BOJ: 1967번 트리의 지름 풀이

문제 설명

문제

트리(tree)는 사이클이 없는 무방향 그래프이다. 트리에서는 어떤 두 노드를 선택해도 둘 사이에 경로가 항상 하나만 존재하게 된다. 트리에서 어떤 두 노드를 선택해서 양쪽으로 쫙 당길 때, 가장 길게 늘어나는 경우가 있을 것이다. 이럴 때 트리의 모든 노드들은 이 두 노드를 지름의 끝 점으로 하는 원 안에 들어가게 된다.

이런 두 노드 사이의 경로의 길이를 트리의 지름이라고 한다. 정확히 정의하자면 트리에 존재하는 모든 경로들 중에서 가장 긴 것의 길이를 말한다.

입력으로 루트가 있는 트리를 가중치가 있는 간선들로 줄 때, 트리의 지름을 구해서 출력하는 프로그램을 작성하시오. 아래와 같은 트리가 주어진다면 트리의 지름은 45가 된다.

트리의 노드는 1부터 n까지 번호가 매겨져 있다.

입력

파일의 첫 번째 줄은 노드의 개수 n(1 ≤ n ≤ 10,000)이다. 둘째 줄부터 n-1개의 줄에 각 간선에 대한 정보가 들어온다. 간선에 대한 정보는 세 개의 정수로 이루어져 있다. 첫 번째 정수는 간선이 연결하는 두 노드 중 부모 노드의 번호를 나타내고, 두 번째 정수는 자식 노드를, 세 번째 정수는 간선의 가중치를 나타낸다. 간선에 대한 정보는 부모 노드의 번호가 작은 것이 먼저 입력되고, 부모 노드의 번호가 같으면 자식 노드의 번호가 작은 것이 먼저 입력된다. 루트 노드의 번호는 항상 1이라고 가정하며, 간선의 가중치는 100보다 크지 않은 양의 정수이다.

출력

첫째 줄에 트리의 지름을 출력한다.

접근 방법

트리의 지름은 트리에 존재하는 모든 경로들 중에서 가장 긴 것의 길이니까 단말 노드-단말 노드 거리가 정답의 후보가 될 것 같다는 생각을 했다. 왜냐하면 노드-노드-비단말 노드를 트리의 지름이라고 가정하면, 노드-노드-비단말 노드-노드로 연결할 수 있는 다른 노드가 존재하기 때문에 가정은 거짓이 된다.

다음으로 비단말 노드에서 DFS를 통해 거리가 가장 먼 노드를 찾으면 될 것 같다는 생각이 들었다. 모든 비단말 노드에서 DFS를 통해 최대 거리를 찾으면 O(N*(E+V))가 걸릴 것이다.

풀이를 헤매다가 질문 게시판에서 힌트를 얻었다. 한 정점에서 DFS를 통해 가장 먼 정점을 찾은 후, 거기서 다시 DFS를 통해 먼 곳을 찾는 방법이었다.

주요 코드

```
cin >> n;
for (int i = 0; i < n - 1; i++) {
	int x, y, d;
	cin >> x >> y >> d;
	adj[x].emplace_back(y, d);
	adj[y].emplace_back(x, d);
}
```

입력을 받는 부분. `vector<pair<int, int>> adj[10001];` 로 선언하였다.

```
void dfs(int cur, int par, int d) {
    for (auto nxt : adj[cur]) {
        if (nxt.first == par) continue;
        if (d + nxt.second > max_dist) {
            max_dist = d + nxt.second;
            max_node = nxt.first;
        }
        dfs(nxt.first, cur, d + nxt.second);
    }
}
```

dfs 함수. parent는 저장할 필요가 없어서 인자로 넘겨주었고, max_dist와 max_node는 전역 변수로 설정하였다. 한 정점에서 가장 먼 정점을 찾아서 거리와 해당 정점을 기록한다.

```
dfs(1, 0, 0);
dfs(max_node, 0, 0);
cout << max_dist;
```

메인 함수에서 입력을 받고 나면, 위의 코드 세 줄만 더 작성하면 된다. 먼저 dfs로 루트에서 최장 거리를 구하고 그 점에서 다시 최장 거리를 구한 후 출력한다.

느낀점

가장 먼 곳 찾기 x2 ←단순한 아이디어지만 나는 쉽게 떠올리진 못했다.

그리고 구현을 할 때도 어떤 자료구조를 이용하는게 좋을까 하는 고민이 많이 됐던 문제이다.