내 풀이(stack 사용): 우선 내가 생각한 답안의 방향은 maze 알고리즘과 상당히 비슷. 블록은 총 네가지 방향으로 회전할 수 있음. 우선 차례대로 블록은 시작위치에서 네가지 방향으로 돌려보고, 만약 블록에 끼워 맞춰지면 stack에 해당 위치와 다음 블록의 모양은 무엇이 되는지 저장. (물론 칸이 채워진 것들은 visited에 검은 칸으로 바꿔 놔야. 만약 계속 배치하다가 네 방향 모두 배치가 되지 않으면, stack에 있는 전의 위치를 pop해주고, 다른 방향으로 블록을 배치. (물론 바꿀 때 전 위치에 놓인 블록의 visited는 다시 흰색으로). 그렇게 해서 시작 위치로까지 돌아왔을 때 배치 실패.

문제집 풀이(stack 대신 재귀 사용, **완전탐색**): 문제집 풀이는 내 풀이와 비슷하지만 재귀를 썼다는 점에서 시간 복잡도를 줄일 수 있었다. 우선

1. 흰칸을 3으로 나눴을 때 나머지가 0이여야 한다.
2. 만약 나머지가 3이면, 모든 흰칸에서 3으로 나눠 들어가야하는 블록(ㅣ모양)이 몇 개 들어갈 수 있는지 판단한다.
3. 가장 왼쪽의, 윗쪽의 칸을 채우는 것으로 시작을 한다. 여기서, 총 네 번의 재귀를 불러올 것이다. (아래, 오른쪽), (오른쪽🡪아래) (아래🡪오른쪽) (아래🡪왼쪽)

* 참고로 다른 블록과 중복되는 일이 없게끔 네가지 모양을 확실히 정한 것이다.
* 채우는 것이 허용이 되면, 해당 칸은 블랙으로 바꾸기(이때 다른 재귀에 영향을 주면 안되니까 매개변수 복사!! 포인터 참조 불가.)
* 만약 채우는 것이 허용이 되지 않으면 바로 버리기.

1. 이런 식으로 재귀가 호출되었을 때, 해당 재귀가 총 N번(블록 개수)불러와졌을 때 총 count 개수가 1 더해진다.
2. 나머지 변수들은 재귀함수가 호출될때마다 복사되어야 하므로 지역변수이지만, 모든 블록을 채울 수 있는 경우의 수는 전역변수에 해당. 얘만!!

6.7 최적화 완전탐색 문제

오늘 해야하는 일 정리해보기