문제 A. 스키테일 암호

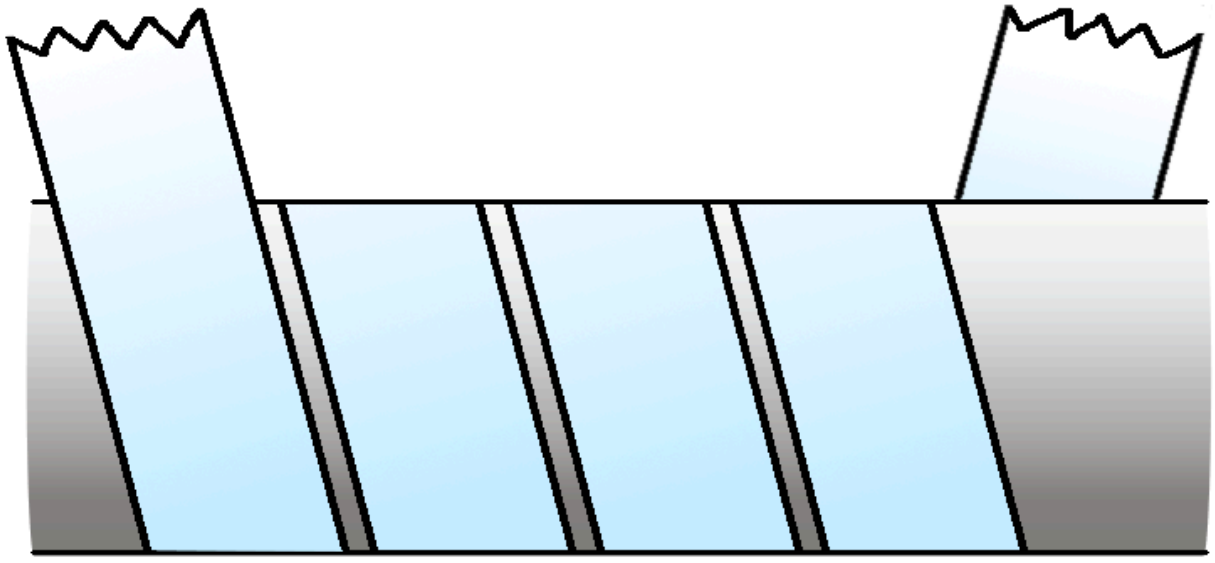
시간 제한: 1초

메모리 제한: 1024 MB

고대 그리스의 옛 나라인 스파르타의 군대에서는 비밀메시지를 전하는 방법으로 스키테일 암호를 사용했다.

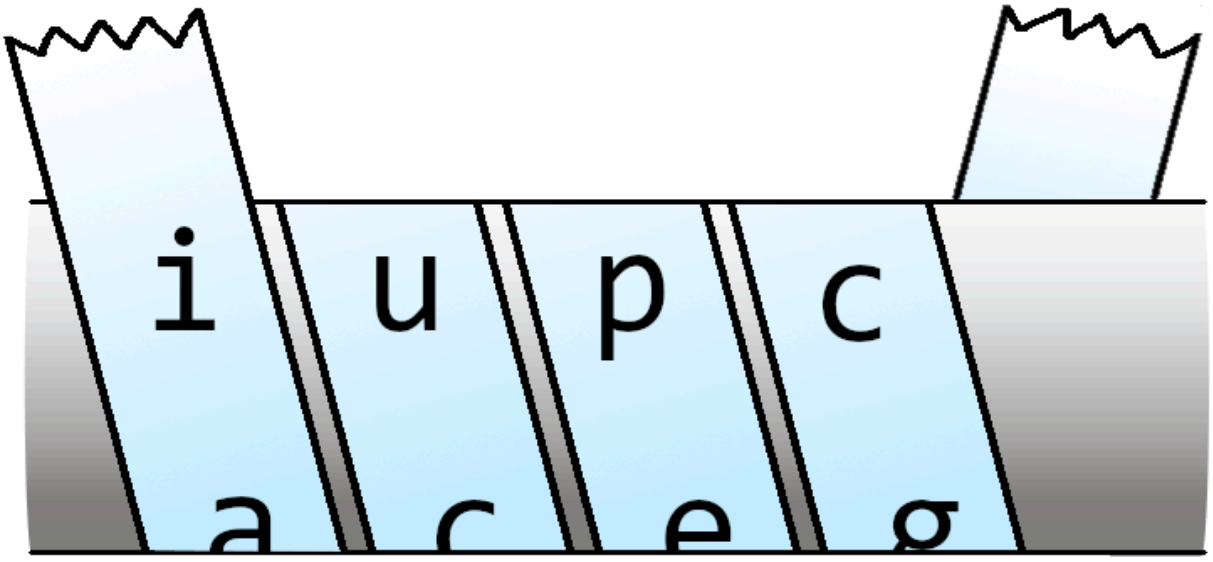
스키테일 암호는 스키테일(Scytale)이라고 하는 정해진 굵기의 원통형 막대에 종이로 된 리본을 위에서 아래로 감은 다음 옆으로 메시지를 적는 방식으로 메세지를 암호화한다. 리본을 풀어 길게 늘어선 글을 읽으면 무슨 뜻인지 전혀 알 수 없지만, 암호화할 때와 같은 굵기의 막대에 감으면 내용을 알 수 있게 된다.

다음은 굵기 3의 막대를 사용하여 "iupc" 라는 문자열을 암호화하는 예시이다.

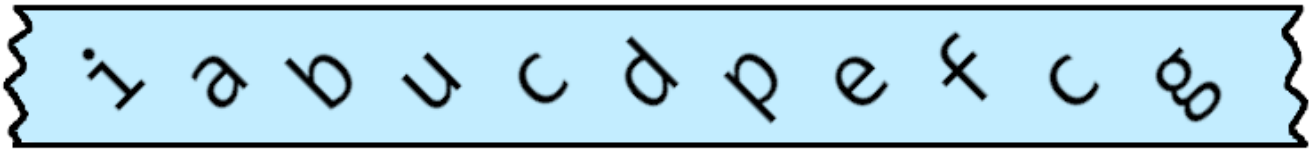


굵기가 X인 막대에 리본을 감고 세로로 글자 X개를 적으면 막대를 한바퀴 돌아오게 된다. 이 막대는 굵기가 3이므로, 세로로 3글자를 적으면 막대를 한바퀴 돌아올 것이다.

암호화하는 문자열을 리본의 가장 왼쪽 끝 부분을 포함하는 가로 한 줄만 사용하여 쓰고, 남은 공간은 아무 문자로나 채운다.



마지막으로 막대에서 리본을 풀면 암호화가 완료된다.



스키테일 암호로 암호화한 문자열과 막대의 굵기가 주어진다. 암호를 해독해 보자!

입력

첫 번째 줄에 막대의 굵기 K가 주어진다. 두 번째 줄에 알파벳 소문자만으로 구성된 암호문 S가 주어진다.

출력

첫 번째 줄에 암호문을 해독한 결과를 출력한다.

제한

$3 \le K \le 10$

$1 \le S$의 길이 $\le 1,000$

예제 입력 1	예제 출력 1
3 iabucdpefcg	iupc

문제 B. 오델로

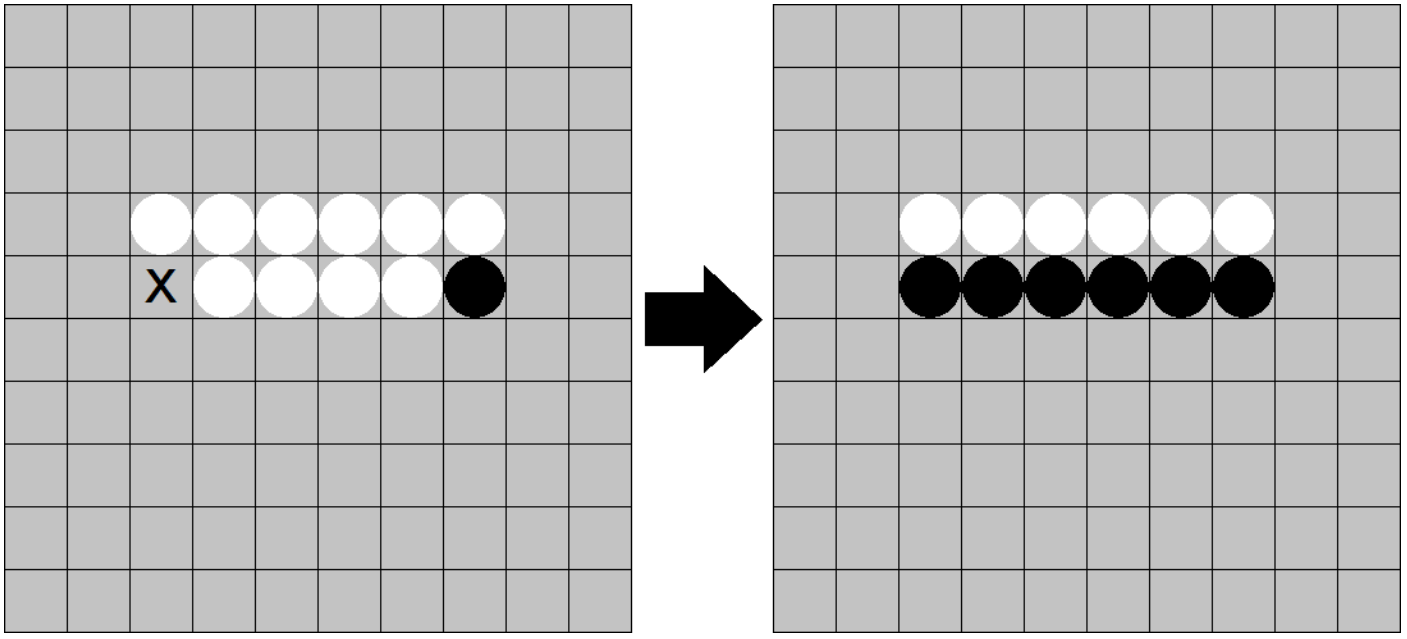
시간 제한: 1 초

메모리 제한: 1024 MB

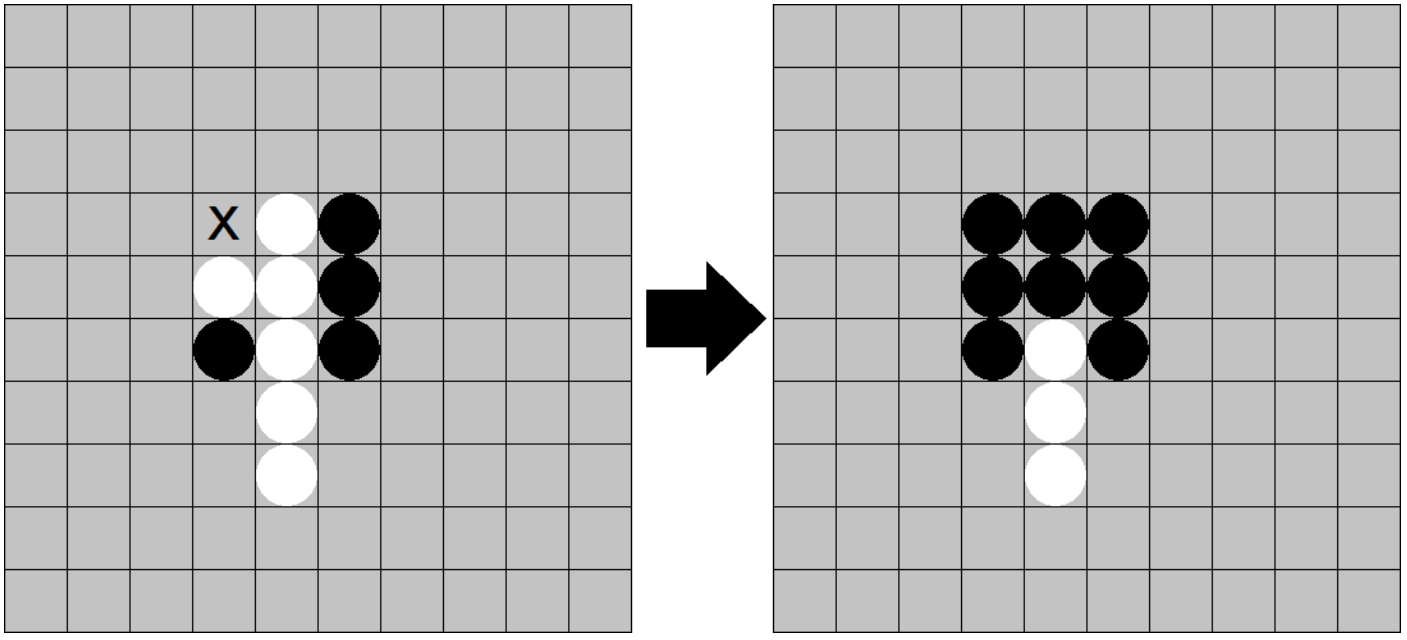
종민이는 CTP에서 만든 AI의 성능을 테스트하기 위해 $N \times N$ 크기의 격자판에서 오델로 게임을 하고 있다.

오델로 판의 가장 왼쪽 위칸의 좌표는 $(0, 0)$, 가장 오른쪽 아래칸의 좌표는 $(N-1, N-1)$ 이다. 오른쪽은 x 좌표가 증가하는 방향, 아래쪽은 y좌표가 증가하는 방향이다.

오델로는 두 플레이어가 번갈아 가며 한 번씩 돌을 놓는 게임이다. 자신이 돌을 놓은 곳과 판에 이미 놓여있는 자신의 돌 사이에 상대편의 돌만이 적어도 한 개 이상 존재한다면, 사이에 있는 상대편의 돌이 모두 뒤집혀 자신의 돌이 된다. 이와 같이 상대편의 돌을 뒤집을 수 있는 곳에만 돌을 놓을 수 있다.



돌은 가로, 세로, 대각선의 총 8방향으로 뒤집을 수 있으며, 동시에 여러 방향에 있는 돌을 뒤집을 수도 있다. 예를 들어, 아래의 그림에서 X 위치에 흑돌을 놓으면 3개의 백돌이 뒤집혀 흑돌이 된다.



만약 어느 곳에 두어도 상대의 돌을 뒤집을 수 없어서 돌을 놓을 수 없다면, 자신의 차례를 건너뛰고 상대의 차례로 넘어가게 된다.

AI의 성능이 너무 좋은 나머지 종민이는 지금까지 모든 게임을 져버리고 말았다. 알고 있는 모든 수 싸움을 걸어도

이길 수 없었던 종민이는 가장 단순한 방법을 써 보기로 하였다. 그 방법은 자신의 차례에 상대의 돌을 가장 많이 뒤집을 수 있는 곳에 돌을 두는 것이다!

현재 종민이는 흑돌로 플레이하고 있다. 어떤 곳에 두어야 가장 많은 백돌을 뒤집을 수 있을까?

입력

첫째 줄에 오델로 판의 크기 N이 주어진다.

둘째 줄부터 N개의 줄에 걸쳐서 오델로 판의 정보를 뜻하는 문자열이 주어진다.

$i+1$번째 줄의 j번째 문자는 오델로 판의 좌표 $(j-1, i-1)$의 상태를 나타낸다. 각 문자의 의미는 다음과 같다.

'.'은 빈 공간을 나타낸다.

'W'는 백돌을 나타낸다.

'B'는 흑돌을 나타낸다.

빈 공간이 적어도 한 칸 이상 존재함이 보장된다.

출력

첫째 줄에 흑돌을 놓았을 때 가장 많은 백돌을 뒤집을 수 있는 곳을 (x, y) 좌표 순으로 공백으로 구분하여 출력한다.

둘째 줄에 그 좌표에 돌을 두었을 때 뒤집히는 백돌의 개수를 출력한다.

만약 흑돌을 놓을 수 없다면, 대신 첫째 줄에 "PASS"를 출력한다.

가장 많은 백돌을 뒤집을 수 있는 좌표가 여러 개일 경우엔 y좌표가 가장 작은 것을, y좌표도 같다면 그 중 x좌표가 가장 작은 것을 출력한다.

제한

$3 \le N \le 500$

예제 입력 1	예제 출력 1
3 .B. .W. ...	1 2 1

예제 입력 2	예제 출력 2
5 .WWWB BWWBB WBWBB WBBWB W...B	0 0 6

예제 입력 3	예제 출력 3
8 .WWWWWW. WWWWWWWW WWWWWWWW WWBWWWWW WWWWWWWW WWWWWWWW WWWWWWWW	PASS

.WWWWWW.	

문제 C. 균형 삼진법

시간 제한: 1초

메모리 제한: 1024 MB

균형 삼진법은 밑이 3이고, 자릿수가 0, 1, -1로 이루어진 기수법이다.

이를 이용해 별도의 부호를 사용하지 않고서도 모든 정수를 유일한 방법으로 나타낼 수 있다.

십진수를 입력 받아 균형 삼진법으로 출력하는 프로그램을 작성하시오.

입력

십진법으로 나타낸 정수 N이 주어진다.

출력

문제의 정답을 출력한다. 자릿수가 -1이라면 대신 'T'를 출력한다.

제한

$-10^9 \le N \le 10^9$

예제 입력 1	예제 출력 1
10	101

예제 입력 2	예제 출력 2
5	1TT

예제 입력 3	예제 출력 3
-8	T01

힌트

$$10 = 1\times3^2 + 0\times3^1 + 1\times3^0$$

$$5 = 1\times3^2 + (-1)\times3^1 + (-1)\times3^0$$

$$-8 = (-1)\times3^2 + 0\times3^1 + 1\times3^0$$

문제 D. Aging

시간 제한: 2 초

메모리 제한: 1024 MB

우선순위 스케줄링은 우선순위가 높은 프로세스가 리소스를 먼저 사용하도록 하는 스케줄링 기법이다. 프로세스의

우선순위를 적절하게 할당할 수 있다면 우선순위가 높은 프로세스를 먼저 처리하는 것이 합리적일 것이라는 고찰로부터 고안되었다.

하지만 우선순위가 높은 프로세스들이 계속해서 요청된다면 우선순위가 낮은 프로세스는 순위에 밀려 영원히 실행되지 않을 수도 있게 된다. 이런 현상을 starvation이라고 한다. 프로세스 간 우선순위 책정이 항상 정확하게 이루어진다는 보장이 없기 때문에 우선순위가 낮다고 해서 실행되지 않아도 되는 것은 아니다. 우선순위가 높은 프로세스를 먼저 처리하지만 모든 요청된 프로세스가 고루 실행될 수 있도록 하려면 starvation을 해결해야 한다.

aging은 starvation을 근본적으로 해결하는 방법이다. aging은 실행되지 않고 준비 큐에 대기 중인 프로세스들의 우선순위를 점진적으로 증가시킨다. 이렇게 하면 실행되지 못하고 큐에 남은 프로세스는 어느 순간부터는 최고 우선순위를 가지게 되고 반드시 실행될 수 있다.

용광이는 운영체제 과목의 자유 과제로 aging이 적용된 우선순위 스케줄러를 제출해서 고득점을 노려보기로 했다. 용광이가 구현할 우선순위 스케줄러는 다음과 같은 조건을 만족한다.

각 프로세스에 입력된 순서대로 번호를 부여한다. 첫 번째로 입력된 프로세스에는 1번, 두 번째로 입력된 프로세스에는 2번, ..., K번째로 입력된 프로세스에는 K번을 부여한다.

현재 실행되고 있는 프로세스가 없다면, 지금까지 실행 요청된 프로세스 중 우선순위가 가장 높은 프로세스가 실행된다.

우선순위가 같은 프로세스가 여러 개라면 실행 시간이 짧은 프로세스가 먼저 실행된다.

우선순위가 같고 실행 시간이 같은 프로세스가 여러 개라면 부여된 번호가 작은 프로세스가 먼저 실행된다.

실행 요청되었지만 실행되지 않은 프로세스는 단위 시간당 우선순위가 1씩 증가한다.

한 프로세스가 실행 중인 동안에는 다른 프로세스가 요청되어도 실행을 중단하지 않는다.

용광이는 우선순위 스케줄러를 완성했지만 자기가 만든 스케줄러가 정확하게 동작하는지는 알 수 없었다. 이래서는 과제 제출을 할 수 없다.

용광이가 고득점의 꿈을 버리지 않도록 스케줄러가 올바르게 동작하는지 확인해주는 프로그램을 만들어보자.

입력

첫 번째 줄에는 프로세스의 개수 N이 주어진다.

다음 N개의 줄에는 i번 프로세스의 정보를 나타내는 3개의 정수 t_i, p_i, b_i가 공백으로 구분되어 주어진다.

t_i는 i번 프로세스가 실행 요청된 시점이며, 비내림차순으로 주어진다.

p_i는 i번 프로세스의 초기 우선순위이며, $p_i < p_j$면 i번 프로세스보다 j번 프로세스의 우선순위가 더 높음을 뜻한다.

b_i는 i번 프로세스의 실행 시간이다.

출력

프로세스가 실행되는 순서대로, 해당 프로세스의 번호를 공백으로 구분하여 출력한다.

제한

$1 \le N \le 3 \times 10^5$

$0 \le t_i \le 3 \times 10^5$

$0 \le p_i \le 10^3$

$0 \le b_i \le 10^3$

예제 입력 1	예제 출력 1
3 0 0 5 1 2 3 2 1 2	1 2 3

예제 입력 2	예제 출력 2
5 0 1000 8 0 1 4 0 2 3 3 5 3 9 11 2	1 3 5 4 2

문제 E. 두 반으로 나누기

시간 제한: 1 초

메모리 제한: 1024 MB

종민이는 CTP 초등학교 5학년 3반 담임 교사로 일하고 있다. 최근 들어 CTP 초등학교에 전학을 오는 5학년 학생들이 많아져서, 대책으로 종민이가 담당하는 3반을 2개 분반으로 나누기로 했다.

종민이는 이전부터 자신이 담당하는 반의 학생들로부터 고통을 받고 있었다. 5학년 3반 친구들은 수업 시간임에도 불구하고 친구들끼리 놀면서 수업을 듣기를 거부하는 일이 많기 때문에, 5학년 3반의 분반은 종민이에게 매우 희소식이었다.

종민이는 5학년 3반의 모든 친한 친구인 두 학생을 서로 다른 분반에 배치하여 고통을 줄이려고 한다. 하지만 그런 배치가 불가능한 경우도 있기 때문에, 이를 위해 한가지 묘책을 생각해 냈다. 그것은 바로 친한 친구 관계인 두 학생이 서로 험담을 한다는 소문을 퍼뜨려 두 친구를 절교시키는 것이다! 친한 친구인 두 학생이 절교하면 더 이상 친한 친구가 아니게 된다.

종민이는 이런 식으로 절교시킬 친한 친구 쌍의 리스트를 만들었다. 물론 친한 친구 쌍들을 마구 절교시키다가 학생들끼리 파벌이 나뉘어 싸움이 나는 것은 원하지 않기 때문에, 해당 리스트의 모든 친한 친구 쌍을 절교시켜도 학생들끼리 파벌이 나뉘지는 않도록 했다. 파벌이 나뉘지 않는다는 말은, 친한 친구 관계인 두 학생을 모두 간선으로 이은 그래프가 연결 그래프임을 뜻한다.

종민이는 리스트에 쓰인 차례대로 친한 친구인 두 학생을 절교시킨 다음, 두 분반에 모두 친한 친구인 두 학생이 없도록 학생들을 배치하여 평화를 찾으려고 한다. 하지만 선생으로서 마지막 양심이 남아있었던 종민이는, 친한 친구 쌍을 절교시키는 일을 최소한으로 하려고 한다. 최소 몇 쌍의 친한 친구를 절교시켜야 각 분반에 친한 친구인 두 학생이 없도록 학생들을 나눌 수 있을까?

입력

첫째 줄에 N, M, K가 공백으로 구분되어 주어진다.

N은 5학년 3반의 학생 수이다.

M은 5학년 3반의 친한 친구 쌍의 수이다.

K는 종민이가 절교시킬 리스트에 포함되어 있는 친한 친구 쌍의 개수이다.

둘째 줄부터 M개의 줄에 걸쳐, i번째 친한 친구 쌍을 나타내는 u_i, v_i가 공백으로 구분되어 주어진다. 이는 u_i번 학생과 v_i번 학생이 친한 친구임을 뜻한다. 중복된 관계는 주어지지 않는다.

$M + 2$번째 줄부터 K개의 줄에 걸쳐 R_i가 주어진다. 이는 i번째로 절교시킬 친한 친구 쌍이 R_i번째 친한 친구 쌍임을

뜻한다. 입력되는 R_i는 모두 서로 다름이 보장된다.

출력

첫째 줄에 절교시켜야하는 최소 친한 친구 쌍의 개수를 출력한다.

둘째 줄에 두 분반의 각 학생 수를 오름차순으로 출력한다.

만약 리스트의 모든 친한 친구 쌍을 절교시켜도 두 분반으로 나눌 수 없다면, 대신 첫째 줄에 "-1"을 출력한다.

제한

$2 \le N \le 10^5$

$N - 1 \le M \le min\left(\frac{N(N-1)}{2}, 2 \times 10^5\right)$

$0 \le K \le M - N + 1$

$1 \le u, v, \le N$

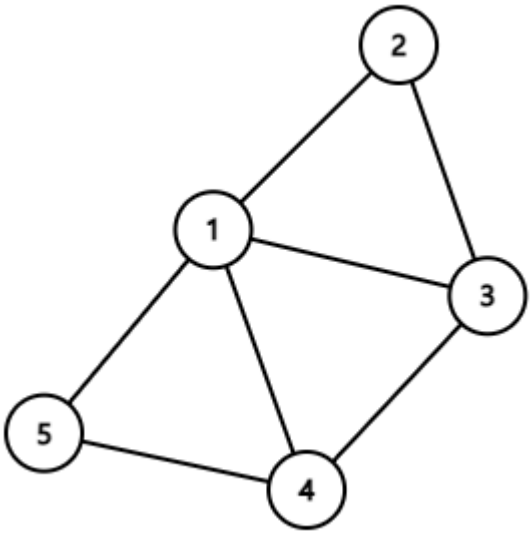
$1 \le R_i \le M$

예제 입력 1	예제 출력 1
5 7 3 1 2 1 3 1 4 1 5 2 3 3 4 4 5 5 3 7	2 2 3

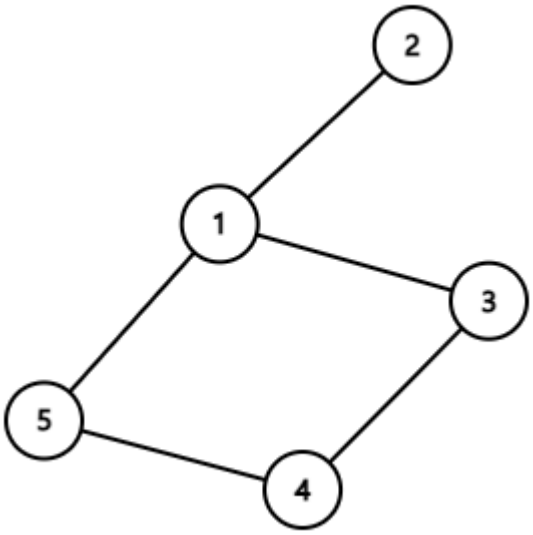
예제 입력 2	예제 출력 2
5 7 2 1 2 1 3 1 4 1 5 2 3 3 4 4 5 2 3	-1

힌트

1번 예제의 초기 친한 친구 관계를 그래프로 나타내면 다음과 같다.



종민이는 5번째, 3번째, 그리고 7번째 친한 친구 쌍을 차례대로 절교시키려고 한다. 이 중 5번째와 3번째, 두 쌍을 절교시키면 한 분반에는 1, 4번 학생을, 나머지 분반에는 2, 3, 5번 학생을 배치함으로써 각 분반에 친한 친구 쌍이 없도록 나눌 수 있다.



이보다 적은 쌍을 절교시키면서 문제의 조건을 만족하도록 하는 것은 불가능하다.

문제 F. IUPC와 비밀번호

시간 제한: 0.5 초

메모리 제한: 1024 MB

치훈이는 인하대학교 빡코딩 대회, IUPC (Inha University Ppakcoding Contest)에 문제를 출제하고 있다.

대회에 낼 문제들은 극비사항이기 때문에, 치훈이는 문제가 저장된 컴퓨터에 비밀번호를 설정해 놓았다. 치훈이가 비밀번호를 만드는 방식은 다음과 같다.

알파벳 소문자로만 이루어진 문자열 S를 준비한다.

문자열의 각 문자들의 위치를 랜덤하게 섞는다.

섞인 문자열의 앞과 뒤에 임의의 문자열을 각각 추가한다. 추가하는 문자열의 길이는 0일 수도 있다.

마지막으로 임의의 위치에 있는 문자 1개를 골라 다른 문자로 바꾼다.

그런데 어느 날, 치훈이는 문자열 문제를 풀다가 실수로 컴퓨터 비밀번호와 다른 문자열들을 섞어 버려서 어떤 문자열이 비밀번호였는지 알 수 없게 되어 버렸다.

지금 치훈이에게 N개의 비밀번호 후보 문자열이 있다. IUPC의 성공적인 개최를 위해 각 비밀번호 후보 문자열이 비밀번호일 가능성이 있는지 알아내 주자!

입력

첫째 줄에 S가 주어진다.

두번째 줄에 비밀번호 후보 문자열의 개수 N이 주어진다.

다음 N개의 줄에 걸쳐, i번째 비밀번호 후보 문자열 T_i이 각각 주어진다.

출력

N개의 줄에 걸쳐, 각 T_i가 비밀번호일 가능성이 있는지 여부를 출력한다.

가능하면 "YES", 불가능하면 "NO"를 출력한다.

제한

$1 \le S$의 길이 $\le 2,000$

$1 \le N \le 2,000$

$1 \le T_i$의 길이 $\le 2,000$

입력으로 주어지는 모든 문자열은 알파벳 소문자로만 이루어져 있다.

예제 입력 1	예제 출력 1
abc 4 defabcghi bda cba ac	YES YES NO NO

문제 G. 최단최단경로

시간 제한: 2 초

메모리 제한: 1024 MB

선노가 사는 도시엔 1번부터 N번까지, 총 N개의 지하철 역이 있다. 이 지하철은 M개의 노선으로 이루어져 있고, 각각의 노선은 서로 다른 2개의 역을 편도로 운행한다.

선노는 x번 역에 살고 있고, 회사는 y번 역에 있다. 매일 최단경로로 출근을 하던 선노는 환승하는 것이 너무 귀찮아져서 회사에 최단거리로 출근하면서도, 동시에 가장 환승을 적게 하는 경로인 최단최단경로로 출근을 하려고 한다. 즉, 최단최단경로는 최단경로이면서 동시에 가장 적은 개수의 노선을 사용하는 경로이다.

지하철 노선이 주어질 때, 최단최단경로로 출근할 때의 이동 거리와 거치는 노선의 수, 그리고 서로 다른 최단최단경로의 개수를 구해보자.

입력

첫번째 줄에 N, M, x, y가 공백으로 구분되어 주어진다.

두번째 줄부터 M개의 줄에 각 지하철 노선을 의미하는u, v, w가 공백으로 구분되어 주어진다. 이는 u번 역부터에서 v번 역까지 이동하는 지하철 노선이 존재하며, 이 노선을 사용했을 때 이동 거리가 w임을 의미한다.

어떤 역에서 특정 역으로 향하는 노선이 여러 개일 수 있다. 이 때 해당 노선들은 모두 서로 다른 노선이다.

출력

x에서 y까지 갈 수 있는 경로가 없다면,

첫 번째 줄에 "-1"을 출력한다.

x에서 y까지 갈 수 있는 경로가 있다면,

첫 번째 줄에 x에서 y로 최단최단경로로 이동할 때의 이동 거리를 출력한다.

두 번째 줄에 x에서 y로 최단최단경로로 이동할 때 거치는 지하철 노선의 개수를 출력한다.

세 번째 줄에 x에서 y로 이동할 때의 서로 다른 최단최단경로의 개수를 $10^9 + 9$로 나눈 나머지를 출력한다. 이동한 역의 순서가 동일하더라도 사용한 노선이 다르면 서로 다른 경로로 간주한다.

제한

$$2 \le N \le 100,000$$

$$2 \le M \le 300,000$$

$$1 \le x,\ y \le N,\ x \ne y$$

$$1 \le u,\ v \le N,\ u \ne v$$

$$1 \le w \le 200,000$$

모든 노선은 단방향이다.

예제 입력 1	예제 출력 1
5 6 1 5 1 2 3 2 4 1 1 3 2 3 5 4 4 5 2 3 2 1	6 2 1

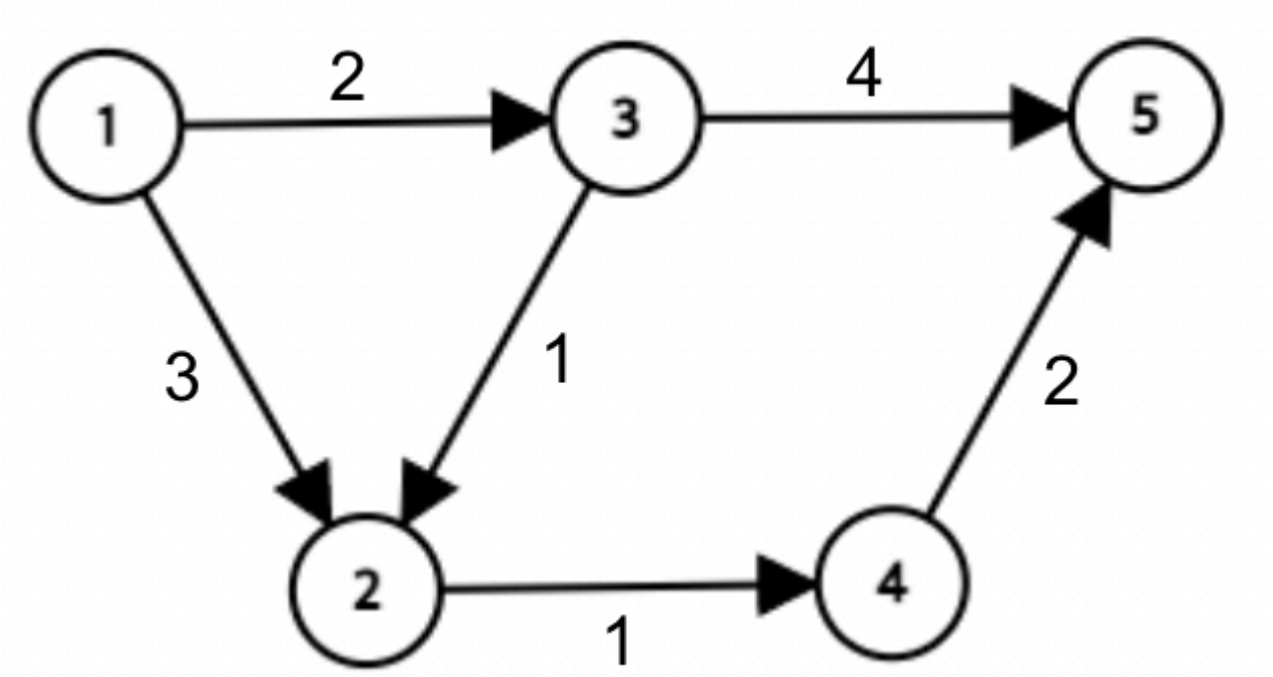
예제 입력 2	예제 출력 2
4 6 1 4 1 2 1 1 2 1 2 4 2 2 4 2 1 3 2 3 4 1	3 2 5

예제 입력 3	예제 출력 3
5 4 1 5 1 2 3	-1

2 4 1 1 3 2 3 2 1	

힌트

지하철 노선이 다음과 같고 선노가 사는 곳이 1번 역, 회사는 5번 역에 있다고 하면, 최단최단경로로 이동할 때 이동 거리는 6, 거치는 노선 수는 2, 그런 경로의 개수는 1이다.



문제 H. 난민

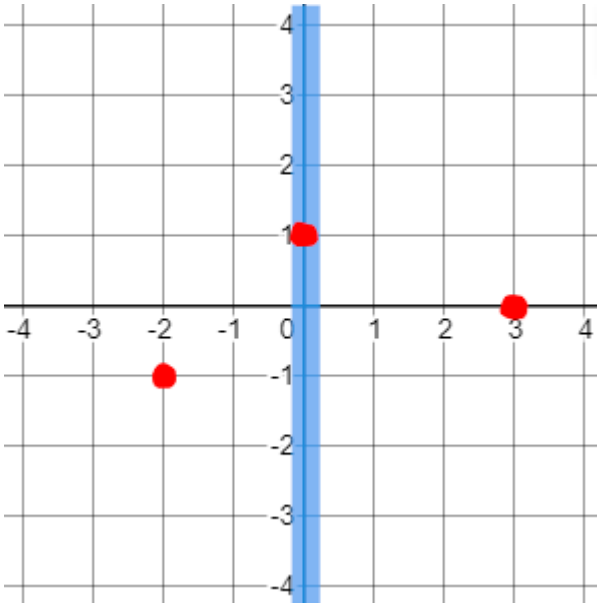
시간 제한: 1 초

메모리 제한: 1024 MB

A나라에서 내전이 일어났다!

이 때문에 많은 내전 난민들이 B나라로 오게 되었고, B나라는 이들을 위해 난민촌을 건설해 주었다.

다음은 난민촌을 좌표평면으로 나타낸 그림이다. 파란색 선은 강, 빨간색 점은 난민의 거주 위치를 나타낸다.



난민촌에는 직선 $x = 0$을 따라 흐르는 강이 하나 있다.

물을 얻기 위해서는 이 강을 이용해야만 하는데, 수질이 좋지 않아 정수시설을 설치하려고 한다.

하지만 예산 문제로 정수시설은 강의 한 지점에만 설치할 수 있기 때문에, 각 난민들이 정수시설에 도착하기 위한 이동거리의 합이 최소가 되는 위치에 정수시설을 설치하려고 한다.

지금 이 순간에도 난민들이 이주해 오고 있다. 난민촌에 사람들이 이주해 올 때마다, 정수시설을 설치해야 하는 위치와 그 때의 이동거리의 합을 갱신해 알려주자.

두 좌표 (x_1, y_1)와 (x_2, y_2)사이의 거리는 $|x_1 - x_2| + |y_1 - y_2|$로 계산한다.

입력

첫째 줄에 난민의 수 N이 주어진다.

이어서 N개의 줄에, i번째로 이주해온 난민의 위치 (x_i, y_i)가 공백으로 구분되어 주어진다.

출력

문제의 답을 공백으로 구분하여 N줄에 걸쳐 출력한다.

i번째 줄에, 1번째 부터 i번째까지 이주해온 난민들이 정수시설까지 이동하는 거리의 합이 최소가 되도록 하는 정수시설의 y좌표와, 그 때의 이동거리의 합을 공백으로 구분하여 출력한다.

만약 이를 만족하는 정수시설의 위치가 여러 곳이라면, y좌표가 가장 작은 지점을 출력한다.

제한

$1 \le N \le 10^5$

$-10^9 \le x_i, y_i \le 10^9$, 모든 좌표는 정수

예제 입력 1	예제 출력 1
3 3 0 -2 -1 0 1	0 3 -1 6 0 7

문제 I. 판치기

시간 제한: 1 초

메모리 제한: 1024 MB

판치기는 N개의 동전을 바닥에 놓고, 임의의 동전들을 뒤집는 것을 반복하여 모두 뒷면이 보이는 상태로 바꾸면 이기는 게임이다.

판치기 경력 20년에 빛나는 치훈이는 판치기 최고의 극의, "K-뒤집기"를 시전할 수 있게 되었다. "K-뒤집기"는 원하는 서로 다른 K개의 동전을 한 번에 뒤집는 능력이다.

초기 동전의 상태가 주어진다. "K-뒤집기"만 사용해 게임을 이기려면 최소 몇 번 사용해야 이길 수 있을까?

입력

첫째 줄에 N, K가 주어진다.

두번째 줄에 초기 동전의 상태를 나타내는 문자열 S가 주어진다.

S의 i번째 문자가 'H'면 i번째 동전이 앞면, 'T'면 i번째 동전이 뒷면이 보이는 상태를 나타낸다.

출력

첫째 줄에 문제의 답을 출력한다.

모두 뒷면이 보이는 상태로 바꿀 수 없다면 대신 -1을 출력한다.

제한

$$1 \le N \le 3,000$$

$$1 \le K \le N$$

예제 입력 1	예제 출력 1
5 3 HHHHH	3

예제 입력 2	예제 출력 2
3 2 THT	-1

문제 J. 사탕나무

시간 제한: 1 초

메모리 제한: 1024 MB

안즈는 사탕나무를 생일선물로 받았다.

사탕나무는 N개의 사탕을 트리 형태로 이은 것을 말한다. 각 사탕은 길이가 1인 간선으로 연결되어있고, 임의의 두 사탕 사이의 최단 경로는 유일하다.

안즈는 이 사탕나무에서 기준이 되는 사탕을 하나 골라, 그 사탕과의 최단거리가 K이하인 모든 사탕을 다

먹어버리려고 한다!

그런데 안즈는 문득 기준으로 어떤 사탕을 골라야 사탕을 가장 많이 먹을 수 있을지 궁금해졌다.

하지만 그 순간 안즈는 매우 귀찮아졌기 때문에, 여러분에게 해결을 부탁했다.

입력

첫째 줄에 N과 K가 주어진다.

이어서 N-1개의 줄에, 사탕나무의 간선을 이루는 두 사탕 번호 u, v가 공백으로 구분되어 주어진다.

주어지는 입력은 트리임이 보장된다.

출력

안즈가 먹을 수 있는 최대 사탕 개수를 출력한다.

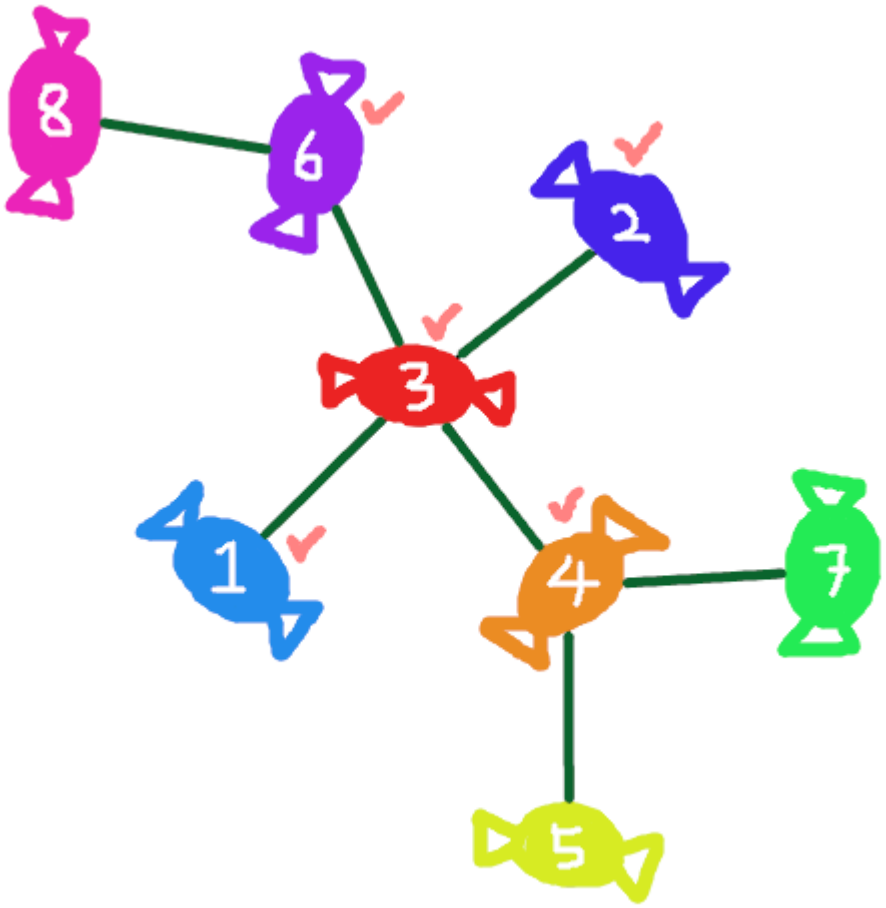
제한

$3 \le N \le 10^5$

$1 \le K \le 20$

$1 \le u,\ v \le N$

예제 입력 1	예제 출력 1
8 1 1 3 2 3 3 4 4 5 3 6 4 7 6 8	5



기준이 되는 사탕을 3번 사탕으로 정하면 총 5개의 사탕을 먹을 수 있다.

문제 K. 꿀벌 승연이

시간 제한: 1 초

메모리 제한: 1024 MB

이세계의 승연이는 꿀벌이다.

승연이가 사는 벌집은 조금 특이한 구조로 되어있다. 벌집은 N행 M열의 격자로 이루어져 있고, 각 칸은 정육각형 모양이다. 같은 행에 위치한 두 칸을 비교했을 때, 짝수 번째 열의 칸은 홀수 번째 열의 칸보다 반 칸 아래에 위치해 있다. 아래 그림은 3행 4열의 격자로 이루어진 벌집의 예시이다.

두 육각형 칸이 하나의 변을 공유하고 있다면 서로 인접하다고 한다. 육각형 칸의 각 변에는 인접한 칸으로 이동할 수 있는 문이 있는데, 벌집 내 교통 정리를 위해 각 칸에서는 아래쪽, 오른쪽 위, 오른쪽 아래 칸으로만 이동할 수 있다. 또, 벌집에는 구멍 칸이 있을 수도 있는데, 구멍 칸으로는 이동할 수 없다.

승연이는 최근 꿀벌들 사이에 급속도로 전염되고 있는 변이 바이러스를 피하기 위해 벌집콕 생활을 하고 있다. 심심함이 극에 달한 승연이는 벌집의 1행 1열 칸에서 N행 M열 칸까지 바깥으로 나가지 않고 이동하는 경로의 개수가 알고 싶어 졌다.

자가 격리 중인 승연이가 더 이상 답답하지 않도록 경로의 개수를 세서 알려주자!

입력

첫째 줄에 N, M이 공백으로 구분되어 주어진다.

다음 줄에는 구멍 칸의 개수 K가 주어진다.

이어서 K개 줄에 구멍 칸의 정보 x_i, y_i가 공백으로 구분되어 주어진다. 이는 i번째 구멍 칸이 x_i행 y_i열에 존재함을 뜻한다. 모든 구멍 칸의 위치는 서로 다르다.

1행 1열 칸 또는 N행 M열 칸이 구멍인 경우는 없음이 보장된다.

출력

벌집의 1행 1열 칸에서 N행 M열 칸으로 이동하는 경로의 개수를 $10^9 + 7$로 나눈 나머지를 출력한다.

제한

$2 \le N,\ M \le 1,000$

$$0 \le K \le \left(NM - 2,\ 10^4\right)$$

$1 \le x_i \le N$

$1 \le y_i \le M$

예제 입력 1	예제 출력 1
3 4 1 2 3	20

문제 L. Calculate! 3

시간 제한: 2 초

메모리 제한: 1024 MB

도현이는 인경호에서 즐겁게 헤엄치는 오리, 인덕이를 바라보다가 흥미로운 사실을 발견했다. 그것은 인경호 수면 위에서 인덕이가 멈추는 지점을 정점으로, 인덕이의 동선을 간선으로 나타내면 놀랍게도 트리가 그려진다는 사실이다!

이 사실에 매우 감명한 도현이는 인덕이가 그리는 트리를 노트에 옮겨 그린 다음, 인덕이가 트리의 각 간선을 헤엄칠 때 날개짓하는 횟수를 세어 해당 간선의 가중치로 적어 넣었다. 그리고 문득 인덕이가 이 트리로 뭔가 메시지를 보내는 것 같다고 생각한 도현이는, 이 트리에서 얻을 수 있는 모든 서로 다른 경로를 기록했다. 그리고 각 경로에 포함된 간선들의 가중치를 모두 XOR 연산하여 계산 결과가 같은 경로가 몇 개씩 있는지 세기 시작했다.

그런데 XOR 연산하기에 몰두한 도현이를 보고 동우의 장난기가 발동해버렸다. 동우는 도현이의 사각을 노려 간선의 가중치를 조금씩 바꾸기 시작했다.

동우: (이 간선 가중치를 바꿔볼까 ㅋㅋ)

도현: 어...? 계산 결과가 30인 경로는 분명 7개였는데... 계산 실수했나?

우둔한 도현이는 가중치가 바뀌었다는 사실도 모르고 XOR 연산에 몰두했다.

동우는 도현이가 XOR 연산을 잘 계산하고 있는 것인지 궁금했다. 하지만 XOR 연산을 할 줄 모르는 동우는 이 상황을 쿼리로 주면 결과를 알려주는 프로그램이 있으면 좋겠다고 생각했다.

이 프로그램을 구현해보자.

입력

첫째 줄에 두 개의 정수 N, M이 주어진다. N은 트리의 정점 개수, M은 쿼리의 개수이다.

다음 N-1개의 줄에는 트리에 존재하는 간선이 "a b c"의 형태로 주어진다. 이는 초기 트리의 a번 정점과 b번 정점 사이의 가중치가 c임을 나타낸다.

다음 M개의 줄에는 한 줄에 하나씩 쿼리가 주어진다. 쿼리의 입력은 다음 형태를 따른다.

1 a b c: 정점 a, b를 연결하는 간선의 가중치를 c로 바꾼다. a와 b를 연결하는 간선이 존재함이 보장된다.

2 c: XOR 연산 결과가 c인 서로 다른 경로의 개수를 출력한다.

출력

2번 쿼리가 주어질 때마다 경로 XOR 합이 c인 경로의 개수를 한 줄에 출력한다.

경로는 적어도 하나의 간선을 포함해야 하며, 역방향인 두 경로는 같은 경로로 취급한다. 즉, 정점 A에서 출발해 정점 B로 도착하는 경로와 정점 B에서 출발해 정점 A로 도착하는 두 경로는 같은 경로이므로, 1번만 센다.

제한

$3 \leq N < 100,000$

$1 \leq M \leq 200,000$

$1 \leq a, b \leq N$

$1 \leq c \leq 30$

예제 입력 1	예제 출력 1
5 7 1 2 1 1 3 2 3 4 1 3 5 3 2 1 1 3 1 1 2 1 1 5 3 2 2 3 1 1 3 3 2 2	3 4 2 3

노트

XOR 연산은 배타적 논리합의 정의를 따른다.