Problem A. 녜힁

해조는 2019년 8월 3일에 오픈베타가 열리는 대작 MMORPG '패스 오브 여로'에서 사전예약을 하려고 한다. 해조는 예쁜 두 글자 닉네임을 갖고 싶었지만 발빠른 다른 유저들이 선점하여 가질 수 없었다.

해조가 선점된 닉네임의 규칙을 살펴본 결과 특정 소설에서 순서대로 두 글자를 따온 닉네임들은 전부 등록되어 있었고 그 외의 두 글자 닉네임은 모두 만들 수 있었다. 예를 들어 소설의 내용이 '나랏말싸미듕귁에달아'이라고 해보자. 해조가 발견한 규칙에 따르면 '나미'는 이미 있는 닉네임이고, '아싸'는 언급된 글에서 '아'보다 '싸'가 앞에 있어서 소설에서 만들 수 없는 단어이므로 생성할 수 있는 닉네임이다.

해조는 만들 수 있는 두 글자 닉네임 중에서 사전 순으로 K번째로 앞서는 것을 닉네임으로 정하려고 한다. 해조는 Q개의 K를 정하여 닉네임을 구한 다음 그 중에서 가장 예쁜 것을 선택할 예정이다. 해조를 도와 소설의 내용이 주어졌을 때 만들 수 있는 닉네임 중에서 사전 순으로 K번째로 앞선 것을 구하는 프로그램을 작성하여라.

편의상 모든 글자는 1 이상 M 이하의 정수로 표현하며, 닉네임 AB가 닉네임 CD보다 사전 순으로 앞선다는 것은 $A < C$이거나, $A = C$이고 $B < D$임을 의미한다.

입력 형식

첫 번째 줄에는 소설의 길이 $N(1 \le N \le 100\,000)$, 글자의 종류 수 $M(1 \le M \le 1\,000\,000)$, 쿼리의 수 $Q(1 \le Q \le 100\,000)$가 주어진다.

두 번째 줄에는 소설의 내용에 해당하는 N개의 1 이상 M 이하의 정수가 주어진다.

세 번째 줄부터 Q개의 줄에는 가지려는 닉네임의 사전 순 번호 $K(1 \le K \le M^2)$가 주어진다.

출력 형식

Q개의 줄에 걸쳐 만들 수 있는 두 글자 닉네임 중 사전 순으로 K번째로 앞선 것을 출력한다. 앞글자와 뒷글자 사이는 공백으로 구분한다.

만약 만들 수 있는 닉네임의 개수가 K개보다 적다면 `-1 -1`을 출력한다.

앞선 쿼리는 이후의 쿼리에 영향을 주지 않는다.

예제

표준 입력(stdin)	표준 출력(stdout)
5 5 3 1 2 3 4 5 6 12 18	3 3 5 2 -1 -1

Problem B. 비트베리

비트베리는 국내 최다 사용자를 확보하고 있는 간편암호화폐 지갑이다. 비트베리의 가장 큰 특징 중 하나는 카카오 계정으로 지갑을 만들고, 전화번호로 암호화폐를 주고받을 수 있는 점이다.

페카즈는 비트베리의 특징을 이용해 자신이 보유한 다양한 종류의 암호화폐를 친구 빈센트와 상호 교환하고자 한다. 현재 페카즈의 비트베리 지갑 속에는 P개의 *비트*와 Q개의 *베리*가 들어 있다. 페카즈의 친구 빈센트는 엄청난 부자여서, 자신의 비트베리 지갑 속에 비트, 베리, 그리고 또다른 단위인 *코인*과 *비트코인*을 각각 10^{100}개씩 가지고 있다.

페카즈는 빈센트와 아래의 거래를 원하는 순서대로 원하는 만큼 반복할 수 있다.

- 빈센트에게 비트 A개를 주고 코인을 B개 받는다.
- 빈센트에게 코인 B개를 주고 비트를 A개 받는다.
- 빈센트에게 베리 C개를 주고 코인을 D개 받는다.
- 빈센트에게 코인 D개를 주고 베리를 C개 받는다.
- 빈센트에게 비트 1개와 코인 1개를 주고 비트코인을 1개 받는다.

물론 거래를 하기 위해 빈센트에게 줘야 하는 암호화폐가 부족하다면 거래를 진행할 수 없다.

페카즈는 최선의 거래를 하여 자신이 갖고 있는 비트코인의 개수를 최대화하고자 한다. 페카즈가 만들 수 있는 비트코인의 최대 개수를 출력하는 프로그램을 작성하라.

입력 형식

첫 번째 줄에 테스트 케이스의 수 T ($1 \le T \le 1\,000$)가 주어진다.

다음 T개의 줄에는 테스트 케이스가 한 줄에 하나씩 주어진다. 각 줄에는 하나의 테스트 케이스를 구성하는 여섯 개의 정수 P, Q, A, B, C, D ($0 \le P, Q \le 10\,000$, $1 \le A, B, C, D \le 10\,000$)가 공백 하나씩을 사이에 두고 주어진다.

출력 형식

각각의 테스트 케이스마다, 페카즈가 만들 수 있는 비트코인의 최대 개수를 한 개의 줄에 출력한다.

예제

표준 입력(stdin)	표준 출력(stdout)
3 2019 8 3 11 16 13 2019 7 27 2019 8 3 8 3 2019 7 29 2018	1584 1992 0

Problem C. 컨테이너

"크리컨테이너제작소"에서는 컨테이너를 제작해 파는데, 1톤짜리 컨테이너와 2톤짜리 컨테이너의 두 가지 종류를 취급한다. 이번에 컨테이너를 대량 발주받아 총 N개의 컨테이너를 제작했고, 제작된 순서대로 창고에 일렬로 늘어놓았다.

창고는 먼저 제작된 컨테이너가 먼저 출고되기 쉬운 형태인데, 안타깝게도 컨테이너가 제작된 순서와 컨테이너가 출고되어야 하는 순서는 다를 수 있다. 그래서 늘어선 컨테이너를 재배열해 컨테이너가 출고되어야 하는 순서와 맞추려고 한다. 무게가 같은 컨테이너는 서로 구별할 수 없기 때문에, 무게의 순서를 맞추면 된다.

컨테이너의 재배열에는 기계의 힘을 빌린다. 기계는 인접한 위치에 있는 컨테이너를 여럿 선택해서 순서를 뒤집을 수 있는데, 크기 문제로 최대 세 개의 컨테이너만 선택할 수 있다. 예를 들어, 선택된 컨테이너의 무게가 순서대로 [2,1]이라면 뒤집어서 [1,2]가, [2,1,1]이라면 뒤집어서 [1,1,2]가, [2,2,1]이라면 뒤집어서 [1,2,2]가 되는 식이고, 넷 이상의 컨테이너를 선택하여 뒤집을 수는 없다.

이렇게 기계를 사용하면 비용이 발생하고, 비용은 선택된 컨테이너의 무게의 합에 기계를 한 번 가동하는 데 드는 비용 C를 더한 만큼이다. 그러므로, [2,1]을 뒤집는 비용은 $3+C$, [1,1,2]을 뒤집는 비용은 $4+C$이다.

실제 예를 들어 보자. 컨테이너가 늘어선 순서대로 무게를 나열했을 때 [2,2,1,2,1,2,1]라고 하고, 출고되어야 하는 순서대로 무게를 나열하면 [1,2,1,2,1,2,2]라고 하자. 다음은 $C = 0$일 때와 $C = 2$일 때 각각의 최적의 재배열 방법이다. 선택한 컨테이너는 진하게 표시되어 있다.

$C = 0$		$C = 2$	
[2,2,1,2,1,**2**,**1**]	비용 $3+0=3$	[**2**,**2**,**1**,2,1,2,1]	비용 $5+2=7$
[2,2,1,**2**,**1**,**1**,2]	비용 $4+0=4$	[1,2,**2**,**2**,**1**,2,1]	비용 $5+2=7$
[2,**2**,**1**,**1**,1,2,2]	비용 $4+0=4$	[1,2,1,2,**2**,**2**,**1**]	비용 $5+2=7$
[**2**,**1**,1,2,1,2,2]	비용 $3+0=3$	[1,2,1,2,1,2,2]	재배열 완료
[1,2,1,2,1,2,2]	재배열 완료		
총 비용 14		총 비용 21	

둘의 방법을 서로 바꿔보면, 서로가 최적이 아니게 되는 것을 알 수 있다.

$C = 0$		$C = 2$	
[**2**,**2**,**1**,2,1,2,1]	비용 $5+0=5$	[2,2,1,2,1,**2**,**1**]	비용 $3+2=5$
[1,2,**2**,**2**,**1**,2,1]	비용 $5+0=5$	[2,2,1,**2**,**1**,**1**,2]	비용 $4+2=6$
[1,2,1,2,**2**,**2**,**1**]	비용 $5+0=5$	[2,**2**,**1**,**1**,1,2,2]	비용 $4+2=6$
[1,2,1,2,1,2,2]	재배열 완료	[**2**,**1**,1,2,1,2,2]	비용 $3+2=5$
		[1,2,1,2,1,2,2]	재배열 완료
총 비용 15		총 비용 22	

N, C, 컨테이너가 제작된 순서대로 무게를 나열한 것, 컨테이너가 출고되어야 하는 순서대로 무게를 나열한 것이 주어진다. 이 때 총 비용이 최소화 되도록 기계를 사용하여 컨테이너를 재배열하는 방법을 구하는 프로그램을 작성하라.

입력 형식

첫 번째 줄에는 제작된 컨테이너의 수를 나타내는 정수 N $(1 \le N \le 500)$과 기계를 사용하는 데 드는 기본 비용을 나타내는 정수 C $(0 \le C \le 1\,000)$가 공백 하나를 사이에 두고 주어진다.

두 번째 줄에는 1 또는 2로 이루어진 길이 N인 문자열이 주어진다. 이 문자열의 i번째 문자는 i번째로 제작된 컨테이너의 무게를 뜻한다. 이 순서는 현재 컨테이너가 늘어서 있는 순서와도 같다.

세 번째 줄에는 1 또는 2로 이루어진 길이 N인 문자열이 주어진다. 이 문자열의 i번째 문자는 i번째로 출고되어야 하는 컨테이너의 무게를 뜻한다. 이 순서대로 컨테이너가 늘어서 있도록 해야 한다.

주어진 두 문자열에서 등장하는 1의 개수는 서로 같음이 보장된다. 역시 2의 개수도 서로 같음이 보장된다.

출력 형식

첫 번째 줄에 기계를 사용한 횟수를 나타내는 정수 K를 출력한다.

다음 K개의 줄에는 기계를 사용하는 순서대로 두 정수 i, j $(1 \le i < j \le N,\ j - i \le 2)$가 공백 하나를 사이에 두고 출력되어야 한다. 이는 기계로 i번째에 있는 컨테이너에서 j번째에 있는 컨테이너까지 선택하여 위치를 뒤집었음을 나타낸다.

기계를 사용하는 데 든 총 비용이 최소가 되어야 정답으로 인정된다.

예제

표준 입력(stdin)	표준 출력(stdout)
5 2 11221 21112	2 1 3 4 5
7 0 2212121 1212122	4 6 7 4 6 2 4 1 2
7 2 2212121 1212122	3 1 3 3 5 5 7

Problem D. 룰렛

종영이가 오랜 기간 동안 준비한 "옥토끼나라 놀이공원"이 곧 개장을 앞두고 있다. 이 놀이공원에는 N개의 놀이기구가 있고, 두 놀이기구 사이를 잇는 도로들이 $N-1$개 있다. 사람들은 이 도로들을 이용해 자신이 있는 놀이기구에서 다른 원하는 놀이기구로 반드시 이동할 수 있다. 또한 이 놀이공원에는 다음에 어떤 놀이기구를 탈지 고민하는 사람들을 위해 종영이의 야심작 "룰렛 시스템"이 도입되었다.

놀이공원의 각 놀이기구 옆에는 룰렛이 하나씩 있다. 룰렛은 여러 칸으로 나뉘어 있고, 각 칸에는 현재 놀이기구와 도로로 직접 연결된 다른 놀이기구나 집이 그려져 있다. 룰렛은 아주 정확해서 어떤 그림이 나올 확률은 그 그림이 그려진 칸의 수와 비례한다. 놀이기구를 타고 나온 사람들은 이 룰렛을 돌려서 다른 놀이기구가 나오면 그 놀이기구로 이동하고, 집이 나오면 더 이상 놀이기구를 타지 않고 집으로 돌아간다.

종영이는 더 많은 수익을 얻고자 놀이기구를 하나 골라서 그 옆에 식당을 짓기로 했다. 사람들은 집으로 돌아가기 전까지는 계속해서 놀이기구를 타기 때문에, 이 놀이기구를 끝으로 더 이상 놀이기구를 타지 않고 집에 돌아가는 사람이 많을수록 식당은 더 큰 수익을 낼 수 있을 것이다. 종영이는 식당이 낼 수 있는 수익이 너무 작다면 놀이공원의 입구를 바꾸는 것까지도 고려하고 있다.

따라서 식당과 입구의 위치를 잘 정하기 위해서는 입구로 들어온 사람이 식당 옆에 있는 놀이기구를 타고 나서 집에 돌아갈 확률을 구해야 한다. 종영이를 위해 두 개의 놀이기구의 번호 S, E가 주어지면 S번 놀이기구에서 출발했을 때 마지막으로 E번 놀이기구를 타고 집으로 돌아갈 확률을 구해주자.

입력 형식

첫 줄에 놀이기구의 수 N과 쿼리의 수 Q가 주어진다. ($2 \le N \le 100\,000$, $1 \le Q \le 500\,000$)

그 후 N개의 수가 주어지는데, i번째 수 A_i는 i번 놀이기구의 룰렛의 칸 수를 의미한다. ($2 \le A_i \le 10^8$)

그 후 $N-1$개의 줄에 걸쳐 한 줄에 하나씩 u, v, p_1, p_2가 주어진다. 이는 u번 놀이기구와 v번 놀이기구를 잇는 도로가 있으며, u번 놀이기구의 룰렛에서 v번 놀이기구로 가는 결과가 나오는 칸의 수가 p_1이며, v번 놀이기구의 룰렛에서 u번 놀이기구로 가는 결과가 나오는 칸의 수가 p_2라는 뜻이다. ($1 \le p_1 < A_u$, $1 \le p_2 < A_v$)

도로 정보들로 주어진 i번째 놀이기구에 있는 룰렛의 칸의 수의 합은 A_i 미만임이 보장되며, 합을 S_i라 했을 때 i번째 놀이기구에서 집으로 돌아가는 결과가 나오는 칸의 수는 $A_i - S_i$개다.

그 후 Q개의 줄에 걸쳐 한 줄에 하나씩 쿼리에 해당하는 S와 E가 공백으로 구분되어 주어진다. ($1 \le S, E \le N$)

어떠한 S와 E를 잡아도 쿼리의 답이 $\frac{p}{q}$(p와 q는 서로소인 정수) 꼴이라는 것을 증명할 수 있으며, 이때 $p, q \not\equiv 0 \pmod{10^9+7}$임이 보장되는 입력이 주어진다.

출력 형식

i번째 답이 B_i일 때, $B_i = \frac{P_i}{Q_i}$ 꼴이라 하자. (P_i와 Q_i는 서로소인 정수)

Q개의 답을 줄바꿈으로 구분하여 출력하는데, i번째 답으로는 $V \cdot Q_i \equiv P_i \pmod{10^9+7}$을 만족하는 0 이상 10^9+6 이하의 정수 V를 출력한다. 입력 제약 조건을 만족하는 모든 입력에서 이러한 V가 유일하게 존재함을 증명할 수 있다.

예제

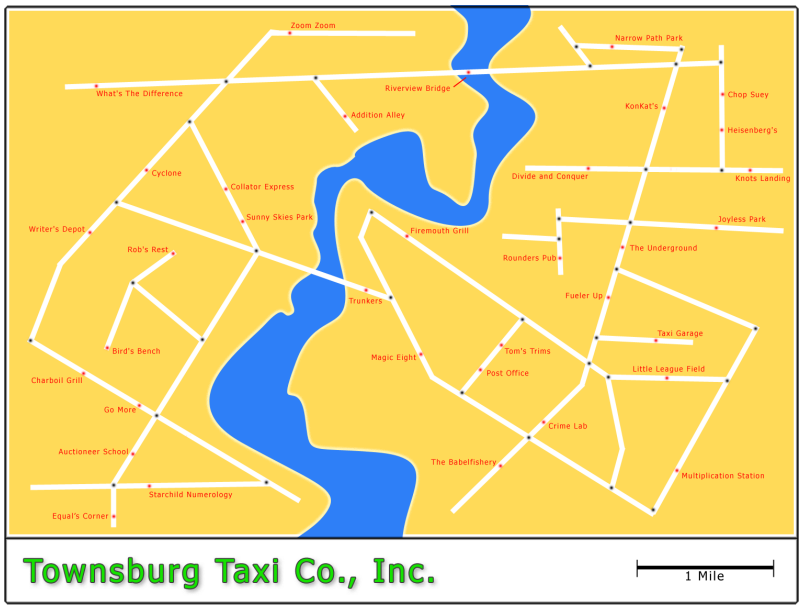
표준 입력(stdin)	표준 출력(stdout)
2 4 2 3 1 2 1 1 1 1 1 2 2 1 2 2	200000002 800000006 400000003 600000005

참고

1번 놀이기구를 탄 후 집으로 돌아갈 확률은 $\frac{1}{2}$이며, 2번 놀이기구로 이동할 확률도 $\frac{1}{2}$이다. 2번 놀이기구를 탄 후 집으로 돌아갈 확률은 $\frac{2}{3}$이며, 1번 놀이기구로 이동할 확률은 $\frac{1}{3}$이다. 1번 놀이기구에서 출발했을 때 마지막으로 1번 놀이기구를 타고 집으로 돌아갈 확률은 $\frac{1}{2} + \frac{1}{12} + \frac{1}{72} + \cdots = \frac{3}{5}$이다.

Problem E. Taxi

Taxi는 프로그래밍 언어로, 가상의 도시에서 택시가 이동하면서 승객을 태우는 과정으로 코드가 표현된다는 점이 특징이다. 이 도시의 지도는 다음과 같다.



(image from https://bigzaphod.github.io/Taxi/)

이 언어로 "Hello, world!"를 출력하는 프로그램을 작성하면 다음과 같다.

```
"Hello, World!" is waiting at the Writer's Depot.
Go to Writer's Depot: west 1st left, 2nd right, 1st left, 2nd left.
Pickup a passenger going to the Post Office.
Go to the Post Office: north 1st right, 2nd right, 1st left.
Go to the Taxi Garage: north 1st right, 1st left, 1st right.
```

택시는 항상 `Taxi Garage`에서 출발하고, 프로그램 종료 시에도 이 곳에 있어야 하며 이때 타고 있는 승객이 없어야 한다. 택시는 연료 A갤런을 채울 수 있으며, 처음 출발할 때는 연료가 가득 찬 상태이다. 각 지점에서는 특정 목적지로 가려는 승객을 태울 수 있고, 이 승객은 택시가 목적지에 도착하면 즉시 내린다. 이때 승객은 택시를 타고 이동한 거리 당 B원을 지불한다. 또한 택시는 승차 정원이 있기 때문에 승객은 최대 세 명까지만 태울 수 있다.

(원래의 설정과는 다르지만) 계산의 편의를 위해 이 도시에는 1마일 간격으로 수평 및 수직 도로가 있으며, 모든 지점은 두 도로가 만나는 지점에만 있다고 가정하자. 그러면 모든 지점은 정수 좌표로 나타낼 수 있으며, 두 지점 간의 거리는 맨해튼 거리(Manhattan distance)로 계산할 수 있다. 즉, 두 지점의 좌표가 (x_1, y_1), (x_2, y_2)일 경우 거리는 $|x_1 - x_2| + |y_1 - y_2|$이다. 단, 택시가 주행 중 특정 지점을 지나더라도 그 지점이 목적지인 승객이 내릴 수는 없으며, 정확히 목적지에 도착해야만 승객이 내릴 수 있다.

택시는 1갤런 당 C마일을 갈 수 있는데, 주행 도중 연료가 떨어지면 안 된다. 따라서 주행을 계속하기 위해 연료가 0 미만이 되기 전에 주유소에서 연료를 충전해야 한다. 입력되는 지점 중 세 곳은 주유소로, 이 곳에 도착하면 자동으로 연료를 채우며, 세 주유소의 연료 1갤런 당 가격은 다를 수 있다. 연료를 가득 채울 만큼의 돈이 있다면 그만큼을 지불하고 연료를 가득 채우고, 그렇지 않다면 가지고 있는 모든 돈을 지불하여 연료를

채운다. 연료를 가득 채우기 위해 필요한 돈이 정수로 나누어떨어지지 않는 경우 소수점 이하는 절사한다. 즉, 연료를 가득 채우기 위해 1,234.567...원이 필요한데 현재 1,234원 이상이 있을 경우 1,234원을 지불하고 연료를 가득 채운다. 만약 주유소가 목적지인 승객이 있다면 승객이 먼저 내리면서 요금을 지불한 후 연료를 채운다.

이제 이 명세를 바탕으로, 도시의 정보와 택시의 이동 경로가 주어졌을 때, 택시가 모든 규칙을 만족하면서 운행을 완료하는지를 판단하는 프로그램을 작성해보자.

입력 형식

첫 줄에 정수 A, B, C가 입력되며, 각각 택시의 연료 용량, 승객이 1마일 당 지불하는 돈, 1갤런 당 갈 수 있는 거리이다. ($1 \le A \le 100$, $1 \le B \le 100$, $1 \le C \le 100$)

다음 줄에 지점의 개수 N이 입력된다. ($4 \le N \le 100$)

이후 N줄에 걸쳐 i번째 지점의 이름 D_i와 정수 좌표 x_i, y_i가 입력된다. ($0 \le x_i, y_i \le 100$) 이름에 들어갈 수 있는 문자는 알파벳과 아포스트로피('), 공백이다. 이름은 1글자 이상 30글자 이하로, 첫 글자와 마지막 글자는 공백이 될 수 없으며, 공백이 두 번 이상 연속한 경우도 없다. 모든 지점의 이름은 다르며, 이름이 `Taxi Garage`인 지점은 항상 존재한다. 이름에 들어가는 알파벳은 대소문자를 구분한다.

이후 세 줄에 걸쳐 주유소의 이름 G_i와 갤런 당 가격 P_i가 입력된다. 세 주유소의 이름은 모두 다르며, 각각 앞에서 입력된 지점 중 하나이다. 갤런 당 가격은 1 이상 100 이하의 정수이다. `Taxi Garage`는 주유소가 될 수 없다.

다음으로 문장의 개수 K가 입력된다. ($1 \le K \le 10\,000$)

이후 K줄에 걸쳐 문장이 입력되며, 다음 두 형식 중 하나이다.

- `Go to (X).`: X로 이동한다. X는 입력된 지점 중 하나이다.
- `Pickup a passenger going to (X).`: X로 이동하는 승객을 태운다. X는 입력된 지점 중 하나로, 택시가 현재 위치한 지점과 `Taxi Garage`는 목적지가 될 수 없다.

출력 형식

규칙에 맞게 운행을 완료한 경우, 최종적으로 번 돈의 액수를 출력한다.

운행에 실패한 경우, 다음 문자열 중 하나를 출력한다.

- `OUT OF GAS`: 이동 중 연료가 떨어졌을 때
- `CAPACITY FULL`: 정원보다 많은 승객을 태우려고 할 때
- `NOT IN GARAGE`: 마지막 지점이 `Taxi Garage`가 아닐 때
- `REMAINING PASSENGER`: 마지막 지점이 `Taxi Garage`이지만 아직 내리지 못한 승객이 있을 때

예제

표준 입력(stdin)	표준 출력(stdout)
50 20 10 5 Taxi Garage 50 50 Zoom Zoom 13 66 Fueler Up 39 27 What's The Difference 95 32 Go More 2 34 Fueler Up 55 Go More 33 Zoom Zoom 17 7 Go to Go More. Pickup a passenger going to Fueler Up. Go to Zoom Zoom. Go to Fueler Up. Go to What's The Difference. Go to Go More. Go to Taxi Garage.	700
10 20 10 4 Zoom Zoom 30 40 Fueler Up 100 0 Taxi Garage 0 0 Go More 0 100 Go More 40 Zoom Zoom 50 Fueler Up 60 3 Go to Go More. Go to Fueler Up. Go to Taxi Garage.	OUT OF GAS

표준 입력(stdin)	표준 출력(stdout)
30 40 50 5 Station A 0 0 Station B 10 10 Taxi Garage 20 20 Station C 30 30 KonKat's 40 40 Station A 10 Station B 10 Station C 10 8 Go to Station A. Pickup a passenger going to KonKat's. Go to Station B. Pickup a passenger going to KonKat's. Pickup a passenger going to Station A. Go to Station C. Pickup a passenger going to KonKat's. Go to Taxi Garage.	CAPACITY FULL
30 30 30 4 Taxi Garage 1 2 A 3 4 B 5 6 C 7 8 A 10 B 20 C 30 1 Go to A.	NOT IN GARAGE

표준 입력(stdin)	표준 출력(stdout)
30 30 30 4 Taxi Garage 1 2 A 3 4 B 5 6 C 7 8 A 10 B 20 C 30 3 Go to A. Pickup a passenger going to B. Go to Taxi Garage.	REMAINING PASSENGER

참고

첫 번째 입력 데이터에 대해, 각 문장을 처리한 다음 택시의 상태는 다음과 같다. 처음에는 연료 50갤런, 돈 0원으로 시작한다.

- `Go to Go More.`: `Taxi Garage`에서 64마일 이동한다. 연료는 6.4갤런 소비한다. 주유소이지만 돈이 없으므로 연료를 채우지 않는다. (43.6갤런/0원)
- `Pickup a passenger going to Fueler Up.`: `Fueler Up`으로 가는 승객을 태운다. (43.6갤런/0원)
- `Go to Zoom Zoom.`: 43마일 이동하면서 연료는 4.3갤런 소비한다. 역시 연료를 채우지 않는다. (39.3갤런/0원)
- `Go to Fueler Up.`: 65마일 이동한다. 승객이 내리면서 2160원을 지불한다. 연료를 가득 채우기 위해 17.2*55=946원이 필요하므로 이를 지불하고 연료를 가득 채운다. (50갤런/1214원)
- `Go to What's The Difference.`: 61마일 이동한다. (43.9갤런/1214원)
- `Go to Go More.`: 95마일 이동한다. 연료를 가득 채우기 위해 15.6*33=514원(원단위 절사)이 필요하므로 이를 지불하고 연료를 가득 채운다. (50갤런/700원)
- `Go to Taxi Garage.`: 64마일 이동한다. (43.6갤런/700원)

이동 중 연료가 떨어지거나 정원을 초과한 적이 없고, 마지막으로 `Taxi Garage`에 있으며 아직 내리지 않은 승객이 없으므로 번 돈의 액수인 700을 출력하면 된다.

Problem F. 공의 합집합

3차원 좌표공간에 n개의 공(ball)이 있다. 이 중 i $(1 \le i \le n)$번째 공은 겉표면이 중심이 $(x_i, 0, 0)$이고 반지름은 r_i인 구 형태이며, 이 공이 차지하는 영역은 $S_i = \{(x, y, z) \in \mathbb{R}^3 : (x - x_i)^2 + y^2 + z^2 \le {r_i}^2\}$이다.

모든 공의 합집합 $S_1 \cup S_2 \cup \cdots \cup S_n$의 부피를 구하는 프로그램을 작성하라.

입력 형식

첫 번째 줄에 공의 개수 n $(1 \le n \le 300\,000)$이 주어진다.

다음 n개 줄에는 공의 정보가 한 줄에 하나씩 주어지는데, 이 중 i $(1 \le i \le n)$번째 줄에는 두 개의 정수 x_i $(0 \le x_i \le 10^6)$와 r_i $(1 \le r_i \le 10^6)$가 공백 하나를 사이로 두고 주어진다.

출력 형식

모든 공의 합집합 $S_1 \cup S_2 \cup \cdots \cup S_n$의 부피는 $\frac{p}{q}\pi$ (단, p와 q는 서로소인 두 자연수)로 나타낼 수 있다. 이때, 첫 번째 줄에 $V \cdot q \equiv p \pmod{10^9 + 7}$을 만족하는 0 이상 $10^9 + 6$ 이하의 정수 V를 출력하라. 입력 제약 조건을 만족하는 모든 입력에서 이러한 V가 유일하게 존재함을 증명할 수 있다.

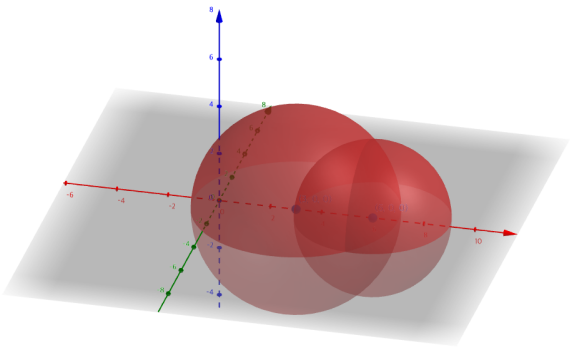
예제

표준 입력(stdin)	표준 출력(stdout)
1 1 4	333333421
2 3 4 6 3	100

참고

첫 번째 예제 설명: 반지름이 4인 공의 부피는 $\frac{4}{3} \cdot 4^3 \cdot \pi = \frac{256}{3}\pi$이다. $V \cdot 3 \equiv 256 \pmod{10^9 + 7}$의 해를 구하면 $V \equiv 333\,333\,421 \pmod{10^9 + 7}$이다.

두 번째 예제 설명: 그림은 아래와 같으며, 합집합의 부피는 100π이다.

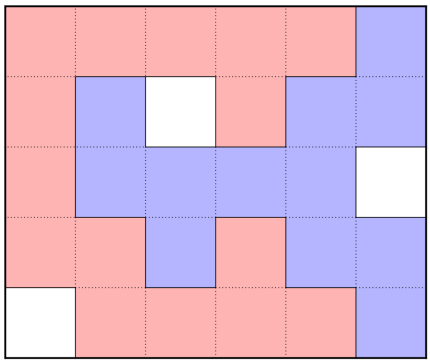


Problem G. 땅다람쥐

근우는 땅다람쥐를 연구하고 있다. 근우가 연구하는 땅다람쥐는 흔히 알려진 땅다람쥐와는 달리, 단 하나의 커다란 굴을 파고 살아가며 아주 독립적인 동물이다. 편의상 땅이 격자로 나뉘어 있다고 가정할 때, 땅다람쥐가 파는 굴은 다음과 같은 특징을 가진다.

- 두 개의 칸이 상하좌우로 인접해 있으면 이동할 수 있다고 할 때, 땅다람쥐 한 마리의 굴에서 어떤 두 개의 칸을 고르더라도 같은 칸을 두 번 이상 방문하거나 굴을 벗어나지 않고 둘 사이를 이동하는 경로가 유일하게 존재한다.
- 여러 마리의 땅다람쥐가 있을 때, 어떤 두 땅다람쥐의 굴도 같은 칸을 공유하지 않는다.

근우는 땅에서 두 개의 칸을 골라 땅다람쥐를 한 마리씩 올려놓았다. 두 땅다람쥐는 각자 처음 놓인 칸에서 굴을 파기 시작해 인접한 칸들로 확장시켜 나갔다. 그러나 더 이상 확장할 수 없는 상태가 되었음에도 불구하고 어떤 굴에도 포함되지 않은 칸들이 남아 있었다.



두 땅다람쥐가 굴을 판 뒤

근우는 땅다람쥐들이 계획적으로 굴을 판다면 모든 칸이 두 땅다람쥐의 굴 중 하나에 포함될 수 있을 거라고 생각했다. 근우의 가설이 맞을지 확인해 보자.

입력 형식

첫 줄에 땅의 세로 길이와 가로 길이를 의미하는 정수 N과 $M(2 \le N, M \le 100)$이 주어진다.

두 번째 줄에 첫 번째 땅다람쥐를 올려놓은 위치를 의미하는 정수 X와 $Y(1 \le X \le N, 1 \le Y \le M)$가 주어진다. 이는 X번째 줄 Y번째 칸에 땅다람쥐를 올려놓았음을 의미한다.

세 번째 줄에 두 번째 땅다람쥐를 올려놓은 위치가 같은 형식으로 주어진다. 두 땅다람쥐의 위치는 서로 다르다.

출력 형식

만약 모든 칸을 두 땅다람쥐의 굴에 포함시킬 수 있다면 첫 줄에 `YES`를 출력한다.

다음 N개의 줄에 걸쳐 굴을 판 결과를 각 줄에 M글자의 문자열로 출력한다. 첫 번째 땅다람쥐의 굴에 속하는 칸은 `#`, 두 번째 땅다람쥐의 굴에 속하는 칸은 `.`으로 나타낸다. 가능한 답이 여러 개 있다면 그중 아무것이나 출력한다.

만약 모든 칸을 포함시키는 것이 불가능하다면 첫 줄에 `NO`를 출력한다.

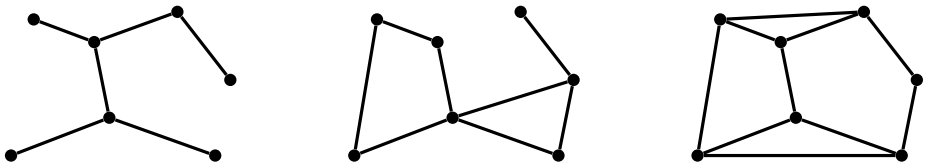
예제

표준 입력(stdin)	표준 출력(stdout)
3 4 2 1 3 3	YES #### #..# ##.#
2 4 1 2 1 3	NO

Problem H. 죽은 선인장의 사회

최근, 애완 그래프를 찾는 사람이 많아지고 있다. 이런 사람들에게는 그래프 중에서 특히 귀여운 외형을 가진 *트리*(Tree)에 대한 수요가 많고, 수요의 대부분을 차지한다. 쿠는 예전부터 애완 그래프를 파는 업자였는데, 그래프 중에서도 특히 *캑터스*(Cactus)만 취급하는 별종이었다.

트리와 캑터스는 그래프의 종류 중 하나인데, 트리는 모든 정점이 연결되어 있으면서, 단순 사이클이 만들어지지 않는 그래프를 뜻하고, 캑터스는 모든 정점이 연결되어 있으면서, 단순 사이클이 만들어지지만, 서로 다른 두 단순 사이클에 동시에 들어가는 간선이 없는 그래프를 뜻한다. 예를 들어 보자.

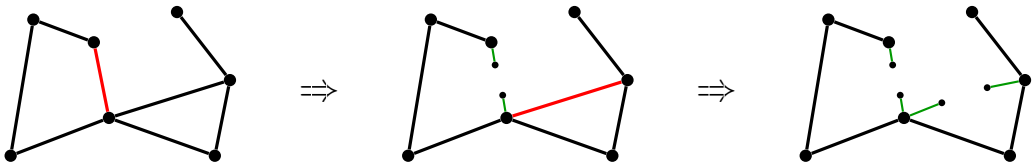


왼쪽에서 첫 번째 그림은 트리, 두 번째 그림은 캑터스, 세 번째 그림은 트리도 캑터스도 아닌 그래프를 나타낸다. 단순 사이클이 많아질수록 끔찍해지는 외형을 볼 수 있을 것이다.

쿠는 지난 25년간 애정으로 캑터스만을 다뤄왔지만, 이제는 지쳐버렸다. 애완 그래프에 관심을 가지는 사람들이 많아져 가게에 손님이 많아졌어도, 다들 캑터스는 모양이 징그럽다면서 떠나버리기 일쑤라 여전히 장사는 되지 않고, 생활은 점점 힘들어지고 있다. 그래서 쿠는 이제 냉엄한 현실과 타협하여 캑터스 판매업을 포기하고 새로운 시작을 하기로 했다. 바로 캑터스의 몇몇 간선을 잘라내 트리로 만들어 파는 트리 판매업을 하는 것이다.

그래프는 아주 섬세한 생물이기 때문에, 간선의 중간을 그냥 자르면 잘라낸 곳 부터 썩어들어간다. 그러나 간선이 정점과 연결된 두 끝부분을 깨끗하게 잘라내면, 잘라낸 곳의 상처가 아물고 끝난다. 그래서 간선은 전체를 잘라내야 그래프가 썩지 않는다. 또한, 간선을 잘라내 연결이 끊어져 그래프가 여러 부분으로 나뉘면, 나뉜 그래프는 모두 썩어버린다. 그래서 모든 정점이 연결되어 있도록 철저히 관리 하면서 간선을 잘라내야 한다.

간선을 잘라내면, 그 간선이 연결되어 있던 두 정점에는 상처가 하나씩 생긴다. 그래프의 억센 생명력은 이 상처를 곧바로 자가치유하여, 새로운 간선 하나와 그 끝의 새 정점 하나를 자라나게 한다. 각 간선과 정점에는 *회복 수치*가 있는데, 잘라낸 간선의 회복 수치가 R_e, 상처가 난 정점의 회복 수치가 R_v일 때, 상처 부위에서는 길이가 $R_e + R_v$인 간선과 그 끝의 정점이 자라서 상처가 아문다.



위의 그림에서 빨간색 간선이 잘라내는 간선이고, 초록색 간선이 새롭게 자란 간선이다. 이렇게 캑터스에 있는 모든 단순 사이클에서 간선을 정확히 하나씩 잘라내면 트리가 된다. 사람들은 트리의 *지름*이 작은 트리를 좋아하는 경향이 있는데, 트리의 지름이란 정점과 정점 사이의 최단 거리 중 최댓값을 말한다. 쿠는 잘 팔리는 트리를 만들기 위해 만들어지는 트리의 지름이 최소화되도록 캑터스의 간선을 잘라 낼 것이다. 캑터스가 주어질 때, 쿠가 만들어낼 트리의 지름이 어떻게 될지 구하는 프로그램을 작성하라.

입력 형식

첫 번째 줄에는 캑터스의 정점 수를 나타내는 정수 N ($3 \leq N \leq 100\,000$)과 간선 수를 나타내는 정수 M ($N \leq M \leq 150\,000$)이 공백 하나를 사이에 두고 주어진다. 각 정점에는 1에서 N사이의 번호가 붙고, 각 간선에도 1에서 M사이의 번호가 붙는다.

두 번째 줄에는 각 정점의 회복 수치를 나타내는 N개의 정수 $R_{v,1}, R_{v,2}, \cdots, R_{v,N}$ ($0 \leq R_{v,i} \leq 10^9$)가 공백 하나로 구분되어 주어진다. $R_{v,i}$는 i번 정점의 회복 수치를 나타낸다.

다음 M개의 줄의 i번째 줄에는 i번 간선의 정보를 나타내는 네 정수 $A_i, B_i, L_i, R_{e,i}$ ($1 \leq A_i, B_i \leq N$, $A_i \neq B_i$, $1 \leq L_i, R_{e,i} \leq 10^9$)가 공백 하나로 구분되어 주어진다. A_i와 B_i는 i번 간선이 잇고 있는 두 정점의 번호를 나타내며, L_i는 i번 간선의 길이, $R_{e,i}$는 i번 간선의 회복 수치를 나타낸다. 주어진 그래프는 캑터스인 것이 보장되며, 같은 두 정점을 연결하는 간선이 여러 번 주어지지 않는다.

출력 형식

첫 번째 줄에 주어진 캑터스의 간선을 적절히 잘라 트리를 만들었을 때 가능한 최소의 지름을 가지는 트리의 지름을 출력한다.

예제

표준 입력(stdin)	표준 출력(stdout)
3 3 1 2 3 3 1 2 3 1 2 1 2 2 3 3 1	10
5 6 1 2 3 4 5 1 2 6 1 1 3 5 4 2 3 4 2 1 4 3 6 1 5 2 3 4 5 1 5	22

Problem I. 미로

민제는 미로 게임장을 운영하고 있다. 게임장에 있는 미로는 N행 M열 크기의 격자 모양이고, 각 칸에는 상, 하, 좌, 우 네 방향 중 하나가 쓰여 있다. 미로 안을 돌아다닐 때는 네 방향으로 이동할 수 있으나, 자신이 위치한 칸에 적혀 있는 방향이나 미로를 벗어나는 방향으로는 이동할 수 없다.

민제는 미로에 참가자가 한 번 들어갔다 나올 때마다 미로를 청소하는데, 매번 $N \times M$개의 칸을 모두 청소하다가 골병이 들고 말았다. 민제는 참가자가 돌아다니지 않은 칸은 굳이 청소할 필요가 없다는 걸 깨닫고, 앞으로는 참가자가 지나갔을 가능성이 있는 칸만 청소하기로 했다.

각 참가자가 미로 탐험을 시작한 위치와 끝낸 위치가 주어질 때, 민제가 청소해야 할 칸의 수를 구하는 프로그램을 작성하여라.

입력 형식

첫 번째 줄에는 미로의 크기 $N, M (1 \le N, M \le 1\,000)$, 쿼리의 수 $Q (1 \le Q \le 300\,000)$가 주어진다.

두 번째 줄부터 N개의 줄에는 미로의 형태에 해당하는 M개의 문자가 주어진다. 각 문자는 `U`, `D`, `L`, `R` 중 하나이며 이는 미로의 해당 칸에서 각각 위, 아래, 왼쪽, 오른쪽으로 이동할 수 없음을 의미한다.

$N+2$번째 줄부터 Q개의 줄에는 시작점의 행 번호와 열 번호, 도착점의 행 번호와 열 번호가 주어진다. 시작점과 도착점이 같을 수 있으며 참가자가 도착점에 처음 도달한 다음에도 미로 탐험을 계속 할 수 있음에 유의하여라.

출력 형식

Q개의 줄에 걸쳐 각 줄에 쿼리에 대한 정답을 출력하는데, 시작점에서 도착점까지 이동할 수 없으면 0을 출력하고 그렇지 않으면 민제가 청소해야 할 칸의 수를 출력한다.

예제

표준 입력(stdin)	표준 출력(stdout)
5 5 5 DDDDD RDDDL RRDLL RUUUL UUUUU 1 1 5 5 2 2 5 5 3 3 5 5 4 4 5 5 5 5 5 5	0 14 20 14 5

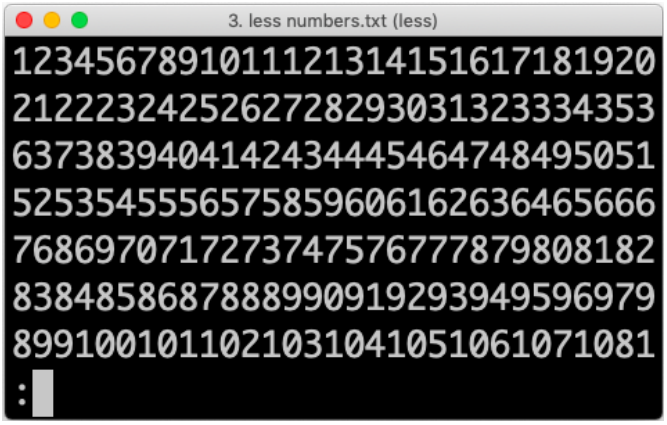
Problem J. 격자 속의 숫자

모든 자연수를 아래처럼 빈칸 없이 왼쪽부터 오른쪽으로 차례대로 1부터 나열한 문자열 S를 생각하자.

$$12345678910111213141516171819202122232425\cdots$$

S와 관련된 문제는 이 세상에 이미 많이 있다. 최근의 어떤 필기시험에서는 S의 $2 \cdot 10^6$번째 자리를 손으로 구하는 문제가 출제된 바 있다.

문자열 S는 가로로 무한히 길쭉하지만, 이 문제가 인쇄된 종이나 이 문제가 띄워져 있는 모니터는 가로 폭이 고정되어 있다. 만약 S를 모든 숫자의 글자 폭이 동일한 글꼴로 인쇄하거나 적절한 편집기로 읽는다면, 아래와 같이 모든 줄에 M개의 숫자가 순서대로 나타나 있는 형태가 될 것이다.



이러한 방식으로 S의 첫 $10^9 \cdot M$글자를 $10^9 \times M$ 크기의 행렬 A에 순서대로 썼다고 하자. 위의 그림에는 $M = 31$일 때 A의 1행부터 7행까지가 나타나 있다.

2차원 행렬이 있으니, 왠지 2차원 부분합을 구하고 싶지 않은가? 행렬 A의 직사각형 영역 $[r_1, r_2] \times [c_1, c_2]$이 주어질 때, 이 영역에 포함된 숫자들의 합, 즉

$$\sum_{i=r_1}^{r_2} \sum_{j=c_1}^{c_2} A_{i,j}$$

의 값을 구하는 프로그램을 작성하라. 한 번만 구하기는 아쉬우므로, Q번 구하도록 하자.

입력 형식

첫 번째 줄에 정수 M $(1 \le M \le 100\,000)$과 질의의 수 Q $(1 \le Q \le 1\,000)$가 주어진다.

다음 Q개의 줄에는 네 개의 정수 r_1, c_1, r_2, c_2 $(1 \le r_1 \le r_2 \le 10^9,\ 1 \le c_1 \le c_2 \le M)$이 주어진다.

출력 형식

각각의 질의마다 $\sum_{i=r_1}^{r_2} \sum_{j=c_1}^{c_2} A_{i,j}$의 값을 한 줄에 하나씩 순서대로 출력한다.

예제

표준 입력(stdin)	표준 출력(stdout)
31 3 1 1 7 31 4 4 5 5 5 25 6 30	946 23 79

Problem K. 옥토끼는 통신교육을 풀어라!!

UCPC World Finals 2020을 준비하는 옥토끼는 여름학교에 가기 위하여 통신교육 문제를 푼다. 그러나 악덕 조교 tncks0121은 옥토끼가 오랫동안 문제를 풀지 않으면 '옥통풀'을 외치며 독촉한다.

옥토끼는 N개의 문제를 모두 풀어야 하며 각 문제를 푸는데 T_i분이 걸린다. 옥토끼는 멀티태스킹 능력이 발달하여 한 번에 동시에 두 개의 문제를 풀 수 있다. 한 번 풀기 시작한 문제는 도중에 풀이를 중단하지 않는다. 옥토끼가 항상 문제를 풀고 있을 필요는 없으며 문제를 푸는 순서에는 제약이 없다.

tncks0121은 옥토끼가 문제를 해결한 시점만 볼 수 있을 뿐, 옥토끼가 지금 문제를 풀고 있는지 쉬고 있는지 모른다. 문제 하나를 풀고 나서 다음 문제를 풀기까지 걸리는 시간이 길어지면 tncks0121의 독촉이 심해지기 때문에, 옥토끼는 이 간격의 최댓값이 최소가 되도록 계획을 세워 모든 문제를 풀려고 한다. 옥토끼를 도와 문제를 푸는 전략을 짜는 프로그램을 작성하여라. tncks0121은 통신교육이 시작되자마자 독촉을 시작하므로, 옥토끼가 첫 문제를 푸는 데 걸리는 시간도 고려해야 함에 유의하여라.

입력 형식

첫 번째 줄에는 문제의 수 N $(1 \le N \le 100\,000)$이 주어진다.

두 번째 줄에는 옥토끼가 각 문제를 푸는데 걸리는 시간 T_i $(1 \le T_i \le 10^9)$가 주어진다.

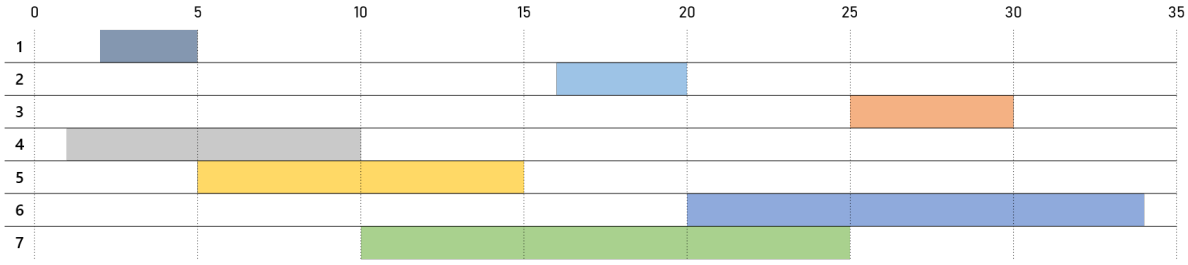
출력 형식

첫 번째 줄에 옥토끼가 문제 하나를 풀고 나서 다음 문제를 풀기까지 걸리는 최대 시간의 최솟값을 출력한다.

예제

표준 입력(stdin)	표준 출력(stdout)
7 3 4 5 9 10 14 15	5

참고



옥토끼가 위와 같이 문제를 풀면 tncks0121은 옥토끼가 5, 10, 15, 20, 25, 30, 34분에 문제를 풀었으므로 최대 5분 동안 쉰 것으로 간주한다.

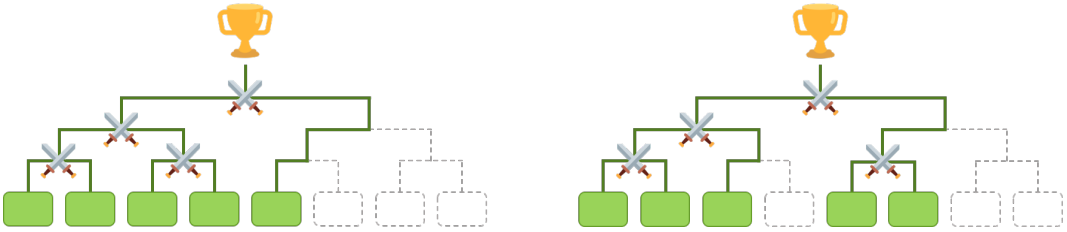
Problem L. 대진표

수빈이는 예전부터 UCPC의 대회 형식이 ICPC와 같다는 것이 마음에 들지 않았다. 그래서 전대프연의 회장이 되자마자 UCPC를 ICPC와 차별화된 토너먼트 방식의 대회로 바꾸겠다고 선언했다.

수빈이가 바꾼 새로운 UCPC의 진행 방식은 다음과 같다.

1. 참가한 팀의 수보다 크거나 같은 가장 작은 2의 거듭제곱 꼴의 수를 찾고, 그 수만큼 빈 슬롯을 일렬로 나열한다.
2. 참가한 팀들을 슬롯들에 적절히 배정한다. 이때 두 개 이상의 팀을 같은 슬롯에 배정할 수는 없다.
3. 슬롯들을 앞에서부터 두 개씩 짝짓는다. 만약 짝지어진 두 슬롯 모두에 팀이 배정되어 있다면 두 팀이 경기를 치르고, 패배한 팀의 슬롯이 삭제된다. 그렇지 않다면 경기를 치르지 않고, 비어 있는 슬롯 하나가 삭제된다.
4. 마지막 한 팀이 남을 때까지 3번 과정을 반복한다. 마지막으로 남은 팀이 우승 팀이 된다.

수빈이는 팀을 배정할 때 가장 앞의 슬롯부터 차례대로 한 팀씩 배정하려고 했으나, 이러면 공평하지 않은 대진표가 만들어진다는 것을 발견했다. 예를 들어 5개의 팀이 대회에 참가했을 때, 첫 번째 슬롯에 배정된 팀은 우승하기 위해 세 번의 경기를 치러야 하는 반면 5번째 슬롯에 배정된 팀은 한 경기만 이기면 우승을 차지할 수 있다.



공평하지 않은 대진표와 공평한 대진표

수빈이는 우승하기 위해 가장 많은 경기를 치러야 하는 팀과 가장 적은 경기를 치러야 하는 팀의 경기 수가 많아야 한 경기 차이가 나도록 팀을 배정하려고 한다. 또한 그런 배치가 여러 개 있다면 팀이 배정된 슬롯들의 번호를 **내림차순으로** 정렬했을 때 이 수열이 사전순으로 가장 앞서는 배치를 고르려고 한다. 즉 마지막 팀이 배정된 슬롯의 번호가 작을수록 좋은 배치이다.

그러나 대회를 열기 위해서는 대진표를 짜는 것 외에도 할 일이 너무나도 많다! UCPC가 원활히 진행될 수 있도록 바쁜 수빈이 대신 좋은 대진표를 만들어 주자.

입력 형식

첫 줄에 대회에 참가한 팀의 수를 의미하는 정수 $N(1 \le N \le 1\,000\,000)$이 주어진다.

출력 형식

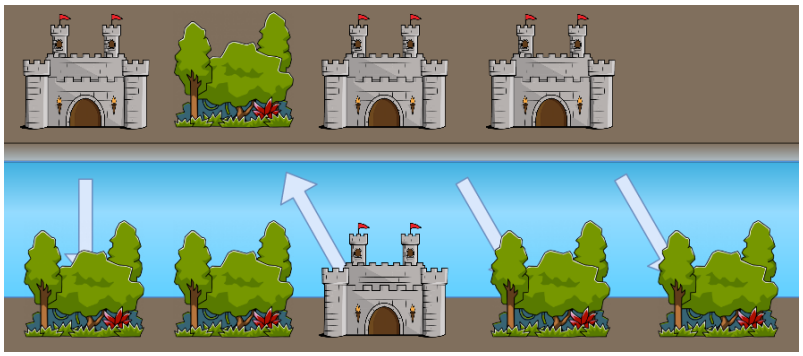
첫 줄에 수빈이가 원하는 대로 팀을 배정한 결과를 나타내는 문자열을 출력한다. 문자열의 길이는 슬롯의 개수와 같아야 하며, 팀이 배정된 슬롯은 #, 비어 있는 슬롯은 . 로 나타낸다.

예제

표준 입력(stdin)	표준 출력(stdout)
3	###.
5	###.##..
9	###.##..####....

Problem M. Uncrossed Knights' Tour

중세 왕국 수비니움은 일직선으로 뻗은 강의 변에 N개의 성과 M개의 자연경관이 있는 것으로 유명하다. 수비니움 강은 매우 올곧아서 다른 왕국에 비해 미터법이 빠르게 발달했는데 국왕은 서쪽 국경의 좌표를 1로 두고 동쪽 국경의 좌표를 10^9으로 두어 나머지 지역의 좌표를 계산하였다. 강을 기준으로 남쪽인지 북쪽인지에 따라, 그리고 지역의 좌표에 따라 수비니움 왕국은 `U1 - U1000000000`과 `D1 - D1000000000`의 주소를 매겼다.



N명의 기사들은 여름 휴가철이 되어 강 건너에 있는 자연경관으로 여행을 가려고 한다. 기사들은 자신의 성에서 원하는 여행지까지 배를 타고 곧바로 건너가는데, 기사들은 곧게 뻗은 강만큼이나 직설적인 성격을 갖고 있어 일직선으로 강을 건너간다. 기사들은 다른 기사들로부터 방해받지 않고 여행하기를 원하므로 기사들은 서로 다른 지역으로 여행을 가야 하며 기사들의 여행경로는 교차하면 안된다.

어떤 기사들은 북쪽에 살고 있고 어떤 기사들은 남쪽에 살고 있어서 자칫 잘못하면 경로가 꼬일 수 있다. N명의 기사들이 모두 건너편의 자연경관으로 떠날 수 있는 방법이 있는지 찾고, 있다면 기사들이 갈 여행지를 배정하는 프로그램을 작성하여라.

입력 형식

입력의 첫 번째 줄에는 기사의 수 N, 자연경관의 수 M이 주어진다. $(1 \le N \le 10^4, 1 \le M \le 10^6)$

입력의 두 번째 줄에는 1, 2, ..., N번째 기사가 살고 있는 성의 주소가 주어진다.

입력의 세 번째 줄에는 1, 2, ..., M번째 자연경관의 주소가 주어진다.

주소는 `[방향][좌표]` $(1 \le$좌표$\le 10^9)$ 형태로 주어지며 모든 기사와 자연경관은 서로 다른 곳에 있다.

출력 형식

만약 N명의 기사에게 여행지를 배정할 수 없다면 -1을 출력한다.

만약 여행지를 배정할 수 있다면 1, 2, ..., N번째 기사가 가야 할 자연경관의 번호를 출력한다.

가능한 답이 여러 개이면 그중 아무거나 출력한다.

예제

표준 입력(stdin)	표준 출력(stdout)
4 5 U1 D3 U3 U4 D1 D2 U2 D4 D5	1 3 4 5