Computing Foundations for Data Science
HW #11 Sample Solution

- 예시코드와 다른 방법으로도 구현 가능합니다.

```
from collections import deque
# from typing import List
def P1_ans(room: List[list]) -> int:
    m = len(room)
    n = len(room[0])
    queue = deque([])

    # 처음 방귀 위치 queue에 넣기
    for i in range(m):
        for j in range(n):
            if room[i][j] == 1:
                # (row 좌표, col 좌표, 시간-1)
                queue.append((i, j, 1))
    # BFS
    while len(queue) != 0:
        node = queue.popleft()

        # 아직 visit하지 않은 곳
        if 0 <=room[node[0]][node[1]] <= 1:
            # 방귀 시간 update
            room[node[0]][node[1]] = node[2]

            # 상하좌우에 대해
            adj = [(0,1), (0,-1), (1,0), (-1,0)]
            for i in adj:
                ax = node[0] + i[0]
                ay = node[1] + i[1]

                # 비어있는 공간인 경우에 bfs 진행
                if 0<=ax<m and 0<=ay<n and room[ax][ay] == 0:
                    # tuple 의 세번째 값에 시간 + 1 을해서 queue 에 넣음
                    queue.append((ax, ay, node[2] + 1))

    # 하나라도 0이 있으면 -1을 return
    # 아니면 최대값 return (방귀가 퍼진 마지막 시간)
    ans = 0
    for i in range(m):
        for j in range(n):
            if room[i][j] == 0:
```

```
                return -1
            elif room[i][j] > 0:
                ans = max(ans, room[i][j] - 1)
    return ans


# from typing import List
def P2_ans(n: int, edges: List[tuple]) -> int:
    graph = [[0 for _ in range(n)] for _ in range(n)]

    # adjacency matrix
    for e in edges:
        graph[e[0]-1][e[1]-1] = 1
        graph[e[1]-1][e[0]-1] = 1

    visit = [0] * n

    stack = []
    stack.append(0)
    ans = 0

    # dfs
    while len(stack) != 0:
        v = stack.pop()
        if visit[v] == 0:
            ans += 1
            visit[v] = 1

            # neighborhood 중 방문하지 않은 곳 stack 에 넣음
            for idx in range(n):
                if graph[v][idx] == 1 and visit[idx] == 0:
                    stack.append(idx)

    return ans


# from typing import List
def P3_ans(image: List[list]) -> int:
    ##### Write your Code Here #####
    # 1로 되어 있는 부분을 지나가면서 0으로 변경하는 함수
    def onetozero(x, y):
        if 0<= x < len(image) and 0 <= y < len(image[0]) and image[x][y] == 1:
            image[x][y] = 0
            onetozero(x+1, y)     # 현재 위치 기준 오른쪽으로
            onetozero(x-1, y)     # 현재 위치 기준 왼쪽으로
            onetozero(x, y+1)     # 현재 위치 기준 위쪽으로
            onetozero(x, y-1)     # 현재 위치 기준 아래쪽으로
            onetozero(x+1, y+1)   # 현재 위치 기준 오른쪽 위로
```

```
            onetozero(x+1, y-1)   # 현재 위치 기준 오른쪽 아래로
            onetozero(x-1, y+1)   # 현재 위치 기준 왼쪽 위로
            onetozero(x-1, y-1)   # 현재 위치 기준 왼쪽 아래로
        else:
            return

    count = 0
    for i in range(len(image)):
        for j in range(len(image[0])):
            if image[i][j] == 1:   # 1일 때 (글자일 때)
                onetozero(i, j)    # 이미 지나온 곳이기 때문에 0으로 변경
                count += 1         # 글자 수 1개 추가

    return count
    ##### End of your code #####


def P4_ans(world):
    def dfs(world, visit, m, n, i, j):
    # 같은 섬의 땅을 dfs로 체크하기
        if 0<=i<m and 0<=j<n and world[i][j] == 1 and visit[i][j] == 0:
            visit[i][j] = 1      # visit matrix의 0을 1로 바꾸기
            dfs(world, visit, m, n, i+1, j)   # 현재 위치 기준 오른쪽으로
            dfs(world, visit, m, n, i-1, j)   # 현재 위치 기준 왼쪽으로
            dfs(world, visit, m, n, i, j+1)   # 현재 위치 기준 위로
            dfs(world, visit, m, n, i, j-1)   # 현재 위치 기준 아래로

    m = len(world)
    n = len(world[0])
    ans = 0
    visit = [[0 for _ in range(n)] for _ in range(m)]
    for i in range(m):
        for j in range(n):
            #처음 방문한 땅이면 개수 추가
            if world[i][j] == 1 and visit[i][j] == 0:
                ans += 1
                dfs(world, visit, m, n, i, j)
    return ans
```