# K진 트리

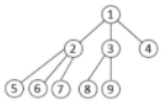
| 시간 제한 | 메모리 제한 | 제출 | 정답 | 맞은 사람 | 정답 비율 |
|---|---|---|---|---|---|
| 1 초 | 256 MB | 3255 | 821 | 620 | 25.769% |

## 문제

각 노드가 자식을 최대 K개 가질 수 있는 트리를 K진 트리라고 한다. 총 N개의 노드로 이루어져 있는 K진 트리가 주어진다.

트리는 "적은 에너지" 방법을 이용해서 만든다. "적은 에너지" 방법이란, 이전 깊이를 모두 채운 경우에만, 새로운 깊이를 만드는 것이고, 이 새로운 깊이의 노드는 가장 왼쪽부터 차례대로 추가 한다.

아래 그림은 노드 9개로 이루어져 있는 3진 트리이다.




노드의 개수 N과 K가 주어졌을 때, 두 노드 x와 y 사이의 거리를 구하는 프로그램을 작성하시오.

## 입력

첫째 줄에 N ($1 \le N \le 10^{15}$)과 K ($1 \le K \le 1\,000$), 그리고 거리를 구해야 하는 노드 쌍의 개수 Q ($1 \le Q \le 100\,000$)가 주어진다.

다음 Q개 줄에는 거리를 구해야 하는 두 노드 x와 y가 주어진다. ($1 \le x, y \le N, x \ne y$)

## 출력

총 Q개의 줄을 출력한다. 각 줄에는 입력으로 주어진 두 노드 사이의 거리를 출력한다.

### 예제 입력 1 복사

```
7 2 3
1 2
2 1
4 7
```

### 예제 출력 1 복사

```
1
1
4
```

### 예제 입력 2 복사

```
9 3 3
8 9
5 7
8 4
```

### 예제 출력 2 복사

```
2
2
3
```

# 풀이

1. 두 노드 a, b가 주어졌을 때, 거리를 구하세요.
   => 공통 조상 까지의 거리의 합을 통해 구할 수 있다.

2. LCA에서 공통조상을 찾기 위해서는 각 노드의 depth를 구해놓고, 그 비교를 통해
   공통 조상을 찾게 된다.

3. " 이 새로운 깊이의 노드는 가장 왼쪽부터 차례대로 추가 한다. " 문제에 제시 된 이 문장으로

완전 K진 트리임을 알 수 있다.

4. 완전 K진 트리에서 번호가 높음 => 최소한 레벨이 같거나 더 높음

2진 트리에서 부모를 구하는 공식

x = x / 2

K진 트리에서 부모를 구하는 공식

ex) 3진

5 -> (( 5 - 2 ) / K) + 1 = 2
6 -> (( 6 - 2 ) / K) + 1 = 2
7 -> (( 7 - 2 ) / K) + 1 = 2

=> ((x - 2) / K ) + 1

두 노드를 비교하여 같지 않을 때( 공통 조상을 못 찾았을 때 )

두 숫자를 비교하여 큰 숫자 ( 레벨이 더 크다 ) 를 공식에 의해 부모 노드로 옮긴다.

공통 조상을 만났을 때 까지 연산 횟수를 출력하여 답을 도출한다.

# Error

K == 1 일 경우 공통 조상을 찾기 위해 N번 실행해야 하여 시간초과가 발생한다.

이 경우는 두 수의 차이가 답이므로 예외처리 해준다.

```
if(K == 1) {
  System.out.println(Math.abs(x - y));
  continue;
}
```

# 코드

```
package Algo_Study_BOJ;

import java.io.BufferedReader;
import java.io.InputStreamReader;
```

```java
import java.util.StringTokenizer;

public class Main_K진트리 {
  public static void main(String[] args) throws Exception {
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

    StringTokenizer str = new StringTokenizer(br.readLine());

    long N = Long.parseLong(str.nextToken());
    int K = Integer.parseInt(str.nextToken());
    int Q = Integer.parseInt(str.nextToken());

    for(int i = 0; i < Q; i++) {

      str = new StringTokenizer(br.readLine());

      long x = Long.parseLong(str.nextToken());
      long y = Long.parseLong(str.nextToken());

      if(K == 1) {
        System.out.println(Math.abs(x - y));
        continue;
      }

      int cnt = 0;

      while(x != y) {

        if(x > y) {
          x = ( x - 2 ) / K + 1;
        }
        else {
          y = ( y - 2) / K + 1;
        }
        cnt++;
      }
      System.out.println(cnt);
    }
  }
}
```