완전탐색
/DFSBFS

Programmers

#거리두기 확인하기

25.07.29

Problem

거리두기 확인하기

~ 22:25 | 문제 풀기
~ 22:30| 힌트 공개
~ 22:55 | 2차 풀기
~ 23:00 | 풀이공개

코딩테스트 연습 〉 2021 카카오 채용연계형 인턴십 〉 **거리두기 확인하기**

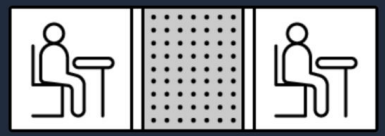
거리두기 확인하기

개발자를 희망하는 죠르디가 카카오에 면접을 보러 왔습니다.

코로나 바이러스 감염 예방을 위해 응시자들은 거리를 둬서 대기를 해야하는데 개발 직군 면접인 만큼
아래와 같은 규칙으로 대기실에 거리를 두고 앉도록 안내하고 있습니다.

1. 대기실은 5개이며, 각 대기실은 5x5 크기입니다.
2. 거리두기를 위하여 응시자들 끼리는 맨해튼 거리[1]가 2 이하로 앉지 말아 주세요.
3. 단 응시자가 앉아있는 자리 사이가 파티션으로 막혀 있을 경우에는 허용합니다.

예를 들어,



위 그림처럼 자리 사이에 파티션이 존재한다면 맨해튼 거리가 2여도 거리두기를 **지킨 것입니다.**

위 그림처럼 파티션을 사이에 두고 앉은 경우도 거리두기를 **지킨 것입니다.**

위 그림처럼 자리 사이가 맨해튼 거리 2이고 사이에 빈 테이블이 있는 경우는 거리두기를 **지키지 않은 것입니다.**

응시자가 앉아있는 자리(`P`)를 의미합니다.

빈 테이블(`O`)을 의미합니다.

파티션(`X`)을 의미합니다.

Hint

Step 1. 문제 분석

💡 문제 요약

개발자를 희망하는 죠르디가 카카오에 면접을 보러 왔다.
코로나 바이러스 감염 예방을 위해, 응시자들은 대기실에서 일정한 거리를 두고 앉아야 한다.
각 대기실은 5x5 크기로, 총 5개의 대기실이 주어진다.

규칙:

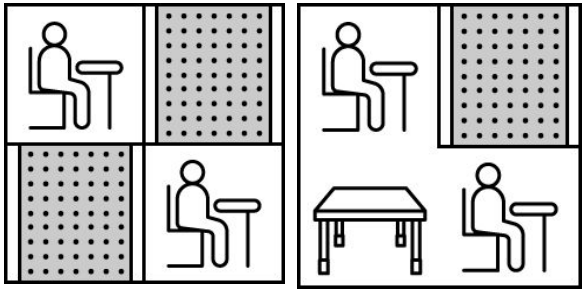
- 응시자끼리는 맨해튼 거리 **2 이하로 앉으면 안 된다.**
- 단, 거리 2 이내라도 **파티션(X) 으로 막혀 있다면 예외적으로 허**용한다.

문제 요구사항:

- 5개의 대기실이 2차원 문자열 배열 형태로 주어진다.
- 각 대기실에 대해 **거리두기를 지키고 있다면 1, 위반한 사람이 단 한 명이라도 있다면 0을 반환**한다.
- 최종적으로 1차원 배열에 5개의 값으로 결과를 반환한다.

문자 의미:

- **P**: 사람이 앉아있는 자리
- **O**: 빈 테이블
- **X**: 파티션



💡 Hint

Step 1. 문제 분석

💡 문제 유형

- **어떤 방식으로 풀 것인가?**
  - 각 P 위치에서 시작해, 맨해튼 거리 2까지 탐색하며 다른 P를 만나면 조건을 확인한다.
  - DFS처럼 깊이 단위보다는 **BFS로 레벨 단위 탐색**이 더 안전하다
- 하지만 이 문제는 **크기가 작아(5x5) 완전탐색으로도 충분히 빠르게 풀 수 있다.**
- **결론**
  - **BFS** 또는 **완전탐색**

💡 Hint

Step 2. 접근 방식

💡 풀이 아이디어

- **완전 탐색**
  - P 좌표를 기준으로 최대 **맨해튼 거리 2 이내의 모든 위치를 확인.**
  - **거리 1** → 다른 P 있으면 바로 위반.
  - **거리 2** → 중간 위치에 파티션이 있는지 확인해서 예외 처리.
  - **방향은 총 12개** (직선 1, 직선 2, 대각선 2 포함).
- **BFS**
  - 모든 **P** 좌표에서 **BFS 수행**
  - **최대 거리 2까지만** 탐색
  - 탐색 도중:
    i. **O** → 통과 가능, 계속 BFS
    ii. **X** → 탐색 종료
    iii. **P** → 거리 2 이내에서 발견되면 위반
  - 모든 **P** 좌표에 대해 **BFS 결과 이상 없으면 → 거리두기 준수**

Hint

Step 2. 접근 방식

코드 플로우

완전탐색

- 전체 places 순회
- 각 대기실(5x5) 내부 순회
  - 좌표 (i, j) 가 'P'이면 다음 12방향 검사
    - (1칸 거리): dx/dy = ±1
      해당 좌표가 P면 바로 위반 → return 0
    - (2칸 직선):
      중간에 파티션이 없다면 → 위반 → return 0
    - (2칸 대각선):
      대각선 양 옆(중간 경유지) 중 하나라도 파티션이 아니면 → 위반 → return 0
- 하나라도 위반 → 대기실 결과 0
- 끝까지 이상 없으면 → 대기실 결과 1

BFS

- 전체 places 순회
- 각 대기실(5x5)에서 모든 P 좌표 찾기
- 각 P마다 BFS 수행
  - 큐에 현재 좌표 저장 + 거리 0부터 시작
  - 4방향 탐색 (상하좌우)
    - 방문하지 않은 O는 큐에 추가 (거리 +1)
    - 방문하지 않은 P는 거리 ≤ 2 이면 → 위반 → return 0
    - X는 건너뜀
    - 거리 > 2 이면 탐색 중단
- 하나라도 위반 → 대기실 결과 0
- 모두 통과 → 대기실 결과 1

✨Solution 완전 탐색 풀이

```python
def solution(places):
    def is_safe(place):
        for i in range(5):
            for j in range(5):
                if place[i][j] != 'P':
                    continue

                for dx, dy in dirs:
                    ni, nj = i + dx, j + dy
                    if not (0 <= ni < 5 and 0 <= nj < 5):
                        continue
                    if place[ni][nj] != 'P':
                        continue

                    dist = abs(dx) + abs(dy)
                    if dist == 1:
                        return 0  # 거리 1에서 바로 P면 위반
                    elif dist == 2:
                        # 파티션 여부 확인
                        if dx == 0:  # 수평
                            if place[i][j + dy // 2] != 'X':
                                return 0
                        elif dy == 0:  # 수직
                            if place[i + dx // 2][j] != 'X':
                                return 0
                        else:  # 대각선
                            if place[i][nj] != 'X' or place[ni][j] != 'X':
                                return 0
        return 1

    dirs = [
        (-1, 0), (1, 0), (0, -1), (0, 1),  # 거리 1
        (-2, 0), (2, 0), (0, -2), (0, 2),  # 일직선 거리 2
        (-1, -1), (-1, 1), (1, -1), (1, 1)  # 대각선 거리 2
    ]

    answer = []
    for place in places:
        answer.append(is_safe(place))

    return answer
```

✨Solution

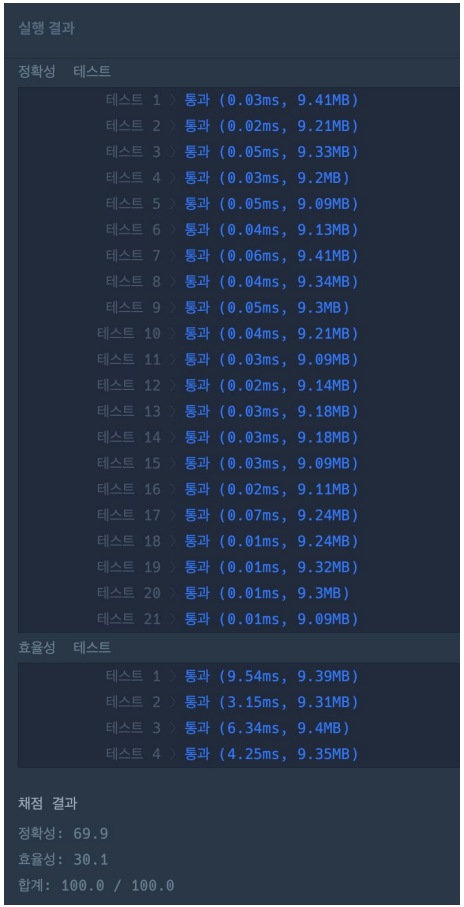
BFS 풀이

출처: [프로그래머스] LEVEL2 거리두기 확인하기 (Python)

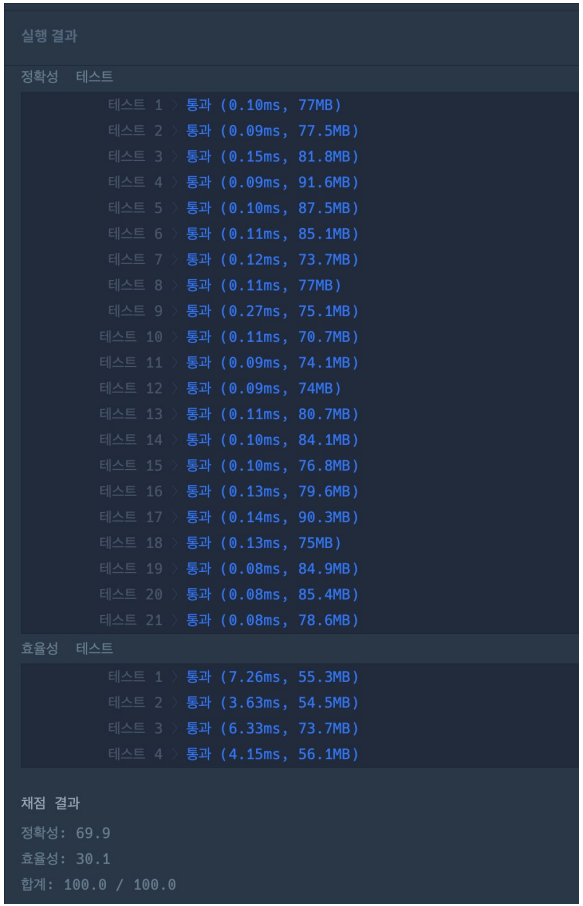
```python
from collections import deque

def bfs(p):
    start = []

    for i in range(5):  # 시작점이 되는 P 좌표 구하기
        for j in range(5):
            if p[i][j] == 'P':
                start.append([i, j])

    for s in start:
        queue = deque([s])  # 큐에 초기값
        visited = [[0] * 5 for i in range(5)]  # 방문 처리 리스트
        distance = [[0] * 5 for i in range(5)]  # 경로 길이 리스트
        visited[s[0]][s[1]] = 1

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

            dx = [-1, 1, 0, 0]  # 좌우
            dy = [0, 0, -1, 1]  # 상하

            for i in range(4):
                nx = x + dx[i]
                ny = y + dy[i]

                if 0 <= nx < 5 and 0 <= ny < 5 and visited[ny][nx] == 0:

                    if p[ny][nx] == 'O':
                        queue.append([ny, nx])
                        visited[ny][nx] = 1
                        distance[ny][nx] = distance[y][x] + 1

                    if p[ny][nx] == 'P' and distance[y][x] <= 1:
                        return 0
    return 1


def solution(places):
    answer = []

    for i in places:
        answer.append(bfs(i))

    return answer
```

✨Solution

완전 탐색 풀이



BFS 풀이



제한 범위가 적기 때문에 완전탐색으로 풀어도 무방하다. 이 문제의 경우 오히려 완전탐색으로 풀면 시간복잡도가 더 낮게 나온다.
하지만 실제 코딩테스트에서 주변 조건이 바뀌거나 범위가 커진다면 BFS가 유리하기 때문에 BFS 풀이과정을 익히는 것이 좋다

💻 Assignment

백준 #2468. 안전영역

DFS 또는 BFS를 이용하여 풀이

안전 영역

☆

실버 I

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
1 초	128 MB	131232	50497	32886	35.238%

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.

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.