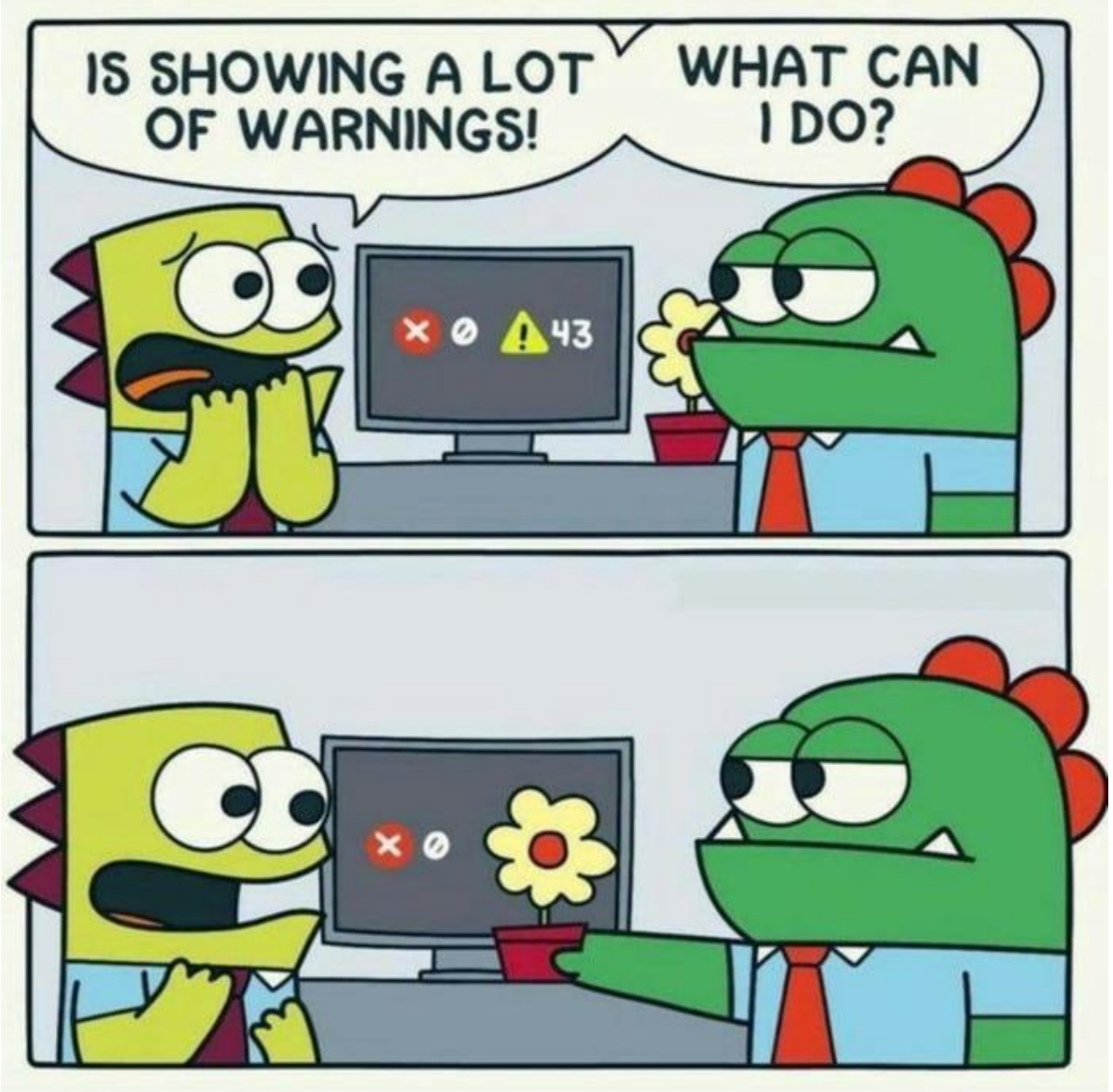
취업 준비 스터디 2주차 멘토링

2022.09.20

발표자 : 박동학

목차

1. 현재 진행상황 공유
2. 문제 분석
3. BFS와 DFS
4. 2차원 배열에서의 이동
5. 실전 시간 복잡도 계산
6. QnA



현재 진행상황 공유

- 현재 스터디원들의 진행 상황 공유 및 회사에 대한 궁금한 점 QnA

현재 마감한 주요 회사

- 카카오 → 9월 24일 코딩테스트
- 라인 -> 9월 24일 코딩테스트
- 삼성그룹 -> 마감 서류 심사중
- 현대차 연구개발 본부 -> 마감 서류 심사중

현재 접수중인 회사

- CJ 그룹, KT
- 우리IFS, 은행들
- LG 전자, LG CNS
- SK CNC, 텔링크

>>>> 대기업이 더 많이 뽑는다.!! 좋은 기업일 수록 사람이 많이 필요

최근 경향

1. 서류의 중요도가 떨어짐

→ 하지만 면접에서 중요하게 작용!! → 성실하게 작성하는 것이 중요

2. 기업들에서 SW 직군을 최대한 빠르게 뽑으려고 전형의 속도를 높이고 있다.

3. 코딩테스트에서 지엽적인 문제보다 빠르고 정확하게 구현하는 것을 요구하고 있다.

→ 그렇다고 알고리즘이 아예 안나오는 것은 아니다.

4. 큰 기업에서 더욱 많이 뽑는다.

→ 스타트업이 들어가기 더 힘들다.

5. 강의나 플랫폼이 좋아져 지원자들의 수준도 높아짐

** 순서 관계가 있는 시뮬레이션 류의 문제가 많이 등장

Ex) Samsung Research 기준 2021년 20명 수준에서 2022년 40명 현재는 100명 채용을 목표로 하고 있음

Ex) 네이버, 카카오도 최대한 많은 기회를 주고자 노력하는 추세

문제 분석

코딩 테스트 문제 분석 – 코딩테스트는 정해진 시간내에 문제를 풀어야 하기 때문에 압박감을 느낄 수 있지만 문제를 제대로 분석하는 것이 오히려 빠르게 푸는 지름길입니다.

코딩테스트 연습 > 2022 KAKAO BLIND RECRUITMENT > **신고 결과 받기**

신고 결과 받기

문제 설명

문제 설명

신입사원 무지는 게시판 불량 이용자를 신고하고 처리 결과를 메일로 발송하는 시스템을 개발하려 합니다. 무지가 개발하려는 시스템은 다음과 같습니다.

- 각 유저는 한 번에 한 명의 유저를 신고할 수 있습니다.
  - 신고 횟수에 제한은 없습니다. 서로 다른 유저를 계속해서 신고할 수 있습니다.
  - 한 유저를 여러 번 신고할 수도 있지만, 동일한 유저에 대한 신고 횟수는 1회로 처리됩니다.
- k번 이상 신고된 유저는 게시판 이용이 정지되며, 해당 유저를 신고한 모든 유저에게 정지 사실을 메일로 발송합니다.
  - 유저가 신고한 모든 내용을 취합하여 마지막에 한꺼번에 게시판 이용 정지를 시키면서 정지 메일을 발송합니다.

다음은 전체 유저 목록이 ["muzi", "frodo", "apeach", "neo"]이고, k = 2(즉, 2번 이상 신고당하면 이용 정지)인 경우의 예시입니다.

유저 ID	유저가 신고한 ID	설명
"muzi"	"frodo"	"muzi"가 "frodo"를 신고했습니다.
"apeach"	"frodo"	"apeach"가 "frodo"를 신고했습니다.
"frodo"	"neo"	"frodo"가 "neo"를 신고했습니다.
"muzi"	"neo"	"muzi"가 "neo"를 신고했습니다.
"apeach"	"muzi"	"apeach"가 "muzi"를 신고했습니다.

1. 한번에 한명씩
2. 인당 신고 횟수 제한 X But 동일 유저에 대한 신고는 1회로 간주
3. 정해진 K번 이상 신고당하면 이용 정지 → 메일 발송
4. 모든 내용은 한꺼번에 마지막에 처리

문제 분석

각 유저별로 신고당한 횟수는 다음과 같습니다.

유저 ID	신고당한 횟수
"muzi"	1
"frodo"	2
"apeach"	0
"neo"	2

위 예시에서는 2번 이상 신고당한 "frodo"와 "neo"의 게시판 이용이 정지됩니다. 이때, 각 유저별로 신고한 아이디와 정지된 아이디를 정리하면 다음과 같습니다.

유저 ID	유저가 신고한 ID	정지된 ID
"muzi"	["frodo", "neo"]	["frodo", "neo"]
"frodo"	["neo"]	["neo"]
"apeach"	["muzi", "frodo"]	["frodo"]
"neo"	없음	없음

따라서 "muzi"는 처리 결과 메일을 2회, "frodo"와 "apeach"는 각각 처리 결과 메일을 1회 받게 됩니다.

이용자의 ID가 담긴 문자열 배열 `id_list` , 각 이용자가 신고한 이용자의 ID 정보가 담긴 문자열 배열 `report` , 정지 기준이 되는 신고 횟수 `k` 가 매개변수로 주어질 때, 각 유저별로 처리 결과 메일을 받은 횟수를 배열에 담아 return 하도록 solution 함수를 완성해주세요.

문제 분석

제한사항

- 2 ≤ `id_list` 의 길이 ≤ 1,000 배열의 길이가 최대 1000개다
  - 1 ≤ `id_list` 의 원소 길이 ≤ 10 배열 속 원소의 길이는 10이다.
  - `id_list` 의 원소는 이용자의 id를 나타내는 문자열이며 알파벳 소문자로만 이루어져 있습니다.
  - `id_list` 에는 같은 아이디가 중복해서 들어있지 않습니다. 중복 없고, 알파벳 소문자만 있다.
- 1 ≤ `report` 의 길이 ≤ 200,000 리포트에는 20만개까지 온다.
  - 3 ≤ `report` 의 원소 길이 ≤ 21
  - `report` 의 원소는 "이용자id 신고한id"형태의 문자열입니다.
  - 예를 들어 "muzi frodo"의 경우 "muzi"가 "frodo"를 신고했다는 의미입니다.
  - id는 알파벳 소문자로만 이루어져 있습니다.
  - 이용자id와 신고한id는 공백(스페이스)하나로 구분되어 있습니다.
  - 자기 자신을 신고하는 경우는 없습니다.
- 1 ≤ `k` ≤ 200, `k` 는 자연수입니다.
- return 하는 배열은 `id_list` 에 담긴 id 순서대로 각 유저가 받은 결과 메일 수를 담으면 됩니다.

입출력 예

id_list	report	k	result
`["muzi", "frodo", "apeach", "neo"]`	`["muzi frodo","apeach frodo","frodo neo","muzi neo","apeach muzi"]`	2	[2,1,1,0]
`["con", "ryan"]`	`["ryan con", "ryan con", "ryan con", "ryan con"]`	3	[0,0]

문제 분석

실제로 제가 푼 풀이 입니다. (여러분들도 IDE사용 가능시 푼 내용을 저장하길 권장드립니다.

당시 점수 --> 486/700 1(OK), 2(OK), 3(OK), 4(OK), 5(정확성), 6(정확성), 7(못품) → 당시 컷 3.5 ~ 4.5

```python
def solution(id_list, report, k):  # 문제의 제한 시간은 10초(정확성만 본다.) --> 10초? --> 무지성 풀이
    answer = []

    sum_report = defaultdict(int)  # 신고 당한 횟수를 이름으로 저장하기 위해
    who_report = defaultdict(list)  # 누가 신고했는지 저장 --> 동일인물은 1번만 신고 가능
    cnt_report = defaultdict(int)  # 신고한 사람들에게 보낼 메일의 수

    report_list = []
    for element in report:  # 모든 신고를 봐야한다. 안 볼수는 없다. --> 최대 20만개
        from_report, to_report = element.split()
        report_list.append([from_report, to_report])

    for element in report_list:  # report_list도 최대 20만개
        if element[0] not in who_report[element[1]]:  # not in은 모두 검사하기 때문에 O(n)소요 | 이미 신고한게 아니면
            who_report[element[1]].append(element[0])  # 새로 신고한 것에 추가하고
            sum_report[element[1]] += 1  # 신고당한 사람 count +1

    for key in sum_report.keys():  # 전체를 다 돌면서 (여기서 key는 신고당한 사람의 이름)
        if sum_report[key] >= k:  # 지정한 k보다 크면
            for element in who_report[key]:  # 신고한 사람들에게 보낼 메일수 +1
                cnt_report[element] += 1

    for name in id_list:  # 정답 배열 생성
        answer.append(cnt_report[name])

    return answer
```

문제 분석

id_list	report	k	result
["muzi", "frodo", "apeach", "neo"]	["muzi frodo","apeach frodo","frodo neo","muzi neo","apeach muzi"]	2	[2,1,1,0]

Sum_report → {'frodo': 2, 'neo': 2, 'muzi': 1}

Who_report → {'frodo': ['muzi', 'apeach'], 'neo': ['frodo', 'muzi'], 'muzi': ['apeach']}

Cnt_report → {'muzi': 2, 'apeach': 1, 'frodo': 1}

["con", "ryan"]	["ryan con", "ryan con", "ryan con", "ryan con"]	3	[0,0]

Sum_report → {'con': 1}

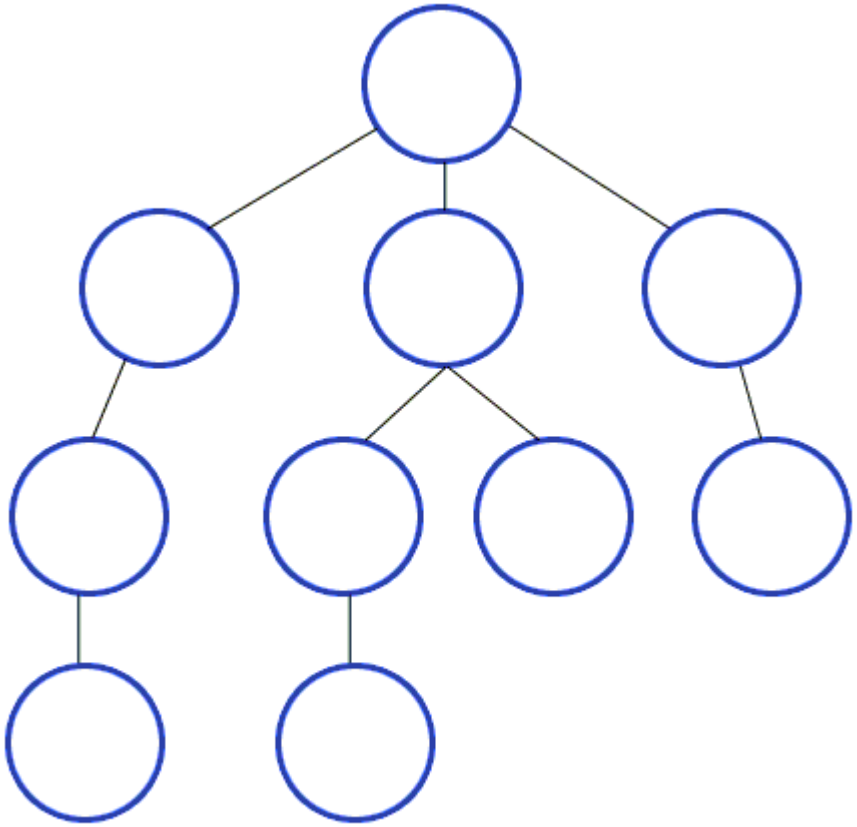
Who_report → {'con': ['ryan']}

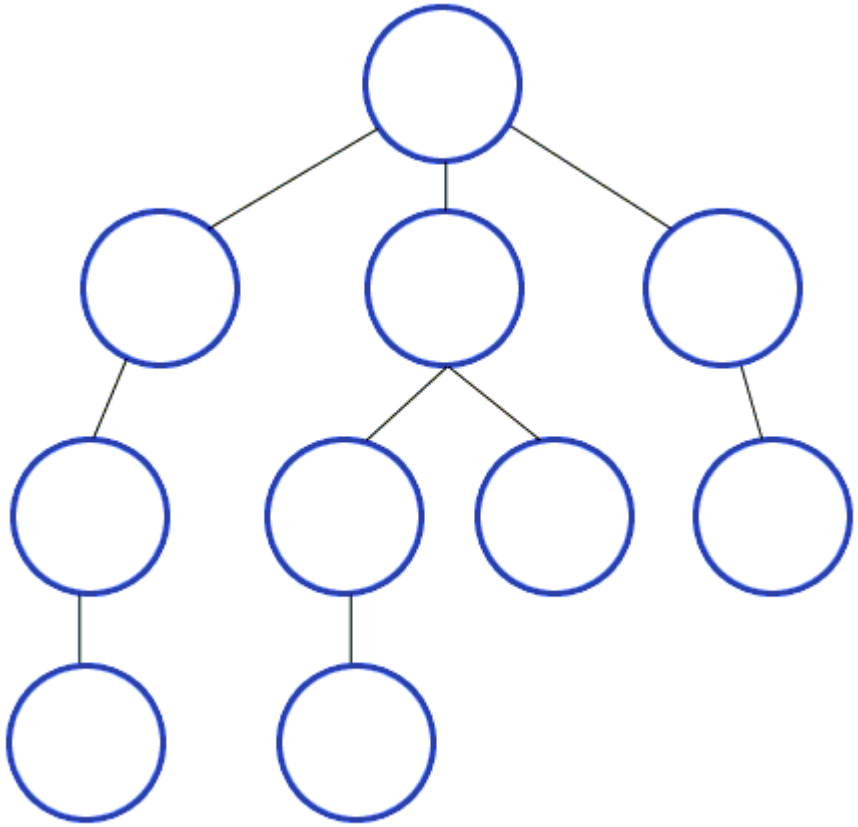
Cnt_report → {}

```
for name in id_list:  # 정답 배열 생성
? answer.append(cnt_report[name])
```

→ ? Cnt_report가 비어있는데 answer.append(Cnt_report[name])을 하면 [0,0]이 나온다!! WHY!!!???

BFS, DFS ➔ Stack와 Queue 그리고 recursive

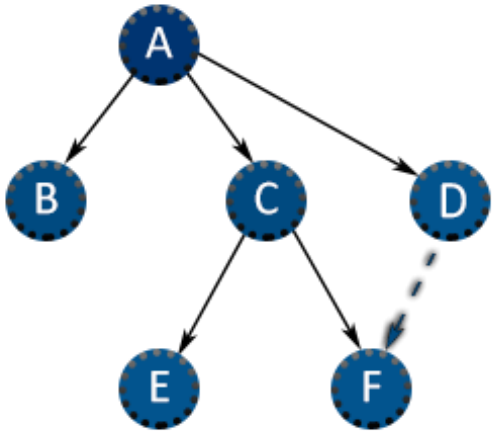


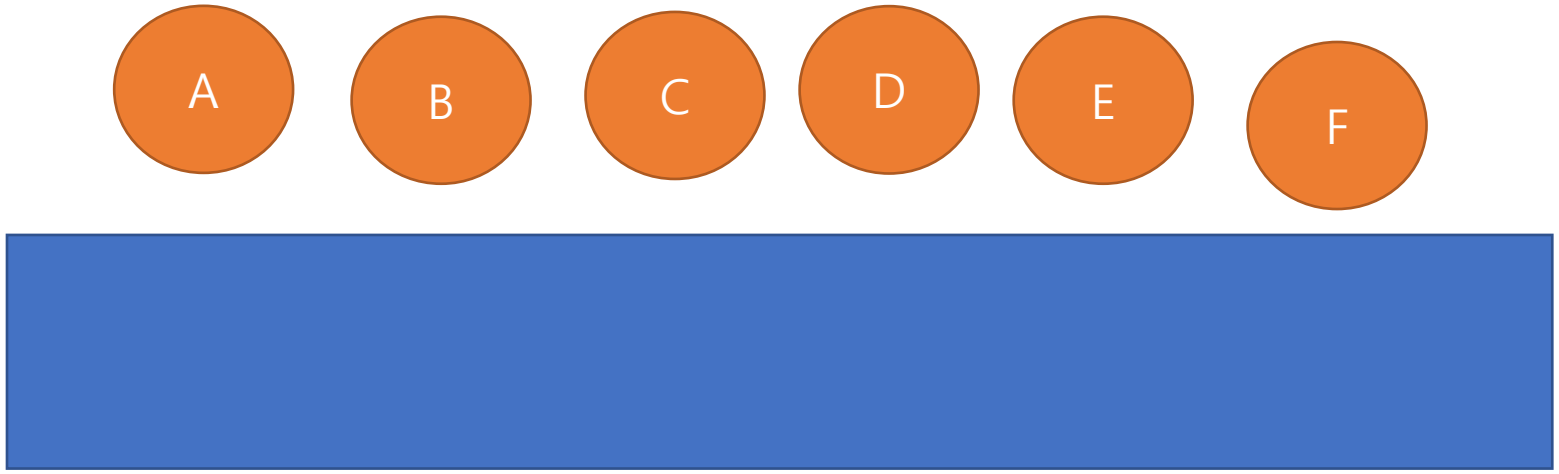


BFS, DFS → Stack와 Queue 그리고 recursive

```
def bfs(graph, start, visited):
    queue = deque([start])
    visited[start] = True

    while queue:
        v = queue.popleft()
        for i in graph[v]:
            if not visited[i]:
                queue.append(i)
                visited[i] = True
```





Graph→

{

A : [B,C,D],

B : [],

C : [E, F],

D : [],

E : [],

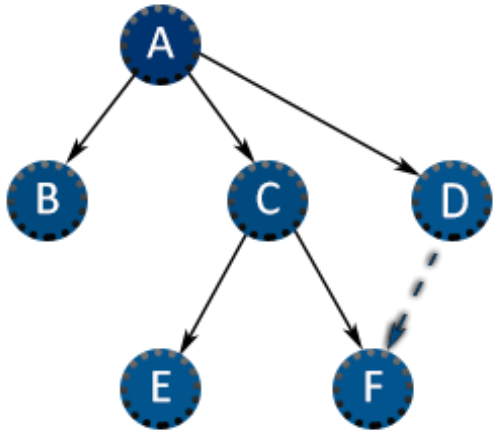
F : [],

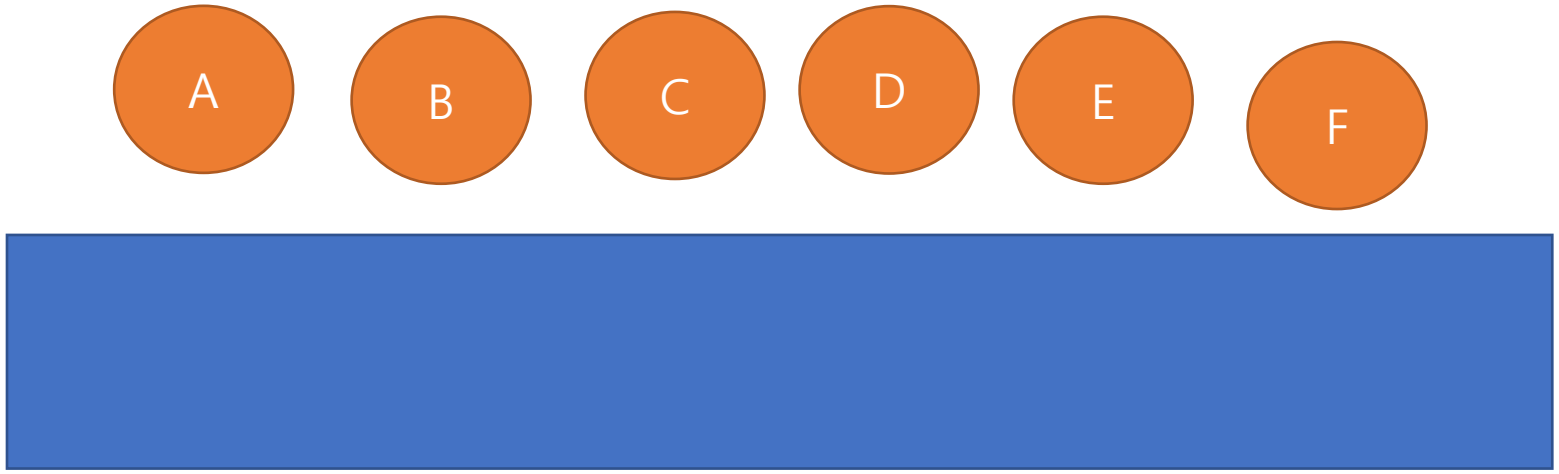
}

BFS, DFS → Stack와 Queue 그리고 recursive

```python
def dfs(graph, v, visited):
    visited[v] = True

    for i in graph[v]:
        if not visited[i]:
            dfs(graph, i, visited)
```





Graph→

{

A : [B,C,D],

B : [],

C : [E, F],

D : [],

E : [],

F : [],

}

2차원 배열에서의 이동

상황 : 2차원 배열이 주어지고 상하좌우를 살펴가면서 탐색해라 or 이동해라 ?

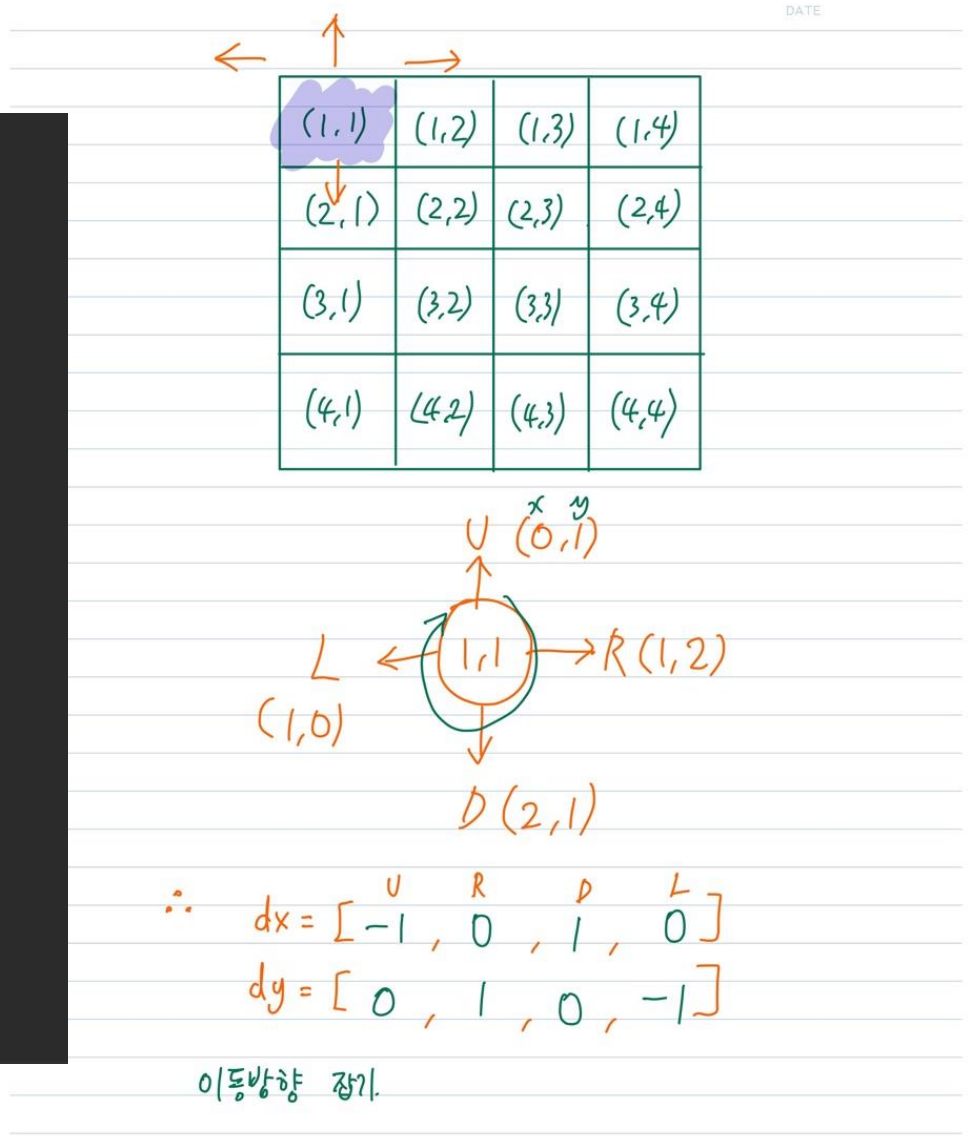
편안한 방법이 있다!!!

```python
#거의 외우듯이 쓰게 될 방법

#상 우 하 좌
dx = [-1, 0, 1, 0]
dy = [0, 1, 0, -1]
x,y = 0, 0
X, Y = 100, 100

for direct in range(4): #4가지 방향에 대해서
    nx = x + dx[direct] #다음 X
    ny = y + dy[direct] #다음 Y

    if 0 <= x < X and 0 <= y < Y: #범위를 벗어나지 않았다면
        if (#원하는 조건):
            #이동 시켜야 하면 이동, Q에 넣어야 하면 넣기
```



미로 탈출 문제

NxN 크기의 직사각형 형태의 미로에 갇혀 있다. 현재 나의 위치는 (0,0)이고 출구는 (N,N)이다. 이때 벽으로 막혀 있는 부분이 있고 이는 0로 표시한다. 미로는 반드시 탈출할 수 있는 형태로 주어지며 이때 탈출을 위한 이동 거리를 구하시오. (출발, 도착지 포함), 단, 대각선으로는 이동 불가능

입력 예시 첫 줄에는 N이 주어지고 다음 N개의 줄에는 미로정보가 주어짐

```
5
101010
111111
000001
111111
111111
```

미로 탈출 문제

NxN 크기의 직사각형 형태의 미로에 갇혀 있다. 현재 나의 위치는 (0,0)이고 출구는 (N,N)이다. 이때 벽으로 막혀있는 부분이 있고 이는 0로 표시한다. 미로는 반드시 탈출할 수 있는 형태로 주어지며 이때 탈출을 위한 이동 거리를 구하시오. (출발, 도착지 포함), 대각선 이동 X

입력 예시 첫 줄에는 N이 주어지고 다음 N개의 줄에는 미로정보가 주어짐

```
5
101010
111111
000001
111111
111111
```

```python
from collections import deque

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

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

def bfs(x, y):
    queue = deque()
    queue.append((x,y))
    while queue:
        x, y = queue.popleft()
        for i in range(4):
            nx = x + dx[i]
            ny = y + dy[i]

            if nx < 0 or ny < 0 or nx >= N or ny >= N:
                continue
            if graph[nx][ny] == 0:
                continue
            if graph[nx][ny] == 1:
                graph[nx][ny] = graph[x][y] + 1
                queue.append((nx, ny))
    return graph[N-1][N-1]

print(bfs(0,0))
```

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

Sort with Python

```python
a = [1,3,2,5,4,8,6,5,4,7]
print(a)
a.sort()
print(a)
a.sort(reverse=True)
print(a)
```

```
[1, 3, 2, 5, 4, 8, 6, 5, 4, 7]
[1, 2, 3, 4, 4, 5, 5, 6, 7, 8]
[8, 7, 6, 5, 5, 4, 4, 3, 2, 1]
```

```python
a = [[1, "a"], [2032, "a"], [2, "c"], [5, "f"], [0, "ZZZ"]]
print(a)
a.sort()
print(a)
a.sort(reverse=True)
print(a)
a.sort(key=lambda x: (-x[0], x[1]))
print(a)
a.sort(key=lambda x: (x[1], x[0]))
print(a)
print(ord("Z"), ord("a"))
```

```
[[1, 'a'], [2032, 'a'], [2, 'c'], [5, 'f'], [0, 'ZZZ']]
[[0, 'ZZZ'], [1, 'a'], [2, 'c'], [5, 'f'], [2032, 'a']]
[[2032, 'a'], [5, 'f'], [2, 'c'], [1, 'a'], [0, 'ZZZ']]
[[2032, 'a'], [5, 'f'], [2, 'c'], [1, 'a'], [0, 'ZZZ']]
[[0, 'ZZZ'], [1, 'a'], [2032, 'a'], [2, 'c'], [5, 'f']]
90 97
```

실전 시간 복잡도 계산

문제 설명

[본 문제는 정확성과 효율성 테스트 각각 점수가 있는 문제입니다.]

N x M 크기의 행렬 모양의 게임 맵이 있습니다. 이 맵에는 내구도를 가진 건물이 각 칸마다 하나씩 있습니다. 적은 이 건물들을 공격하여 파괴하려고 합니다. 건물은 적의 공격을 받으면 내구도가 감소하고 내구도가 0이하가 되면 파괴됩니다. 반대로, 아군은 회복 스킬을 사용하여 건물들의 내구도를 높이려고 합니다.

적의 공격과 아군의 회복 스킬은 항상 직사각형 모양입니다.
예를 들어, 아래 사진은 크기가 4 x 5인 맵에 내구도가 5인 건물들이 있는 상태입니다.

	0	1	2	3	4
0	5	5	5	5	5
1	5	5	5	5	5
2	5	5	5	5	5
3	5	5	5	5	5

첫 번째로 적이 맵의 **(0,0)부터 (3,4)까지 공격하여 4만큼** 건물의 내구도를 낮추면 아래와 같은 상태가 됩니다.

	0	1	2	3	4
0	1	1	1	1	1
1	1	1	1	1	1
2	1	1	1	1	1
3	1	1	1	1	1

실전 시간 복잡도 계산

두 번째로 적이 맵의 **(2,0)부터 (2,3)까지 공격하여 2만큼** 건물의 내구도를 낮추면 아래와 같이 4개의 건물이 파괴되는 상태가 됩니다.

	0	1	2	3	4
0	1	1	1	1	1
1	1	1	1	1	1
2	-1	-1	-1	-1	1
3	1	1	1	1	1

마지막으로 적이 맵의 **(0,1)부터 (3,3)까지 공격하여 1만큼** 건물의 내구도를 낮추면 아래와 같이 8개의 건물이 더 파괴되어 총 10개의 건물이 파괴된 상태가 됩니다. **(내구도가 0 이하가 된 이미 파괴된 건물도, 공격을 받으면 계속해서 내구도가 하락하는 것에 유의해주세요.)**

	0	1	2	3	4
0	1	0	0	0	1
1	3	2	0	0	1
2	1	0	-2	-2	1
3	3	2	0	0	1

최종적으로 총 10개의 건물이 파괴되지 않았습니다.

건물의 내구도를 나타내는 2차원 정수 배열 `board` 와 적의 공격 혹은 아군의 회복 스킬을 나타내는 2차원 정수 배열 `skill` 이 매개변수로 주어집니다. 적의 공격 혹은 아군의 회복 스킬이 모두 끝난 뒤 파괴되지 않은 건물의 개수를 return하는 solution함수를 완성해 주세요.

실전 시간 복잡도 계산

제한사항

- 1 ≤ `board` 의 행의 길이 (= `N`) ≤ 1,000
- 1 ≤ `board` 의 열의 길이 (= `M`) ≤ 1,000
- 1 ≤ `board` 의 원소 (각 건물의 내구도) ≤ 1,000
- 1 ≤ `skill` 의 행의 길이 ≤ 250,000
- `skill` 의 열의 길이 = 6
- `skill` 의 각 행은 `[type, r1, c1, r2, c2, degree]` 형태를 가지고 있습니다.
  - type은 1 혹은 2입니다.
    - type이 1일 경우는 적의 공격을 의미합니다. 건물의 내구도를 낮춥니다.
    - type이 2일 경우는 아군의 회복 스킬을 의미합니다. 건물의 내구도를 높입니다.
  - (r1, c1)부터 (r2, c2)까지 직사각형 모양의 범위 안에 있는 건물의 내구도를 degree 만큼 낮추거나 높인다는 뜻입니다.
    - 0 ≤ r1 ≤ r2 < `board` 의 행의 길이
    - 0 ≤ c1 ≤ c2 < `board` 의 열의 길이
    - 1 ≤ degree ≤ 500
    - type이 1이면 degree만큼 건물의 내구도를 낮춥니다.
    - type이 2이면 degree만큼 건물의 내구도를 높입니다.
- 건물은 파괴되었다가 회복 스킬을 받아 내구도가 1이상이 되면 파괴되지 않은 상태가 됩니다. 즉, 최종적으로 건물의 내구도가 1이상이면 파괴되지 않은 건물입니다.

정확성 테스트 케이스 제한 사항

- 1 ≤ `board` 의 행의 길이 (= `N`) ≤ 100
- 1 ≤ `board` 의 열의 길이 (= `M`) ≤ 100
- 1 ≤ `board` 의 원소 (각 건물의 내구도) ≤ 100
- 1 ≤ `skill` 의 행의 길이 ≤ 100
  - 1 ≤ degree ≤ 100

효율성 테스트 케이스 제한 사항

- 주어진 조건 외 추가 제한사항 없습니다.

입출력 예

board	skill	result
[[5,5,5,5,5],[5,5,5,5,5], [5,5,5,5,5],[5,5,5,5,5]]	[[1,0,0,3,4,4],[1,2,0,2,3,2], [2,1,0,3,1,2],[1,0,1,3,3,1]]	10
[[1,2,3],[4,5,6],[7,8,9]]	[[1,1,1,2,2,4],[1,0,0,1,1,2], [2,2,0,2,0,100]]	6

아!! 그냥 시뮬레이션이구나 -- > 50점/100점

행의 길이 1000 열의 길이 1000 → 1,000,000

Skill 길이 250,000

→ 모든 배열의 전체 다돌면서 실제로 적용하면 ?

→ N = 250,000,000,000 → 2500억번의 연산

정확성 테스트에서는 ?

100 * 100 * 100 ➔ 100만 → 가능

QnA