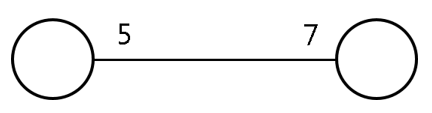
**(중상) [기출P-0070] 조약돌 게임**

|  |  |
| --- | --- |
| 시간 제한 | **30**개의 Test Input 입력시 C/C++ **1**초 / Java **1.5**초 |
| 메모리 제한 | Stack : **5** Mbytes  /  Total : **256** Mbytes |
| 제출횟수 제한 | **10** 회 |
| 코드길이 제한 | **128** Kbytes |
| 채점 | 답안을 제출하면 제공된 Sample input에 대한 결과를 판정해서 실시간으로 알려주며  그 의미는 다음과 같다.  **Submit**: **Sample input**에 대하여 1개 이상 정답을 도출한 경우 (Pass를 의미하지는 않음)  **Fail**: **Sample input**에 대하여 정답을 전혀 도출하지 못한 경우  ※ Sample input 파일은 문제 지문 바로 아래에서 다운로드 받을 수 있음  ※ 최종결과는 별도의 채점용 **Test input**으로 판단하므로 시험중 결과와 다를 수 있음 |

조약돌 N 개와 연결선 M 개가 있는 게임이 있다. 아래 그림과 같이 한 연결선은 서로 다른 두 개의 조약돌을 연결한다.   
연결선에는 두 개의 점수가 있어서 각각 한쪽 조약돌에 붙어 있다. 아래의 경우 점수는 5점과 7점이며 5점은 왼쪽 조약돌에, 7점은 오른쪽 조약돌에 붙어 있다.



게임은 혼자서 하는 것이며, 조약돌을 하나씩 가져가는 방식으로 진행된다. 한 조약돌을 가져가면 그 조약돌에 붙어 있는 연결선들에서 해당 조약돌에 붙어 있는 점수들이 모두 합산된다.   
점수를 가져온 연결선들은 모두 없어진다. 따라서, 위의 그림에서 왼쪽 조약돌을 먼저 가져가는 경우 연결선에서 5점이 합산되고 7점은 사라지게 된다.

철수가 게임을 하려고 조약돌들과 연결선들을 보니 다음의 조건이 성립함을 알게 되었다.

(조건)

한 조약돌에서 시작해서 연결선을 따라 움직일 때, 낮은 점수를 먼저 보고 높은 점수를 보게 되는 (위의 그림에서 왼쪽 조약돌에서 오른쪽 조약돌로 움직이는 것을 말함) 것을 반복해서 원래 조약돌로 돌아올 수 있는 경우는 없다.

위와 같은 게임에서 **받을 수 있는 최대의 점수**와, 그 때 **조약돌을 가져가는 순서**를 계산하는 프로그램을 작성하라.

**[입력]**

입력의 제일 첫째 줄에는 파일에 포함된 케이스의 수 T가 주어진다.   
각 케이스의 첫째 줄에는 조약돌의 개수 N, 연결선의 개수 M, 출력을 위한 정수 K가 주어진다. (2 ≤ N ≤ 100,000, 1 ≤ M ≤ 200,000, 1 ≤ K ≤ N)   
조약돌들은 1 번부터 N 번까지 번호가 붙어 있다. 다음 M 개의 줄에는 연결선에 대한 정보가 4개의 정수로 주어진다.   
주어진 정보가 a b p q 라면, a 번 조약돌과 b 번 조약돌을 연결선이 연결하며, a 번 쪽 점수는 p, b 번 쪽 점수는 q 라는 뜻이다.   
두 점수 p 와 q 가 같을 수 있다. 모든 점수는 1 이상 109 이하의 정수이다.   
단 하나의 조약돌에 연결선이 붙어 있는 경우는 없다. 하지만, 동일한 쌍의 조약돌을 두 개 이상의 연결선이 연결할 수는 있다.

**[출력]**

각 테스트 케이스의 답을 순서대로 표준출력으로 출력하며, 각 케이스마다 줄의 시작에 “#x”를 출력하여야 한다. 이때 x는 케이스의 번호이다.   
같은 줄에, 얻을 수 있는 최대 점수를 출력하고, 또 다음과 같이 **정의**되는 조약돌의 번호를 출력한다.

**\* 정의**

최대 점수를 얻을 수 있도록 조약돌을 가져가는 순서는 여러가지가 있을 수 있는데, 그들 중 **사전 순으로 가장 빠른 것에서 K번째로 등장하는 조약돌의 번호**.

**\* 참고**

조약돌 번호의 서로 다른 순서 두 개 X, Y가 있을 때, 사전순으로 X가 Y보다 빠르다는 뜻은 X와 Y를 앞에서부터 보았을 때 최초로 다른 값에서 그 값이 작은 경우를 말한다.

즉, 예를 들면, [3, 5, 2, 4, 1] 과 [3, 5, 1, 4, 2] 를 사전순으로 비교하면 두번째 것이 더 빠르다.

**[입출력 예]**

(입력)

2  
4 3 2  
1 2 2 1  
2 3 3 4  
3 4 5 4  
2 1 2  
1 2 5 7

(출력)

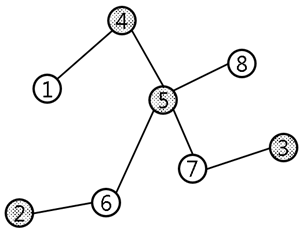
#1 11 3  
#2 7 1

(sample\_input.txt에 대한 출력)  
#1 11 1  
#2 11 2  
#3 414 9  
#4 398 9  
#5 394 2  
#6 193723 21  
#7 203040 80  
#8 46303730 134  
#9 1996702640 751  
#10 132985900835033 71476

**(중상) [기출P-0055] 고양이**

|  |  |
| --- | --- |
| 시간 제한 | **30** 개의 Test Input 입력시 C/C++ **1**초, Java **1.5**초 |
| 메모리 제한 | Stack : **5** Mbytes  /  Total : **256** Mbytes |
| 코드길이 제한 | **128** Kbytes |
| 채점 | 답안을 제출하면 Test Input에 대한 결과를 판정해서 실시간으로 알려준다.  모든 케이스에 대해 정답일 경우 **Pass**, 그 외의 경우 **Fail** 이다. |

어떤 도시에 N개의 집이 있고 두 집을 양방향으로 연결하는 길이 N-1 개 있다. 집들은 1번부터 N번까지 번호가 붙어 있다. 임의의 집에서 다른 집으로 길들을 이용해서 갈 수 있는 방법이 존재한다. N개의 집들 중 일부에는 고양이가 한 마리씩 살고 있을 수 있다. 고양이들이 밤에 싸우는 일이 잡아서 큰 문제가 되었다. 도시에서는 길들 중 일부를 끊어서 고양이들이 싸우지 않도록 만들려고 한다. 고양이가 싸우지 않는 조건은 길들을 끊은 이후, 각 고양이가 자신이 살고 있는 집에서 출발하여 길들을 이용해서 갈수 있는 다른 집에 살고 있는 고양이가 없어야 한다. 아래 예를 보자.



집들 중 고양이가 살고 있는 집은 짙게 표시하였다.

모든 길이 연결된 상태에서 4,5번 집을 잇는 길, 5,6번을 잇는 길, 5,7번을 잇는 길을 끊는다면 고양이들은 싸우지 않는다.

또한, 모든 길이 연결된 상태에서 4,5번 집을 잇는 길, 6,2번을 잇는 길, 7,3번을 잇는 길을 끊어도 고양이들은 싸우지 않는다.

물론 모든 길을 다 끊는 경우는 당연히 아무 고양이도 싸우지 않게 될 것이다.

집들의 연결 상태와 고양이들이 살고 있는지 여부를 입력으로 받아 어떤 고양이도 싸우지 않도록 **길을 끊는** **방법의 개수**를 출력하는 프로그램을 작성하라.

**[입력]**

입력 파일의 제일 첫째 줄에는 파일에 포함된 케이스의 수 T가 주어진다. 단, T ≤ 30이다. 케이스의 첫 줄에 집의 개수 N이 주어진다. (1 ≤ N ≤ 300,000) 다음 줄에 N개의 0/1 값이 주어진다. 이 값들은 집 번호의 순서대로 각 집에 고양이가 살고 있는지 여부를 표시한 것이다. 값이 0이면 고양이가 살고 있지 않다는 뜻이며, 1이면 고양이가 살고 있다는 뜻이다. 다음 N-1개의 줄에 길들에 대한 정보가 두 집의 번호로 주어진다.

**[출력]**

각 테스트 케이스의 답을 순서대로 표준출력으로 출력하며, 각 케이스마다 줄의 시작에 “#x”를 출력하여야 한다. 이때 x는 케이스의 번호이다. 같은 줄에 길을 끊는 방법의 개수를 1,000,000,007로 나눈 나머지를 출력한다.

**[입출력 예]**(입력)

2 ← 2 test cases in total

8 ← 1st case

0 1 1 1 1 0 0 0

1 4

5 4

8 5

5 7

2 6

5 6

7 3

3 ← 2nd case

1 0 1

1 2

2 3

(출력)

#1 36

#2 3   
  
(sample\_input.txt 에 대한 출력)   
#1 1   
#2 1   
#3 757665211   
#4 336766282   
#5 177226326   
#6 945696062   
#7 263749668   
#8 174768395   
#9 288058885   
#10 339931503   
#11 198177204   
#12 464334604   
#13 33626811   
#14 952805906   
#15 962498739