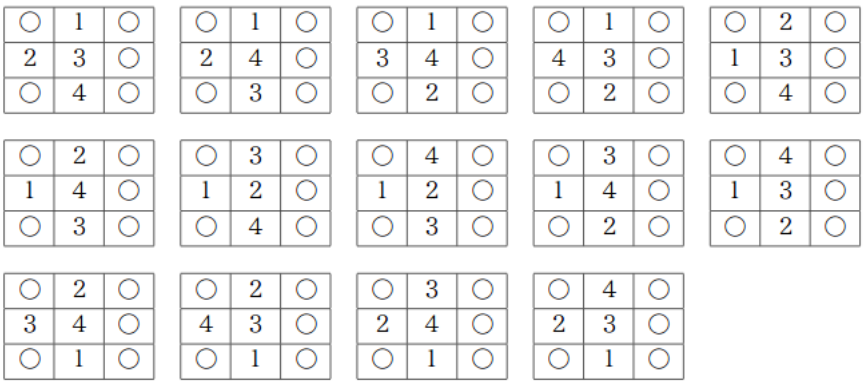
Examples

standard input	standard output
3 oxo xxo oxo	14

이 입력 예제에서는, 게임의 초기상태가 다음과 같다. (말이 위치해 있는 칸을 ○로 표시한다.)



다음 각각의 표에 쓰인 방법으로 말을 놓으면, 목적을 달성할 수 있다. (번호는 말을 놓는 순서를 의미한다.)



목적을 달성하는 순서는 위의 14가지가 있으므로, 14를 출력한다.

standard input	standard output
10 ooxooxoxoo xooxxxoxxx oxoxoooooo	149022720

입력 예제 2는 모든 Subtask의 제한조건을 만족한다.

standard input	standard output
10 ooxoxxoxoo oxxxxxoxxx oxooxoxoxo	0

게임의 초기상태에 대해서, 목적을 달성하지 못할 수도 있다.

standard input	standard output
20 oxooxoxooxoxooxoxoxo oxxxoxoxxxooxxxxxoox oxooxoxooxooxooxoxoo	228518545

This page is intentionally left blank

Problem D. Employment

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	512 megabytes

당신은 Just Odd Inventions사를 아는가? 이 회사의 업무는, "그저 기묘한 발명(just odd inventions)"를 하는것이다. 여기서는 줄여서 JOI사라고 부르겠다. JOI사의 작업을 확대하기 위해, 새로운 사원을 고용하기로 했다. 회사에는 후보자가 N명 있다. 후보자 각각은 1부터 N까지의 번호가 붙어있고, 각각의 후보자는 **평가치**라 불리는 하나의 정수가 정해져 있다. 이번 고용은, 평가치가 특정한 수치 이상의 후보자를 전원 채용한다. 새로이 채용된 사원을 몇명을 몇개의 그룹으로 나눈다. 새로이 채용된 사람의 그룹은 다음의 조건을 만족하도록 만든다.

- 후보자 a와 후보자 b $(a < b)$가 둘 다 채용되었을 때, 이 사원이 같은 그룹에 있으면, 후보자 c $(a \leq c \leq b)$가 전원 채용된 경우이고, 그 경우에 한한다.

JOI사의 인사담당인 당신은, 쿼리를 총 M개 처리하는 것으로, 이번 고용에서 만들어 질 그룹의 수를 예상하기로 했다. j번째 쿼리는, 다음의 2종류 중 하나이다.

- 평가치가 B_j 이상의 후보자를 전원 채용할 때, 그룹의 갯수를 구하여라. 이 종류의 쿼리를 **해답 쿼리**라고 한다.
- 후보자 C_j의 평가치를 D_j로 갱신한다. 이 쿼리를 **갱신 쿼리**라고 한다.

M개의 쿼리에 대한 정보가 주어졌을 때, 각각의 해답 쿼리에 대응하는 그룹의 갯수를 구하는 프로그램을 작성하여라.

Input

첫째 줄에는, 정수 N, M이 공백으로 구분되어 들어온다. 이것은, 회사의 후보자가 N명이고, 당신이 M개의 쿼리를 처리하는 것을 의미한다. 다음 N개의 줄의 i번째 줄$(1 \leq i \leq N)$에는, 정수 A_i가 들어온다. 이것은 쿼리를 처리하기 이전, 후보자 i의 평가치 A_i를 의미한다. 다음 M개의 줄의 j번째 줄$(1 \leq j \leq M)$에는, 2개 혹은 3개의 정수가 공백으로 구분되어 들어온다. 첫번째의 정수를 T_j라 하면, 이 행의 내용은 다음 중 하나를 만족한다.

1. $T_j = 1$일 때. 그 행은 정수 T_j, B_j가 공백으로 구분되어 들어온다. 이것은 j번째의 쿼리가, 평가치가 B_j이상인 후보자를 전원 합격시켰을 경우 그룹의 갯수를 구하는 쿼리를 의미한다.
2. $T_j = 2$일 때. 그 행은 정수 T_j, C_j, D_j가 공백으로 구분되어 들어온다. 이것은, j번째 쿼리가, 후보자 C_j의 평가치를 D_j로 갱신하는 쿼리를 의미한다.

Output

각각의 해답 쿼리에 대해, 그룹의 갯수를 한 줄에 출력하여라.

Constraints

모든 입력데이터는 다음의 조건을 만족한다.

- $1 \leq N \leq 200\ 000$
- $1 \leq M \leq 200\ 000$
- $1 \leq A_i \leq 1\ 000\ 000\ 000$ $(1 \leq i \leq N)$
- $1 \leq T_j \leq 2$ $(1 \leq j \leq M)$

- $1 \leq B_j \leq 1\ 000\ 000\ 000$ $(1 \leq i \leq M)$
- $1 \leq C_j \leq N$ $(1 \leq j \leq M)$
- $1 \leq D_j \leq 1\ 000\ 000\ 000$ $(1 \leq j \leq M)$
- $T_j = 1$인 j $(1 \leq j \leq M)$이 적어도 하나 존재한다.

Subtask 1 (10 points)

다음의 조건을 만족한다.

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

Subtask 2 (30 points)

- $T_j = 1$ 을 만족한다.

Subtask 3 (60 points)

추가 제한조건이 없다.

Examples

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

1. 첫번째 쿼리는 해답쿼리이다. 평가치가 5이상인 후보자 1, 후보자 2, 후보자 4를 채용할 때, 후보자 1과 후보자 2로 이루어진 그룹과, 후보자 4로 이루어진 그룹 2개의 그룹이 만들어지므로, 2를 출력한다.
2. 두번째 쿼리는 갱신 쿼리이다. 후보자 4의 평가치를 1로 갱신한다.
3. 세번째 쿼리는 해답쿼리이다. 평가치가 5이상인 후보자 1과 후보자 2를 채용할 때, 후보자 1과 후보자 2로 이루어진 1개의 그룹이 만들어지므로, 1을 출력한다.
4. 네번째 쿼리는 해답 쿼리이다. 평가치가 3 이상인 후보자 1, 후보자 2, 후보자 3, 후보자 5를 채용한 경우, 후보자 1, 후보자 2, 후보자 3으로 이루어진 그룹과, 후보자 5로 이루어진 그룹 2개의 그룹이 만들어지므로, 2를 출력한다.

standard input	standard output
7 5 13 19 1 15 13 1 19 1 20 1 1 1 6 1 11 1 17	0 1 3 3 2

입력 예제 2는 Subtask 2의 제한을 만족한다.

standard input	standard output
10 5 8 10 15 2 2 8 5 12 11 4 1 5 2 8 4 1 12 2 5 11 1 16	2 1 0

This page is intentionally left blank

Problem E. Sandwich

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	256 megabytes

JOI군은, IOI친목회에 참석해 있다. 친목회에는 샌드위치가 R행 C열의 직사각형 모양의 격자를 따라 배치되어있다. 샌드위치는, 두 변이 격자의 한 칸 길이의 직각이등변삼각형으로 되어있어서, 각각의 격자에 2개의 샌드위치가 빗변이 맞닿은 형태로 위치되어있다. 다음은 샌드위치의 배열의 예를 표시하고 있다.

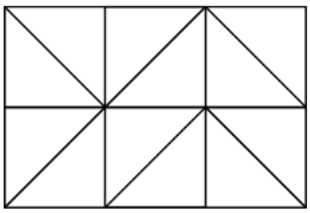


Figure 1: 샌드위치 배열의 예

이하의 두 조건을 동시의 만족하는 샌드위치는 자를 수 **없다**.

- 빗변이, 아직 잘리지 않은 샌드위치와 맞닿아 있는 경우.
- 빗변 이외의 두 변 중 적어도 하나가 아직 잘리지 않은 샌드위치와 맞닿아 있는 경우.

이 이외의 샌드위치는 자를 수 있다.

샌드위치가 전부 잘리지 않은 상태를 초기상태라고 하자. 초기상태에서, 어떤 샌드위치를 자르기 위해서는, 다른 샌드위치를 몇 개 잘라야만 할 수도 있다. 샌드위치의 위치에 따라서는, 자를 수 없는 샌드위치가 있을 수도 있다.

JOI군은, 같은 격자에 위치해있는 2개의 샌드위치를 모두 먹고싶다고 생각했다. 어떤 격자에 있는 샌드위치를 먹고 싶은가는, 아직 결정하지 않았다.

초기상태로 부터, 어떤 격자의 2개의 샌드위치를 모두 자를 때, 잘라야 하는 샌드위치의 갯수의 최솟값이 궁금했다.

샌드위치의 위치가 주어질 때, 각각의 격자에 대해, 그 격자의 2개의 샌드위치를, 초기상태로부터 샌드위치를 자를 수 있는지를 판단하고, 만약 자를수 있다면, 자를 필요가 있는 샌드위치의 갯수의 최솟값을 구하는 프로그램을 작성하라. 단, 샌드위치의 갯수는, 원하는 2개의 샌드위치를 포함해서 센다.

Input

첫째 줄에는, 정수 R, C가 공백으로 구분되어 들어온다. 이것은, 샌드위치가 R행, C열의 직사각형 모양의 격자로 이루어진 것을 의미한다.

다음 R개의 줄의 i번째 줄($1 \leq i \leq R$)에는, C문자로 되어있는 문자열이 들어온다. 각 문자는 'N' 혹은 'Z' 이다. 각 문자열의 왼쪽부터 j번째 문자($1 \leq j \leq C$)는, 위에서 부터 i번째, 왼쪽에서 부터 j번째의 격자의 샌드위치의 배열을 의미한다. 'N', 'Z'는 다음의 배열을 의미한다.

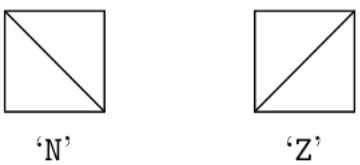


Figure 2: 각 칸의 샌드위치의 배열

Output

출력은 R줄로 한다. i번째($1 \leq i \leq R$)에는, C개의 정수를 공백으로 구분하여 출력하라. j번째($1 \leq j \leq C$) 정수는, 위에서 부터 i번째, 왼쪽에서 부터 j번째의 격자의 샌드위치 2개를 모두 자를 때, 잘라야 하는 샌드위치의 갯수의 최솟값을 출력하여라. 단, 자를 수 없는 경우에는 -1을 출력하여라.

Constraints

모든 입력데이터는 다음의 조건을 만족한다.

- $1 \leq R \leq 400$
- $1 \leq C \leq 400$

Subtask 1 (35 points)

다음의 조건을 만족한다.

- $R \leq 50$
- $C \leq 50$

Subtask 3 (65 points)

추가 제한조건이 없다.

Examples

standard input	standard output
2 3 NZN ZZN	10 8 2 8 6 4

입력 예제 1의 샌드위치의 배열은, 본문의 그림 1에 대응된다. 예를들면, 위에서 2번째, 왼쪽에서 2번째 격자의 2개의 샌드위치를 자르려면, 다음 순서로 샌드위치를 자르면 된다.

- 위에서 1번째, 왼쪽에서 3번째 격자의 오른쪽 위의 샌드위치를 자른다.
- 위에서 1번째, 왼쪽에서 3번째 격자의 왼쪽 아래의 샌드위치를 자른다.
- 위에서 2번째, 왼쪽에서 3번째 격자의 오른쪽 위의 샌드위치를 자른다.
- 위에서 2번째, 왼쪽에서 3번째 격자의 왼쪽 아래의 샌드위치를 자른다.
- 위에서 2번째, 왼쪽에서 2번째 격자의 오른쪽 아래의 샌드위치를 자른다.
- 위에서 2번째, 왼쪽에서 2번째 격자의 왼쪽 위의 샌드위치를 자른다.

총 6개의 샌드위치를 자르게 되고, 이것이 최솟값이므로 6을 출력한다.

standard input	standard output
2 2 NZ ZN	-1 -1 -1 -1

이 경우, 어떤 샌드위치도 자르는것이 불가능하다.

standard input	standard output
5 5 NZZZN NNNZN NNZNN NZNNN NZZZN	10 12 14 16 2 8 -1 -1 -1 4 6 -1 -1 -1 6 4 -1 -1 -1 8 2 16 14 12 10

This page is intentionally left blank

Problem F. Toilets

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 seconds
Memory limit:	256 megabytes

국제정보올림피아드 일본대회의 대회장소에는 화장실이 2개 있다. 한쪽은 여성전용 화장실이고, 다른쪽은 남녀공용 화장실이다. 여성은 어느쪽도 이용할 수 있지만, 남성은 남녀공용 화장실만 이용할 수 있다.

경기가 종료되었으므로, $2N$명의 선수가 화장실을 이용하기 위해 일렬로 섰다. 선수들은 여성이나 남성중 하나이다. 선수는 다음 규칙에 따라, 순서대로 화장실을 이용한다.

- 줄의 첫 선수가 여성일 경우, 선수는 비어있는 화장실을 이용한다. 단 화장실이 모두 비어있는 경우는, 여성전용 화장실에 들어간다.
- 줄의 첫 선수가 남성일 경우, 다음 규칙에 따라 화장실을 이용한다.
  - 남녀공용 화장실이 비어있을 경우, 줄의 첫 선수는 남녀공용 화장실에 들어간다.
  - 남녀공용 화장실이 비어있지 않은 경우, 여성전용 화장실이 비어있다면, 줄에 선 여성 중 가장 앞에 선 선수가 열에서 나와서 여성전용 화장실을 들어간다.

모든 선수가 화장실에 들어가 나오는 데 까지 1분이 걸린다. 선수가 화장실에 들어가는데 걸리는 시간은 없다고 하자.

열을 미리 정렬해서, N분 후에 모두가 화장실 사용을 마칠 수 있게 하고 싶다. 열의 정렬에 대해, 선수들의 불만족도를 다음과 같이 정의하자.

- 어떤 선수의 **불만족도**는, 줄에 자신보다 뒤에 서있다가, 정렬 후에 자신의 앞에 오게 된 선수의 수이다.

불만족도의 정의해서, 실제 화장실에 들어갈 때 발생하는 순서의 바뀜은 고려하지 않는다.

N분 후의 시점에 모두가 화장실 이용을 끝내도록 잘 정렬하는 중에, 선수의 불만족도의 최댓값을 최소화하고 싶다.

화장실에 서있는 $2N$명의 선수의 정보가 주어질 때, N분 후에 모두가 화장실 이용을 종료하도록 정렬할 수 있는지를 판단하고, 만약 가능하다면, 선수의 불만족도의 최댓값 중에서, 최솟값을 구하는 프로그램을 작성하여라.

Input

첫째 줄에는 정수 N이 쓰여 있다. 이것은, $2N$명의 선수가 줄에 서있다는 것을 의미한다.

둘째 줄에는, 정수 M이 쓰여 있다. M의 값과, 다음 M개의 줄의 데이터는, 열에 서 있는 선수의 정보를 의미한다.

다음 M개의 줄의 i번째 줄($1 \leq i \leq M$)에는, 문자열 S_i와 정수 K_i가 공백으로 구분되어 들어온다.

이 데이터 에서, 선수의 열을 의미하는 길이 $2N$의 문자열 X를 다음의 방법으로 결정한다.

- 문자열 X는, 문자열 $X_1, \cdots, X_M$을 순서대로 붙여 만든 문자열이다.
- 문자열 X_i $(1 \leq i \leq M)$은, 문자열 S_i를 K_i개 연속해서 붙여 만든 문자열이다.

이 방법으로 정한 문자열 X 중 왼쪽에서 j번째 $(1 \leq j \leq 2N)$ 문자는, 줄의 앞에서 j번째에 서 있는 선수의 성별을 의미한다. 이 문자가 '`M`' 이면 남성을 의미하고, '`F`'이면 여성을 의미한다.

Output

선수의 불만도의 최댓값을 최소로한 값을 첫째 줄에 출력하여라. 단, 어떻게 배치해도 N분 후에 전원이 화장실 사용을 마치는 것이 불가능 할 경우, -1을 출력하여라.

Constraints

모든 입력데이터는 다음의 조건을 만족한다.

- $1 \leq N \leq 1\ 000\ 000\ 000\ 000\ 000\ 000\ (= 10^{18})$
- $1 \leq M \leq 100\ 000$
- $1 \leq K_i \leq 2N\ (1 \leq i \leq M)$
- $1 \leq$ (문자열 S_i의 길이) $\leq 2N\ (1 \leq i \leq M)$
- 문자열 $S_i\ (1 \leq i \leq M)$의 각 문자는, '`M`' 혹은, '`F`' 중의 하나이다.
- (문자열 S_1의 길이) + (문자열 S_2 의 길이) + $\cdots$ + (문자열 S_M의 길이) $\leq 200\ 000$
- 입력 데이터로부터 정해진 X의 길이는 $2N$이다.

Subtask 1 (14 points)

다음의 조건을 만족한다.

- $N \leq 10$
- $M = 1$
- $K_1 = 1$

Subtask 2 (22 points)

다음의 조건을 만족한다.

- $N \leq 100\ 000$
- $M = 1$
- $K_1 = 1$

Subtask 3 (64 points)

추가 제한조건이 없다.

Examples

standard input	standard output
6 1 FFFMMMMMMFFF 1	2

입력 예제 1에선, 12명의 선수가 일렬로 줄을 서 있다. 다음의 방법으로 줄을 정렬하자.

1. 앞에서 부터 4번째의 선수를, 2명 만큼 앞으로 옮긴다.
2. 앞에서 부터 5번째의 선수를, 2명 만큼 앞으로 옮긴다.

이 정렬 방법을 사용하면, 선수의 불만족도의 최댓값은 2이다. 정렬한 후, 선수의 줄을 의미하는 문자열은, `FMMFFMMMMFFF`가 된다. ('`M`은 남성을, '`F`'는 여성을 의미한다). 정렬한 후의 열에 대해, 앞에서부터 i번째 $(1 \leq i \leq 12)$ 선수를, 선수 i라고 하자. 선수는 다음과 같이 화장실을 이용한다.

1. 선수 1과 선수 2가 화장실을 이용한다.
2. 1분 후에, 선수 3과 선수 4가 화장실을 이용한다.
3. 그 다음 1분 후에, 선수 5와 선수 6이 화장실을 이용한다.
4. 그 다음 1분 후에, 선수 7과 선수 10이 화장실을 이용한다.
5. 그 다음 1분 후에, 선수 8과 선수 11이 화장실을 이용한다.
6. 그 다음 1분 후에, 선수 9와 선수 12가 화장실을 이용한다.

선수의 불만족도의 최댓값을 2보다 작게 하는 정렬방법은 존재하지 않으므로, 2를 출력한다.

standard input	standard output
6 1 MMFFMMMMFFMF 1	-1

6분 후에 모두가 화장실을 이용하도록 정렬하는 방법은 존재하지 않는다.

standard input	standard output
6 1 MFFFMFMMFFFM 1	0
6 4 M 1 F 2 FM 2 MFFFM 1	0

입력 예제 3과 4에서는, 입력 데이터로 부터 결정된 문자열 X가 MFFFMFMMFFFM으로 같다.

This page is intentionally left blank

Problem G. Dungeon 2

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

당신은 Just Ordinary Inventions사를 아는가? 이 회사의 업무는, "그저 평범한 발명(just ordinary inventions)"를 하는것이다.

JOI군은, Just Ordinary Invention사가 개발한 최신 게임을 하고 있다.

이 게임은, 몇개의 방과 몇개의 길로 구성된 던전을 탐험하는 게임이다. 길은 던전 내의 서로 다른 2개의 방을 연결하고 있으며, 양방향으로 이동 가능 하다. 어떤 다른 2개의 방에 대해서도, 그들을 연결하는 길은 최대 하나 밖에 없고, 양쪽에 같은 방이 있는 길은 존재하지 않는다, 또한 던전에 있는 어떤 2개의 방도, 몇개의 길을 이용하면 서로 이동할 수 있다는 것을 알고 있다. 각 방끼리는 매우 유사하며, 동일한 갯수의 길이 나오는 방 끼리는,방 모습을 보는 것 만으로는 구별할 수 없다.

이 게임에서는 공략을 돕기 위해 각 방마다 표적과 받침대가 준비되어있다. 방에서 나오는 길은, 표적을 기준으로 첫번째, 두번째, $\cdots$로 셀 수 있다. 게임 중 던전의 구조가 바뀌지는 않는다. 따라서, 같은 방에서 같은 번호의 길을 통해 이동하면 항상 같은 방에 도착한다. 받침대는 플레이어가 색을 변경할 수 있는 구슬이 하나 있다. 보석의 색은 색 1, 색 2, $\cdots$, 색 X중 하나 이며, 게임 시작시 각 방 보석의 색은 색 1이다. 보석의 색은 플레이어가 색을 변경하지 않는 한 바뀌지 않는다.

JOI군은, 만약 던전에 구조, 즉 던전의 방들이 어떻게 길로 연결되어 있는가에 대해 알면, 이 게임은 간단히 공략이 될 것이라 느꼈다. 하지만, JOI군이 여러가지로 시도해봐도 던전의 구조를 알 수 없었다. 그래서 당신은 JOI군을 대신해서 던전의 구조를 결정하는 프로그램을 작성하기로 했다.

던전을 탐험하고, 던전의 구조를 결정하는 프로그램을 작성하여라. 단, JOI군은 던전의 구조를 완전히 아는것을 원치 않았기 때문에, 프로그램은 던전의 구조를 직접 답하는 대신, 1이상 R이하의 정수 i에 대해, "최소 i개의 길을 이용해서 2개의 방을 이동할 수 있는 쌍이 몇개 있는가" (단, 방의 순서만 바꾼 것은 같은 쌍으로 본다.)를 답해야 한다. 던전을 탐험하기 위해, 당신에게는 다음의 라이브러리가 제공된다.

- 현재 있는 방에서, 나가는 길이 몇개 있는지를 알 수 있다.
- 현재 방 받침대에 장식되어 있는 보석의 색을 알 수 있다.
- 현재 방 받침대에 장식되어 있는 보석의 색을 지정한 색으로 바꿀 수 있다. (같은 색으로도 바꿀 수 있다.) 그 후, 방에서 나와, 길을 하나 선택하고, 그 길을 이용해 다른 방으로 이동한다.
- 마지막으로 사용한 길이, 현재 있는 방에서 몇번째 길인지 알 수 있다.

Interaction Protocol

당신은, JOI군에게 답할 방법을 담은 1개의 프로그램을 작성해야 한다. 프로그램은 `dungeon2.h`를 `include` 해야 한다. 프로그램에는, 다음의 함수가 구현되어 있어야 한다.

- `void Inspect(Int R)` 이 함수는, 처음 1번만 실행된다.
  - 인자 `R`은, 1이상 R이하의 각 정수 i에 대해, "최소 i개의 길을 이용해서 2개의 방을 이동할 수 있는 쌍이 몇개 있는가" (단, 방의 순서만 바꾼 것은 같은 쌍으로 본다.)를 답해야 하는 것을 의미한다.

또한, 당신은 다음 함수를 불러 질문에 대한 답을 해야 한다.

- `void Answer(int D, int A)`
  - 인자 `D`, `A`는, 이 함수 호출을 할 때, "최소 D개의 길을 이용해서 2개의 방을 이동할 수 있는 쌍은 A개가 있다."라는 답을 의미한다.

단, Answer를 호출 할 때는, 다음의 조건을 만족해야 한다.

- D는 1이상 R이하의 정수여야 한다. 이것을 만족하지 않을 경우 **오답** [1]이 된다.
- Answer를 같은 인자 D로 2번 이상 호출하면 안된다. 이것을 만족하지 않은 경우 **오답** [2]가 된다.
- Answer는 정확히 R번 호출되어야 한다. 이것을 만족하지 않은 경우, **오답** [3]이 된다.
- A는 최소 D개의 길을 이용해서 2개의 방을 이동할 수 있는 쌍의 갯수여야 한다. 이것을 만족하지 않은 경우 **오답** [4]가 된다.

추가로, 프로그램 중에 다음 함수를 호출할 수 있다.

- `void Move(int I, int C)`
  - 인자 I는, 플레이어가 이동하기 위해서 고른 길의 번호를 의미한다. 이 함수가 호출된 직후, 플레이어는 현재 있는 방에서 I번째 길을 이용해 다른 방으로 이동한다.
  - 인자 C는, 방을 이동하기 전에, 현재 방 받침대에 있는 보석의 색을 색 C로 바꾸는 것을 의미한다.

  단, Move를 호출할 때, 다음 조건을 만족해야 한다.
  - 인자 I는, 플레이어가 현재 있는 방에서 나올 수 있는 길의 수를 K라 할 때, 1 이상 K이하의 정수여야 한다. 이것을 만족하지 않은 경우 **오답** [5]가 된다.
  - 인자 C는, 보석의 색의 종류가 X종류라고 할 때, 1 이상 X이하의 정수여야 한다. X는 Subtask에 따라 결정된다. 이것을 만족하지 않은 경우 **오답** [6]이 된다.
  - Move를 1 500 000번 이상 호출해서는 안된다. 이것을 만족하지 않은 경우 **오답** [7]이 된다.
- `int NumberOfRoads()`
  - 이 함수는, 플레이어가 현재 있는 방에서 나가는 길의 갯수를 반환한다.
- `int LastRoad()`
  - 이 함수는, 플레이어가 마지막에 사용한 길을, 현재 있는 방에서 몇번째 길인지를 반환한다. 단, Move가 한번도 호출되지 않았을 경우, -1을 반환한다.
- `int Color()`
  - 이 함수는, 플레이어가 현재 있는 방에 위치해있는 보석의 색을 반환한다.

함수 Inspect를 호출 한 후, 답에 대한 판단을 한다. 내부 사용을 위해서 다른 함수를 구현하거나, 글로벌 변수를 선언하는것은 자유이다. 하지만, 당신의 제출은 표준 입출력이나, 다른 함수에 접근하면 안 된다.

작성한 프로그램을 테스트하기 위한 채점 프로그램 샘플이, 콘테스트 사이트에서 다운로드 받을 수 있는 아카이브 안에 있다. 이 아카이브는, 제출해야하는 파일의 샘플도 들어있다.

채점 프로그램 샘플은 1개의 파일이다. 이 파일은 grader.c 혹은 grader.cpp이다. 작성한 프로그램을 테스트 하기 위해서는, 다음의 커맨드를 실행한다.

- C의 경우 `gcc -std=c11 -O2 -o grader grader.c dungeon2.c -lm`
- C++의 경우 `g++ -std=c++11 -O2 -o grader grader.cpp dungeon2.cpp`

컴파일에 성공하면, grader라는 이름의 파일이 생성된다.

실제의 채점 프로그램은, 채점 프로그램 샘플과는 다르므로 주의한다. 채점 프로그램의 샘플은 단일 프로세스로 실행된다. 이 프로그램은, 표준입력에서 입력을 받아서, 표준출력으로 결과를 출력한다.

채점 프로그램 샘플은, 표준입력에서 다음과 같은 데이터를 읽는다.

Input

- 첫째 줄에는, 정수 N, X, R이 공백으로 구분되어 들어온다. 이것은, 던전이 방 1, 방 2, $\cdots$, 방 N의 N개의 방으로 되어 있고, 보석의 색은 X종류 이며, 프로그램이 답해야 하는 값이 R개 임을 의미한다.
- 다음 $2N$개의 줄의 $2i-1$번째 줄 $(1 \leq i \leq N)$에는, 정수 D_i가 들어오고, 방 i에서 D_i개의 나가는 길이 있다는 것을 의미한다. $2i$번째 줄에는 D_i개의 정수 $T_{i1}, T_{i2}, \cdots, T_{iD_i}$가 공백으로 구분되어 들어온다. 이것은, 방 i에서 나가는 j $(1 \leq j \leq D_i)$번째 도로를 써서 이동한 방의 번호가 T_ij라는 것을 의미한다.
- 다음 R개의 줄의 j번째 줄 $(1 \leq j \leq R)$에는, 정수 A_j가 쓰여 있다. 이것은, "최소 j개의 길을 이용해서 2개의 방을 이동할 수 있는 쌍은 A_j개가 있다."는 것을 의미한다. 즉, 각 $j(1 \leq j \leq R)$에 대해, `Answer`의 인수 `D`를 j, 인수 `A`를 A_j로 해서 호출 한 경우, 채점 프로그램이 정답으로 생각하고, 다른 경우 오답으로 생각한다는 것을 의미한다.

채점 프로그램은, 플레이어의 초기 위치를 방 1로 하여, 당신이 작성한 함수를 호출한다.

Output

프로그램의 실행이 정상적으로 종료된 경우, 채점 프로그램 샘플은 표준출력으로 다음의 정보를 첫째 줄에 출력한다. (따옴표는 출력되지 않는다.)

- 정답일 경우, 함수 `Move`를 호출한 횟수가 `"Accepted : #move = 8"`처럼 출력한다.
- 오답일 경우, 오답의 종류를 `"Wrong Answer [1]"`처럼 출력한다.

Constraints

모든 입력데이터는 다음의 조건을 만족한다. N, D_i, T_{ij}의 의미에 대해서는 채점 프로그램의 샘플 입력을 참고하여라.

- $2 \leq N \leq 200$
- $3 \leq X \leq 100$
- $1 \leq R \leq 200$
- $1 \leq D_i \leq N-1$ $(1 \leq i \leq N)$
- $1 \leq T_{ij} \leq N$ 이며 $T_{ij} \neq i$ $(1 \leq i \leq N,\ 1 \leq j \leq D_i)$
- $T_{i1}, T_{i2}, \cdots, T_{iD_i}$는 서로 다르다.
- 각 i, j $(1 \leq i \leq N,\ 1 \leq j \leq D_i)$에 대해, $T_{T_{ij}k} = i$를 만족하는 k $(1 \leq k \leq D_{T_{ij}})$가 존재한다.
- 어떤 2개에 방에 대해서도, 몇개의 길을 쓰면 서로 이동할 수 있다.

이하에서, 입력데이터의 방의 수를 N, 길의 수를 M이라 한다.

Subtask 1 (17 points)

다음의 조건을 만족한다.

- $N \leq 50$
- $M \leq 100$
- $X = 100$

Subtask 2 (50 points)

다음의 조건을 만족한다.

- $N \leq 50$
- $M \leq 100$
- $X = 3$

Subtask 3 (40 points)

다음의 조건을 만족한다.

- $X = 3$을 만족한다.

이 Subtask에서는 다음에 따라 점수가 정해진다.

- 이 Subtask의 모든 테스트데이터에 대해, 다음의 최댓값을 L이라 한다.
  - `Move`의 호출 횟수를 C라고 할 때, $\frac{C}{M}$
- 이 때, 이 Subtask의 득점은
  - $L \leq 14$일 때, 56점.
  - $14 < L \leq 32$일 때, $\lfloor 70 - L \rfloor$점.
  - $32 < L \leq 64$일 때, $\lfloor 54 - \frac{L}{2} \rfloor$점.
  - $64 < L$일 때, 0점.

  여기서 $\lfloor x \rfloor$는 x를 넘지 않는 최대의 정수로 한다.

채점 시스템 상에서, 프로그램이 정상적으로 종료되었을 경우 채점 결과를 `Accepted`로 표시한다. 단 Subtask 3에서 $64 < \frac{C}{M}$인 테스트케이스에 대해서는, 결과가 **오답**으로 표시됨에 주의하라.

Example

다음은 채점 프로그램 샘플이 입력받은 입력 예제와, 그에 대응되는 함수 호출 예이다.

standard input	interaction	
	Function call	Return value
`4 3 3`	`Inspect(3)`	
`1`	`NumberOfRoads()`	`1`
`2`	`LastRoad()`	`-1`
`3`	`Move(1, 2)`	
`1 3 4`	`Color()`	`1`
`2`	`LastRoad()`	`1`
`2 4`	`NumberOfRoads()`	`3`
`2`	`Move(1, 3)`	
`2 3`	`Color()`	`2`
`4`	`Answer(1, 4)`	
`2`	`Answer(2, 2)`	
`0`	`Answer(3, 0)`	

이 예제의 함수 호출은, 반드시 의미 있는 호출은 아니라는 점에 주의하라.

Problem H. Sushi

Input file:	`standard input`
Output file:	`standard output`
Time limit:	9 seconds
Memory limit:	256 megabytes

회전초밥집 JOI는 초밥을 담은 접시를 원모양의 컨베이어 벨트로 운반하고 있다. 컨베이어 벨트는 반시계방향으로 돌고 있다. 현재, 가게 내에는 손님 1부터 손님 N까지 N명의 손님이 있고, 손님은 컨베이어 벨트에 시계 반대방향으로 앉아있다. 손님 N옆에는 손님 1이 앉아있다.

손님은, 한 사람이 한 개씩 접시를 갖고 있다. 각 접시에는 **가치**라고 불리는 숫자가 정해져 있어서, 가게를 나갈 때에 손님은 자신이 가지고 있는 접시의 가치와 같은 양의 돈을 지불한다.

회전초밥집 JOI에서는, 색다른 타임 세일을 하고 있다. 이 타임세일에는, Q번에 나누어 차례대로 요리사로부터 접시가 제공된다. i ($1 \leq i \leq Q$)번째 접시의 내용은 3개의 수 (S_i, T_i, P_i)로 표시된다.

타임세일의 룰은 다음과 같다. 타임 세일이 시작하기 직전, 요리사는 컨베이어 벨트에 있는 접시를 모두 회수한다. 이하의 1　3을 $i = 1, 2, \cdots, Q$까지 반복한다.

1. 요리사가 손님 S_i 앞 위치 컨베이어 벨트에, 가치 P_i의 접시를 놓는다.
2. 접시는 손님 S_i의 위치 부터 손님 T_i의 위치 까지 컨베이어 벨트 위를 이동한다. 각각의 손님은, 자신의 앞에 접시에 대해, 다음 행동을 한다.
   - 만약, 그 접시의 가치가 자신이 갖고 있는 접시의 가치보다 작으면, 자신의 접시와 컨베이어 벨트의 접시를 교환한다.
   - 만약, 그 접시의 가치가 자신이 갖고 있는 접시의 가치 이상이면, 접시를 교환하지 않는다.

   접시가 T_i에 도착한 후, 요리사가 그 접시를 회수한다.

당신은 요리사 밑에서 장인수업을 듣고 있는 제자이고, 가게 접시의 설거지를 맡고 있다. 회전초밥집 JOI의 접시의 가치에 따라 설거지 방법이 다르다. 당신은 설거지를 준비하기 위해서, 타임세일 Q번에 대해, 요리사가 어떤 가치의 접시를 회수하는지 알고 싶다.

각각의 손님이 타임세일 직전에 가지고 있는 접시의 정보와, 타임세일에 제공될 접시의 정보가 주어졌을 때, 요리사가 회수하는 각 접시의 가치를 구하는 프로그램을 작성하라.

(추가 사항) $S_i = T_i$인 경우, 손님 S_i만이, 2번째 행동을 진행하게 된다.

Input

표준 입력으로, 다음의 데이터가 들어온다.

- 첫째 줄에는, 정수 N, Q가 공백으로 구분되어 들어온다. 이는, 손님의 수가 N명이며, 타임세일에 제공될 접시의 수가 Q개 라는 것을 의미한다.
- 다음 N개의 줄의 i번째 줄($1 \leq i \leq N$)에는, 정수 X_i가 입력으로 들어온다. 이는, 손님 i가 타임세일 직전에 가치 X_i의 접시를 가지고 있다는 것을 의미한다.
- 다음 Q개의 줄의 i번째 줄에는, 정수 S_i, T_i, P_i가 공백으로 구분되어 들어온다. 이는, io번째 제공되는 접시가 (S_i, T_i, P_i)로 표시됨을 의미한다.

Output

출력은 Q개의 줄로 되어있다. 표준 출력에 i번째 줄 ($1 \leq i \leq Q$)에, i번째로 요리사가 회수하는 접시의 가치를 의미하는 정수를 출력하여라.

Constraints

모든 입력데이터는 다음의 조건을 만족한다.

- $1 \leq N \leq 400\ 000$
- $1 \leq Q \leq 25\ 000$
- $1 \leq X_i \leq 1\ 000\ 000\ 000$ $(1 \leq i \leq N)$
- $1 \leq S_i \leq N$ $(1 \leq i \leq Q)$
- $1 \leq T_i \leq N$ $(1 \leq i \leq Q)$
- $1 \leq O_i \leq 1\ 000\ 000\ 000$ $(1 \leq i \leq Q)$

Subtask 1 (5 points)

다음의 조건을 만족한다.

- $N \leq 2\ 000$
- $Q \leq 2\ 000$

Subtask 2 (15 points)

다음의 조건을 만족한다.

- $S_i = 1$ $(1 \leq i \leq Q)$
- $T_i = N$ $(1 \leq i \leq Q)$

Subtask 3 (80 points)

추가 제한조건이 없다.

Examples

standard input	standard output
6 7 8 6 7 4 5 9 2 4 5 4 1 4 6 2 7 1 5 2 3 4 8 4 3 1 3 1 3	7 9 8 7 8 6 5

손님 1부터 손님 6이 가지고 있는 접시의 가치는, 각각의 접시가 제공된 후 다음과 같이 바뀐다.

- 1번째 접시가 제공된 후, 8, 5, 6, 4, 5, 9
- 2번째 접시가 제공된 후, 8, 5, 6, 4, 4, 5
- 3번째 접시가 제공된 후, 7, 5, 6, 4, 4, 5
- 4번째 접시가 제공된 후, 2, 5, 6, 4, 4, 5

- 5번째 접시가 제공된 후, 2, 5, 6, 4, 4, 5
- 6번째 접시가 제공된 후, 2, 5, 5, 1, 4, 4
- 7번째 접시가 제공된 후, 2, 5, 3, 1, 4, 4

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

입력 예제 2는, Subtask 2의 조건을 만족한다.

standard input	standard output
10 10 19 5 8 17 14 3 9 10 7 6 1 8 4 7 3 2 5 9 10 4 8 3 10 3 6 8 7 4 6 6 3 2 9 12 6 3 7 9 6 3	19 10 14 17 8 10 3 12 7 9

This page is intentionally left blank

Problem I. Telegraph

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

JOI군도는, 태평양에 떠있는 작은 섬나라이다. JOI군도에는, N개의 섬이 있고, 1부터 N까지의 번호가 붙어 있다.

JOI군도는, 섬끼리 통신을 주로 무선통신을 한다. 각각의 섬에는 전파의 발신기와 수신기가 하나씩 있다. 발신기는 전방향으로 전파를 발신하는 것이 가능하지만, 수신기는 특정한 방향에서 오는 전파밖에 받지 못한다. 그래서, 각각의 수신기는, 특정한 하나의 섬에서 밖에 전파를 수신하지 못한다. 단, 수신기의 방향을 바꾸는 것으로, 어느 섬에서 전파를 수신할 것인지 바꾸는 것은 가능하다.

현재, 섬 $i(1 \leq i \leq N)$의 수신기는 섬 $A_i(A_i \neq i)$에서 오는 전파를 수신하는 것이 가능하다. 그리고, 섬 i의 수신기의 방향을 바꾸는데 드는 비용은, 어느 방향으로 바꾸든 C_i이다.

JOI군도에는, 공공사업으로 전보 서비스를 하고 있다. 섬 i $(1 \leq i \leq N)$에서, 전파를 섬 j $(1 \leq j \leq N, j \neq i)$의 수신기가 받는 것이 가능 할 때, 섬 i에서 섬 j에 무선통신을 통해 전보를 보내는 것이 가능하다. 그리고, 전보는 몇개의 섬을 경유해도 된다. 즉, 섬 i, 섬 j, 섬 k $(1 \leq i, j, k \leq N$, i, j, k는 서로 다르다.)에 대해, 섬 i에서 섬 j에 전보를 보낼 수 있고, 섬 j에서 섬 k에 전보를 보낼 수 있으면, 섬 i에서 섬 k에 전보를 보낼 수도 있다. 무선통신 이외에 방법으로 전보를 보내는것은 불가능하다.

JOI군도의 통신장관인 당신은, 임의의 섬에서 임의의 섬으로 정보를 보낼수 있도록 하고 싶다. 그러기 위해서, 몇개의 섬의 수신기의 방향을 바꿔야 할 수도 있다. 몇개의 섬의 수신기의 방향을 바꾸는데 드는 비용은, 각각의 수신기의 방향을 바꾸는 데 쓰는 비용의 총합이다.

임의의 섬에서 임의의 섬으로 전보를 보낼 수 있도록 하기 위해 드는 비용의 최소치를 계산하여라.

JOI 제도의 섬의 수와 각각의 섬의 수신기에 대한 정보가 주어졌을 때, 임의의 섬에서 임의의 섬으로 전보를 보낼 수 있도록 하기 위해 드는 비용의 최소치를 구하는 프로그램을 작성하라.

Input

표준 입력으로 다음의 데이터가 들어온다.

- 첫째 줄에는, 정수 N이 들어온다. 이것은, JOI군도에 N개의 섬이 있다는 것을 의미한다.
- 다음 N개의 줄의 i번째 줄$(1 \leq i \leq N)$에는, 정수 A_i, C_i가 공백으로 구분되어 들어온다. 이는, 섬 i의 수신기는, 현재 섬 A_i에서 전파를 수신하는 것이 가능하고, 방향을 바꾸는데 드는 비용이 C_i라는 것을 의미한다.

Output

표준 출력에, 임의의 섬에서 임의의 섬으로 전보를 보낼 수 있게 하는 비용의 최솟값을 첫째 줄에 출력하여라.

Constraints

모든 입력데이터는 다음의 조건을 만족한다.

- $2 \leq N \leq 100\ 000$
- $1 \leq A_i \leq N$ $(1 \leq i \leq N)$
- $A_i \neq i$ $(1 \leq i \leq N)$
- $1 \leq C_i \leq 1\ 000\ 000\ 000$ $(1 \leq i \leq N)$

Subtask 1 (10 points)

다음의 조건을 만족한다.

- $N \leq 10$

Subtask 2 (30 points)

다음의 조건을 만족한다.

- $N \leq 15$

Subtask 3 (30 points)

다음의 조건을 만족한다.

- $N \leq 3\ 000$

Subtask 4 (30 points)

추가 제한조건이 없다.

Examples

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

섬 2의 수신기의 방향을 바꿔, 섬 4에서 오는 전파를 받을수 있게 한다. 이렇게 하면 임의의 섬에서 임의의 섬으로 전보를 보내는 것이 가능하고, 비용은 4가 된다.

어떻게 수신기의 방향을 바꿔도 비용을 4보다 낮추는 것은 불가능하기 때문에, 4를 출력한다.

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

일단, 섬 1의 수신기의 방향을 바꿔 섬 4에서 오는 전파를 받을수 있게 한다. 그리고, 섬 3의 수신기의 방향을 바꿔 섬 2에서 오는 전파를 받을 수 있게 한다. 이렇게 하면 임의의 섬에서 임의의 섬으로 전보를 보내는 것이 가능하고, 비용은 $2 + 3 = 5$가 된다.

어떻게 수신기의 방향을 바꿔도 비용을 5보다 낮추는 것은 불가능하기 때문에, 5를 출력한다.

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

섬 1과 섬 3의 수신기의 방향을 바꾸면 된다.

standard input	standard output
3 2 1 3 1 1 1	0

어떤 섬의 수신기의 방향도 바꾸지 않아도 된다.

This page is intentionally left blank

Problem J. Dangerous Skating

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 megabytes

JOI군은 대자연의 광대한 스케이트링크에서 아이스스케이트를 타는 취미가 있다.

스케이트링크는 남북으로 R칸, 동서로 C칸으로 이루어진 직사각형으로 표현된다. 북쪽에서 r번째, 서쪽에서 c번째의 칸을 (r, c)라고 표시하자. 각각의 격자는 JOI군이 통과 할 수 있지만, 얼음덩어리가 있어 통과 할 수 없는 칸이 몇 개 있다. 또한, 스케이트 링크의 외벽의 격자에는 모두 얼음덩어리가 있어, 링크 밖으로 나가는것도 불가능 하다. 즉, 격자 $(i, 1), (i, C)(1 \leq i \leq R)$ 과, 격자 $(1, j), (R, j)(1 \leq j \leq C)$에는 얼음덩어리가 있다.

JOI군은 스케이트를 잘 타지 못한다. JOI군이 스케이트 링크 위에서 이동할 때는, 동서남북중 한 방향을 정해, 현재 JOI군이 있는 격자에서 미끄러져, 얼음 덩어리에 부딪히기 바로 전 격자까지 달려가 멈춘다. 링크에서 미끄러져 멈출때 까지를 1번의 이동으로 센다. 이동하고자 하는 칸에 얼음덩어리가 있을 때는, 그 방향으로 이동할 수 없다.

여느날, JOI군이 스케이트를 즐기고 있는 동안, JOI군이 링크에서 이동하면, 그 자리에 얼음덩어리가 생기는 것을 깨달았다. 미끄러진 지점 이외에 얼음 덩어리가 생기는 일은 없다. 이 상황에서 스케이트를 계속 타는것은 위험하므로, JOI군은 최대한 빨리 스케이트 링크를 탈출하고 싶다.

JOI군은, 현재 격자 (r_1, c_1)에 있다. 이 스케이트 링크에서 탈출하기 위해서는 출구의 격자 (r_2, c_2)에서 멈출 필요가 있다. JOI군이 안전하게 스케이트링크에서 탈출 할 수 있도록, 현재 지점에서 이동을 시작해 출구에서 멈추기 위해서는, 적어도 몇번의 이동이 필요한가 계산하는 프로그램을 작성했으면 한다. 스케이트 링크의 상태는, JOI군이 현재 지점에서 출구 격자에 멈추는 것이 불가능 할 수도 있다. JOI군의 이동 중에 출구 격자를 지나친것 만으로는 스케이트링크를 탈출 할 수 없다는 것에 주의해라.

스케이트링크 위의 얼음덩어리의 정보와, JOI군의 현재 위치, 출구 격자의 위치가 주어질 때, JOI군이 현재 격자에서 이동을 시작해서, 출구 격자에 멈추는 것이 가능한지 판단하고, 가능 할 경우, 필요한 이동의 최소횟수를 구하는 프로그램을 작성하여라.

Input

표준 입력으로, 다음의 데이터가 들어온다.

- 첫째 줄에는, 정수 R, C가 공백으로 구분되어 들어온다. 이는, 스케이트 링크의 크기가 남북으로 R칸, 동서로 C칸임을 의미한다.
- 다음 R개의 줄은 각각, C문자로 된 문자열이 들어온다. 각 문자는 '`.`' 혹은 '`#`'이다. 이 R행 중 r번째 $(1 \leq r \leq R)$에서, 왼쪽에서 c번째 문자 $(1 \leq c \leq C)$는, 스케이트 링크의 격자 (r, c)의 초기상태를 의미한다. 이 문자가 '`.`'인 경우에는, 그 격자를 지나갈 수 있다는 것을 의미하고, 이 문자가 '`#`'인 경우에는 그렇지 않다는 것을 의미한다.
- 다음 1개의 줄에는 정수 r_1, c_1이 공백으로 구분되어 들어온다. 이것은, JOI군이 현재 위치가 격자 (r_1, c_1)이라는 것을 의미한다.
- 다음 1개의 줄에는 정수 r_2, c_2가 공백으로 구분되어 들어온다. 이것은, 스케이트링크의 출구가 격자 (r_2, c_2)에 위치함을 의미한다.

Output

표준출력에, JOI군이 현재 위치에서 시작해서, 출구의 격자에 멈추기 위해 필요한 이동횟수의 최솟값을 의미하는 정수를 첫째 줄에 출력하여라. 단, JOI군이 어떻게 해도 출구의 격자에 멈출 수 없을 경우, `-1`을 출력하여라.

Constraints

모든 입력데이터는 다음의 조건을 만족한다.

- $3 \leq R \leq 1\ 000$
- $3 \leq C \leq 1\ 000$
- $1 \leq r_1 \leq R$
- $1 \leq c_1 \leq C$
- $1 \leq r_2 \leq R$
- $1 \leq c_2 \leq C$
- 스케이트링크의 외벽의 칸은 모두 얼음이 있다. 즉, 격자 $(i, 1), (i, C)(1 \leq i \leq R)$과, 격자 $(1, j), (R, j)(1 \leq j \leq C)$에는 얼음덩어리가 있다.
- 격자 (r_1, c_1)과 격자 (r_2, c_2)에는 얼음덩어리가 없다.

Subtask 1 (13 points)

다음의 조건을 만족한다.

- $R \leq 10$
- $C \leq 10$
- JOI군이 현재 위치에서 시작해, 출구가 있는 격자로 나갈 수 있는 경우에는, 필요한 이동의 횟수가 10 번 이내이다.

Subtask 2 (65 points)

다음의 조건을 만족한다.

- $R \leq 200$
- $C \leq 200$

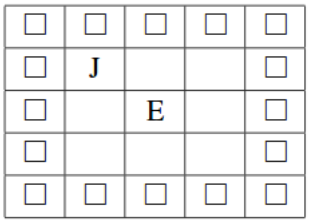
Subtask 3 (22 points)

추가 제한조건이 없다.

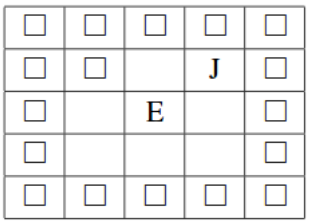
Examples

standard input	standard output
5 5 ##### #...# #...# #...# ##### 2 2 3 3	4

입력 예제 1에서는, 스케이트 링크의 초기상태가 다음과 같다. 흰색 사각형이 적힌 격자는 얼음덩어리를, J가 적힌 격자는 JOI군의 현재 위치를, E가 적힌 격자는 출구를 의미한다.



일단, JOI군이 동쪽 방향으로 이동한 후에, 스케이트링크의 상태는 다음과 같이 된다.



그 후 JOI군이 서쪽으로, 남쪽으로, 북쪽으로 차례로 이동한 경우, 총 4계의 이동으로 출구에서 멈추는 것이 가능하다. 3번 이하로 이동해서 출구에서 멈추는 것이 불가능 하므로, 4를 출력한다.

standard input	standard output
8 6 ###### #..#.# ##...# #....# #.#..# #....# ##...# ###### 4 3 6 4	5
5 5 ##### #.#.# #.#.# #.#.# ##### 2 2 4 4	4
3 3 ### #.# ### 2 2 2 2	0

입력 예제 4에서는, JOI군이 현재 위치가 출구이므로, 필요한 이동 횟수는 0이다.

This page is intentionally left blank

Problem K. Snowy Roads

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

러시아의 많은 지점은 눈이 쌓여있다. 러시아에는 N개의 도시가 있고, 0부터 $N-1$까지의 번호가 붙어있다. 러시아에는 $N-1$개의 도로가 있고, 0부터 $N-2$까지의 번호가 붙어있다. 도로 i $(0 \leq i \leq N-2)$는, 2개의 서로 다른 도시 $A_i, B_i (0 \leq A_i < B_i \leq N-1)$를 양방향으로 잇고 있다. 러시아의 어떤 2개의 다른 도시도, 몇개의 도로를 경유하면 이동하는 것이 가능하다.

각 도로의 강설 상황은 매일 바뀐다. 어떤 날짜에 대해서도, 도로의 강설 상황은 눈이 내리거나, 눈이 내리지 않는것 중 하나이다. 하루 안에 강설상황이 변하는 일은 없다.

Anya와 Boris는 러시아의 통신국에서 일하고있다. Anya는 도로정보 관리부서, Boris는 시민에서 질문을 대답하는 부서에 소속되어 있다. 시민이 하는 질문은, 러시아의 수도인 도로 0부터 다른 어떤 도시까지 이동하기 위해서는, 눈이 내리는 도로를 최소 몇개 거쳐야 하는가 이다. Boris는 보통 질문을 받은 후, Anya와 상호작용을 하여 시민에게 답변 한다.

여기서는, 러시아에서 Q일간 프로그래밍 컨테스트 세계대회가 개최되었다. 대회기간 중에, 통신회선이 혼잡해서, Anya와 Boris가 직접 상호작용하는 것이 어려울 것이라 예상된다. 그래서, Anya와 Boris는 다음의 방법으로 상호작용을 하는것으로 했다.

- Anya는, 하루의 시작에 그 날의 강설정보를 모아, 중간 서버에 데이터를 전송한다.
- Boris는, 시민으로부터 질문을 받아, 중간 서버와 상호작용 하는 것으로 답을 한다.

단, Anya 와 Boris가 상호작용을 하는데에는 제약이 있다.

- 중간 서버의 용량은 $L = 1000$비트이다. Anya는 중간 서버에 최대 L비트의 정보만 저장 할 수 있다.
- 중간 서버에 저장된 데이터는, 하루의 시작에 모두 0으로 초기화 된다.
- Boris는 중간서버와 1번 주고받기를 할 때, 지정한 1비트의 정보를 읽는 것이 가능하다.
- 질문에 대답하기 위해 Boris가 중간 서버와 상호작용 할 수 있는 것은, 각 질문에 대회 최대 20번 까지이다.

통신국장과 아는 사이인 당신은, Anya와 Boris에게 전략을 알려주게 되었다.

시민에게서 받은 질문을 바르게 대답하도록 Anya와 Boris의 전략을 구현한 프로그램을 작성하여라.

Interaction Protocol

당신은, 같은 프로그래밍 언어로 2개의 파일을 제출해야 한다.

첫번째 파일은 `Anya.c`혹은 `Anya.cpp`라는 이름이다. 이 파일은 Anya의 전략이 구현된 파일이고, 다음의 2개의 함수가 구현되어 있어야한다. 프로그램은 `Anyalib.h`를 `include`해야 한다.

- `void InitAnya(int N, int A[], int B[])`

  이 함수는, 각 테스트케이스에 대해 한 번만 호출된다.

  - 인자 `N`은 도시의 갯수를 의미한다.
  - 인자 `A[]`와 `B[]`는 각각의 길이 $N-1$인 배열이고, 도로의 연결정보를 의미한다. `A[i]`와 `B[i]` $(0 \leq i \leq N-2)$는 도로 i가 도시 `A[i]`와 `B[i]`를 양방향으로 잇는 것을 의미하는 정수이고, $0 \leq$ `A[i]` ¡ `B[i]` $\leq N-1$을 만족한다.

- `void Anya(int N, int C[])`

  이 함수는, `InitAnya`가 호출 된 후, Q번 호출된다. 이 함수는, 하루의 시작에 도로의 강설 상황을 갱신한 후, Anya가 중간 서버에 저장할 비트를 정하는 것에 대응된다.

  - 인자 `C[]`의 길이는 $N-1$이고, 도로의 강설정보를 의미한다. `C[i]`$(0 \leq i \leq N-2)$는 도로 i의 강설정보를 의미하는 0이나 1인 숫자이고, `C[i]` $= 1$이면, 도로 i에 눈이 내린 것을, `C[i]` $= 0$이면, 도로 i에 눈이 내리지 않은 것을 의미한다.

    함수 `Anya`의 중에 이하의 함수를 호출할 수 있다.

    * `void Save(int place, int bit)`
      이 함수는 Anya가 중간 서버의 비트를 저장하는 작업을 의미한다.
      · 인자 `place`는 비트를 쓸 위치를 의미한다. `place`는 0 이상 $L-1$이하의 정수여야 한다. 이 범위 외의 값을 정해서 함수를 호출 할 경우, **오답** [1]이 된다. 또, 이 함수는 각 `Anya`의 호출에 대해, 같은 인자 `place`를 2회 이상 호출할 수 없다. 같은 인자로 2회 이상 호출한 경우 **오답** [2]가 된다.
      · 인자 `bit`는, 써 넣은 비트를 의미하는 정수이고, 0이나 1이어야 한다. 이 범위 외의 값을 정해서 함수를 호출 할 경우, **오답** [3]이 된다.

    `Save`를 호출 한 후, 중간서버의 `place`번째 비트는 `bit`이 된다. `Save`를 호출 하는 중에 오답이 된 경우, 그 시점에서 프로그램을 종료한다.

    함수 `Anya`가 호출 되기 직전에, 반드시, 중간서버의 비트는 0으로 초기화된다. 즉, 함수 `Save`를 통해 저장되지 않은 장소에는, 함수 `Anya`가 종료 될 때, 0이 적혀 있다.

두번째 파일은 `Boris.c`혹은 `Boris.cpp`라는 이름이다. 이 파일은 Anya의 전략이 구현된 파일이고, 다음의 2개의 함수가 구현되어 있어야한다. 프로그램은 `Borislib.h`를 `include`해야 한다.

- `void InitBoris(int N, int A[], int B[])`

  이 함수는, 각 테스트케이스에 대해 한 번만 호출된다.

  - 인자 `N`은 도시의 갯수를 의미한다.
  - 인자 `A[]`와 `B[]`는 각각의 길이 $N-1$인 배열이고, 도로의 연결정보를 의미한다. `A[i]`와 `B[i]` $(0 \leq i \leq N-2)$는 도로 i가 도시 `A[i]`와 `B[i]`를 양방향으로 잇는 것을 의미하는 정수이고, $0 \leq$ `A[i]` ¡ `B[i]` $\leq N-1$을 만족한다.

- `void Boris(int city)`

  이 함수는, `InitAnya`가 호출 된 후, 몇번이든 호출 된다. 이함수는 시민의 질문에 대응하는 Boris의 행동에 대응된다.

- 인자 `city`는 시민의 질문을 의미한다. `city`는 1 이상 $N-1$이하의 정수이다. 이것은 도시 0 부터 도시 `city`까지 이동하려면, 눈이 쌓인 도로를 최소 몇 개 지나야 하는지를, 시민이 질문하고 있는 것을 의미한다.

- 함수 `Boris`는, 시민의 질문에 대한 답을 0 이상 $N-1$이하의 정수로 반환해야 한다. 이 범위 외의 값을 반환 할 경우, **오답** [4]가 된다. 답이 올바르지 않을 경우 **오답** [7]이 된다.

  함수 `Boris`의 중에 이하의 함수를 호출할 수 있다.

- `int Ask(int place, int bit)`
  이 함수는 Boris가 중간 서버에서 비트를 읽는 작업을 의미한다.
  * 인자 `place`는 비트를 읽을 위치를 의미한다. `place`는 0 이상 $L-1$이하의 정수여야 한다. 이 범위 외의 값을 정해서 함수를 호출 할 경우, **오답 [5]**가 된다.

  함수 `Ask`의 반환값은, 중간 서버의 `place`번째 비트를 의미하는 정수이고, 0 혹은 1이다. 또, 함수 `Ask`는 함수 `Boris`의 **각 호출에 대해, 최대 20번 만 호출 할 수 있다.** 20번을 초과해 호출을 한 경우, **오답 [6]**이 된다.

`Ask`를 호출 하는 중에 오답이 된 경우, 그 시점에서 프로그램을 종료한다.

채점은 이하의 순서로 진행된다. 오답이라고 판정된 경우에는, 그 시점에 프로그램이 종료된다.

1. 도로정보를 알리기 위해, `InitAnya`와 `InitBoris`가 1번씩 순서대로 호출된다.
2. $i = 1, \cdots, Q$에 대해, 순서대로 이하의 작업을 한다.
   (a) 함수 `Anya`를 1회 부른다. 이것은, 하루의 시작에 도로의 강설정보를 갱신 한 후에, Anya가 중간 서버에 저장하는 비트열을 정하는 것에 대응된다.
   (b) 함수 `Boris`를 D_i회 부른다. D_i는 i일에 대해 시민이 한 질문의 횟수 이다. 그 중 j번째 $(1 \leq j \leq D_i)$ 호출을 의미하는 수는 R_{ij} $(1 \leq R_{ij} \leq N-1)$이다. j번째 호출에 대해, 함수 `Boris`의 반환값이, 도시 0부터 도시 R_{ij}까지 이동하기 위해 건너야 하는 눈이 쌓인 길의 최솟값과 일치하지 않은 경우, 오답이 된다.
3. 한번도 오답이라고 판정되지 않은 경우, 정답이 된다.

- 실행시간 계산 ·사용 메모리 계산의 대상은, 채점의 순서의 1, 2번이다. 정답인 경우에는, 순서 2에서 `Anya`가 Q번, $Boris$가 합계 $D_1 + \cdots + D_Q$번 호출된다.
- Anya나 Boris는, $D_1, \cdots, D_Q$에 대해서 알 수 없다.
- 함수 `Boris`는, 어떤 시점에 함수 `Anya`가 중간 서버에 존재하고 있는 데이터를 갱신했는가에 대한 정보를 모른다.
- 내부 사용을 위해서 다른 함수를 구현하거나, 글로벌 변수를 선언하는것은 자유이다. 하지만, 당신이 제출한 2개의 프로그램은 링크되어 하나의 실행파일이 되므로, 각 파일의 모든 글로벌 변수와 내부 함수(`InitAnya, Anya, InitBoris, Boris`를 제외)를 `static`으로 선언하여, 다른 파일과의 충돌을 막을 필요가 있다. 채점시에는, 이 프로그램을 Anya측, Boris측의 2개의 프로세스로 하여 실행하므로, Anya측과 Boris측이 프로그램 중에 글로벌 변수를 공유하는 것은 불가능하다.
- 당신의 제출은 표준 입출력이나, 다른 함수에 접근하면 안 된다.

작성한 프로그램을 테스트하기 위한 채점 프로그램 샘플이, 콘테스트 사이트에서 다운로드 받을 수 있는 아카이브 안에 있다. 이 아카이브는, 제출해야하는 파일의 샘플도 들어있다.

채점 프로그램 샘플은 1개의 파일이다. 이 파일은 `grader.c` 혹은 `grader.cpp`이다. 작성한 프로그램 `Anya.c`와 `Boris.c`, 혹은 `Anya.cpp`와 `Boris.cpp`라고 할 때, 작성 한 프로그램을 테스트 하기 위해서는, 다음의 커맨드를 실행한다.

- C의 경우 `gcc -std=c11 -O2 -o grader grader.c Anya.c Boris.c -lm`
- C++의 경우 `g++ -std=c++11 -O2 -o grader grader.cpp Anya.cpp Boris.cpp`

컴파일에 성공하면, `grader`라는 이름의 파일이 생성된다.

실제의 채점 프로그램은, 채점 프로그램 샘플과는 다르므로 주의한다. 채점 프로그램의 샘플은 단일 프로세스로 실행된다. 이 프로그램은, 표준입력에서 입력을 받아서, 표준출력으로 결과를 출력한다.

채점 프로그램의 샘플은, 채점의 순서에 따라 함수를 호출한다. 다음이 실제 채점 프로그램과 다름을 주의하라.

- **채점 프로그램의 샘플은, 오답[7]을 판정하지 않는 대신에, 각 질문에 대한 함수 `Boris`의 반환값을 출력한다.**

Input

채점 프로그램 샘플은, 표준입력에서 다음과 같은 데이터를 읽는다.

- 첫째 줄에는, 정수 N이 입력으로 들어온다. 이것은 도시의 수가 N개 있다는 것을 의미한다.
- 다음 $N-1$개의 줄의 $I+1$번째 줄 $(0 \leq i \leq N-2)$에는, 정수 A_i, B_i가 공백으로 구분되어 들어온다. 이것은 도로 i가 도시 A_i와 도시 B_i를 양방향으로 잇는 것을 의미한다.
- 다음 줄에는 정수 Q가 쓰여 있다. 이것은 대회가 Q일간 개최되었다는 것을 의미한다.
- 다음 Q개의 줄의 i번째 줄 $(1 \leq i \leq Q)$에는, 길이 $N-1$의 문자열 S_i와 D_i+1개의 정수가 공백으로 구분되어 들어온다. S_i는 i번째 날의 강설정보를 의미하고, S_i에서 왼쪽으로 $j+1$번째 $(0 \leq j \leq N-2)$ 문자가, '`1`'이면, 도로 j가 눈이 내린 것을, '`0`'이면, 도로 j에 눈이 내리지 않은 것을 의미한다. 다음 D_i+1개의 정수 중 가장 첫 정수는 D_i이다. 다음 D_i개의 정수는 $R_{i1}, \cdots, R_{iD_i}$이다. 이것은, i번째 날에 대해, j번째 $(1 \leq j \leq D_i)$ 질문이, 도시 0부터 도시 R_ij까지 이동하기 위해 눈이 쌓인 도로를 건너야 하는 수의 최소치에 관련이 있다는 것을 의미한다.

Output

채점 프로그램의 샘플은 표준 출력으로 이하의 정보를 출력한다. (따옴표는 실제로 출력되지 않는다.)

- 프로그램의 실행 중에 오답이라고 판단되는 경우, 오답의 종류가 "`Wrong Answer[1]`"처럼 출력되어, 실행이 종료된다.
- i번째 $(1 \leq i \leq Q)$의 `Anya`의 호출 후 호출된 D_i개의 `Boris`의 호출에 대해 모두 오답이라고 판된되지 않았을 경우, 그 시점에 한 줄에 D_i개의 정수를 공백으로 구분해서 출력한다. 출력된 정수 중 j번째 $(1 \leq j \leq D_i)$ 정수는, `Boris`(R_ij)의 반환값이다.

실행한 프로그램이 여러개의 오답을 일으켰을 경우, 표시되는 것은 그 중 하나 뿐이다.

Constraints

모든 입력데이터는 다음의 조건을 만족한다.

- $2 \leq N \leq 500$
- $1 \leq Q \leq 500$
- $0 \leq A_i < B_i \leq N-1$ $(0 \leq i \leq N-2)$
- $1 \leq D_j$ $(1 \leq j \leq Q)$
- $D_1 + \cdots + D_Q \leq 500$
- 어떤 2개의 서로 다른 도시 사이에도, 몇개의 도로를 경유하면 갈 수 있다.

Subtask 1 (15 points)

다음의 조건을 만족한다.

- $N \leq 20$

Subtask 2 (5 points)

다음의 조건을 만족한다.

- $N \leq 100$

Subtask 3 (35 points)

다음의 조건을 만족한다.

- $A_i = i$ $(0 \leq i \leq N-2)$
- $B_i = i+1$ $(0 \leq i \leq N-2)$

Subtask 4 (45 points)

추가 제한조건이 없다.

Example

다음은 채점 프로그램 샘플이 입력받은 입력 예제와, 그에 대응되는 함수 호출 예이다.

standard input	interaction			
	함수 호출	반환값	함수 호출	반환값
	`InitAnya(...)`			
		(없음)		
	`InitBoris(...)`			
		(없음)		
	`Anya(...)`			
			`Save(0, 1)`	
				(없음)
			`Save(1, 0)`	
				(없음)
			`Save(500, 1)`	
				(없음)
`5`		(없음)		
`0 1`	`Boris(2)`			
`1 2`			`Ask(0)`	
`1 4`				`1`
`2 3`			`Ask(1)`	
`2`				`0`
`0101 3 2 4 3`			`Ask(2)`	
`1110 1 4`				`0`
		`1`		
	`Boris(4)`			
		`0`		
	`Boris(3)`			
		`2`		
	`Anya(...)`			
		(없음)		
	`Boris(4)`			
			`Ask(0)`	
				`0`
		`2`		

이 예제의 함수 호출은, 반드시 의미 있는 호출은 아니라는 점에 주의하라. 이 때, `InitAnya(...)`, `InitBoris(...)`, 첫번째와 2번째의 `Anya(...)`에 전달되는 인수는 다음과 같다.

인자	InitAnya(...)	InitBoris(...)	Anya(...)	
			함수호출	반환값
N	5	5	1번째	2번째
A	{0, 1, 1, 2}	{0, 1, 1, 2}		
B	{1, 2, 4, 3}	{1, 2, 4, 3}		
C			{0, 1, 0, 1}	{1, 1, 1, 0}

Problem L. Worst Reporter 2

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 second
Memory limit:	256 megabytes

때는, 21XX년, Problem Solving은 마인드 스포츠의 하나로 널리 인지되어 있고, TV나 신문 등 미디아에 오르는 일도 많다. 당신은 JOI신문사의 기사이고, Problem Solving 기사를 담당하고 있다. 어느 날, N명의 선수가 참가하는 국제적인 Problem Solving 대회가 개최되었다. 이 대회에 대해 기사를 쓰기 위해, 당신은 다음의 정보를 알았다.

- 국제 정보올림피아드 처럼, 이 대회는 몇개의 나라에서 온 선수들이 참여한다. 나라에는 1부터 N까지의 번호가 붙어있다. 하나의 나라에서 여러명의 선수가 참가 할 수도 있다. 또한, 선수가 참가하지 않은 나라가 있을 수도 있다.
- 이 대회의 시간제한은 5시간이다.
- 대회 중에 선수가 획득한 점수는, 그 후 줄어들지 않는다.
- 대회 개최후 2시간이 경과한 시점에, 동점인 선수는 없었다. 그 시점의 순위표는 i등 $(1 \leq i \leq N)$인 선수는 A_i 나라 출신이고, 그 선수의 점수는 B_i점 이었다.
- 대회가 끝날 때, 동점의 선수는 없었다. 대회가 끝날 때 순위표에 따르면 i등 $(1 \leq i \leq N)$인 선수는 C_i 나라 출신이고, 그 선수의 점수는 D_i점 이었다.

그러나, 기사를 쓰려 보니, 순위표의 출신국의 표시에 결함이 있었다는 것이 밝혀졌다. 선수의 출신국의 정보가 잘못되었을 가능성이 있다. 표시된 선수의 점수는 올바르다는 것을 알고 있다.

여기서, 당신은, 알고 있는 정보로 최소한의 수정을 가해서, 순위표의 정보를 모순이 없게(같은 선수의 국적이 대회 기간동안 변하지 않고, 획득한 순서의 감소가 없게) 추측하기로 했다. 즉, $2N$개의 값 $A_1, \cdots, A_N, C_1, \cdots C_N$중 최대한 적은 갯수의 값을 변경해서, 다음을 만족하도록 하고 싶다.

- $1, 2, \cdots, N$의 순열 $x_1, x_2, \cdots, x_N$에 대해, 각 $i = 1, 2, \cdots, N$에 대해, $A_i = C_{x_i}$ 이고, $B_i \leq D_{x_i}$가 성립하는 것이 존재한다.

당신은, 알고 있는 정보로, 최소 몇번의 수정을 가해야 하는가.

대회의 참가자수와, 대회 개최후 2시간 경과한 시점과 대회 종료 시점의 순위표에 대한 정보가 주어졌을 때, 순위표에 모순이 없게 만드는데 필요한, 출신국 정보의 변경 횟수의 최솟값을 구하는 프로그램을 작성하여라.

Input

표준 입력으로 다음의 데이터가 들어온다.

- 첫째 줄에는, 정수 N이 들어온다. 이것은, 대회의 참가자 수가 N명이라는 것을 의미한다.
- 다음 N개의 줄의 i번째 줄$(1 \leq i \leq N)$에는, 정수 A_i, B_i가 공백으로 구분되어 들어온다. 이는, 대회 개최 후 2시간 경과한 시점의 순위표가 i등인 선수가 국가 A_i 출신으로 표시되었고, 획득 점수가 B_i 점이었다는 것을 의미한다.
- 다음 N개의 줄의 i번째 줄$(1 \leq i \leq N)$에는, 정수 C_i, D_i가 공백으로 구분되어 들어온다. 이는, 대회 종료 시점의 순위표가 i등인 선수가 국가 C_i 출신으로 표시되었고, 획득 점수가 D_i점이었다는 것을 의미한다.

Output

표준 출력에, 순위표에 모순이 없는 상황을 만들기 위해 필요한, 출신국 정보의 변경갯수의 최솟값을 첫째 줄에 출력하여라.

Constraints

모든 입력데이터는 다음의 조건을 만족한다.

- $2 \leq N \leq 200\ 000$
- $1 \leq A_i \leq N$ $(1 \leq i \leq N)$
- $0 \leq B_i \leq 1\ 000\ 000\ 000$ $(1 \leq i \leq N)$
- $B_i > B_{i+1}$ $(1 \leq i \leq N-1)$
- $1 \leq C_i \leq N$ $(1 \leq i \leq N)$
- $0 \leq D_i \leq 1\ 000\ 000\ 000$ $(1 \leq i \leq N)$
- $D_i > D_{i+1}$ $(1 \leq i \leq N-1)$
- $A_1, \cdots, A_N, C_1 \cdots, C_N$의 값을 몇개 수정하는 것으로, 순위표에 모순이 없는 상황을 만들 수 있다.

Subtask 1 (15 points)

다음의 조건을 만족한다.

- $N \leq 16$

Subtask 2 (15 points)

다음의 조건을 만족한다.

- $N \leq 50$

Subtask 3 (30 points)

다음의 조건을 만족한다.

- $N \leq 5\ 000$

Subtask 4 (40 points)

추가 제한조건이 없다.

Examples

standard input	standard output
3 3 500 2 200 1 100 1 1000 3 700 3 400	1

C_3의 값을 2로 수정하면, 다음과 같이 모순이 없는 순위표가 된다:

- 대회 개최후 2시간 경과한 시점에 500점으로 1위 였던 나라 3의 선수는, 대회 종료 후 700점으로 2위가 되었다.

- 대회 개최후 2시간 경과한 시점에 200점으로 2위 였던 나라 2의 선수는, 대회 종료 후 400점으로 3위가 되었다.
- 대회 개최후 2시간 경과한 시점에 100점으로 3위 였던 나라 1의 선수는, 대회 종료 후 1000점으로 1위가 되었다.

여기서, C_2의 값을 2로 수정한 경우, 나라 3 출신의 선수가 대회 개최후 2시간 경과한 시점에서 500점을 획득했음에도 불구하고, 대회 종료 시점에서 400점을 획득했기 때문에, 모순된 순서표가 된다. 1개 보다 더 작은 수정을 가해 모순이 없는 순위표를 만드는 것이 불가능 하므로, 1을 출력한다.

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

이 경우, 출신국 정보의 수정이 없어도 모순이 없는 순위표가 된다. 대회 개최후 2시간 경과시점의 점수에서 점수를 획득하지 못한 선수가 있을수도 있음에 주의하여라. 또, 순의표에 대해, 같은 나라의 전수가 여럿 있을 가능성이 있음에 주의하여라.

standard input	standard output
6 1 70 4 50 1 30 2 20 1 10 3 0 6 100 2 90 1 80 2 60 4 40 1 10	3

이 입력에 대해서는, A_1의 값을 2로 수정하고, A_6의 값을 4로 수정하고, C_1의 값을 4로 수정하는 것으로, 모순이 없는 순위표가 된다.

This page is intentionally left blank