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

도전 문제

#알튜비튜

/<> 14500번 : 테트로미노 - Gold 4

문제

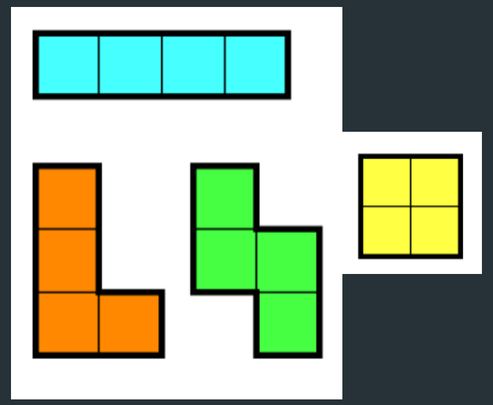
- 정사각형 4개를 이어 붙인 폴리오미노는 테트로미노라고 함
- n*m 각각의 칸에는 정수가 하나 쓰여있음
- 테트로미노가 놓인 칸에 쓰인 수들의 합의 최댓값을 출력

제한 사항

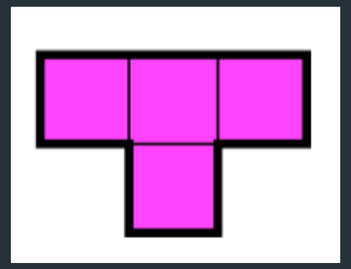
- 세로 크기 N, 가로 크기 M, ($4 \le N, M \le 500$)

테트로미노의 특징

테트로미노의 특징



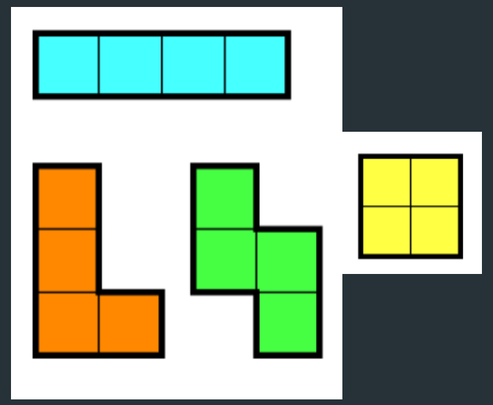
DFS 함수로 설계 가능한 테트로미노



DFS 함수로 설계 불가능
→ 해당 테트로미노에 대해 다른 설계가 필요함!

테트로미노의 특징

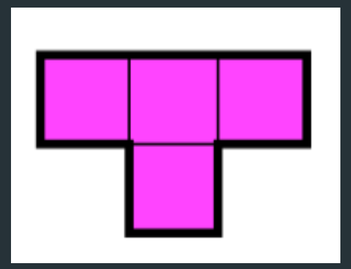
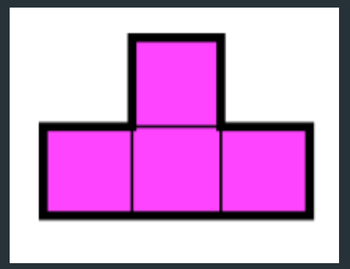
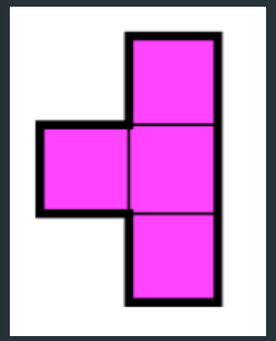
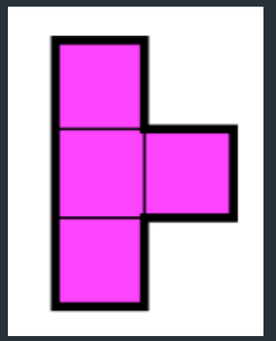
테트로미노의 특징 DFS 함수로 설계 가능한 테트로미노



- DFS 함수로 설계 가능하다는 의미는?
- 정사각형 4개로 이루어져 있기 때문에, 상하좌우 방향으로 탐색을 하면 됨
  → 즉, depth가 4일 때, 값을 리턴하면 됨

테트로미노의 특징

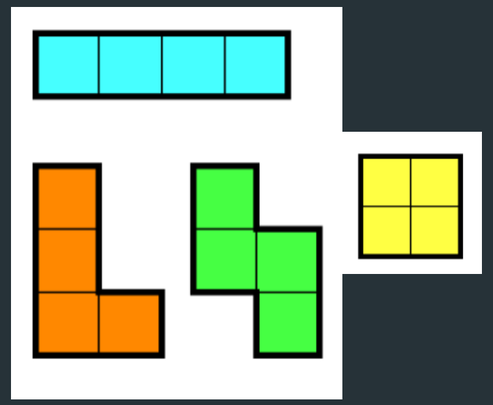
테트로미노의 특징 ‘ㅜ’ 테트로미노로 만들 수 있는 모양 4가지

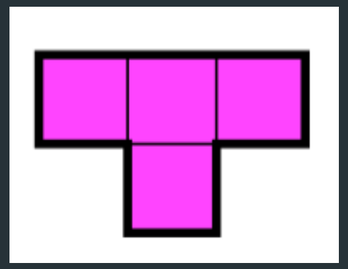





- DFS 함수로 설계 불가능하다는 의미는?
- DFS는 특성상, 한 번 방문한 곳을 더 이상 방문하지 않음
- 그러나, ㅜ 모양은 한 번 방문한 곳을 거치지 않고서는 모양을 만들 수 없음
  → 따라서, 다른 테트로미노와는 다르게 다른 처리가 필요함
- 즉, depth가 4일 때, 값을 리턴하면 됨

풀이

문제 풀이의 핵심





- 테트로미노를 DFS로 탐색한다는 아이디어
- DFS로 탐색하지 못하는 테트로미노에 대해 예외 처리 필요

도전 문제

#알튜비튜

/<> 14502번 : 연구소 - Gold 4

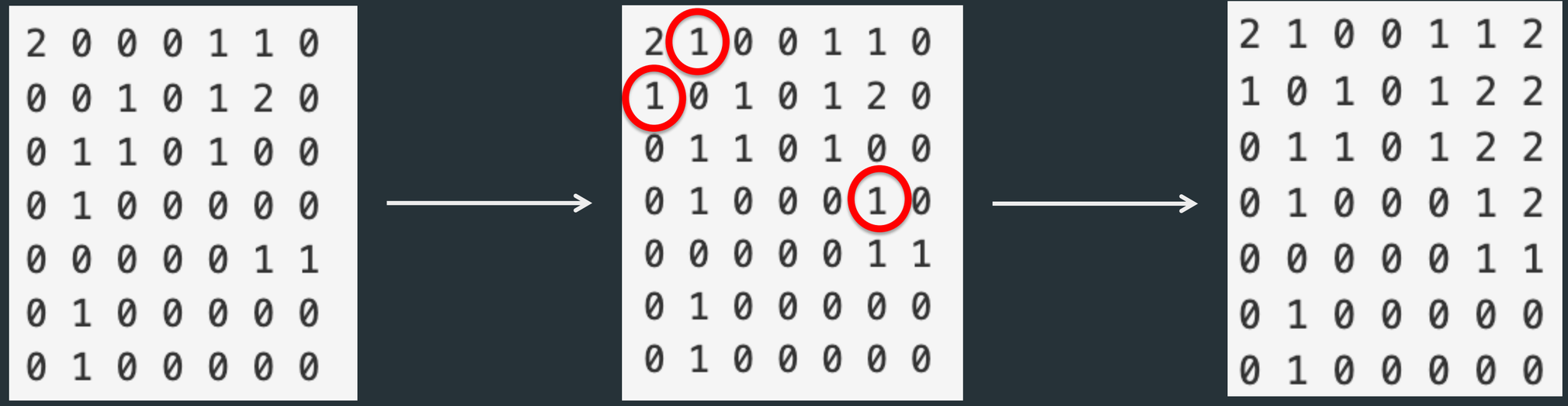
문제

- 연구소는 크기가 N×M인 직사각형으로 나타낼 수 있으며, 직사각형은 1×1 크기의 정사각형으로 나누어져 있음
- 연구소는 빈 칸, 벽으로 이루어져 있으며, 일부 칸에 바이러스가 존재
- 벽을 3개 세워서 바이러스가 더이상 퍼질 수 없는 지역(안전지역) 의 최대 개수를 구하는 문제

제한 사항

- 세로 크기 N, 가로 크기 M, (3 ≤ N, M ≤ 8)

예제



연구소
0: 빈 칸, 1: 벽, 2:바이러스

벽을 3개 세운 후

바이러스가 퍼진 후

풀이

문제 풀이 방법

- 1. 빈 칸(0)에 벽 3개 세울 수 있는 경우 생각
- 2. 벽 3개를 세우고, 해당 경우 바이러스가 퍼질 수 있는 지역 BFS로 탐색
- 3. 바이러스가 퍼지고 나서 안전지대 탐색

문제 풀이 방법

1. 빈 칸(0)에 벽 3개 세울 수 있는 경우 생각

```
void wall(int cnt) {
    if (cnt == 3) {
        bfs(); //벽을 3개 세우면 안전지대 탐색
        return;
    }
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            if (li[i][j] == 0) {
                li[i][j] = 1;
                wall(cnt + 1);
                li[i][j] = 0;
            }
        }
    }
}
```

문제 풀이 방법

2. 벽 3개를 세우고, 해당 경우 바이러스가 퍼질 수 있는 지역 BFS로 탐색

```
void bfs() {
    queue<pair<int, int>> q;
    vector<vector<int>> tmp_map = li;
    for (auto& v : virus) {
        q.push(v);
    }

    while (!q.empty()) {
        int x = q.front().first;
        int y = q.front().second;
        q.pop();

        for (int i = 0; i < 4; i++) {
            int newx = x + dx[i];
            int newy = y + dy[i];
            if (0 <= newx && newx < n && 0 <= newy && newy < m && tmp_map[newx][newy] == 0) {
                tmp_map[newx][newy] = 2;
                q.push(make_pair(newx, newy));
            }
        }
    }
    check_safe_zone(tmp_map);
}
```

문제 풀이 방법

3. 바이러스가 퍼지고 나서 안전지대 탐색

```
void check_safe_zone(const vector<vector<int>>& tmp_map) {
    int count = 0;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            if (tmp_map[i][j] == 0) {
                count++;
            }
        }
    }
    ans = max(ans, count);
}
```

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

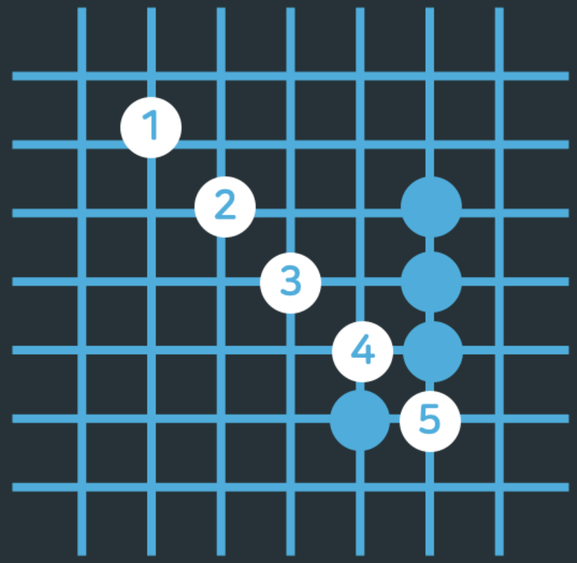
예제 입력

```
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

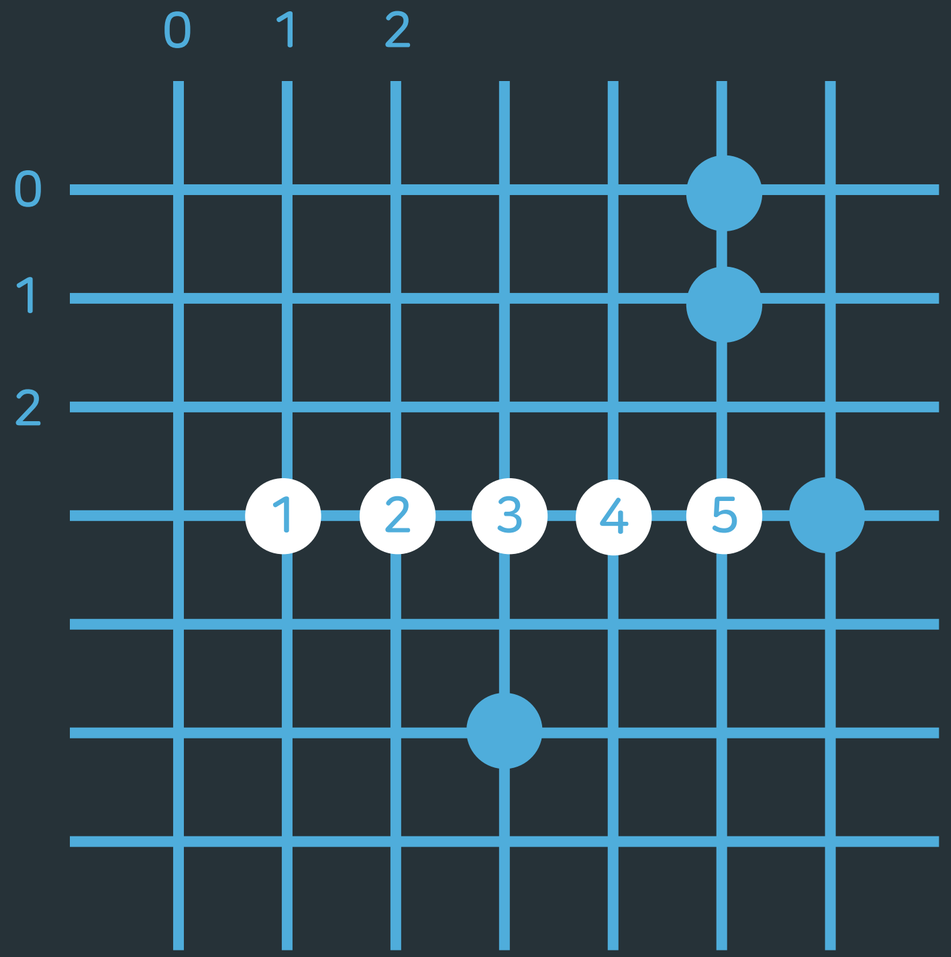
#알튜비튜

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

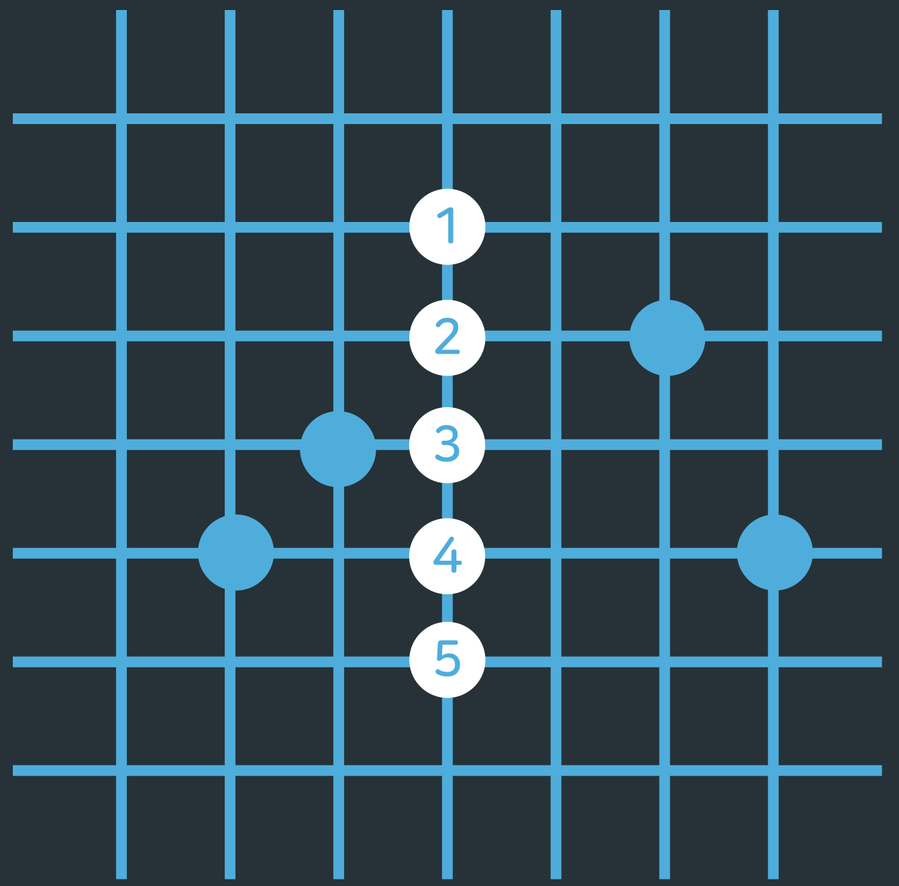
- → (0, 1) : {0, 0}, {0, 1}, {0, 2}, {0, 3}, {0, 4}, {0, 5}, …
- ↓ (1, 0) : {0, 0}, {1, 0}, {2, 0}, {3, 0}, {4, 0}, {5, 0}, …
- ↘ (1, 1) : {0, 0}, {1, 1}, {2, 2}, {3, 3}, {4, 4}, {5, 5}, …
- ↗ (-1, 1) : {0, 0}, {-1, 1}, {-2, 2}, {-3, 3}, {-4, 4}, {-5, 5}, …

⇒ 방법 1: 브루트 포스
⇒ 방법 2: DFS

접근 방법

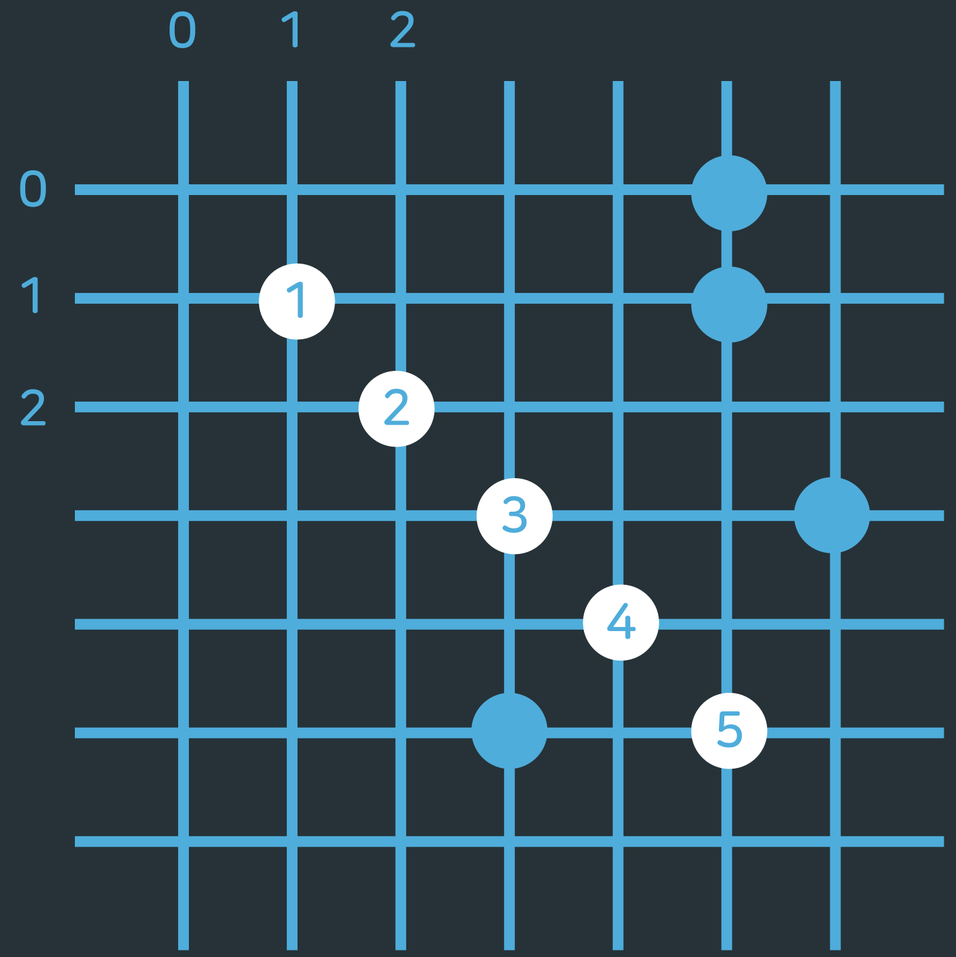


→ (0, 1)

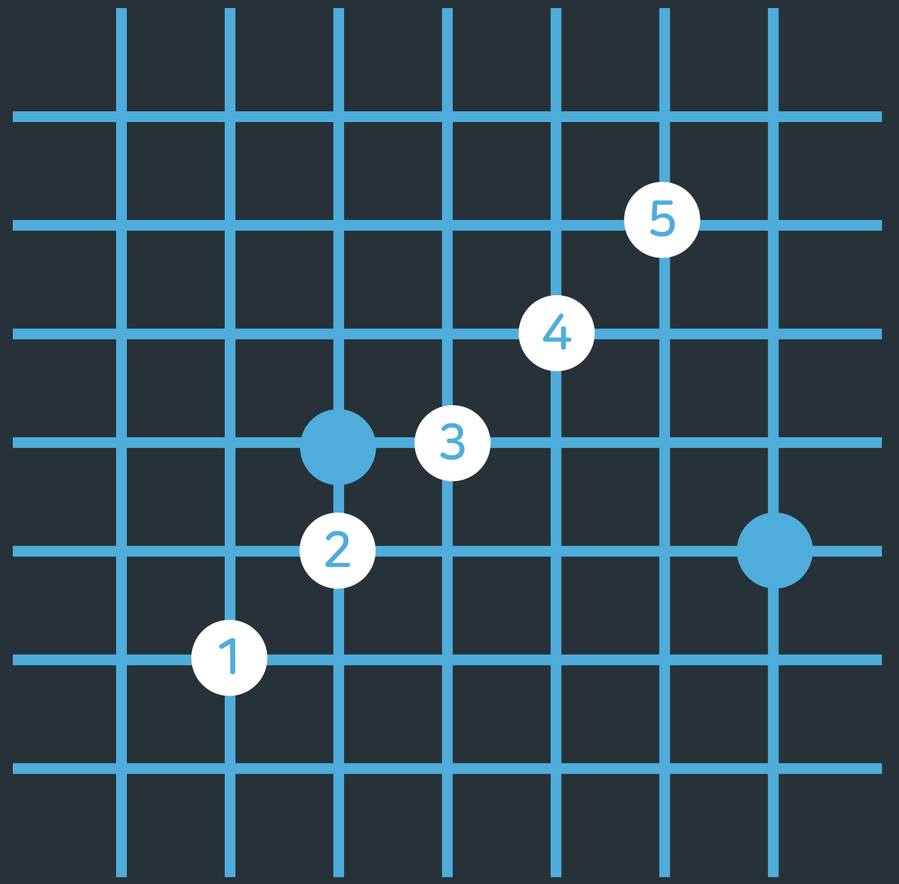


↓ (1, 0)

접근 방법



↘ (1, 1)



↗ (-1, 1)

접근 방법

#알튜비튜

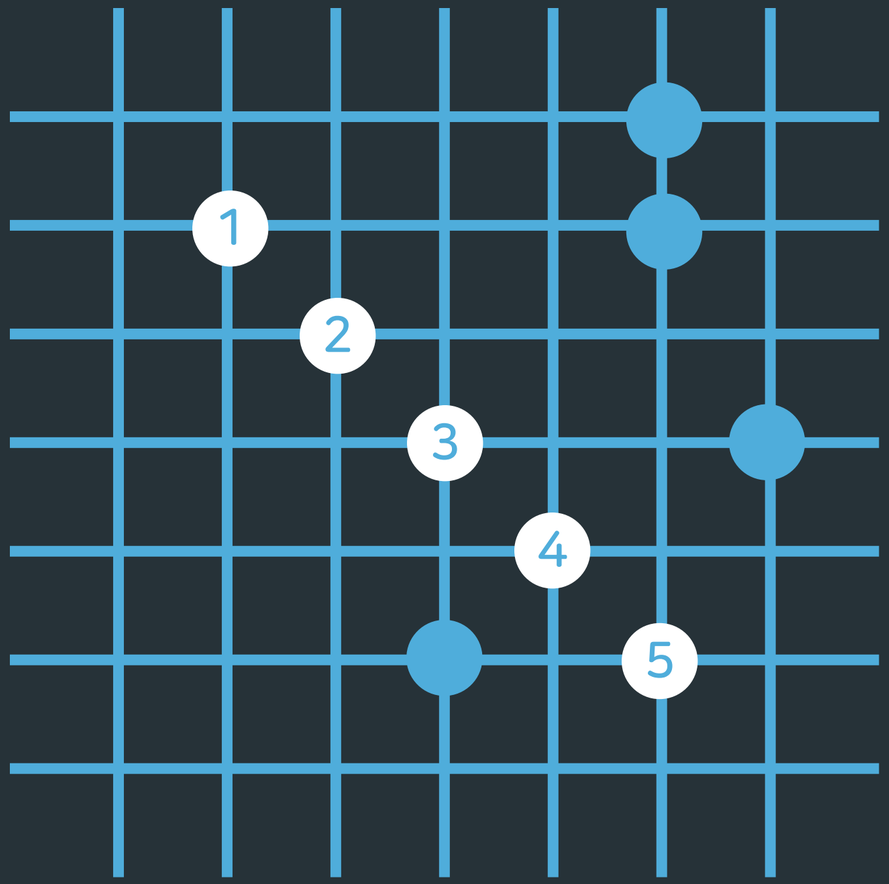
- 탐색하는 방향으로 바둑알이 6개 이상 존재하는 경우 오목 실패

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

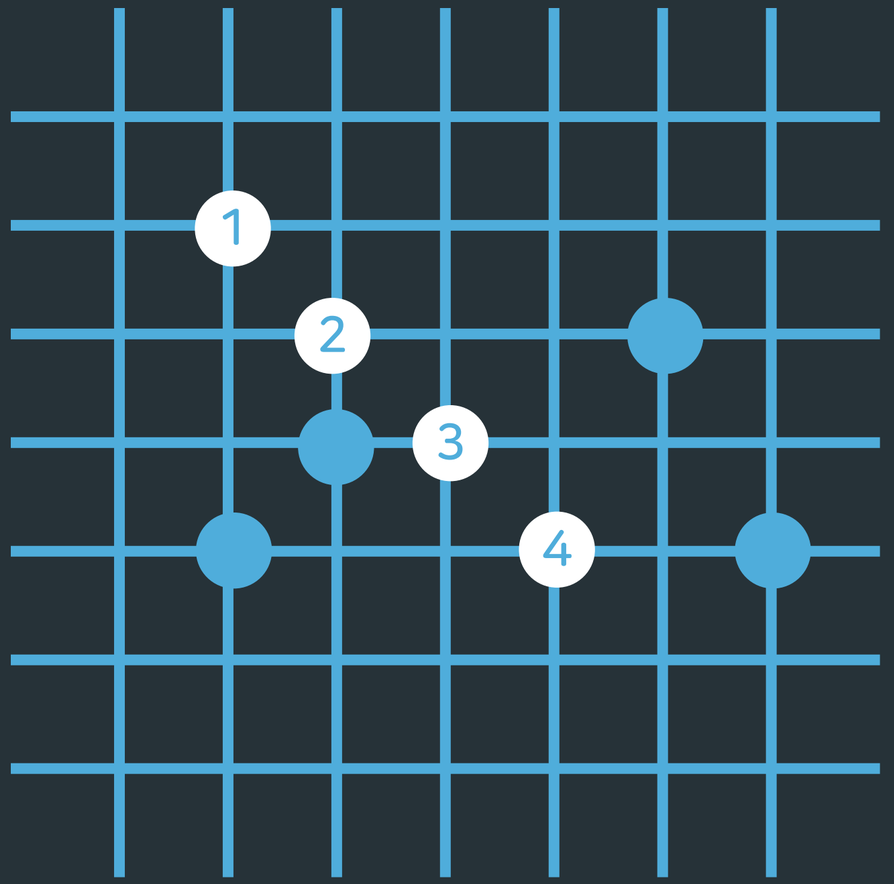
- 탐색 방향으로 5알이 연속이면 무조건 승리?

  🙅‍♀️ 탐색 반대 방향으로 같은 색의 바둑알이 있으면 실패!

접근 방법



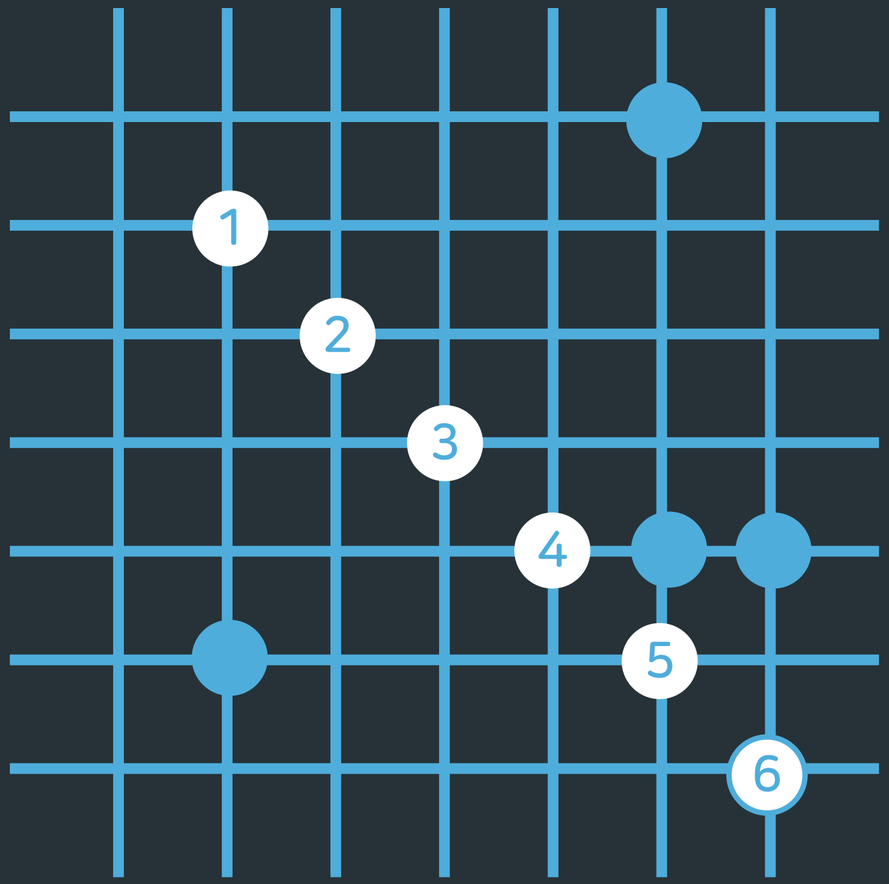
성공



실패

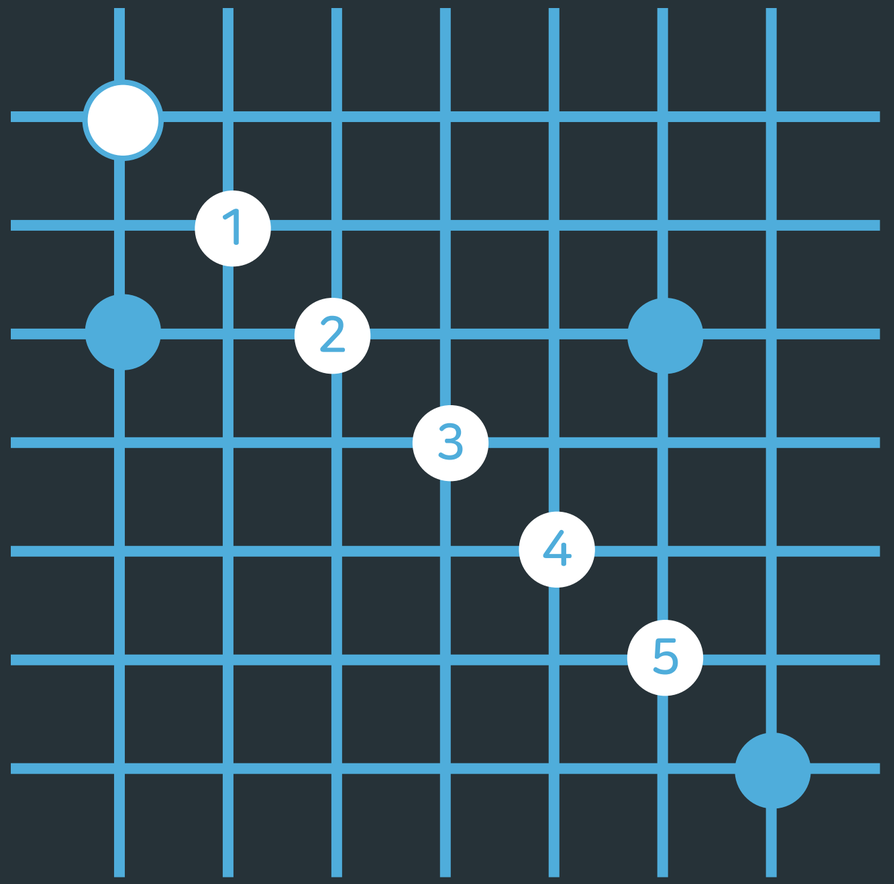
: 4알 연속

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