퍼즐 조각 채우기

# Index	84021
CreatedAt	@September 28, 2022
Person	A Ally Hyeseong Kim
Status	Done
Tags	BFS/DFS Python
UpdatedAt	@September 28, 2022

References

https://school.programmers.co.kr/learn/courses/30/lessons/84021

1. Breadth First Search

```
def solution(game_board, table):
    def search_shape(board, start, color):
        shape = []

        stack = [start]
        while stack:
            row, col = stack.pop()
            shape.append((row, col))
            for dr, dc in [(-1, 0), (1, 0), (0, -1), (0, 1)]:
                if row + dr < 0 or row + dr > len(board) - 1 or col + dc < 0 or col + dc > len(board[0]) - 1:
                    continue
                if board[row + dr][col + dc] != color:
                    continue
                board[row + dr][col + dc] = 1 - color
                stack.append((row + dr, col + dc))

        start = [len(board), len(board[0])]
        for s in shape:
            start[0] = min(start[0], s[0])
            start[1] = min(start[1], s[1])
        for i in range(len(shape)):
            shape[i] = (shape[i][0] - start[0], shape[i][1] - start[1])

        return board, set(shape)

    def rotate(origin):
        rotated_shapes = list()

        rotated = list(origin)
        for _ in range(4):
            n = 0
            for r in rotated:
                n = max(n, r[0])
            for i in range(len(rotated)):
                rotated[i] = (rotated[i][1], n - rotated[i][0])
            rotated_shapes.append(set(rotated))

        return rotated_shapes

    # table 도형 탐색 + rotate
    shapes = dict()
    rotates = dict()
    for i in range(len(table)):
        for j in range(len(table[0])):
            if table[i][j]:
                table[i][j] = 0
                table, shape = search_shape(table, [i, j], 1)
                shapes[len(shape)] = shapes.get(len(shape), []) + [rotate(shape)]

    # game_board 도형 탐색
    game_shapes = dict()
    for i in range(len(game_board)):
        for j in range(len(game_board[0])):
            if game_board[i][j] == 0:
                game_board[i][j] = 1
                game_board, shape = search_shape(game_board, [i, j], 0)
                game_shapes[len(shape)] = game_shapes.get(len(shape), []) + [shape]

    # game
    visited = dict()
    result = 0
    for cur_len in game_shapes:
        count = 0
        visited = dict()
        for shape in game_shapes[cur_len]:
            found = False
            for i in range(len(shapes.get(cur_len, []))):
                if visited.get(i, False):
                    continue
                for trial in shapes[cur_len][i]:
                       if shape == trial or not trial:
                            visited[i] = True
                            count += 1
                            found = True
                            break
                if found:
                    break
        result += cur_len * count

    return result
```