# 백준 1240번 - 노드사이의 거리

| 시간제한 | 메모리 제한 | 제출 | 정답 | 맞은 사람 | 정답 비율 |
|---|---|---|---|---|---|
| 2초 | 128MB | 1832 | 969 | 762 | 53.586% |

## 문제

N(2≤N≤1,000)개의 노드로 이루어진 트리가 주어지고 M(M≤1,000)개의 두 노드 쌍을 입력받을 때 두 노드 사이의 거리를 출력하라.

## 입력

첫째 줄에 노드의 개수 N이 입력되고 다음 N-1개의 줄에 트리 상에 연결된 두 점과 거리(10,000 이하의 정수)를 입력받는다. 그 다음 줄에는 거리를 알고 싶은 M개의 노드 쌍이 한 줄에 한 쌍씩 입력된다.

## 출력

M개의 줄에 차례대로 입력받은 두 노드 사이의 거리를 출력한다.

## 예제 입력1

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

## 예제 출력1

```
2
7
```

# 출처

출처

# 알고리즘 분류

- 그래프 이론
- 그래프 탐색
- 트리
- 너비 우선 탐색
- 깊이 우선 탐색

# 접근 방법

노드의 개수가 최대 $10^3$ 이하이고 Query의 수가 $10^3$ 인것을 고려한다면, 충분히 $O(NM)$의 시간 복잡도를 가지는 Naive한 탐색으로 문제를 해결할 수 있다는 것을 알 수 있다.

또한, 주어지는 edge는 $N-1$ 이고 트리 형태의 입력만 주어진다는 정보를 병합하면, Tree상에 존재하는 모든 노드들은 Connected되어 있다는 것을 알 수 있다. 또한, 모든 노드들이 Connected되기 위하여 필요한 최소 Edge의 개수가 $N-1$ 이므로 불필요한 Edge가 추가되므로서 Cycle이 발생하는 경우는 없다고 볼 수 있다.

주어진 정보들을 이용하여, 기본적인 그래프 탐색 알고리즘으로 문제를 해결할 수 있다.

# 소스코드

```cpp
#define FASTIO cin.tie(0)->sync_with_stdio(false), cout.tie(0)
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
int N, M;
ll ans;
bool visited[1001] = {false, };
vector<pair<int, int>> adj[1001];
void DFS(int u, const int target, ll acSum){
    if(u == target){
        ans = acSum;
        return;
    }
    for(auto [next, cost] : adj[u]){
        if(visited[next]) continue;
        visited[next] = true;
        DFS(next, target, acSum + cost);
    }
}
int main(void){
```

```
    FASTIO;
    cin >> N >> M;
    // cycle이 발생하지 않는다는 전제 조건
    for(int i=0; i<N-1; i++){
        int u, v, cost;
        cin >> u >> v >> cost;
        adj[u].push_back({v, cost});
        adj[v].push_back({u, cost});
    }
    // DFS로 풀만함
    while(M--){
        ans = 0;
        memset(visited, false, sizeof(visited));
        int u, v;
        cin >> u >> v;
        visited[u] = true;
        DFS(u, v, 0);
        cout << ans << '\n';
    }
    return 0;
}
```