8주차 주간학습보고서(5/10~5/16)

분류 1	전공		
팀 명	정융팡팡	제출자	김도연
학습 일시	2021.05.10 17:00 ~ 2021.05.10 19:59	마감 일시	2021.05.10 19:59
제출 일시	2021.05.11 10:25	수정 일시	2021.05.11 10:25

[2021-1 광운스터디그룹]

주간학습보고서

학습주제	DFS & BFS(심화)
학습장소	기념관 2층 로비
학습방법 및 학습개요	지난 주에 풀었던 DFS & BFS 알고리즘을 복습하기 위해 처음 한 시간 동안 백준에서 새로운 문제를 풀어보는 연습을 했다. DFS & BFS 문제(이전 주 문제 풀이) 1. 2667_단지번호붙이기 2. 2468_안전영역 3. 7562_나이트의이동 4. 2151_거울 설치(공동문제) 시간 제한을 두고 풀었던 문제 1389_케빈 베이컨의 6단계 법칙
다음 주 계획	**주제 : 그래프** **일시 : 5.17** **장소 : 기념관 2층 로비** **보고서 제출자 : 김도연** 다음주 문제: 1. 10451_순열 사이클 2. 11725_트리의 부모 찾기 3. 1991_트리 순회 4. 5639_이진 검색 트리

* 학습개요의 내용을 구체적으로 작성해 주세요.

DFS & BFS 문제(이전 주 문제 풀이)
1. 2667_단지번호붙이기

문제

<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>

<김도연>

```python
def dfs(x,y, count):
  # 주어진 범위를 벗어나는 경우에는 즉시 종료
  if x<=-1 or x>=n or y<=-1 or y>=n:
    return False
  if graph[x][y] == 1:
    graph[x][y] = 0
    count=dfs(x-1,y,count)+dfs(x+1,y,count)+dfs(x,y-1,count)+dfs(x,y+1,count)+1
    return count
  return False
  # 현재 노드를 아직 방문하지 않았다면

n = int(input())

graph = []
for i in range(n):
  graph.append(list(map(int, input())))

result = []
for i in range(n):
  for j in range(n):
    # 현재 위치에서 DFS 수행
    count= dfs(i,j, 0)
    if count!=False:
      result.append(count)
print(len(result))
for i in sorted(result):
  print(i)
```

2. 2468_안전영역

문제

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.

어떤 지역의 높이 정보는 행과 열의 크기가 각각 N인 2차원 배열 형태로 주어지며 배열의 각 원소는 해당 지점의 높이를 표시하는 자연수이다. 예를 들어, 다음은 N=5인 지역의 높이 정보이다.

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

이제 위와 같은 지역에 많은 비가 내려서 높이가 4 이하인 모든 지점이 물에 잠겼다고 하자. 이 경우에 물에 잠기는 지점을 회색으로 표시하면 다음과 같다.

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

물에 잠기지 않는 안전한 영역이라 함은 물에 잠기지 않는 지점들이 위, 아래, 오른쪽 혹은 왼쪽으로 인접해 있으며 그 크기가 최대인 영역을 말한다. 위의 경우에서 물에 잠기지 않는 안전한 영역은 5개가 된다(꼭짓점으로만 붙어 있는 두 지점은 인접하지 않는다고 취급한다).

또한 위와 같은 지역에서 높이가 6이하인 지점을 모두 잠기게 만드는 많은 비가 내리면 물에 잠기지 않는 안전한 영역은 아래 그림에서와 같이 네 개가 됨을 확인할 수 있다.

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

이와 같이 장마철에 내리는 비의 양에 따라서 물에 잠기지 않는 안전한 영역의 개수는 다르게 된다. 위의 예와 같은 지역에서 내리는 비의 양에 따른 모든 경우를 다 조사해 보면 물에 잠기지 않는 안전한 영역의 개수 중에서 최대인 경우는 5임을 알 수 있다.

어떤 지역의 높이 정보가 주어졌을 때, 장마철에 물에 잠기지 않는 안전한 영역의 최대 개수를 계산하는 프로그램을 작성하시오.

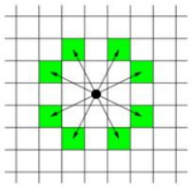
<김도연>

```python
import sys
sys.setrecursionlimit(100000)
r = sys.stdin.readline

# 상 우 하 좌
dx = [-1, 0, 1, 0]
dy = [0, 1, 0, -1]

def dfs(x, y, h):
  for m in range(4):
    nx = x + dx[m]
    ny = y + dy[m]

    if (0 <= nx < N) and (0 <= ny < N) and not visit[nx][ny] and zone[nx][ny] > h:
      visit[nx][ny] = True
      dfs(nx, ny, h)

N = int(r())
zone = [list(map(int, r().split())) for _ in range(N)]

ans = 1
for k in range(max(map(max, zone))):
  visit = [[False]*N for _ in range(N)]
  safe = 0
  for i in range(N):
    for j in range(N):
      if zone[i][j] > k and not visit[i][j]:
        safe += 1
        visit[i][j] = True
        dfs(i, j, k)
  ans = max(ans, safe)

print(ans)
```

3. 7562_나이트의이동

문제

체스판 위에 한 나이트가 놓여져 있다. 나이트가 한 번에 이동할 수 있는 칸은 아래 그림에 나와있다. 나이트가 이동하려고 하는 칸이 주어진다. 나이트는 몇 번 움직이면 이 칸으로 이동할 수 있을까?



입력

입력의 첫째 줄에는 테스트 케이스의 개수가 주어진다.

각 테스트 케이스는 세 줄로 이루어져 있다. 첫째 줄에는 체스판의 한 변의 길이 l(4 ≤ l ≤ 300)이 주어진다. 체스판의 크기는 l × l이다. 체스판의 각 칸은 두 수의 쌍 {0, ..., l-1} × {0, ..., l-1}로 나타낼 수 있다. 둘째 줄과 셋째 줄에는 나이트가 현재 있는 칸, 나이트가 이동하려고 하는 칸이 주어진다.

<이윤희>

```python
import sys
from collections import deque

t = int(sys.stdin.readline())
answer = []

for i in range(t):

    n = int(sys.stdin.readline())
    px, py = map(int, sys.stdin.readline().split())
    tx, ty = map(int, sys.stdin.readline().split())

    is_visited = [[False] * n for _ in range(n)]
    q = deque()
    if px == tx and py == ty:
        pass
    else:
        q.append((px, py))
    is_visited[px][py] = True
    count = 0
    breaker = False

    while q:
        count += 1
        for _ in range(len(q)):
            x, y = q.popleft()
            for a, b in (1, 2), (2, 1), (2, -1), (1, -2), (-1, -2), (-2, -1), (-2, 1), (-1, 2):
                if 0 <= x+a < n and 0 <= y+b < n and not is_visited[x+a][y+b]:
                    if x + a == tx and y + b == ty:
                        # print("정답 찾음")
                        # print(count)
                        breaker = True
                        break
                    q.append((x+a, y+b))
                    is_visited[x+a][y+b] = True

            if breaker:
                break
        if breaker:
            break
    answer.append(count)

for i in range(t):
    print(answer[i])
```

4. 2151_거울 설치(공동문제)

문제

채영이는 거울을 들여다보는 것을 참 좋아한다. 그래서 집 곳곳에 거울을 설치해두고 집 안을 돌아다닐 때마다 거울을 보곤 한다.

채영이는 새 해를 맞이하여 이사를 하게 되었는데, 거울을 좋아하는 그녀의 성격 때문에 새 집에도 거울을 매달만한 위치가 여러 곳 있다. 또한 채영이네 새 집에는 문이 두 개 있는데, 채영이는 거울을 잘 설치하여 장난을 치고 싶어졌다. 즉, 한 쪽 문에서 다른 쪽 문을 볼 수 있도록 거울을 설치하고 싶어졌다.

채영이네 집에 대한 정보가 주어졌을 때, 한 쪽 문에서 다른 쪽 문을 볼 수 있도록 하기 위해 설치해야 하는 거울의 최소 개수를 구하는 프로그램을 작성하시오.

거울을 설치할 때에는 45도 기울어진 대각선 방향으로 설치해야 한다. 또한 모든 거울은 양면 거울이기 때문에 양 쪽 모두에서 반사가 일어날 수 있다. 채영이는 거울을 매우 많이 가지고 있어서 거울이 부족한 경우는 없다고 하자.

거울을 어떻게 설치해도 한 쪽 문에서 다른 쪽 문을 볼 수 없는 경우는 주어지지 않는다.

입력

첫째 줄에 집의 크기 N (2 ≤ N ≤ 50)이 주어진다. 다음 N개의 줄에는 N개의 문자로 집에 대한 정보가 주어진다. '#'는 문이 설치된 곳으로 항상 두 곳이며, '.'은 아무 것도 없는 것으로 빛은 이 곳을 통과한다. '!'은 거울을 설치할 수 있는 위치를 나타내고, '*'은 빛이 통과할 수 없는 벽을 나타낸다.

출력

첫째 줄에 설치해야 할 거울의 최소 개수를 출력한다.

<안희승>

```python
import sys

input = sys.stdin.readline
dx = [0, 1, 0, -1]
dy = [1, 0, -1, 0]

def bfs(x, y, dir):
    q.append([x, y, dir])
    c[x][y][dir] = 1
    ans = []
    while q:
        x, y, dir = q.popleft()
        nx = x + dx[dir]
        ny = y + dy[dir]
        if 0 <= nx < n and 0 <= ny < n:
            if not c[nx][ny][dir] or c[nx][ny][dir] > c[x][y][dir]:
                if a[nx][ny] != '*':
                    c[nx][ny][dir] = c[x][y][dir]
                    if nx == fx and ny == fy:
                        ans.append(c[nx][ny][dir])
                        continue
                    q.append([nx, ny, dir])
                    if a[nx][ny] == '!':
                        turn(nx, ny, dir)

    print(min(ans)-1)

def turn(x, y, dir):
    ndir = [(dir+1) % 4, (dir+3) % 4]
    for d in ndir:
        if not c[x][y][d] or c[x][y][d] > c[x][y][dir] + 1:
            c[x][y][d] = c[x][y][dir] + 1
            q.append([x, y, d])

n = int(input())
q = deque()
c = [[[0]*4 for _ in range(n)] for _ in range(n)]

a, temp = [], []
for i in range(n):
    row = list(input().strip())
    a.append(row)
    for j in range(n):
        if row[j] == '#':
            temp.extend([i, j])
sx, sy, fx, fy = temp

for i in range(4):
    nx = sx + dx[i]
    ny = sy + dy[i]
    if 0 <= nx < n and 0 <= ny < n:
        if a[nx][ny] != '*':
            dir = i
            break

bfs(sx, sy, dir)
```

시간 제한을 두고 풀었던 문제
1. 1389_케빈 베이컨의 6단계 법칙

문제

케빈 베이컨의 6단계 법칙에 의하면 지구에 있는 모든 사람들은 최대 6단계 이내에서 서로 아는 사람으로 연결될 수 있다. 케빈 베이컨 게임은 임의의 두 사람이 최소 몇 단계 만에 이어질 수 있는지 계산하는 게임이다.

예를 들면, 전혀 상관없을 것 같은 인하대학교의 이강호와 서강대학교의 민세희는 몇 단계만에 이어질 수 있을까?

천민호는 이강호와 같은 학교에 다니는 사이이다. 천민호와 최백준은 Baekjoon Online Judge를 통해 알게 되었다. 최백준과 김선영은 같이 Startlink를 창업했다. 김선영과 김도현은 같은 학교 동아리 소속이다. 김도현과 민세희는 같은 학교에 다니는 사이로 서로 알고 있다. 즉, 이강호-천민호-최백준-김선영-김도현-민세희 와 같이 5단계만 거치면 된다.

케빈 베이컨은 미국 헐리우드 영화배우들 끼리 케빈 베이컨 게임을 했을때 나오는 단계의 총 합이 가장 적은 사람이라고 한다.

오늘은 Baekjoon Online Judge의 유저 중에서 케빈 베이컨의 수가 가장 작은 사람을 찾으려고 한다. 케빈 베이컨 수는 모든 사람과 케빈 베이컨 게임을 했을 때, 나오는 단계의 합이다.

방법1) 플로이드-워셜 알고리즘 : 각각의 지점에서 모든 정점에 관한 최단 경로를 구하는 알고리즘
참고한 나무위키(https://namu.wiki/w/%ED%94%8C%EB%A1%9C%EC%9D%B4%EB%93%9C-%EC%9B%8C%EC%85%9C%20%EC%95%8C%EA%B3%A0%EB%A6%AC%EC%A6%98)를요약하자면,
i를 출발해 j로 바로 가는 것보다 I를 출발해 k를 거쳐 j로 가는 게 효율적인 경우(저렴할 경우) 해당 값을 갱신해주는 것이다.
경로 k라 해서 하나의 정점 k만 거치는 것이 아니다. 여러 정점을 거치는 모든 경우의 수가 포함되어 있다.

주가 되는 코드는 다음과 같이 표현할 수 있다.
cost[i][j] = min(cost[i][j], cost[i][k]+cost[k][j])

3중 for 문을 돌아야해서 시간 복잡도는 O(n^3)이다.
주의할 점은 3중 for문 중 경로가 되는 k의 for문이 가장 위에 있어야 누락하지 않고 한 번에 돌 수 있다는 것이다.

```
#입력
N, M = map(int, input().split())
friend_map = [[N for _ in range(N)] for _ in range(N)]

for _ in range(M):
    friend_A, friend_B = map(int, input().split())
    friend_map[friend_A-1][friend_B-1] = 1
    friend_map[friend_B-1][friend_A-1] = 1


#플로이드-워셜 알고리즘
for k in range(N): #경로 for문이 가장 상위 단계여야 누락되지 않는다
    for i in range(N):
        for j in range(N):
            if i == j:
                friend_map[i][j] = 0 #자기 자신과는 친구가 되지 못한다
            else:
                friend_map[i][j] = min(friend_map[i][j],
                                       friend_map[i][k] + friend_map[k][j])


#출력
bacon = []
for row in friend_map:
    bacon.append(sum(row))
print(bacon.index(min(bacon)) + 1)
```

방법2) bfs

```python
import sys
from collections import deque

def bfs(x):
    q = deque()
    q.append((x, 0))
    total = 0

    while q:
        s, d = q.popleft()
        total += d
        for i in arr[s]:
            if not visited[i]:
                visited[i] = True
                q.append((i, d + 1))

    return total


input = sys.stdin.readline
n, m = map(int, input().split())
relations = [list(map(int, input().split())) for _ in range(m)]
arr = [[] for _ in range(n + 1)]
visited = [False for _ in range(n + 1)]
answer = []

for i, j in relations:
    arr[i].append(j)
    arr[j].append(i)

for i in range(1, n + 1):
    visited = [False for _ in range(n + 1)]
    visited[i] = True
    answer.append(bfs(i))

print(answer.index(min(answer)) + 1)
```

* 학습활동 사진(필수)
온라인 활동시 : 시작 화면, 종료 화면(구성원 모두 나온 화면캡쳐)
오프라인 활동시 : 현장 활동 사진(구성원 및 활동 시간이 나오도록 촬영)