고급 소프트웨어 실습

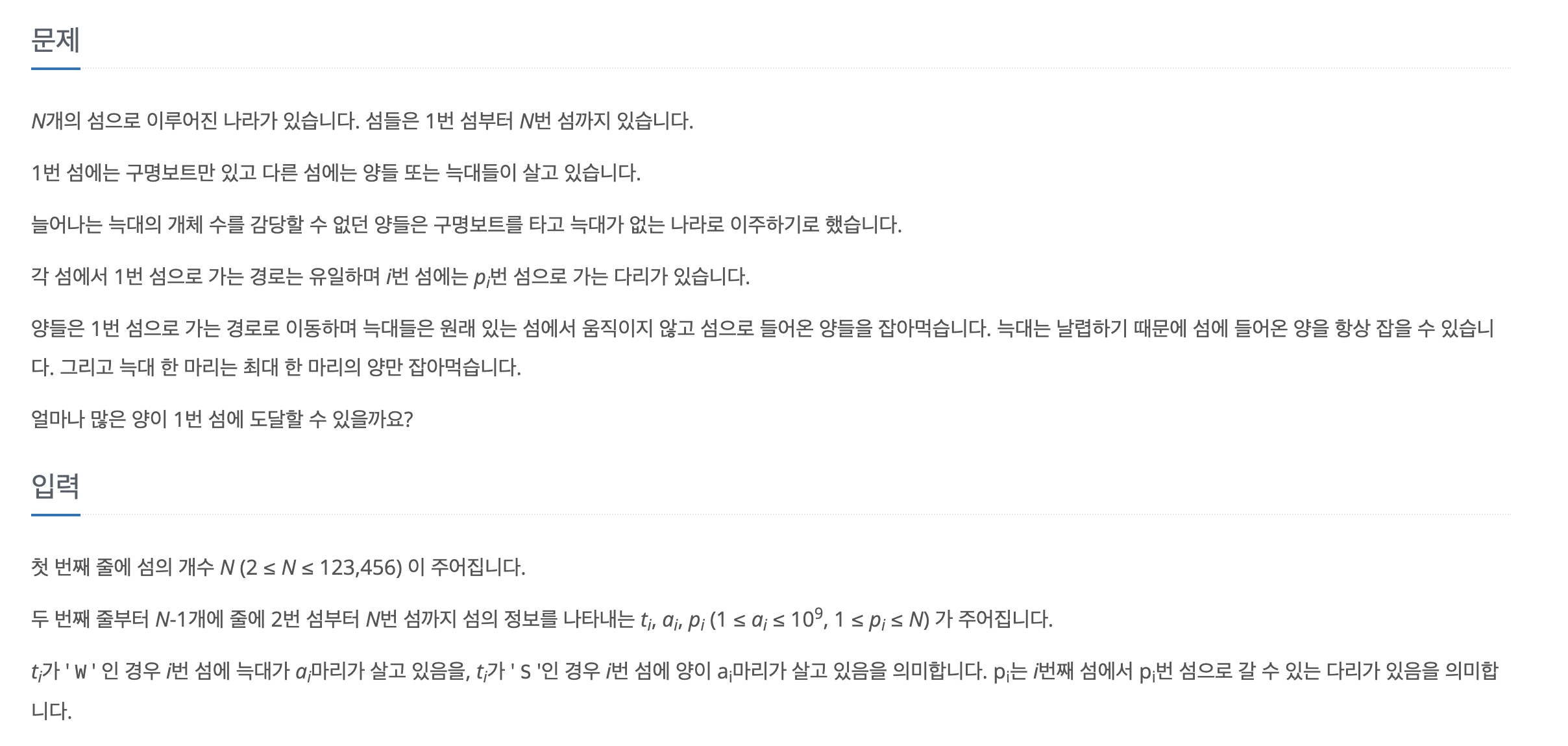
분반: 1

학번: 20191574

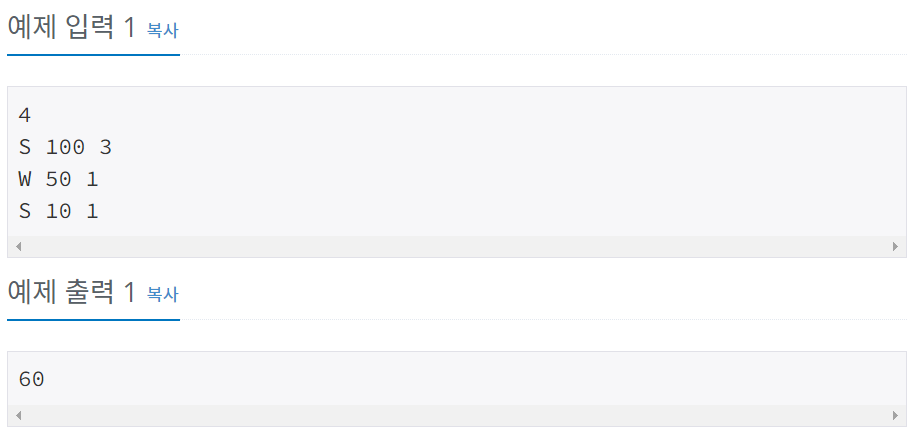
이름: 김예진

DFS와 BFS로 풀 수 있는 문제를 한 가지 씩 예를 들어 설명하시오. 관련된 그래프 구조도 그리시오.

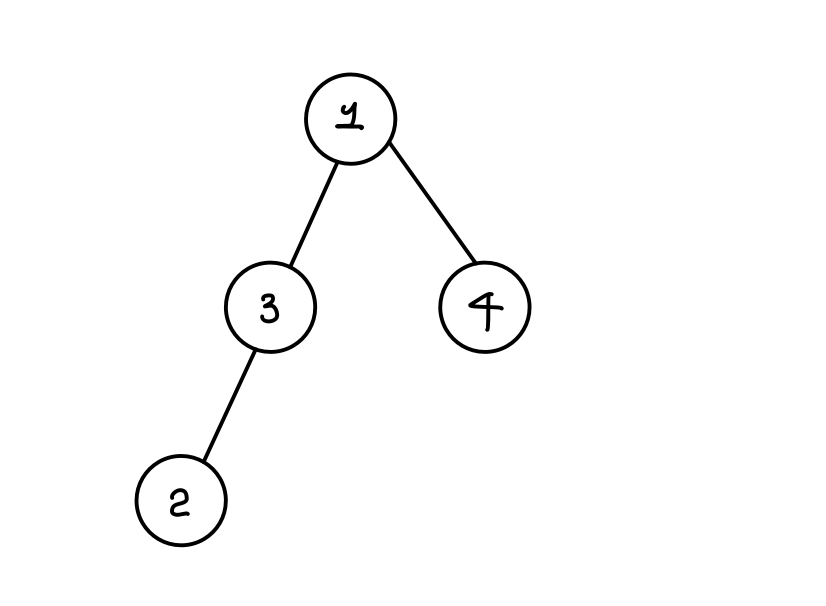
- **DFS**

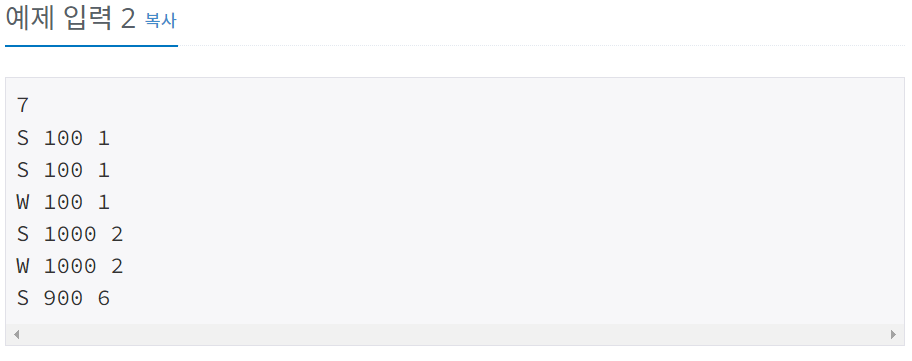


해당 문제는 각 섬에서 root 섬인 1번 섬으로 가는 경로가 1개씩 밖에 없으므로 그래프가 트리 형태를 띄게 된다. 1번 섬을 제외한 다른 섬에서 1번 섬으로 양들이 이동할 때 늑대를 만나면 잡아먹힌다고 할 때 1번 섬에 도착 가능한 양을 세는 문제이다. 이 문제는 각 subtree의 root에서 child tree로부터 생존해 오는 양의 수를 dfs로 count하여 문제를 해결할 수 있다.

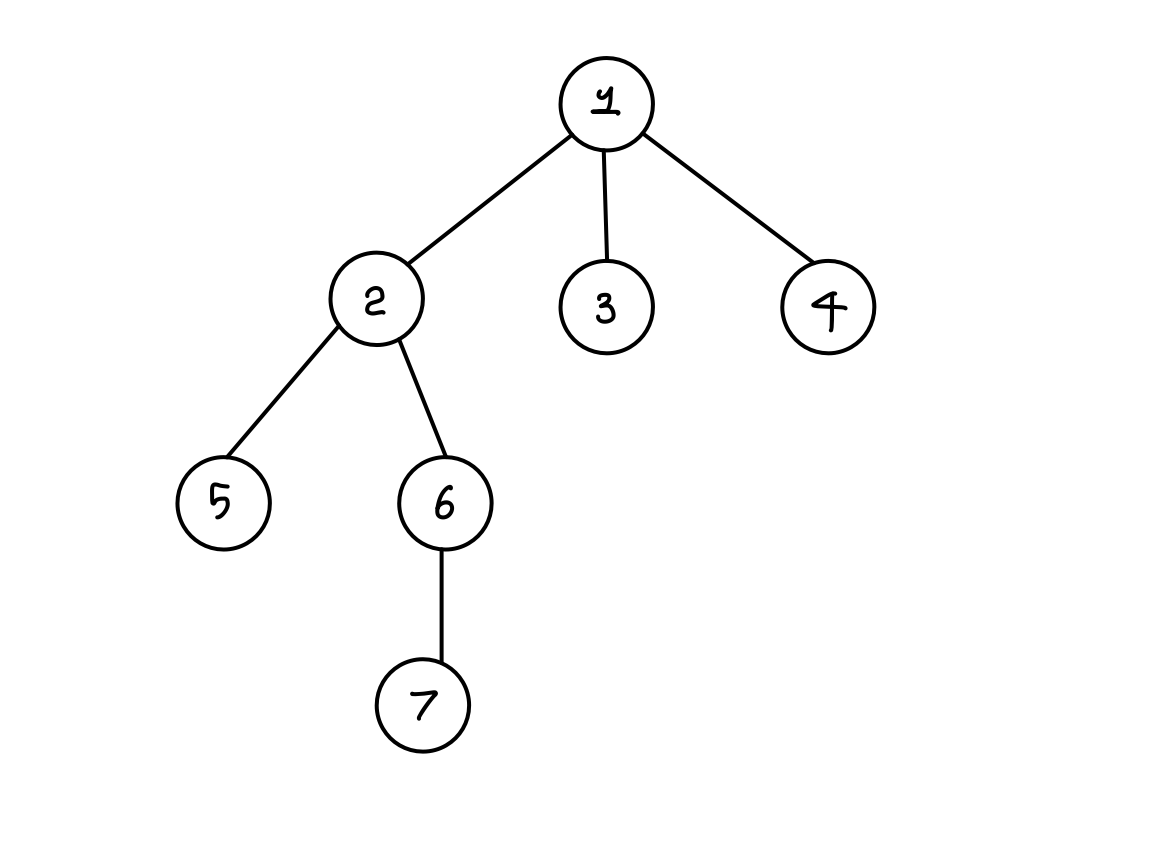


예제 1은 다음과 같은 그래프 형태를 보인다.

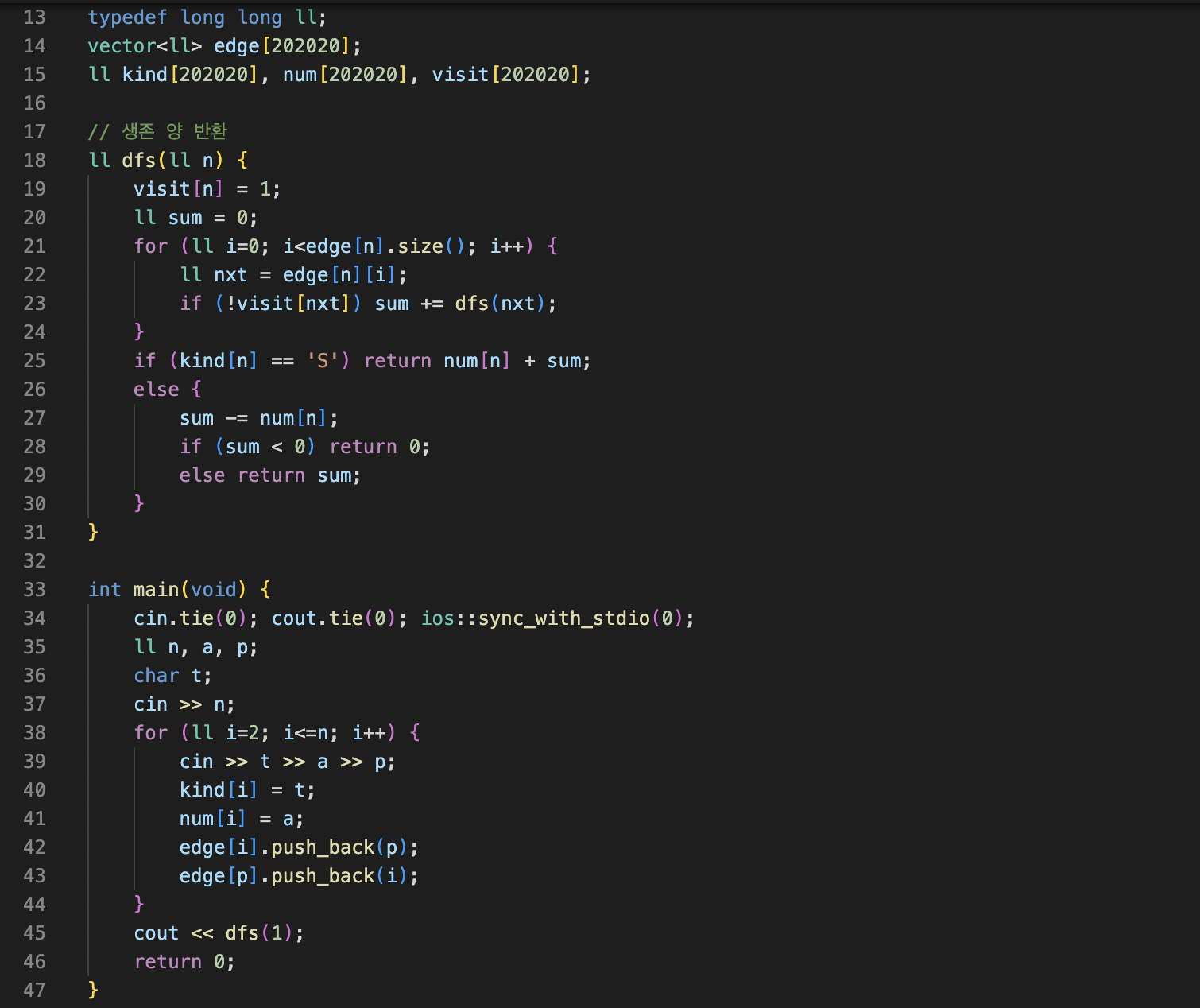




조금 더 복잡한 예제인 예제2는 다음과 같은 그래프 구조를 갖는다.



각 노드에서 깊이 우선 탐색을 하면서 자식들로부터 생존해 온 양의 수 정보를 받고, 현재 노드의 늑대 수, 혹은 양의 수를 계산하여 return 해줌으로써 parent node에 재귀적으로 정보를 전달한다. 코드는 다음과 같이 작성할 수 있다.

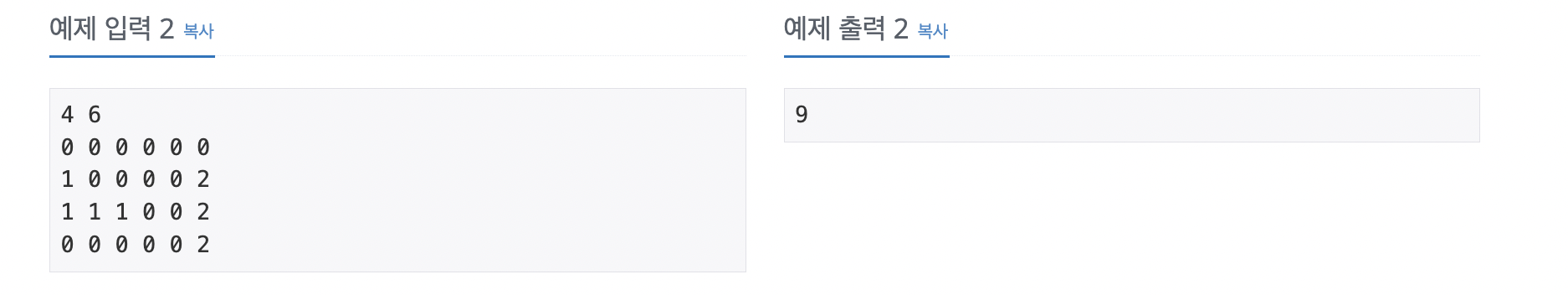


자식들로부터 들어온 양의 수를 재귀적으로 dfs를 호출해서 받고, 현재 subtree의 root에 양이 있다면 이 값을 더해서 return한다. 현재 subtree의 root에 늑대가 있다면 자식 subtree로 부터 넘어오던 양이 늑대에 의해 잡아먹히기 때문에 늑대 수를 반환 값에서 빼준 후 return 한다. 이 때 음수가 된다면 늑대는 더 이상 부모 노드에 영향을 주지 않기 때문에 0으로 초기화해서 return을 해주어야 한다.

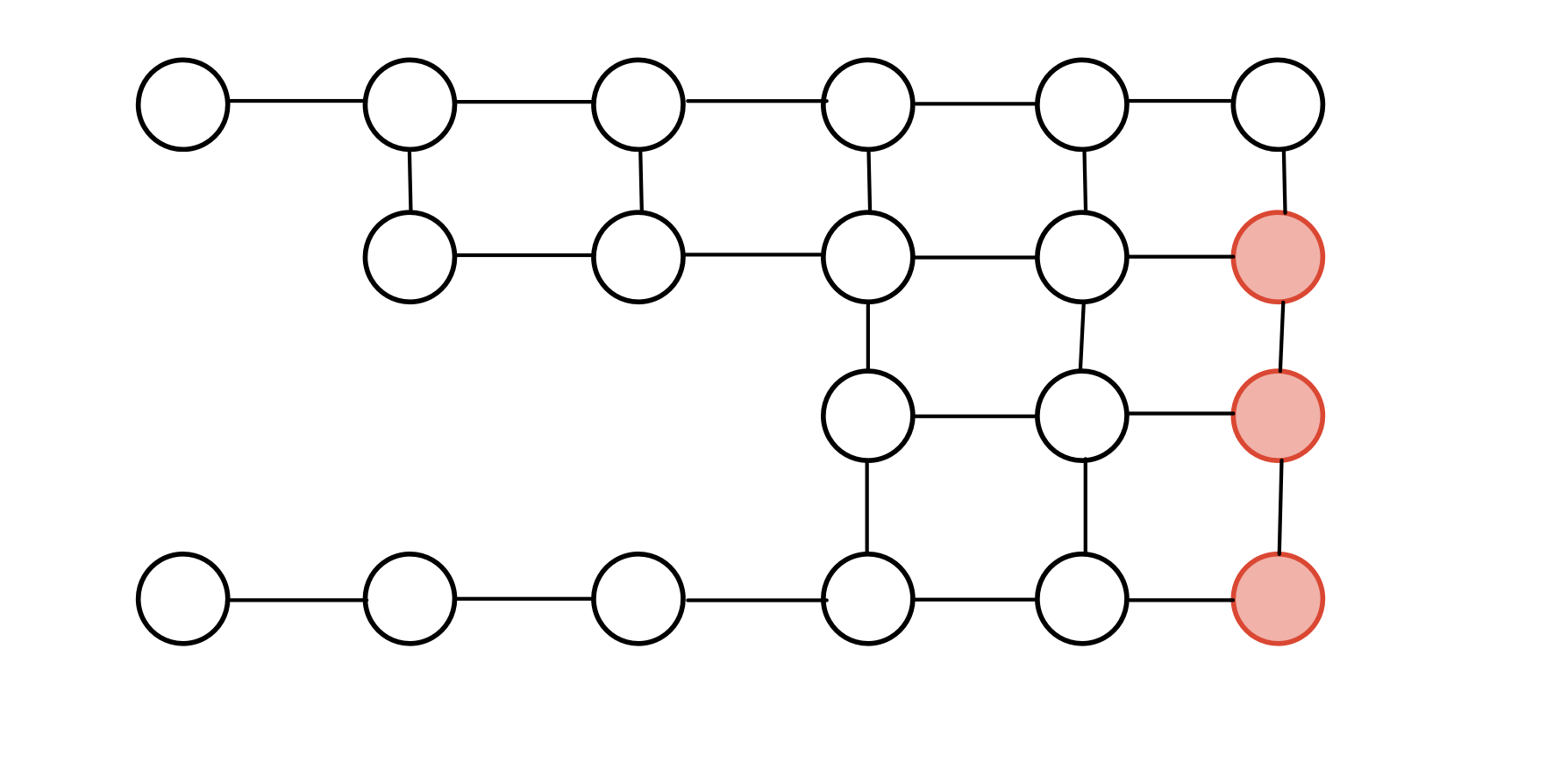
**- BFS**



해당 문제는 그래프가 2차원 배열 구조로 제시되어 있다. 0이 빈 공간, 1이 벽, 2가 바이러스일 때 벽 3개를 추가로 세워 최대한 큰 안전 영역을 취하는 것이 문제의 목적이다. 이 문제는 연구소의 size가 최대 8\*8로 모든 cell에서 3개를 선택하는 경우의 수가 최대 10000 정도이기 때문에 빈 공간에 대한 정보를 미리 따로 저장하여 해당 부분에서 3개의 벽을 선택한 상태로 bfs 알고리즘을 수행하였다. 각 bfs 상황에 대해 안전 영역 max 값을 계산하여 출력함으로써 결과를 도출할 수 있다.



위 예제에 대한 그래프 구조는 다음과 같다.



이 상태의 그래프에서 노드 3개를 적절히 떼어냈을 때 바이러스에 의해 완전히 분리될 수 있는 노드의 수를 세어야 한다.

bfs는 queue를 활용하여 진행하였다. 현재 바이러스가 간선을 타고 뻗어나가므로 바이러스의 위치를 먼저 queue에 넣어두고 간선으로 이동할 수 있는 cell들 중 빈칸이면서 아직 방문하지 않은 노드가 있으면 추가적으로 감염을 시킨다. queue에 추가하고 while 문에서 해당 원소를 pop 했을 때 이 노드와 인접한 노드들에 대해 같은 로직으로 점차 탐색해나가는 방식으로 구현이 가능하다.

코드는 다음과 같이 작성하였다.

