#15591 - MooTube

문제

농부 존은 남는 시간에 MooTube라 불리는 동영상 공유 서비스를 만들었다. MooTube에서 농부 존의 소들은 재밌는 동영상들을 서로 공유할 수 있다. 소들은 MooTube에 1부터 N까지 번호가 붙여진 N (1 ≤ N ≤ 5,000)개의 동영상을 이미 올려 놓았다. 하지만, 존은 아직 어떻게 하면 소들이 그들이 좋아할 만한 새 동영상을 찾을 수 있을지 괜찮은 방법을 떠올리지 못했다.

농부 존은 모든 MooTube 동영상에 대해 "연관 동영상" 리스트를 만들기로 했다. 이렇게 하면 소들은 지금 보고 있는 동영상과 연관성이 높은 동영상을 추천 받을 수 있을 것이다.

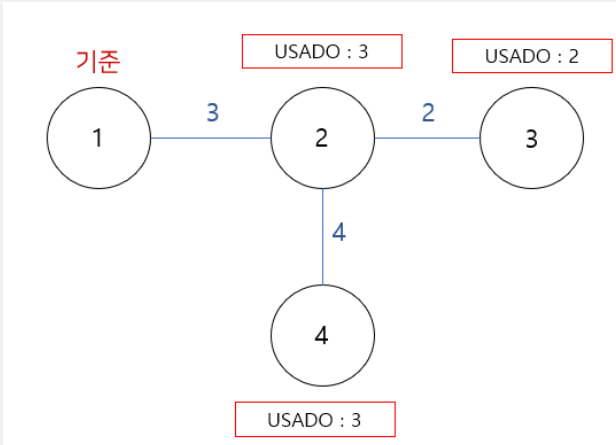
존은 두 동영상이 서로 얼마나 가까운 지를 측정하는 단위인 "USADO"를 만들었다. 존은 N-1개의 동영상 쌍을 골라서 직접 두 쌍의 USADO를 계산했다. 그 다음에 존은 이 동영상들을 네트워크 구조로 바꿔서, 각 동영상을 정점으로 나타내기로 했다. 또 존은 동영상들의 연결 구조를 서로 연결되어 있는 N-1개의 동영상 쌍으로 나타내었다. 좀 더 쉽게 말해서, 존은 N-1개의 동영상 쌍을 골라서 어떤 동영상에서 다른 동영상으로 가는 경로가 반드시 하나 존재하도록 했다. 존은 임의의 두 쌍 사이의 동영상의 USADO를 그 경로의 모든 연결들의 USADO 중 최솟값으로 하기로 했다.

존은 어떤 주어진 MooTube 동영상에 대해, 값 K를 정해서 그 동영상과 USADO가 K 이상인 모든 동영상이 추천되도록 할 것이다. 하지만 존은 너무 많은 동영상이 추천되면 소들이 일하는 것이 방해될까 봐 걱정하고 있다! 그래서 그는 K를 적절한 값으로 결정하려고 한다. 농부 존은 어떤 K 값에 대한 추천 동영상의 개수를 묻는 질문 여러 개에 당신이 대답해주기를 바란다.

N개의 동영상에서 두 쌍씩, 총 N-1개의 동영상 쌍을 골라, 두 동영상 간의 USADO를 계산한다.

각 동영상을 노드로 표현을 했을 경우, 어떤 동영상에서 다른 동영상으로 가는 경로가 반드시 하나 존재하도록 동영상 쌍을 골랐으며, 임의의 두 개의 동영상의 USADO는 그 경로에 있는 모든 연결들의 USADO 중 최솟값이 된다.

예시



어떤 주어진 동영상에 대해, 값 K를 정하여 USADO가 K이상인 동영상들이 몇개인지 구하는 문제.

입력

N(동영상 개수) Q(Q개의 질문)

p(동영상1) q동영상2 초기USADO

...

r(K값) 기준동영상

...

(1 ≤ N, Q ≤ 5000)

(1 ≤ p, q≤ N, 1 ≤ r≤ 1,000,000,000)

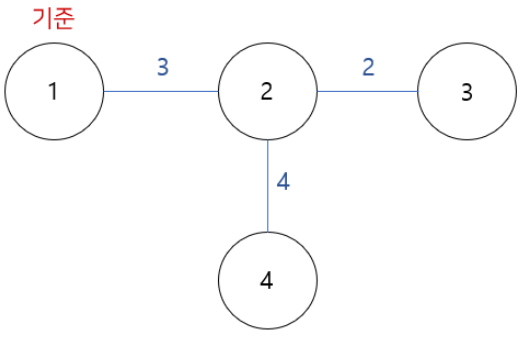
문제 분석

- 각 동영상의 USADO는, 기준 동영상과 목적지 동영상까지의 경로에 있는 USADO의 최솟값이 됨

  ⇒ 그래프 탐색(DFS 또는 BFS)

풀이 방법

기준 동영상 노드로부터 그래프 탐색



- 각 경로에 존재하는 엣지의 최솟값과 현재 방문할 노드와의 엣지값을 비교하여 최솟값 경신 및 각 노드의 USADO를 최솟값으로 부여

- 만약 경로 탐색 중, 최솟값이 이미 K보다 작은 경우, 그 경로에 대해서는 더이상 탐색을 진행하지 않음.

구현

- DFS로 구현
  - ~~Queue 이용하여 BFS 이용하였더니 메모리 초과 나옴. ⇒ 재귀함수 이용하여 DFS 구현~~
  - 오개념: 재귀함수 역시 내부적으로 스택을 사용하기 때문에 재귀를 이용했다고 해서 메모리가 크게 줄지는 않을 것으로 예상. 다만 BFS로 구현할 때, Queue에 불필요한 요소들을 push해서 문제가 발생한 것으로 추정.
- 노드 방문 여부를 판단하는 marking 배열 이용

코드

```
#include <iostream>
#include <vector>
using namespace std;

void initUsados();
void findVideos(pair<int,int> v, int min);

int N,Q,r,cnt;
int v1,v2, u;
vector<vector<pair<int,int>>> videoG;
vector<int> usados;

int main()
{
    int i;

    // graph init
    cin >> N >> Q;

    for(i=0;i<=N;i++){
        videoG.push_back(vector<pair<int,int>>());
    }

    for(i=0;i<N-1;i++){
        cin >> v1 >> v2 >> u ;
        videoG[v1].push_back(pair<int,int>(v2,u));
        videoG[v2].push_back(pair<int,int>(v1,u));
    }


    // find related videos
```

```
    for(i=0;i<Q;i++){
        cin >> r >> v1;
        cnt =0;
        initUsados();

        findVideos(pair<int,int>(v1,-1),-1);

        cout<<cnt<<endl;
    }

    return 0;
}

void initUsados(){
    int j;
    usados.clear();
    for(j=0;j<=N;j++){
        usados.push_back(0);
    }
}

void findVideos(pair<int,int> v, int min){

    if(usados[v.first]!=0){
        return;
    }

    if(min==-1){
        min=v.second;
    }else{
        if(v.second<min)
            min = v.second;
    }

    usados[v.first] = min;
    if(usados[v.first]>=r){
        cnt++;
    }else{
        if(min!=-1) return;
    }

    vector<pair<int,int>>:: iterator it;
    for(it=videoG[v.first].begin(); it != videoG[v.first].end();it++){
        findVideos((*it),min);
    }
}
```

실패 - 메모리 부족

- BFS 시 Queue에 불필요한 요소들 push (ex. 이미 조회한 동영상 등)
- 자료형으로 vector와 list 둘다 사용 ⇒ vector만 사용
- 함수의 변수로 그래프, 마킹배열을 넣음

- 인자들이 함수 스택공간에 복사되기 때문에, 메모리를 많이 차지하게 될 수 밖에 없음.

⇒ 전역변수로 설정