Week 6. DFS 및 BFS (2)

📌 백준

2178 미로탐색

문제

요약 : 배열 N x M에서 (1, 1) → (N, M)까지 최단 경로

1	0	1	1	1	1
1	0	1	0	1	0
1	0	1	0	1	1
1	1	1	0	1	1

미로에서 1은 이동할 수 있는 칸을 나타내고, 0은 이동할 수 없는 칸을 나타낸다. 이러한 미로가 주어졌을 때, (1, 1)에서 출발하여 (N, M)의 위치로 이동할 때 지나야 하는 최소의 칸 수를 구하는 프로그램을 작성하시오. 한 칸에서 다른 칸으로 이동할 때, 서로 인접한 칸으로만 이동할 수 있다.

위의 예에서는 15칸을 지나야 (N, M)의 위치로 이동할 수 있다. 칸을 셀 때에는 시작 위치와 도착 위치도 포함한다.

- **입력**
  - 첫째 줄에 두 정수 N, M(2 ≤ N, M ≤ 100)이 주어진다.
  - 다음 N개의 줄에는 M개의 정수로 미로가 주어진다.
  - 각각의 수들은 **붙어서** 입력으로 주어진다.

- **출력**
  - 첫째 줄에 지나야 하는 최소의 칸 수를 출력한다.
  - 항상 도착위치로 이동할 수 있는 경우만 입력으로 주어진다.

예시 및 출력

예제 입력 1 복사

```
4 6
101111
101010
101011
111011
```

예제 출력 1 복사

```
15
```

예제 입력 2 복사

```
4 6
110110
110110
111111
111101
```

예제 출력 2 복사

```
9
```

예제 입력 3 복사

```
2 25
1011101110111011101110111
1110111011101110111011101
```

예제 출력 3 복사

```
38
```

예제 입력 4 복사

```
7 7
1011111
1110001
1000001
1000001
1000001
1000001
1111111
```

예제 출력 4 복사

```
13
```

구현 아이디어

1. BFS로 탐색 중 N, M에 도달하면 지나간 경로의 합 출력

코드

```
from collections import deque

# (1, 1) -> (N, M) 으로 이동
N, M = [int(n) for n in input().split()]
map = [[int(s) for s in input()] for _ in range(N)]

VISITABLE = 1
VISITED = 2

DX = [0, 0, -1, 1]
DY = [-1, 1, 0, 0]


def bfs():
    while q:
        x, y, cnt = q.popleft()

        for i in range(4):
            nx, ny = x + DX[i], y + DY[i]
            if 0 <= nx < M and 0 <= ny < N and map[ny][nx] == VISITABLE:
                if ny == N - 1 and nx == M - 1:
                    return cnt + 1
                q.append([nx, ny, cnt + 1])
                map[ny][nx] = VISITED


# x, y, cnt
q = deque([])
q.append([0, 0, 1])
map[0][0] = VISITED
print(bfs())
```

2468 안전영역

문제

요약 : 장마철 잠기는 영역에서 살아 남는 섬이 최대인 경우는?

재난방재청에서는 많은 비가 내리는 장마철에 대비해서 다음과 같은 일을 계획하고 있다. 먼저 어떤 지역의 높이 정보를 파악한다. 그 다음에 그 지역에 많은 비가 내렸을 때 물에 잠기지 않는 안전한 영역이 최대로 몇 개가 만들어 지는 지를 조사하려고 한다. 이때, 문제를 간단하게 하기 위하여, 장마철에 내리는 비의 양에 따라 일정한 높이 이하의 모든 지점은 물에 잠긴다고 가정한다.

- **입력**
  - 첫째 줄에는 어떤 지역을 나타내는 2차원 배열의 행과 열의 개수를 나타내는 수 N이 입력된다.
  - N은 2 이상 100 이하의 정수이다.
  - 둘째 줄부터 N개의 각 줄에는 2차원 배열의 첫 번째 행부터 N번째 행까지 순서대로 한 행씩 높이 정보가 입력된다.
  - 각 줄에는 각 행의 첫 번째 열부터 N번째 열까지 N개의 높이 정보를 나타내는 자연수가 빈 칸을 사이에 두고 입력된다. 높이는 1이상 100 이하의 정수이다.

- **출력**
  - 첫째 줄에 장마철에 물에 잠기지 않는 안전한 영역의 최대 개수를 출력한다.

예시 및 출력

예제 입력 1 복사

```
5
6 8 2 6 2
3 2 3 4 6
6 7 3 3 2
7 2 5 3 6
8 9 5 2 7
```

예제 출력 1 복사

```
5
```

예제 입력 2 복사

```
7
9 9 9 9 9 9 9
9 2 1 2 1 2 9
9 1 8 7 8 1 9
9 2 7 9 7 2 9
9 1 8 7 8 1 9
9 2 1 2 1 2 9
9 9 9 9 9 9 9
```

예제 출력 2 복사

```
6
```

구현 아이디어

1. map에서의 max 값을 찾아 max - 1인 각 경우에 대해서 독립적인 map 생성
2. map은 1: 방문가능, 2: 방문 했음, 0: 방문 불가능 으로 정의함
3. map을 가지고 BFS를 수행할 시, 초기 값 생성하는 부분을 map을 탐색하여 각 경우의 수에 대한 BFS를 수행함

코드

```
from collections import deque

N = int(input())
map = [[int(s) for s in input().split()] for _ in range(N)]


VISITABLE = 1
VISITED = 2


DX = [0, 0, -1, 1]
DY = [-1, 1, 0, 0]

result = []

def bfs(map):
    while q:
        x, y = q.popleft()

        for i in range(4):
            nx, ny = x + DX[i], y + DY[i]

            if 0 <= nx < len(map[0]) and 0 <= ny < len(map) and map[ny][nx] == VISITABLE:
                q.append([nx, ny])
                map[ny][nx] = VISITED

# 큐 선언
q = deque([])

# map에서 max 값 찾기
max_num = max([max(n) for n in map])

# max 만큼 M 탐색
for num in range(1, max_num):
    _map = []
    for y in map:
        _map.append([])
        for x in y:
            if x <= num:
                _map[-1].append(0)
            else:
                _map[-1].append(1)

    visited = 0
```

```
    # 큐 초기화
    for y in range(N):
        for x in range(N):
            if _map[y][x] == VISITABLE:
                q.append([x, y])
                bfs(_map)
                visited += 1
    result.append(visited)

if result:
    print(max(result))
else:
    print(1)
```

2667 단지번호붙이기

문제

요약 : 탐색 가능한 범위내 단지를 묶어 각 상태 출력해라

<그림 1>과 같이 정사각형 모양의 지도가 있다. 1은 집이 있는 곳을, 0은 집이 없는 곳을 나타낸다. 철수는 이 지도를 가지고 연결된 집의 모임인 단지를 정의하고, 단지에 번호를 붙이려 한다. 여기서 연결되었다는 것은 어떤 집이 좌우, 혹은 아래위로 다른 집이 있는 경우를 말한다. 대각선상에 집이 있는 경우는 연결된 것이 아니다. <그림 2>는 <그림 1>을 단지별로 번호를 붙인 것이다. 지도를 입력하여 단지수를 출력하고, 각 단지에 속하는 집의 수를 오름차순으로 정렬하여 출력하는 프로그램을 작성하시오.

0	1	1	0	1	0	0
0	1	1	0	1	0	1
1	1	1	0	1	0	1
0	0	0	0	1	1	1
0	1	0	0	0	0	0
0	1	1	1	1	1	0
0	1	1	1	0	0	0

<그림 1>

0	1	1	0	2	0	0
0	1	1	0	2	0	2
1	1	1	0	2	0	2
0	0	0	0	2	2	2
0	3	0	0	0	0	0
0	3	3	3	3	3	0
0	3	3	3	0	0	0

<그림 2>

- **입력**
  - 첫 번째 줄에는 지도의 크기 N(정사각형이므로 가로와 세로의 크기는 같으며 5≤N≤25)이 입력되고, 그 다음 N줄에는 각각 N개의 자료(0혹은 1)가 입력된다..

- **출력**
  - 첫 번째 줄에는 총 단지수를 출력하시오.
  - 그리고 각 단지내 집의 수를 오름차순으로 정렬하여 한 줄에 하나씩 출력하시오.

예시 및 출력

예제 입력 1 복사

```
7
0110100
0110101
1110101
0000111
0100000
0111110
0111000
```

예제 출력 1 복사

```
3
7
8
9
```

구현 아이디어

1. 큐 초기화하는 부분에서 단지의 첫 번째를 찾기 위해 조회하고, BFS를 매번 호출한다.

코드

```
from collections import deque
N = int(input())
map = [[int(s) for s in input()] for _ in range(N)]

VISITABLE = 1
VISITED = 2
```

```
DX = [0, 0, -1, 1]
DY = [-1, 1, 0, 0]


visited = []
q = deque([])


def bfs():
    visited.append(1)

    while q:
        x, y = q.popleft()

        for i in range(4):
            nx, ny = x + DX[i], y + DY[i]
            if 0 <= nx < len(map[0]) and 0 <= ny < len(map) and map[ny][nx] == VISITABLE:
                q.append([nx, ny])
                map[ny][nx] = VISITED
                visited[-1] += 1


# 큐 초기값
for y in range(N):
    for x in range(N):
        if map[y][x] == 1:
            q.append([x, y])
            map[y][x] = VISITED
            bfs()


print(len(visited))
visited = sorted(visited)
for v in visited:
    print(v)
```

5567 결혼식

문제

요약 : 상근이를 기준으로 깊이 2인 친구들의 합은?

상근이는 자신의 결혼식에 학교 동기 중 자신의 친구와 친구의 친구를 초대하기로 했다. 상근이의 동기는 모두 N명이고, 이 학생들의 학번은 모두 1부터 N까지이다. 상근이의 학번은 1이다.

상근이는 동기들의 친구 관계를 모두 조사한 리스트를 가지고 있다. 이 리스트를 바탕으로 결혼식에 초대할 사람의 수를 구하는 프로그램을 작성하시오.

- **입력**
  - 첫째 줄에 상근이의 동기의 수 n (2 ≤ n ≤ 500)이 주어진다.
  - 둘째 줄에는 리스트의 길이 m (1 ≤ m ≤ 10000)이 주어진다. 다음 줄부터 m개 줄에는 친구 관계 ai bi가 주어진다.

    $(1 \le a_i < b_i \le n)$ a_i와 b_i가 친구라는 뜻이며, b_i와 a_i도 친구관계이다.

- **출력**
  - 첫째 줄에 상근이의 결혼식에 초대하는 동기의 수를 출력한다.

예시 및 출력

예제 입력 1 복사

```
6
5
1 2
1 3
3 4
2 3
4 5
```

예제 출력 1 복사

```
3
```

예제 입력 2 복사

```
6
5
2 3
3 4
4 5
5 6
2 5
```

예제 출력 2 복사

```
0
```

구현 아이디어

1. q의 데이터 자료형에 깊이 관련 값을 추가함

코드

```
# BFS
# 인접 리스트를 활용.
from collections import deque
from collections import defaultdict

def bfs(q, rd, visited, cnt):
    while q:
        node, depth = q.popleft()

        for n in rd[node]:
            if visited[n] == False and depth < 2:
                q.append([n, depth + 1])
                visited[n] = True
                cnt += 1

    return cnt


N = int(input())
M = int(input())
relation = []

for _ in range(M):
    relation.append([int(n) for n in input().split()])

cnt = 0
rd = defaultdict(list)
for r1, r2 in relation:
    rd[r1].append(r2)
    rd[r2].append(r1)

visited = [False] * (len(rd) + 1)


# Definition of Queue
q = deque([])
visited[1] = True
for n in rd[1]:
    q.append([n, 1])
    visited[n] = True
    cnt += 1

if not q:
    print(0)
else:
    print(bfs(q, rd, visited, cnt))
```

14502 연구소

문제

요약 : 상근이를 기준으로 깊이 2인 친구들의 합은?

인체에 치명적인 바이러스를 연구하던 연구소에서 바이러스가 유출되었다. 다행히 바이러스는 아직 퍼지지 않았고, 바이러스의 확산을 막기 위해서 연구소에 벽을 세우려고 한다.

연구소는 크기가 N×M인 직사각형으로 나타낼 수 있으며, 직사각형은 1×1 크기의 정사각형으로 나누어져 있다. 연구소는 빈 칸, 벽으로 이루어져 있으며, 벽은 칸 하나를 가득 차지한다.

일부 칸은 바이러스가 존재하며, 이 바이러스는 상하좌우로 인접한 빈 칸으로 모두 퍼져나갈 수 있다. 새로 세울 수 있는 벽의 개수는 3개이며, 꼭 3개를 세워야 한다.

- **입력**
  - 첫째 줄에 지도의 세로 크기 N과 가로 크기 M이 주어진다. (3 ≤ N, M ≤ 8)
  - 둘째 줄부터 N개의 줄에 지도의 모양이 주어진다. 0은 빈 칸, 1은 벽, 2는 바이러스가 있는 위치이다. 2의 개수는 2보다 크거나 같고, 10보다 작거나 같은 자연수이다.
  - 빈 칸의 개수는 3개 이상이다.

- **출력**
  - 첫째 줄에 얻을 수 있는 안전 영역의 최대 크기를 출력한다.

예시 및 출력

예제 입력 1 복사

```
7 7
2 0 0 0 1 1 0
0 0 1 0 1 2 0
0 1 1 0 1 0 0
0 1 0 0 0 0 0
0 0 0 0 0 1 1
0 1 0 0 0 0 0
0 1 0 0 0 0 0
```

예제 출력 1 복사

```
27
```

예제 입력 2 복사

```
4 6
0 0 0 0 0 0
1 0 0 0 0 2
1 1 1 0 0 2
0 0 0 0 0 2
```

예제 출력 2 복사

```
9
```

예제 입력 3 복사

```
8 8
2 0 0 0 0 0 0 2
2 0 0 0 0 0 0 2
2 0 0 0 0 0 0 2
2 0 0 0 0 0 0 2
2 0 0 0 0 0 0 2
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 3 복사

```
3
```

구현 아이디어

1. mpa의 값이 0인 인덱스를 possible_area 리스트에 추가
2. combinations를 사용해서 가능한 조합을 찾음
3. 각 _map에 대해 bfs를 활용해서 탐색함
4. bfs를 통해 바이러스가 퍼질 수 있는 최대를 계산함
5. 0의 수를 탐색해서 0의 개수가 최대 경우를 찾음

코드

```
from itertools import combinations
import copy
from collections import deque

N, M = [int(n) for n in input().split()]
```

```
map = [[int(s) for s in input().split()] for _ in range(N)]

SAFE = 0
WALL = 1
VIRUS = 2
VIRUS_VISITED = 3

DX = [0, 0, -1, 1]
DY = [-1, 1, 0, 0]


# 3개 영역
possible_area = []
for i in range(N):
    for j in range(M):
        if map[i][j] == SAFE:
            possible_area.append([i, j])

search_idx = list(combinations(possible_area, r=3))
q = deque([])
max_safe_area = -1

def bfs():
    while q:
        x, y = q.popleft()

        for i in range(4):
            nx, ny = x + DX[i], y + DY[i]
            if 0 <= nx < M and 0 <= ny < N and _map[ny][nx] == SAFE:
                q.append([nx, ny])
                _map[ny][nx] = VIRUS_VISITED


# search_idx를 이용한 벽 세우기
for area1, area2, area3 in search_idx:
    _map = copy.deepcopy(map)
    _map[area1[0]][area1[1]] = WALL
    _map[area2[0]][area2[1]] = WALL
    _map[area3[0]][area3[1]] = WALL


    # 최대의 0을 찾기
    result = -1
    for y in range(N):
        for x in range(M):
            if _map[y][x] == VIRUS:
                q.append([x, y])
                _map[y][x] = VIRUS_VISITED
                bfs()

    cnt = 0
    for y in range(N):
        for x in range(M):
            if _map[y][x] == SAFE:
                cnt += 1
    if max_safe_area < cnt:
        max_safe_area = cnt
```

```
print(max_safe_area)
```