땅울림
자구 스터디

Made by 규정

13주차

BFS

BFS

1 BFS

지도에서의 탐색 문제(★★★★★)

· 2차원 지도에서의 최단 거리나 연결 요소를 찾는 문제

· 삼성, 라인, 카카오, LG 등 입사 시험, 코딩 대회에 무조건 하나씩은 나오는 유형

· 수많은 응용 문제가 생길 수 있는데 일단 기본적인 틀을 손에 익혀놓는 게 가장 중요

2178_미로 탐색

N×M크기의 배열로 표현되는 미로가 있다.

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.

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

1 미로 탐색

· 지금까지 풀었던 문제들은 그래프의 연결관계가 입력으로 주어졌음
(먼저 그래프를 만들어놓고, 순회하는 방식)

· 지도 문제는 입력으로 지도의 정보만 주어짐
(순회를 하면서 그래프가 만들어지는 방식)

1 미로 탐색

맵에서의 각 칸이 노드가 됨

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

1 미로 탐색

풀이 과정

1. 노드 구조체 생성
2. 입력받아서 맵 생성
3. bfs=〉 각 칸에서 상하좌우를 보고, 1이면 카운트를 1 증가시키면서 그 칸을 큐에 삽입
4. 끝에 도달하는순간 순회 마치고 현재 카운트 출력

1 미로 탐색

카운트 횟수를 맵에 나타내보면

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

1	0	9	10	11	12
2	0	8	0	12	0
3	0	7	0	13	14
4	5	6	0	14	15

1 미로 탐색

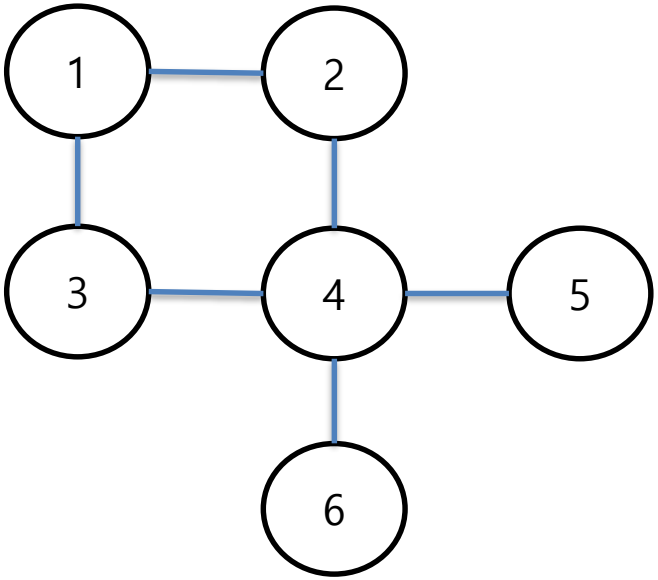
DFS로 하면 안되는 이유

dfs로는 끝까지 도달하는 길을 찾았어도 그게 최단거리인지 모르기 때문에 결국 존재하는 모든 **경로**를 다 찾아봐야한다 =〉 지수 시간복잡도 =〉 시간초과

bfs는 상하좌우 칸을 다 검사하면서 가기 때문에 한 번 방문한 칸은 다시 방문하지 않고, 끝에 도달하면 바로 끝내면 됨 =〉 최악의 경우에도 모든 **칸**만 검사하면 됨 =〉 O(N*M)

최단 경로 문제는 거ㅣ어ㅣㅣㅣㅣ의 다 bfs로 푼다고 보면 됨

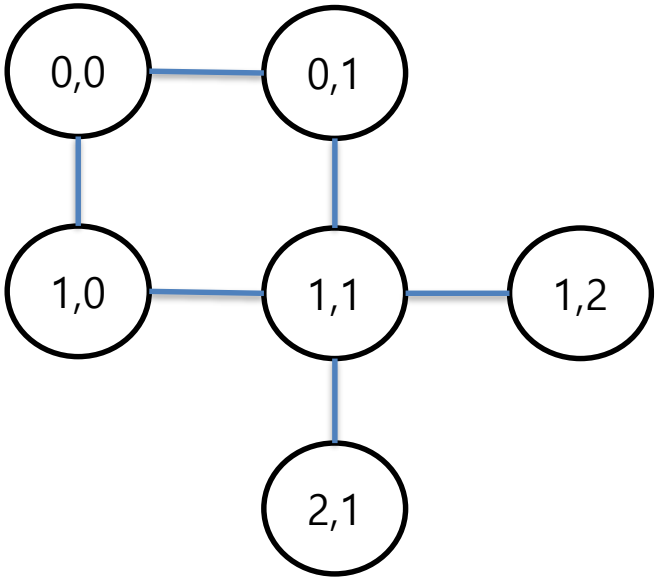
1 미로 탐색



1차원일 때는 노드가 가지고 있는 정보가 노드번호밖에 없어서 int 타입의 큐로 사용할 수 있었지만

```
queue <int> que;
que.push(1);
```

1 미로 탐색



2차원이므로 int만으로 표현 불가능
(pair 사용하거나 구조체 만들어야함)

이 문제에서는 이동횟수까지 세줘야하므로 구조체를 만드는게 편함

```
struct Node
{
    int y, x, cnt;
    Node() {} // 디폴트 생성자
    Node(int _y, int _x, int _cnt)
    {
        y = _y;
        x = _x;
        cnt = _cnt;
    }
};
```

노드에 필요한 정보는 (y,x) 좌표와 이동횟수

생성자 만들어주는게 좋음

1 미로 탐색

bfs 코딩

이제 (0,0)에서 시작해서 **상하좌우 칸을 보면서** 연결되어 있으면 이동횟수를 1 증가시키면서 큐에 넣어주면 됨

but, 상하좌우가 연결되어있는지 어떻게 검사할지가 문제

상하좌우 검사는 어떻게?

if문 네개 쓰면 되지!

```
while (!q.empty()) {
    current_location = q.front();
    q.pop();
    current_row = current_location.first;
    current_column = current_location.second;

    if (current_row == row && current_column == column)
        break;

    //go_top
    if (current_row != 1 && visit[current_row - 1][current_column] == 0 && miro[current_row - 1][current_column] == 1) {
        q.push(make_pair(current_row - 1, current_column));
        visit[current_row - 1][current_column] = visit[current_row][current_column] + 1;
    }
    //go_bottom
    if (current_row != row && visit[current_row + 1][current_column] == 0 && miro[current_row + 1][current_column] == 1) {
        q.push(make_pair(current_row + 1, current_column));
        visit[current_row + 1][current_column] = visit[current_row][current_column] + 1;
    }
    //go_left
    if (current_column != 1 && visit[current_row][current_column - 1] == 0 && miro[current_row][current_column - 1] == 1) {
        q.push(make_pair(current_row, current_column - 1));
        visit[current_row][current_column - 1] = visit[current_row][current_column] + 1;
    }

    //go_right
    if (current_column != column && visit[current_row][current_column + 1] == 0 && miro[current_row][current_column + 1] == 1) {
        q.push(make_pair(current_row, current_column + 1));
        visit[current_row][current_column + 1] = visit[current_row][current_column] + 1;
    }

}
```

ㅁㅅ

for문으로 간단하게 검사

아래, 위, 오른쪽, 왼쪽

```
int my[] = { 1,-1,0,0 };
int mx[] = { 0,0,1,-1 };
```

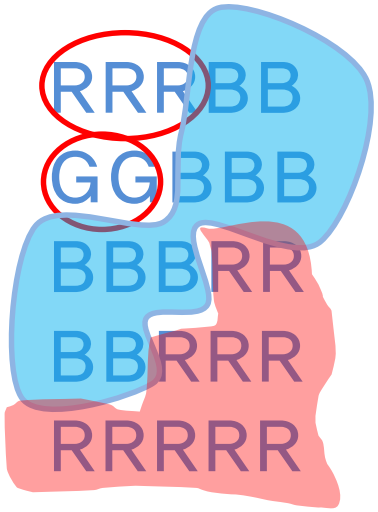
```
while (!q.empty()) {
    Node cur = q.front();
    q.pop();

    for (int i = 0; i < 4; i++) {
        int ny = cur.y + my[i];
        int nx = cur.x + mx[i];
        int nc = cur.cnt + 1;
        if (G[ny][nx] == '1' && ny >= 0 && nx >= 0 && ny < n && nx < m && !visit[ny][nx]) {
            if (ny == n - 1 && nx == m - 1) return nc; 끝에 도달하면 바로 종료
            visit[ny][nx] = true;
            q.push({ ny,nx,nc });
        }
    }
}
```

10026_적록색약

1 적록색약

bfs 코딩



BFS 한번 실행할 때마다 같은 구간은 모두 방문하고 끝남

=〉 모든 칸을 검사하면서 방문했는지 체크하고, 방문하지 않았으면 BFS 실행 =〉 BFS가 실행된 횟수를 세주면 그게 바로 구간의 개수

감사합니다.

Made by 규정