**(중상) [기출P-0053] 공주의 결혼식**

|  |  |
| --- | --- |
| 시간 제한 | **20** 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. |

고대 왕국의 공주가 결혼할 나이가 되어, 공주와 결혼하려는 기사들이 모여들었다. 기사들이 서로 싸우고 있었기 때문에 왕은 아이디어를 냈다.   
기사들은 백마를 탄 기사들 W 명과 흑마를 탄 기사들 B 명이 있었는데, 이들 중 두명을 무작위로 뽑는 것을 반복하기로 했다.

여기서 무작위라고 하는 것은 모든 기사가 두명에 포함될 확률이 동일하다는 것이다.

다음 규칙에 따라 정해진 기사들은 결혼을 포기하고 집으로 돌아가야 한다.

1. 뽑힌 두 명이 같은 색의 말을 타고 있으면 두 명 중 한명이 집에 가고, 다른 한 명은 남아서 다음 번 뽑기의 대상이 된다.   
2. 뽑힌 두 명이 다른 색의 말을 타고 있으면 두 명 모두 집에 가야 한다.

이 규칙을 기사가 한명이 남거나 없을 때까지 반복하고, 마지막에 남는 기사는 공주와 결혼한다.   
마지막에 남는 기사가 없는 경우 공주는 내년에 결혼을 할 것이다.

백마를 탄 기사의 흑마를 탄 기사의 수를 입력 받아, 백마를 탄 기사가 공주와 결혼할 확률, 흑마를 탄 기사가 공주와 결혼할 확률, 그리고 공주가 내년으로 결혼을 미룰 확률을 계산하는 프로그램을 작성하라.

**[입력]**   
첫 줄에 테스트 케이스의 수 T가 주어진다. 각 케이스는 단 한 줄이며 백마를 탄 기사의 수 W와 흑마를 탄 기사의 수 B가 주어진다. (0 ≤ W, B ≤1,000, 단, W+B>0)

**[출력]**   
각 테스트 케이스의 답을 순서대로 표준출력으로 출력하며, 각 케이스마다 줄의 시작에 “#x”를 출력하여야 한다. 이때 x는 케이스의 번호이다.

같은 줄에, 백마를 탄 기사가 공주와 결혼할 확률, 흑마를 탄 기사가 공주와 결혼할 확률, 그리고 공주가 내년으로 결혼을 미룰 확률을 출력한다.

**[주의]**

본 문제는 실수형 변수의 유효숫자 오류가 발생할 수 있다. 따라서 **오차가 발생하였다 하더라도 그 오차가 10-6 이하이면 정답 처리**될 것이다.

소수점 이하 출력의 자릿수는 상관이 없다.

**[입출력 예]**   
(입력)   
3                                              ← 3 test cases in total   
0 1                                           ← 1st case   
1 1                                           ← 2nd case   
2 2                                           ← 3rd case

(출력)   
#1 0.0000000 1.0000000 0.0000000   
#2 0.0000000 0.0000000 1.0000000   
#3 0.1111111 0.1111111 0.7777778

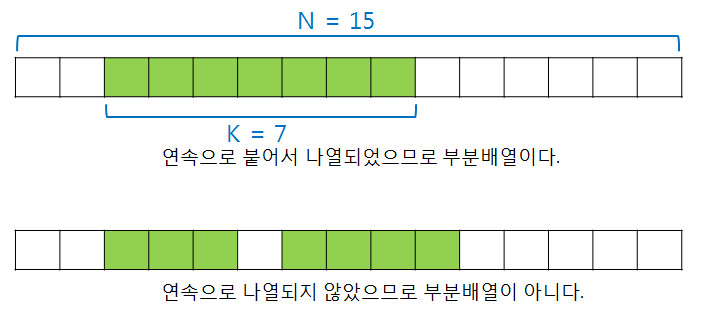
['출력 예'에 대한 설명]   
1번 케이스의 경우 #1 0.0 1.0 0.0 으로 출력해도 정답으로 처리된다.

하지만 3번 케이스의 경우 #3 0.11 0.11 0.78 로 출력하면 허용 오차범위를 초과하므로 오답 처리된다.   
  
  
(sample\_input.txt 에 대한 출력)   
#1 0.0000000000000 1.0000000000000 0.0000000000000   
#2 1.0000000000000 0.0000000000000 0.0000000000000   
#3 0.1111111111111 0.1111111111111 0.7777777777778   
#4 0.7572152553516 0.0291820777528 0.2136026668955   
#5 0.5040124813478 0.1570242759882 0.3389632426640   
#6 0.2903611284817 0.2903611284817 0.4192777430366   
#7 0.9987688563035 0.0000015012242 0.0012296424722   
#8 0.2081372482660 0.4596145098515 0.3322482418825   
#9 0.2944700221868 0.3557400418610 0.3497899359522   
#10 0.3248951733138 0.3248951733138 0.3502096533725

**(중상) [기출P-0049] 부분배열의 중앙값**

|  |  |
| --- | --- |
| 시간 제한 | **55** 개의 Test Input 입력시 C/C++ **1.5**초, Java **2**초 |
| 메모리 제한 | Stack : **1** 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. |

길이가 N 인 정수로 이루어진 배열이 있다. 이 배열 안에서 **연속으로 붙어서 나열된 K 개의 원소**를 취한 것을 길이가 K인 부분배열이라고 정의하도록 하자. 아래 그림은 N = 15 이고 K = 7 인 경우 부분배열을 취하는 예시이다.



이렇게 부분배열을 취한다고 가정하면, 길이가 N 인 배열의 부분배열 중 크기가 K 인 것은 총 N - K + 1 개가 존재한다.

이 경우 길이가 K 인 N - K + 1 개의 부분배열의 중앙값(Median)들 중 가장 큰 값을 출력하는 프로그램을 작성하라.

K 는 홀수이므로 중앙값은 항상 자연수가 됨을 유의하라.

**[제한조건]**

1. 배열의 길이 **N 은 1 이상 100000(십만) 이하의 자연수**이다.

2. 부분배열의 길이 **K 는 1 이상 N 이하의 홀수**이다.

3. 배열의 원소는 **1 이상 1000000000(십억) 이하의 자연수**이다.

4. 중앙값(median)이란 어떤 주어진 값들을 크기의 순서대로 정렬했을 때 가장 중앙에 위치하는 값을 의미한다.    
   예를 들어 1, 2, 100의 세 값이 있을 때, 2가 가장 중앙에 있기 때문에 2가 중앙값이다.

**[입력]**

맨 처음 테스트 케이스의 개수 T가 주어지며, 그 다음 줄부터 T 개의 테스트 케이스가 주어진다. 각 테스트 케이스는 두 줄로 이루어져 있으며 첫 줄에는 배열의 크기 N 과 부분배열의 크기 K 가 공백으로 구분되어 차례대로 주어지고, 그 다음 줄에는 N개의 배열의 원소가 공백으로 구분되어 차례대로 주어진다.

**[출력]**

각각의 테스트 케이스에 대하여 #x (x는 테스트 케이스 번호, 1부터 시작) 을 출력하고 공백을 하나 둔 다음, 길이가 K인 부분 배열의 중앙값들 중 가장 큰 값을 자연수로 출력한다.

**[입출력 예]**

(입력)

3  
5 1  
3 6 2 4 5  
10 3  
1 7 9 5 4 8 8 6 5 1  
10 5  
9 1 2 10 2 9 1 2 3 4

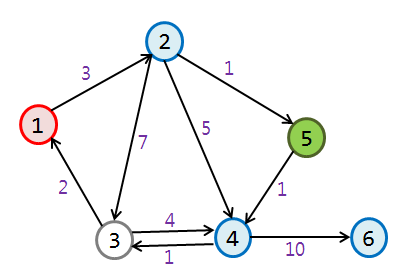
(출력)  
#1 6  
#2 8  
#3 3

(sample\_input.txt에 대한 출력)   
#1 9   
#2 4   
#3 7   
#4 703   
#5 591   
#6 525   
#7 431   
#8 465   
#9 991031   
#10 626001   
#11 517580   
#12 511587   
#13 517558   
#14 503422045   
#15 503613079

**(중상) [기출P-0019] 일방통행**

|  |  |
| --- | --- |
| 시간 제한 | **20**개의 Test Input 입력시 C/C++ **1**초 / Java **1.5**초 |
| 메모리 제한 | Stack : **5** Mbytes  /  Total : **256** Mbytes |
| 제출횟수 제한 | **제한없음** |
| 코드길이 제한 | **128** KB |
| 채점 | 답안을 제출하면 제공된 Sample input 에 대한 결과를 판정해서 실시간으로 알려주며  그 의미는 다음과 같다.  **Submit**: Sample input에 대하여 1개 이상 정답  **Fail** : Sample input에 대하여 오답 또는 런타임 에러, Time out 등  (※ sample input 파일 다운로드는 문제 지문 바로 아래에 있습니다.)  **최종결과는 별도의 채점용 Test input으로 진행되므로 시험 중 판정 결과와 다를 수 있다.** |
| 평가기준 | 정답 처리된 Test Input의 개수, 실행 속도, 메모리 사용량, 구현 코드 |

어느 구역에 N개의 지점과, 임의의 두 지점을 잇는 M개의 **일방통행 도로**로 구성된 도로망이 있다.  
이 도로망의 지점은 적색지점(R), 녹색지점(G), 청색지점(B), 백색지점(W) 이 존재하며 N 개의 지점이  
각각 이 4 종류의 지점 중 하나로 이루어져 있다. 또한 지점간의 구분을 위하여 각 지점에 1번 부터 N 번까지의  
번호가 매겨져 있다. 또한 M 개의 일방통행 도로는 각각 고유의 가중치(도로의 길이) 를 가지고 있다.  
이해를 돕기 위하여 아래에 N = 6, M = 9 인 경우의 예시 중 하나를 그림으로 표현하였다.



현재 1번 지점에 있는 당신은 갑자기 급한 용무가 생겨 신속하게 N번 지점으로 이동해야 한다.  
당신에게는 현재 적색 통행증과 청색 통행증이 각각 한장씩 있으며, 본 도로망의 법규를 지켜야 한다.

도로망의 법규는 아래와 같다.

 1. 통행증은 적색과 청색 두 가지 종류가 있으며 각각 최대 1장씩만 가질 수 있다.

 2. 적색 지점을 떠나 다른 곳으로 이동하고자 한다면, 떠나기 전에 적색 통행증을 제출해야 한다.

 3. 청색 지점을 떠나 다른 곳으로 이동하고자 한다면, 떠나기 전에 청색 통행증을 제출해야 한다.

 4. 백색 지점에 도달하면 적색 통행증과 청색 통행증을 1장씩 받을 수 있다.

    하지만 이미 같은 색 통행증을 가지고 있다면, 법규 1에 의하여 추가로 통행증을 받을 수는 **없다**.

 5. 녹색 지점은 통행증 제출 의무 및 수령 행위가 없으며 자유로이 진입과 진출이 가능하다.

위와 같은 도로망의 법규를 지킨다면, 도로망 내에서 얼마든지 이동이 가능하다고 한다. 이 때, 1 번지점부터 N 번지점까지 이동할 때의 그 최단 거리를 알아내어 출력하는 프로그램을 작성하라.

경우에 따라서는 법규의 제한 등으로 1 번지점부터 N 번지점까지 갈 수 없는 상황이 발생할 수 있다. 그럴 경우 최단 거리 대신 -1 을 출력한다.

위의 법규에서, 통행증 제출하는 시점은 현 지점을 **떠날 때** 이므로 현 지점에 도착해 그냥 머물러 있고 다른 지점으로 가지 않는다면 통행증을 제출할 필요가 없음에 유의하라.

**[제한조건]**1. 지점의 개수 **N 은 2 이상 100000(십만) 이하의 자연수**이다.  
2. 도로의 개수 **M 은 1 이상 300000 이하의 자연수**이며, N\*(N-1) 개를 넘지 않는다.

3. 모든 도로는 일방통행이며 역방향으로 이동할 수 없다.

4. 임의의 두 지점을 잇는 방향이 같은 도로는 최대 1개만 존재할 수 있다.

5. 각 도로의 가중치(경로길이) 는 **1 이상 1000000000(10억) 이하의 자연수**이다.

**[입력]**  
첫 줄에 테스트케이스 개수 T가 주어지며, 그 다음 줄부터 T개의 테스트 케이스가 주어진다.  
각 케이스의 첫 줄에는 지점의 개수 N 과 일방통행 도로의 개수 M 이 공백으로 구분되어 주어진다.  
그 다음 줄에 길이가 N인 문자열이 주어지는데, 문자열의 i번째 문자는 i번 지점의 정보를 알파벳으로 표현한 것으로서  
R은 적색지점, G는 녹색지점, B는 청색지점, W는 백색지점임을 뜻한다. (1 <= i <= N)  
그 다음 줄부터 M개의 줄에 걸쳐 일방통행 도로에 대한 정보가 도로 하나 당 한 줄에 표현된다.  
각 줄은 3 개의 자연수가 공백으로 구분되어 주어지며 첫 번째 자연수는 그 도로의 시작지점 번호, 두 번째 자연수는 그 도로의 도착지점 번호, 세 번째 자연수는 그 도로의 거리(경로길이)를 뜻한다.

**[출력]**  
테스트 케이스 하나당 한줄씩 출력한다.   
각 테스트 케이스에 대하여 #x (x는 테스트케이스 번호, 1부터 시작) 을 출력하고 공백을 하나 둔 다음,  
주어진 조건을 만족하면서 1번 지점에서 N번 지점으로 가는 최단 거리를 출력한다.  
만약 1번 지점에서 N번 지점으로 가는 경로가 존재하지 않으면 최단 거리 대신 -1을 출력한다.

**[입출력 예]**  
(입력)  
3  
6 9  
RBWBGB  
1 2 3  
2 5 1  
3 1 2  
2 3 7  
2 4 5  
5 4 1  
4 3 1  
4 6 10  
3 4 4  
3 2  
RBB  
1 2 10  
2 3 10  
5 5  
BRWBG  
4 5 3  
1 2 1  
3 2 6  
2 3 5  
2 4 2

(출력)

#1 24  
#2 20  
#3 17

(sample\_input.txt에 대한 출력)

#1 -1  
#2 11544  
#3 98884  
#4 68379  
#5 143176  
#6 196480  
#7 -1  
#8 1000000000  
#9 1000000000  
#10 -1

※ Java의 경우 Scanner 를 사용하면 실행속도가 느려질 수 있음에 유의하시오.

※ C++ 의 경우 std::cin 을 사용하려 한다면 std::ios::sync\_with\_stdio(false) 를 설정하고

   cin, cout 을 scanf, printf 와 섞어 사용하지 마시오.