고급소프트웨어실습 7주차 과제

> **학번: 20171665 / 이름: 이선호**

> 실습에서 다루지 않은 문제 중 DFS와 BFS로 풀 수 있는 문제를 각각 한 가지 씩 예를 들어 설명하시오. 관련된 그래프 구조도 그리시오.

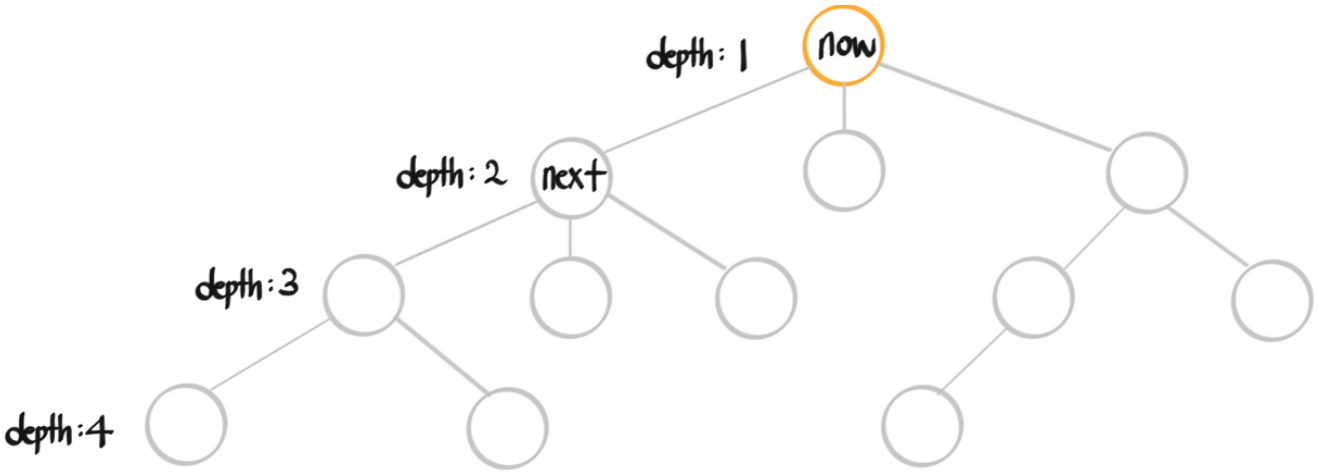
DFS로 풀 수 있는 문제

[문제 링크] https://www.acmicpc.net/problem/19535

DFS로 풀 수 있는 문제 중 하나는 2020년 UCPC 예선전에서 D번으로 출제되었던 'ㄷㄷㄷㅈ' 문제이며, 개인적으로 블로그에 풀이법을 정리하여 올린 적이 있다. (블로그 링크)

N개의 정점으로 구성된 임의의 트리가 주어졌을 때, 4개의 정점으로 이루어진 'ㄷ' 모양의 부분 트리와 'ㅈ' 모양의 부분 트리를 찾을 수 있다. 이 때, 'ㄷ' 모양의 부분 트리의 개수가 'ㅈ' 모양의 부분 트리 개수의 3배한 값보다 크면 'D-트리', 작으면 'G-트리', 같으면 'DUDUDUNGA-트리'로 분류한다. 임의의 트리가 주어졌을 때, 해당 트리가 어떠한 분류에 속하는 트리인지를 구하는 것이 문제이다.

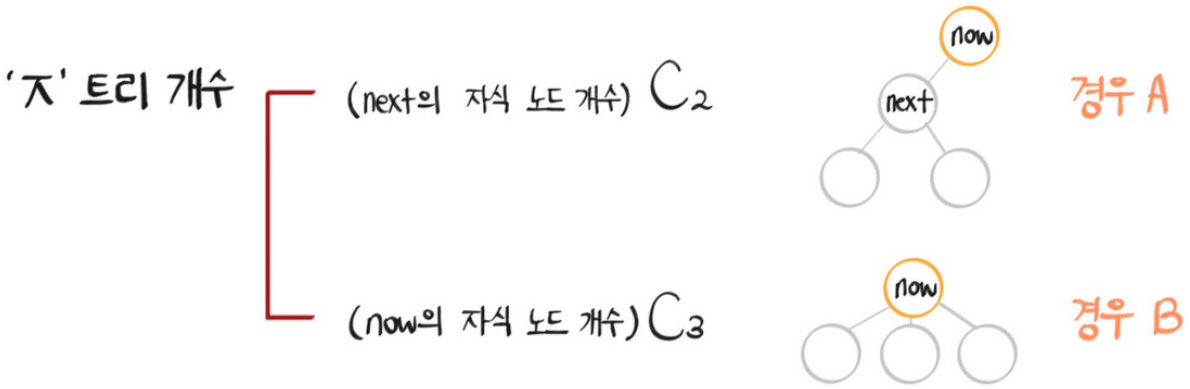
이 문제의 핵심은 바로 'ㄷ' 모양의 부분 트리와 'ㅈ' 모양의 부분 트리의 개수를 세어야 한다는 건데, 각 정점을 DFS(Depth-First Search)로 탐색하는 방법을 사용할 수 있다.



예를 들어, 위와 같이 4개의 depth로 이루어진 트리가 존재하며, 현재 방문 중인 정점을 정점 now, 그의 임의의 자식 정점을 정점 next, 각 정점이 어떠한 depth에 있는지를 저장하는 depth 배열이 있다고 가정한다.

현재 방문 중인 정점 now를 기준으로 만들어지는 'ㄷ'와 'ㅈ' 모양의 부분 트리 개수를 세기 위해 다음과 같이 케이스를 분리할 수 있다.

1. 정점 now를 포함하는 'ㅈ' 모양의 부분 트리 개수
    1. 정점 now가 'ㅈ' 모양의 부분 트리의 끄트머리에 달려 있는 경우
       'ㅈ' 모양의 부분 트리에 속하면서 정점 now와 직접 연결된 정점이 한 개일 때 (A)
    2. 정점 now가 'ㅈ' 모양의 부분 트리의 가운데에 위치하는 경우
       'ㅈ' 모양의 부분 트리에 속하면서 정점 now와 직접 연결된 정점이 세 개일 때 (B)
2. 정점 now를 포함하는 'ㄷ' 모양의 부분 트리 개수
    1. 정점 now가 'ㄷ' 모양의 부분 트리의 끄트머리에 달려 있는 경우
       'ㄷ' 모양의 부분 트리에 속하면서 정점 now와 직접 연결된 정점이 한 개일 때 (C)
    2. 정점 now가 'ㄷ' 모양의 부분 트리의 가운데에 위치하는 경우
       'ㄷ' 모양의 부분 트리에 속하면서 정점 now와 직접 연결된 정점이 세 개일 때 (D)

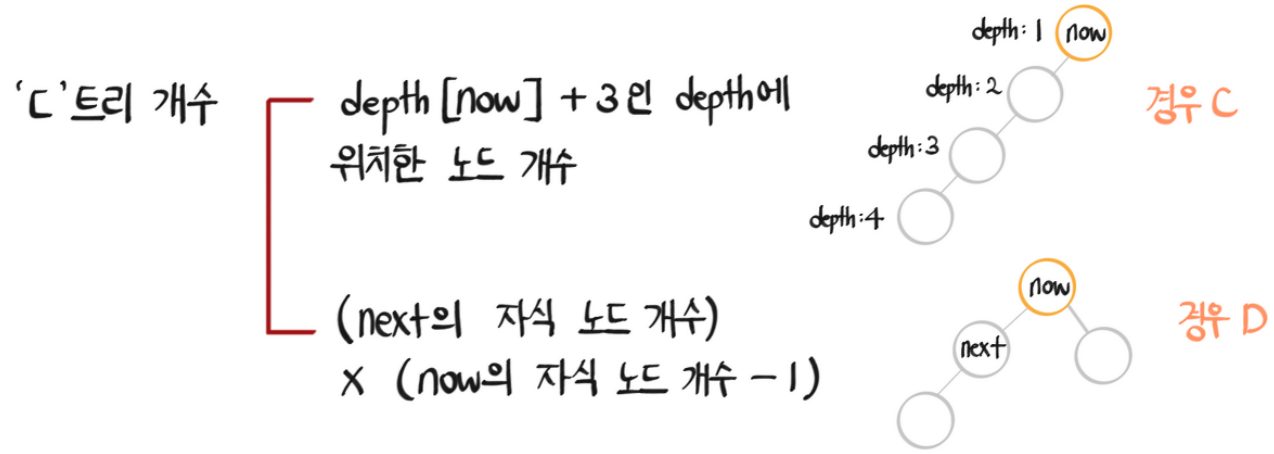


A의 경우에는 정점 next의 자식 정점에서 그의 두 개의 자식 정점을 뽑아야 하므로, 다음과 같은 조합으로 'ㅈ' 모양의 부분 트리 개수를구할 수 있다.

$$\binom{\text{정점 next의 자식 정점 개수}}{2} \tag{1}$$

B의 경우에는 정점 next의 자식 정점에서 그의 세 개의 자식 정점을 뽑아야 하므로, 다음과 같은 조합으로 'ㅈ' 모양의 부분 트리 개수를 구할 수 있다.

$$\binom{\text{정점 next의 자식 정점 개수}}{3} \tag{2}$$



C의 경우에는 현재 정점 now부터 네 개의 depth를 일자로 타고 갈 때 나오는 부분 트리의 개수를 세야 하므로, `depth[now] + 3`인 depth에 몇 개의 정점이 존재하는지를 세어야 한다.

D의 경우에는 현재 정점 now에서 정점 next를 제외한 나머지 자식 정점에서 1개를 뽑고, 정점 next의 자식 정점에서 한 개를 뽑아 구성하는 부분 트리이므로 $(\text{정점 next의 자식 정점 개수}) \times (\text{정점 now의 자식 정점 개수} - 1)$로 'ㄷ' 모양의 부분 트리 개수를 구할 수 있다.

그래서 DFS를 사용하여 전체 그래프를 구성하는 모든 정점을 한 번씩 탐색하면서 각 정점의 자식 정점의 수를 사전에 구해 놓는 것이 필요하다. 이후 다시 DFS로 그래프를 탐색하면서 A, B, C, D의 각 케이스를 만족할 때마다 'ㄷ'와 'ㅈ' 모양의 부분 트리 개수에 더해 나간다.

BFS로 풀 수 있는 문제

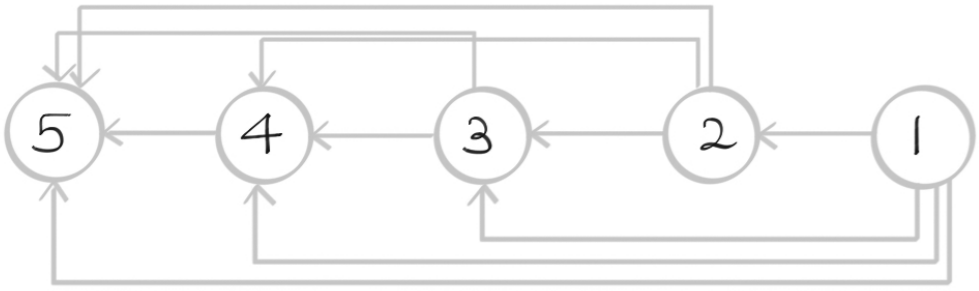
[문제 링크] https://www.acmicpc.net/problem/3665

BFS로 풀 수 있는 문제 중 하나는 2010년 ICPC Nortwestern European 지역문제로 출제되었던 '최종 순위'라는 문제이며, 위의 DFS 문제와 마찬가지로 개인적으로 이 문제도 블로그에 풀이법을 정리하여 올린 적이 있다. (블로그 링크)

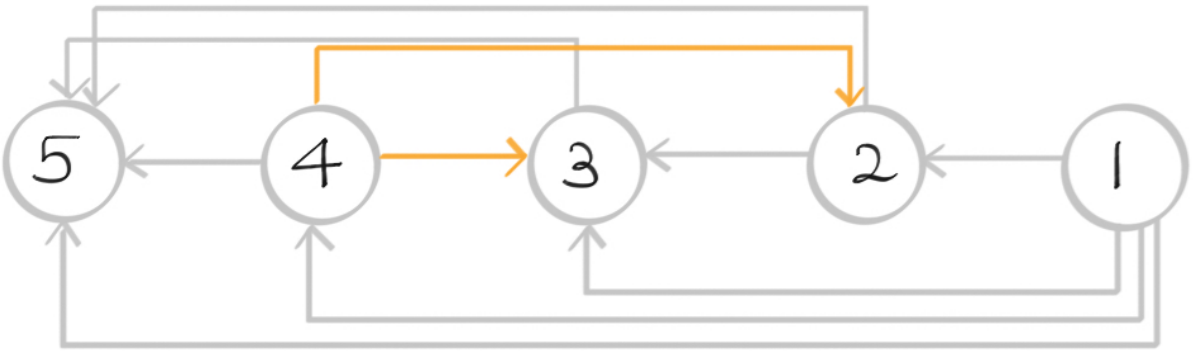
1부터 n까지 번호가 매겨진 팀들이 작년과 올해 대회에 참전했고, 문제에서는 작년 대회에서 각 팀들의 최종 순위가 주어진다. 또한 올해 대회에서 모든 팀 중 상대적인 등수가 바뀐 두 팀들이 쌍으로 주어질 때, 올해 모든 각 팀의 정확한 최종 순위를 구하는 것이 문제이다.

단, 올해 정확한 최종 순위를 구할 수 없는 경우도 존재하는데, 이는 후술할 예정이다.

작년 최종순위

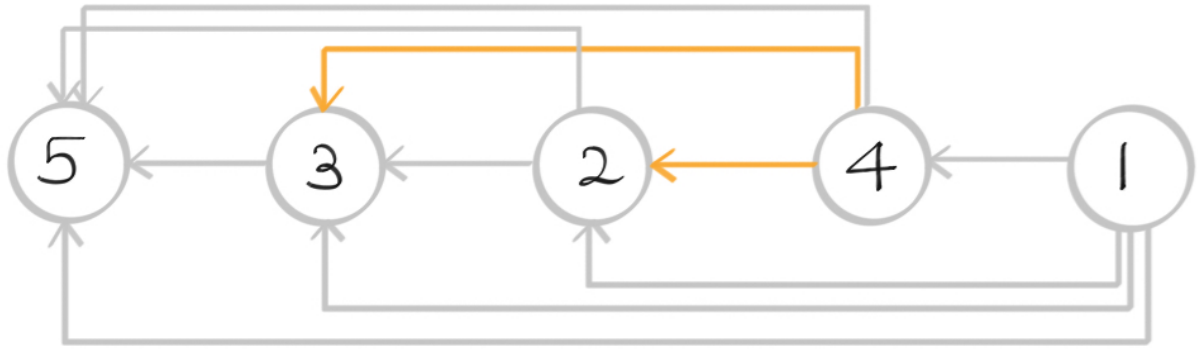


작년 최종 순위를 바탕으로 팀들의 상대적인 순위에 대한 관계를 간선으로 나타냄



올해 상대적 순위가 바뀐 팀 짝의 간선 방향을 reverse 해줌

먼저 팀들의 작년 최종 순위를 기준으로 위상 정렬을 하고, 올해 상대적인 등수가 바뀐 두 팀을 잇는 간선의 방향을 변경해서 다시 위상 정렬이 만족되는지를 확인한다.



다시 related 된 그래프를 바탕으로 Topological Sort 진행

위상정렬을 할 때는 순위가 낮은 정점에서 순위가 높은 정점으로 간선을 잇는데, 위상정렬을 할 때 queue 자료구조를 사용하여 BFS로 탐색하는 방법을 사용할 수 있다. 이를 위해서 모든 각 정점마다 해당 정점으로 간선이 들어오는 개수를 세서 indegree 배열에 구해 놓는다.

첫 방문하는 정점을 queue에 넣고, queue가 빌 때까지 queue의 앞에서 한 원소를 뽑아서 그 원소와 연결된 다른 정점을 탐색하는데, 이때 해당 정점의 indegree 원소 값을 하나씩 줄여 나간다. 만약 해당 정점의 indegree 원소 값이 0이면 해당 정점을 방문한 것으로 처리하기 위해 queue에 넣는다.

그런데 올해 최종 순위로 위상정렬을 할 수 없는 경우도 존재하는데, 이는 크게 두 가지 케이스로 존재한다.

올해 최종 순위를 구했을 때 cycle이 발생하는 것은 일관성이 없다는 것인데, 이는 위상정렬 과정에서 방문하는 정점의 수가 n보다 작은 것이다.

팀들의 최종 순위를 구할 때 어떤 둘 이상의 팀의 순위는 달라질 수 있어서 확실한 순위를 구할 수 없는 경우가 있을 수 있는데, 이는 위상정렬 시 둘 이상의 팀 정렬 순위가 서로 달라질 수 있어서 queue에 둘 이상의 정점이 들어갈 수 있을 때이다.

위의 두 가지 케이스에 관해서는 올해의 최종 순위를 구할 수 없다고 결론을 내리고, 나머지 경우에는 올해 최종 순위를 구성할 수 있으므로 앞서 설명한 queue를 이용해 BFS로 탐색해서 queue에 정점이 들어가는 순서로 최종 순위를 구한다.