알튜비튜

DFS & BFS

오늘은 그래프 탐색 알고리즘인 깊이 우선 탐색(DFS)과 너비 우선 탐색(BFS)를 배웁니다.
앞으로 배울 그래프 알고리즘의 시작이자 코딩테스트에 높은 확률로 한 문제 이상 나오는 알고리즘이죠
.

그래프 탐색

#알튜비튜

DFS (깊이 우선 탐색)

- 최대한 깊게 탐색 후 빠져 나옴
- 한 정점을 깊게 탐색해서 빠져 나왔다면, 나머지 정점 계속 동일하게 탐색
- 스택(stack), 재귀함수로 구현

BFS (너비 우선 탐색)

- 자신의 자식들부터 순차적으로 탐색
- 순차 탐색 이후, 다른 정점의 자식들 탐색
- 큐(queue)로 구현

10026번 : 적록색약 – Gold 5

문제

- 적록색약이 보는 구역 개수와 적록색약이 아닌 사람이 보는 구역의 개수를 찾는 문제
- N*N 각각의 칸에는 R, G, B 중 하나가 쓰여 있음
- 여기서 말하는 구역은 같은 색으로 이루어진 구역
- 상하좌우로 인접해 있는 경우에 두 글자는 같은 구역에 속함

제한 사항

- 세로 크기 & 가로 크기 == N, (1 ≤ N ≤ 100)

도전 문제

10026번 : 적록색약 – Gold 5

예시

적록색약이 보는 구역 => 3개

빨강과 초록이 인접해 있을 때 구역을 구분하지 못한다!

적록색약이 아닌 사람이 보는 구역 => 4개

도전 문제

#알튜비튜

/<> 10026번 : 적록색약 – Gold 5

RRRBB
GGBBB
BBBRR
BBRRR
RRRRR

dfs함수로 설계가 가능합니다!

1. 적록색약이 아닌 사람이 구역을 인식하는 경우
2. 적록색약인 사람이 구역을 인식하는 경우

1번의 경우 어느 한 점에서 상하좌우 다른 색이 나올 때까지 탐색을 하면 됩니다.

2번의 경우 비슷하지만, 어느 한 점이 R 혹은 G 일 때 상하좌우 G 나 R이 나와도 계속 탐색을 하면 됩니다.

-> 쉽게 두 가지의 경우를 나누어 dfs 함수를 2개 짜서 구현할 수 있다.

도전 문제

10026번 : 적록색약 – Gold 5

```
void dfs(int y, int x, char c) {
    for (int i = 0; i < 4; ++i) {
        int ny = y + dy[i];
        int nx = x + dx[i];
        if (ny >= 0 && ny < n && nx>=0 && nx < n) {
            if (!visit[ny][nx] && graph[ny][nx] == c) {
                visit[ny][nx] = true;
                dfs(ny, nx, c);
            }
        }
    }
}
```

적록 색약이 아닌 사람의 경우

```
void dfs2(int y, int x, char c) {
    for (int i = 0; i < 4; ++i) {
        int ny = y + dy[i];
        int nx = x + dx[i];
        if (ny >= 0 && ny < n && nx>=0 && nx < n) {
            if (!visit[ny][nx]) {
                if ((c == 'R' || c == 'G' )&& (graph[ny][nx]=='R' || graph[ny][nx]=='G')) {
                    visit[ny][nx] = true;
                    dfs2(ny, nx, c);
                }
                else if (c == 'B') {
                    if (graph[ny][nx] == 'B') {
                        visit[ny][nx] = true;
                        dfs2(ny, nx, c);
                    }
                }
            }
        }
    }
}
```

적록 색약인 사람의 경우

현재의 위치에 따른 조건문을 추가합니다.

도전 문제

1068번 : 트리 - Gold 5

문제

- 트리에서 리프 노드란, 자식의 개수가 0인 노드를 말한다. 트리가 주어졌을 때, 노드 하나를 지울 것이다. 그 때, 남은 트리에서 리프 노드의 개수를 구하는 프로그램을 작성하시오. 노드를 지우면 그 노드와 노드의 모든 자손이 트리에서 제거된다.
- 첫째 줄에 입력으로 주어진 트리에서 입력으로 주어진 노드를 지웠을 때, 리프 노드의 개수를 출력한다.

제한 사항

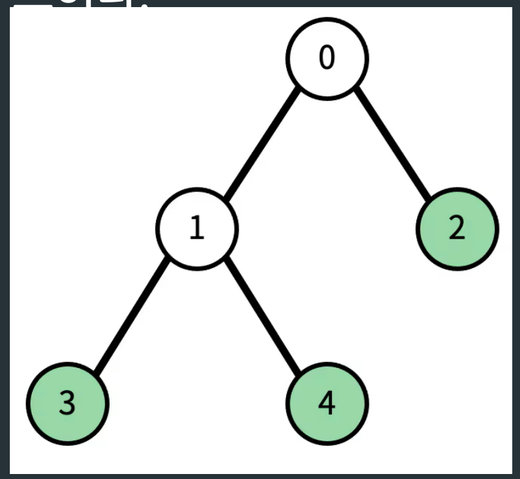
- 트리 노드의 개수: $1 \le N \le 50$
- 0~N-1번 노드의 부모가 주어지는데 루트 노드의 부모는 -1로 주어진다.

도전 문제

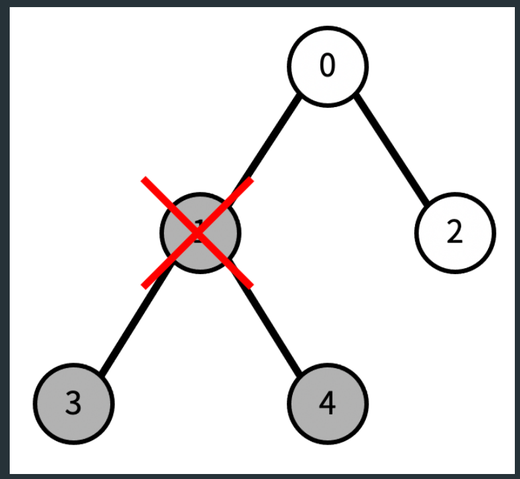
1068번 : 트리 - Gold 5

예시

현재 리프 노드의 개수는 3이다. (초록색 색칠된 노드) 이때, 1번을 지우면, 다음과 같이 변한다. 검정색으로 색칠된 노드가 트리에서 제거된 노드이다.



이제 리프 노드의 개수는 1개이다.



도전 문제

1068번 : 트리 - Gold 5

문제 풀이 방법

- 1. BFS 방법으로 해결하기
- 2. DFS 방법으로 해결하기

도전 문제

#알튜비튜

1068번 : 트리 - Gold 5

문제 풀이 방법 ● 1. BFS 탐색으로 해결하기

```
void bfs(int root) {
    if (root == deleteNode) return;
    queue<int> q;
    q.push(root);

    while (!q.empty()) {
        int cur = q.front();
        q.pop();

        if (edge[cur].empty() || (edge[cur].size() == 1 && edge[cur][0] == deleteNode))
{           answer++;
            continue;
        }

        for (int next : edge[cur]) {
            if (next == deleteNode) continue;
            q.push(next);
        }
    }
}
```

- 인접리스트 배열로 저장
- 루트 노드가 삭제 대상이면 바로 종료
- 큐의 맨 앞 노드를 가져온 후, 큐에서 제거
- 현재 노드가 리프 노드인지 확인 후, 개수 증가
- 현재 노드의 모든 자식 노드를 큐에 추가
- 삭제 대상 노드는 건너뜀

1068번 : 트리 - Gold 5

문제 풀이 방법 ● 2. DFS 탐색으로 해결하기

```
void dfs(int cur) {
    if (cur == deleteNode) return;

    if (edge[cur].empty()) {
        answer++;
        return;
    }

    bool isLeaf = true;
    for (int next : edge[cur]) {
        if (next == deleteNode) continue;
        dfs(next);
        isLeaf = false;
    }

    if (isLeaf) answer++;
}
```

- 인접리스트 배열로 저장
- 삭제할 노드이면 종료
- 현재 노드가 리프 노드인지 확인 후, 개수 증가
- 삭제할 노드는 건너뜀
- 자식이 모두 삭제되었으면 리프 노드라 판단

그래프 탐색

#알튜비튜

DFS (깊이 우선 탐색)

- 최대한 깊게 탐색 후 빠져 나옴
- 한 정점을 깊게 탐색해서 빠져 나왔다면, 나머지 정점 계속 동일하게 탐색
- 스택(stack), 재귀함수로 구현

BFS (너비 우선 탐색)

- 자신의 자식들부터 순차적으로 탐색
- 순차 탐색 이후, 다른 정점의 자식들 탐색
- 큐(queue)로 구현

2615번: 오목 - Silver 1

문제

- 바둑판의 상태가 주어졌을 때, 검은색이 이겼는지, 흰색이 이겼는지 또는 아직 승부가 결정되지 않았는지를 판단하는 문제
- 한 방향으로 같은 색의 바둑알이 연속적으로 다섯 알 놓이면 승리
- 단, 여섯 알 이상이 연속적으로 놓인 경우는 이긴 것이 아님
- 방향은 가로, 세로, 대각선 모두 포함

제한 사항

- 바둑판의 크기는 19 x 19
- 검은 바둑알은 1, 흰 바둑알은 2, 알이 놓이지 않는 자리는 0으로 표시 (한 칸씩 띄어서 표시됨)
- 둘 중 하나가 이겼을 경우, 연속된 다섯 개의 바둑알 중 가장 왼쪽 위에 있는 바둑알의 좌표를 출력

예제 입력1

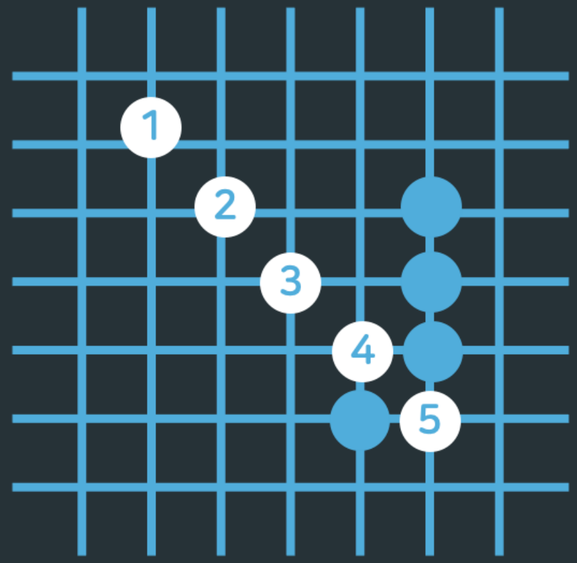
예제 입력

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 2 0 0 2 2 2 1 0 0 0 0 0 0 0 0 0 0
0 0 1 2 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 1 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 2 2 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 2 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

예제 출력

```
1
3 2
```

포인트



① 연속된 다섯 개의 바둑알 중에서 가장 왼쪽 위에 있는 바둑알을 출력해야 해요.
어느 위치부터 어떤 방향으로 탐색을 진행해야 할까요?

② 어떻게 여섯 개의 바둑알이 연속적으로 놓이는 경우를 제외할 수 있을까요?

접근 방법

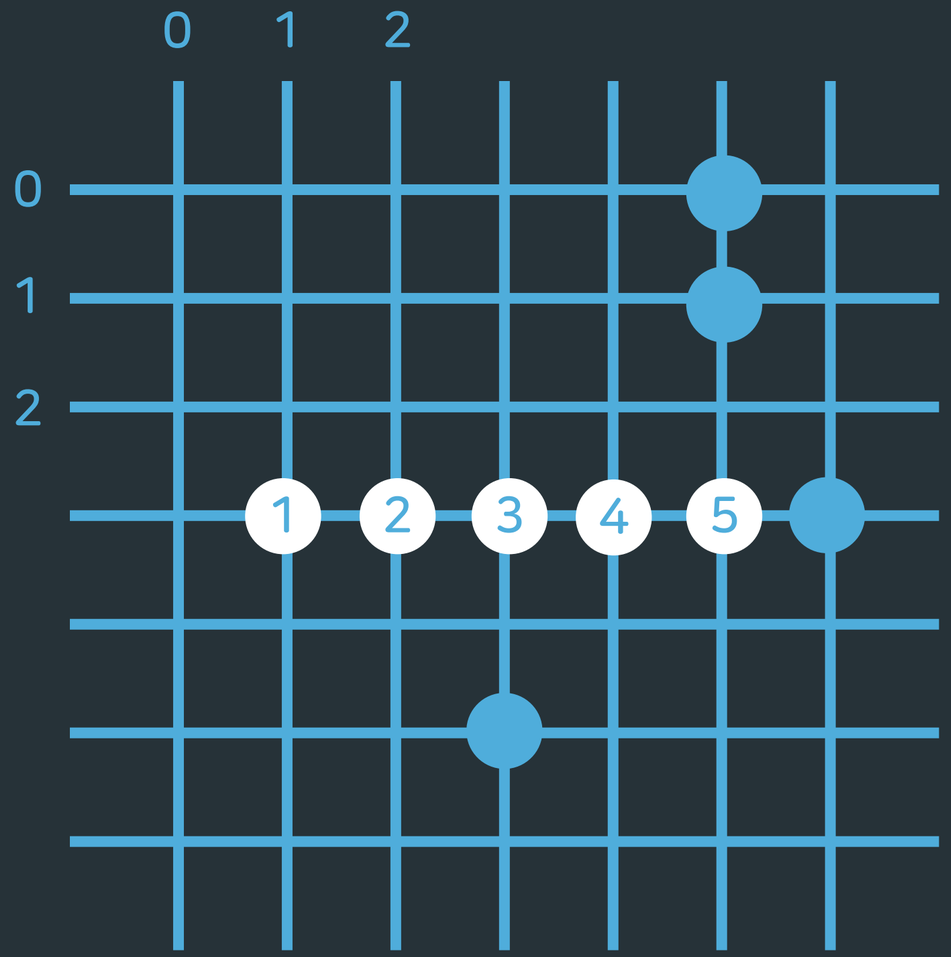
#알튜비튜

- 오목에 성공했을 시 가장 왼쪽 위에 있는 바둑알을 출력
  → 탐색 방향은 아래 4가지뿐

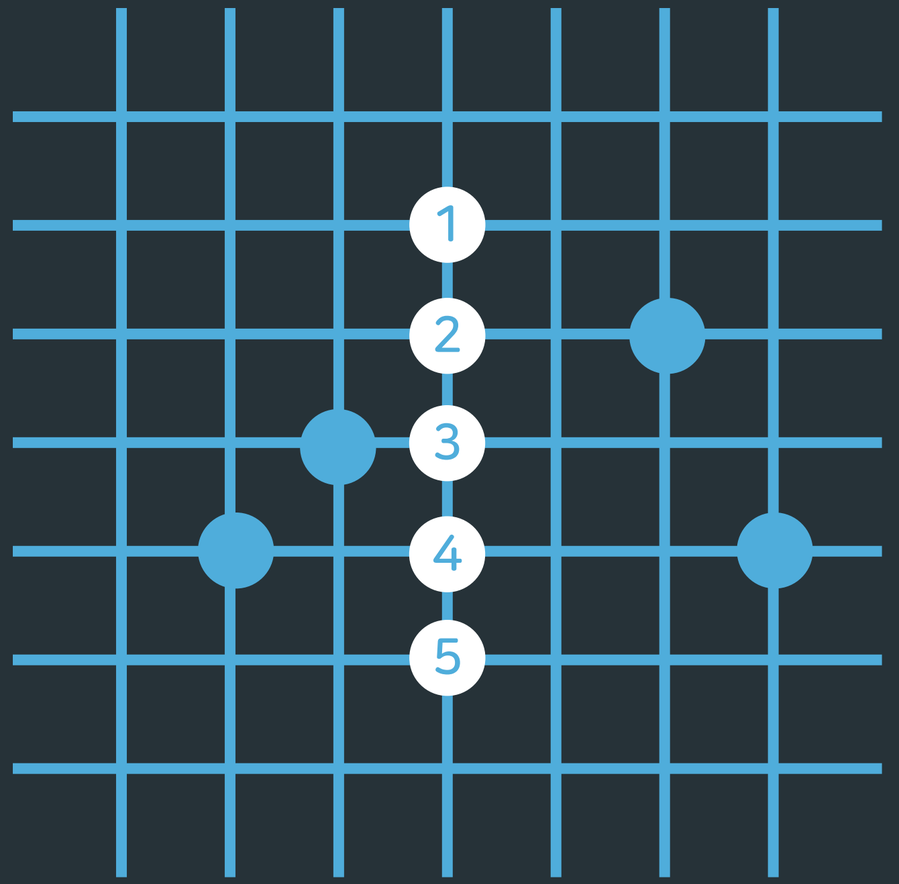
- → (0, 1) : {0, 0}, {0, 1}, {0, 2}, {0, 3}, {0, 4}, {0, 5}, …
- ↓ (1, 0) : {0, 0}, {1, 0}, {2, 0}, {3, 0}, {4, 0}, {5, 0}, …
- ↘ (1, 1) : {0, 0}, {1, 1}, {2, 2}, {3, 3}, {4, 4}, {5, 5}, …
- ↗ (-1, 1) : {0, 0}, {-1, 1}, {-2, 2}, {-3, 3}, {-4, 4}, {-5, 5}, …

⇒ 방법 1: 브루트 포스
⇒ 방법 2: DFS

접근 방법

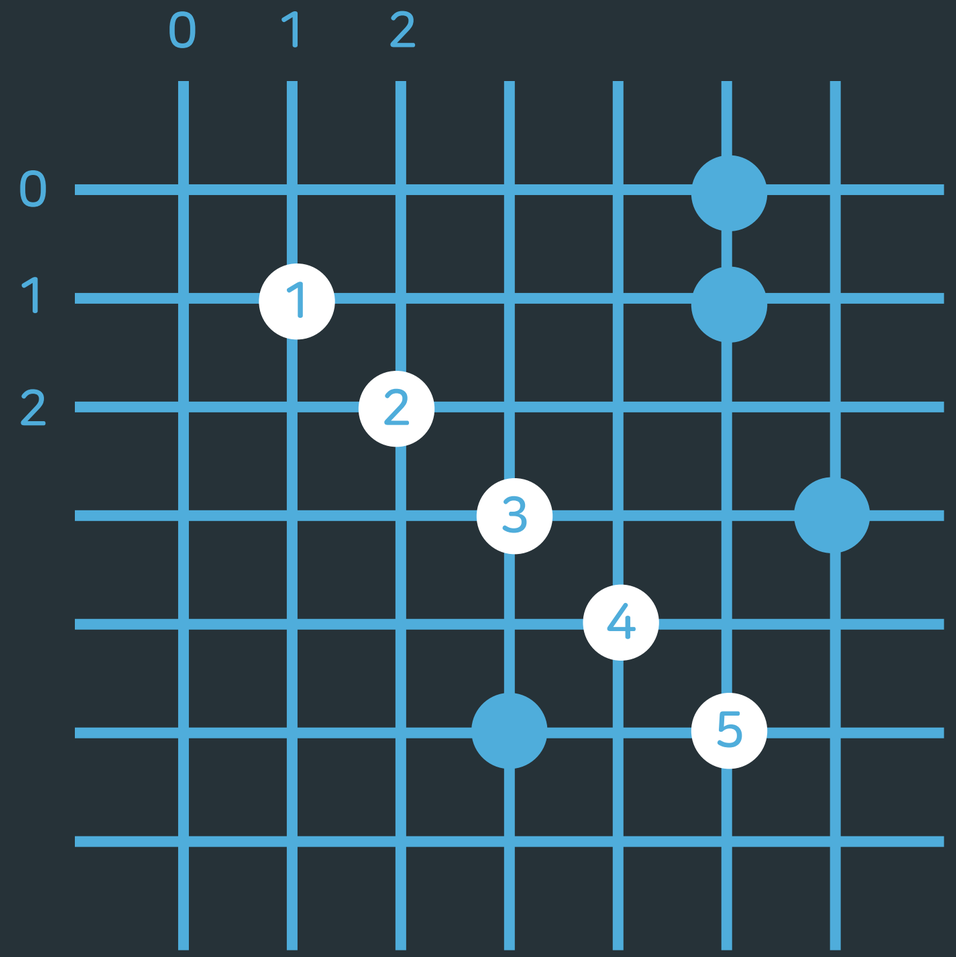


→ (0, 1)

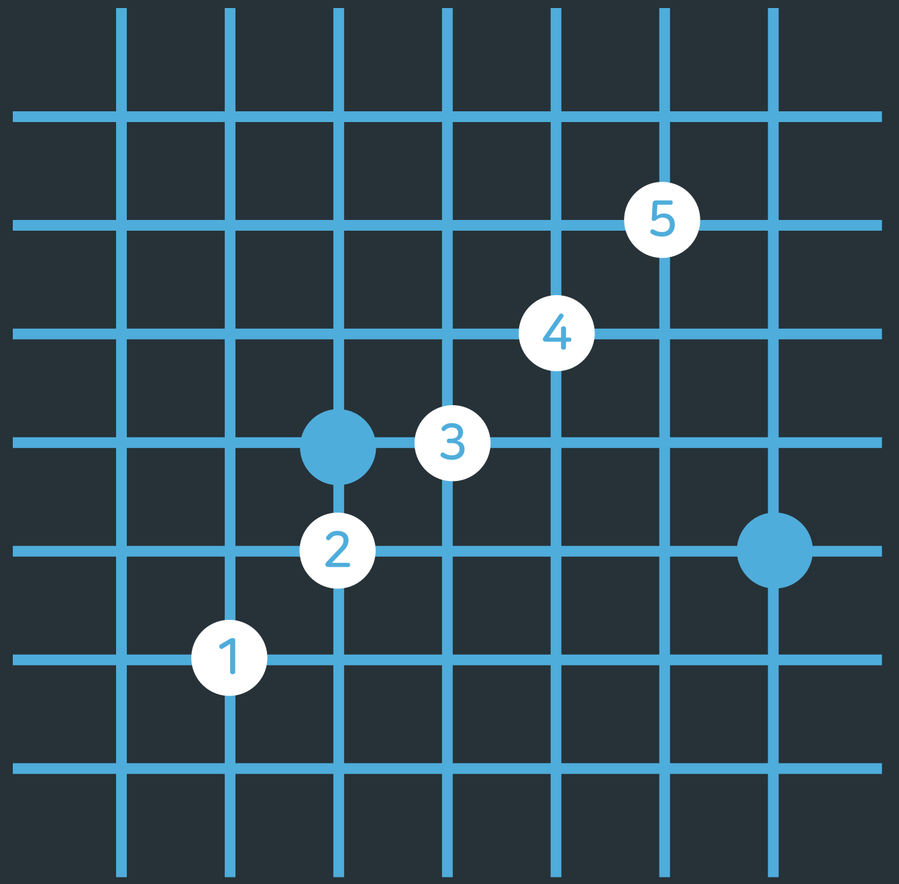


↓ (1, 0)

접근 방법



↘ (1, 1)



↗ (-1, 1)

접근 방법

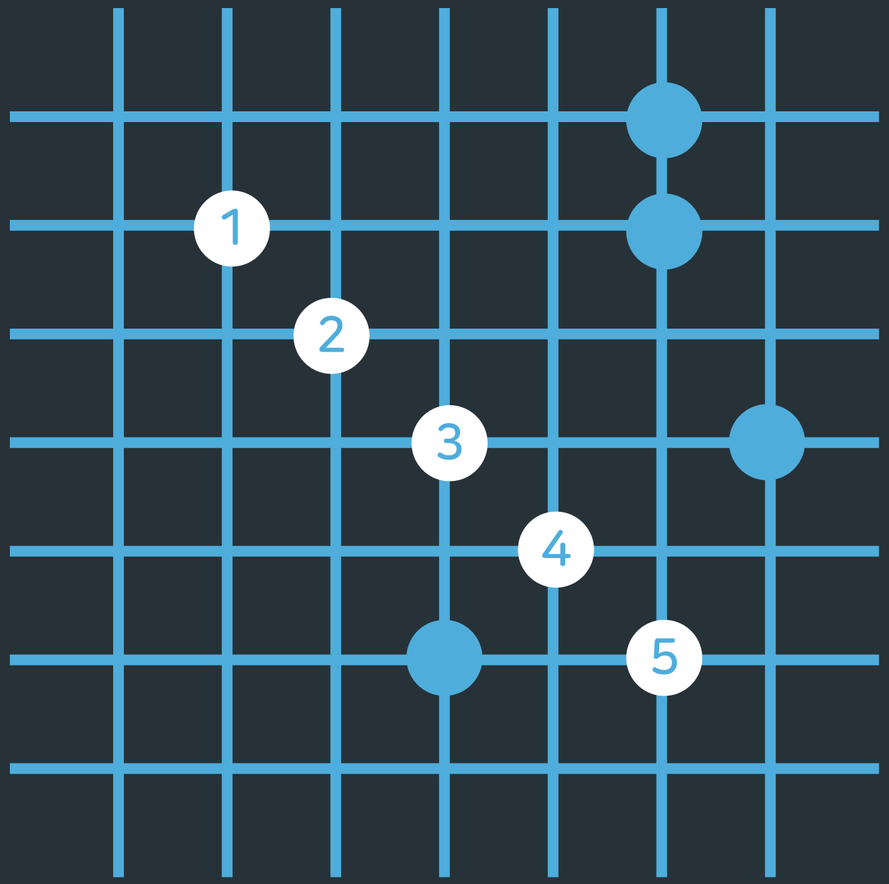
- 탐색하는 방향으로 바둑알이 6개 이상 존재하는 경우 오목 실패

  → 다섯 알까지만 연속한지 확인하고, 여섯 알 째는 연속하지 않는지를 확인

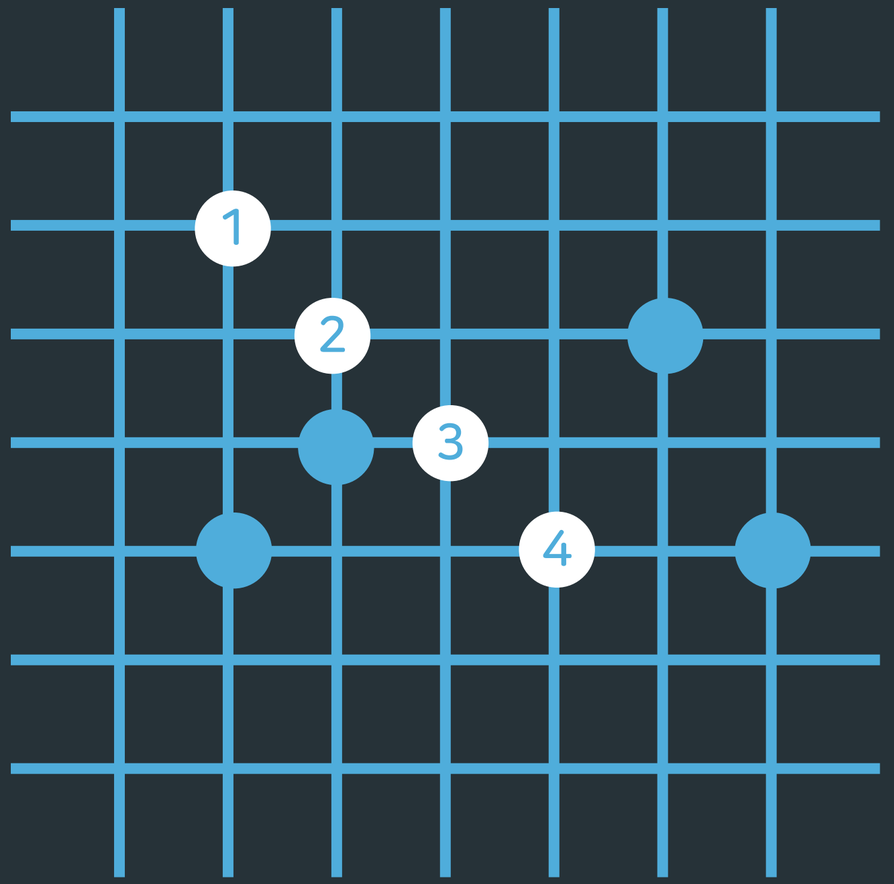
- 탐색 방향으로 5알이 연속이면 무조건 승리?

  ❌ 탐색 반대 방향으로 같은 색의 바둑알이 있으면 실패!

접근 방법



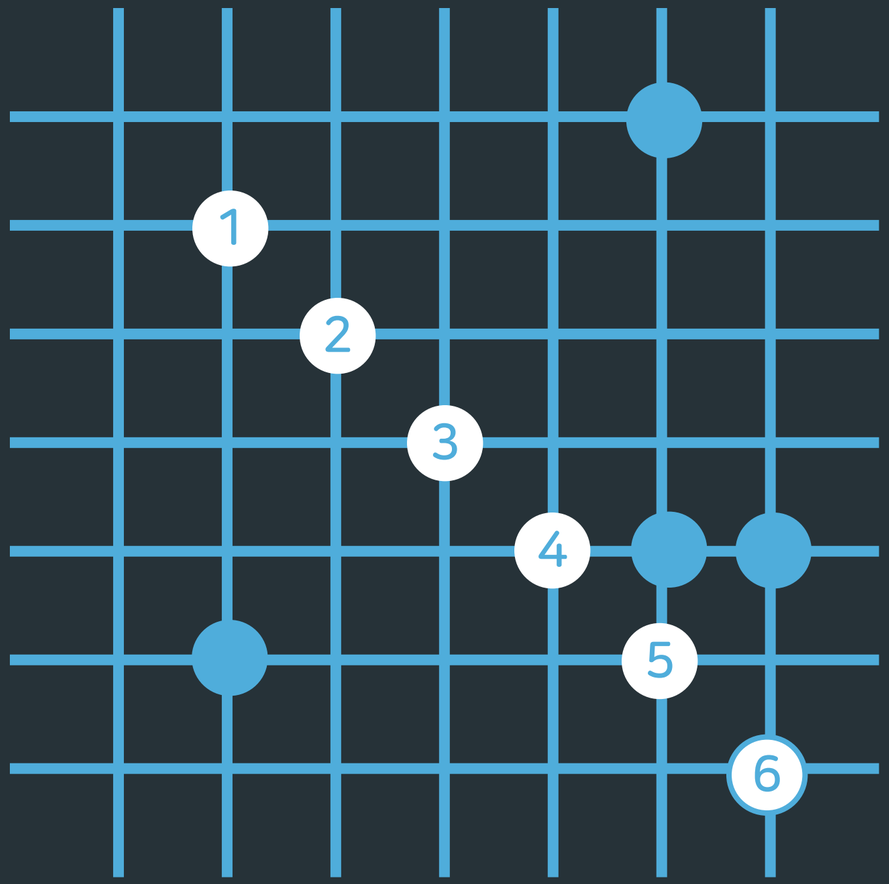
성공



실패

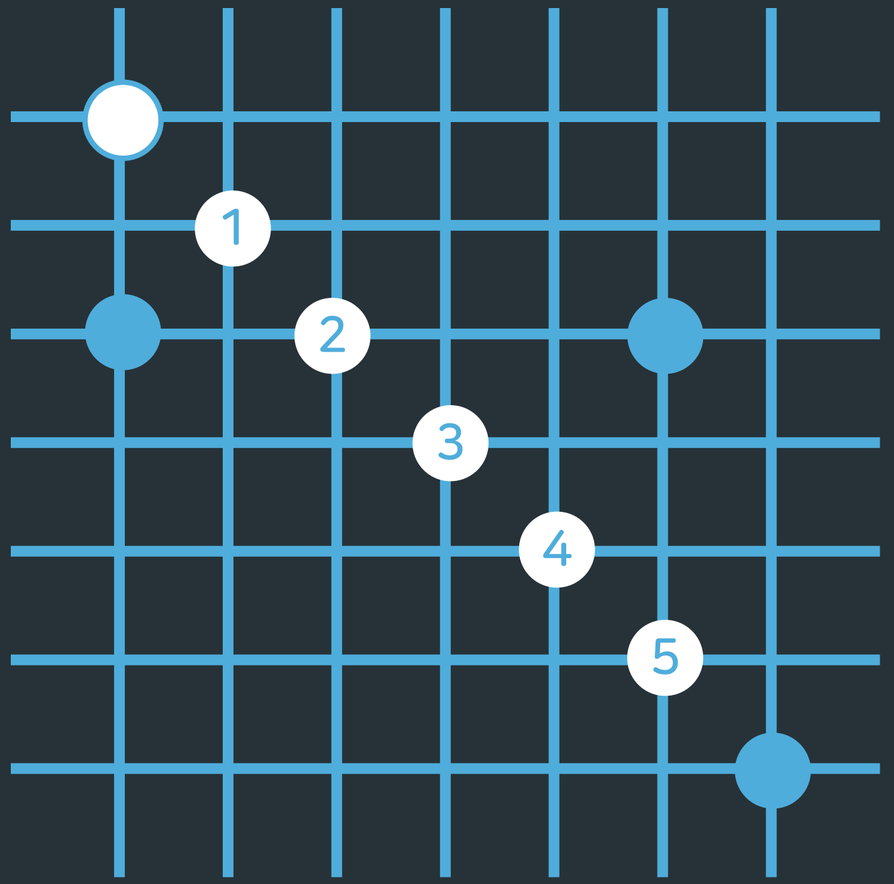
: 4알 연속

접근 방법



실패

: 탐색 방향으로 6알 이상 연속



실패

: 탐색 반대 방향으로 6알 이상 연속

마무리

#알튜비튜

추가로 풀어보면 좋은 문제!

- 1012번 : 유기농 배추 – Silver 2
- 2667번 : 단지번호붙이기 – Silver 1
- 7576번 : 토마토 – Gold 5
- 17471번 : 게리맨더링 – Gold 4