BOJ 10026

적록색약

소프트웨어학과 201921017 이지우

1. 접근

문제

적록색약은 빨간색과 초록색의 차이를 거의 느끼지 못한다. 따라서, 적록색약인 사람이 보는 그림은 아닌 사람이 보는 그림과는 좀 다를 수 있다.

크기가 N×N인 그리드의 각 칸에 R(빨강), G(초록), B(파랑) 중 하나를 색칠한 그림이 있다. 그림은 몇 개의 구역으로 나뉘어져 있는데, 구역은 같은 색으로 이루어져 있다. 또, 같은 색상이 상하좌우로 인접해 있는 경우에 두 글자는 같은 구역에 속한다. (색상의 차이를 거의 느끼지 못하는 경우도 같은 색상이라 한다)

예를 들어, 그림이 아래와 같은 경우에

```
RRRBB
GGBBB
BBBRR
BBRRR
RRRRR
```

적록색약이 아닌 사람이 봤을 때 구역의 수는 총 4개이다. (빨강 2, 파랑 1, 초록 1) 하지만, 적록색약인 사람은 구역을 3개 볼 수 있다. (빨강-초록 2, 파랑 1)

그림이 입력으로 주어졌을 때, 적록색약인 사람이 봤을 때와 아닌 사람이 봤을 때 구역의 수를 구하는 프로그램을 작성하시오.

입력

첫째 줄에 N이 주어진다. (1 ≤ N ≤ 100)

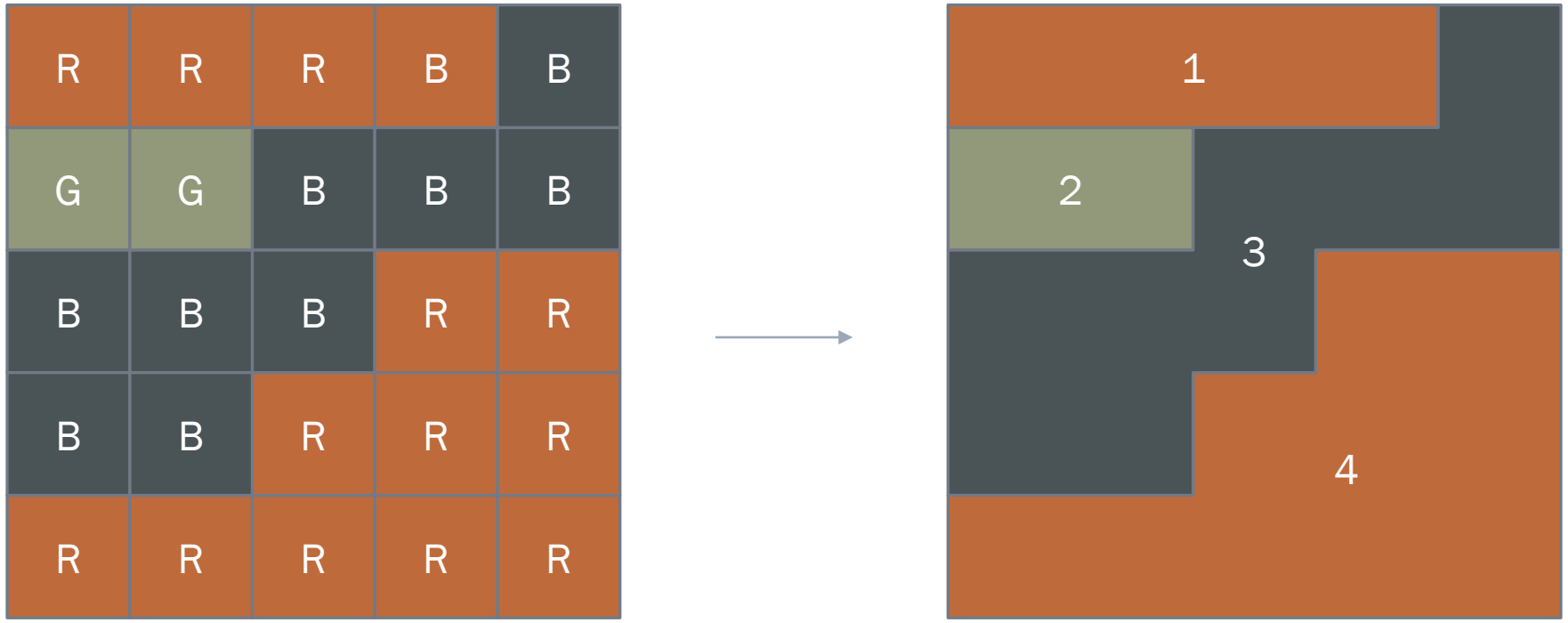
둘째 줄부터 N개 줄에는 그림이 주어진다.

출력

적록색약이 아닌 사람이 봤을 때의 구역의 개수와 적록색약인 사람이 봤을 때의 구역의 수를 공백으로 구분해 출력한다.

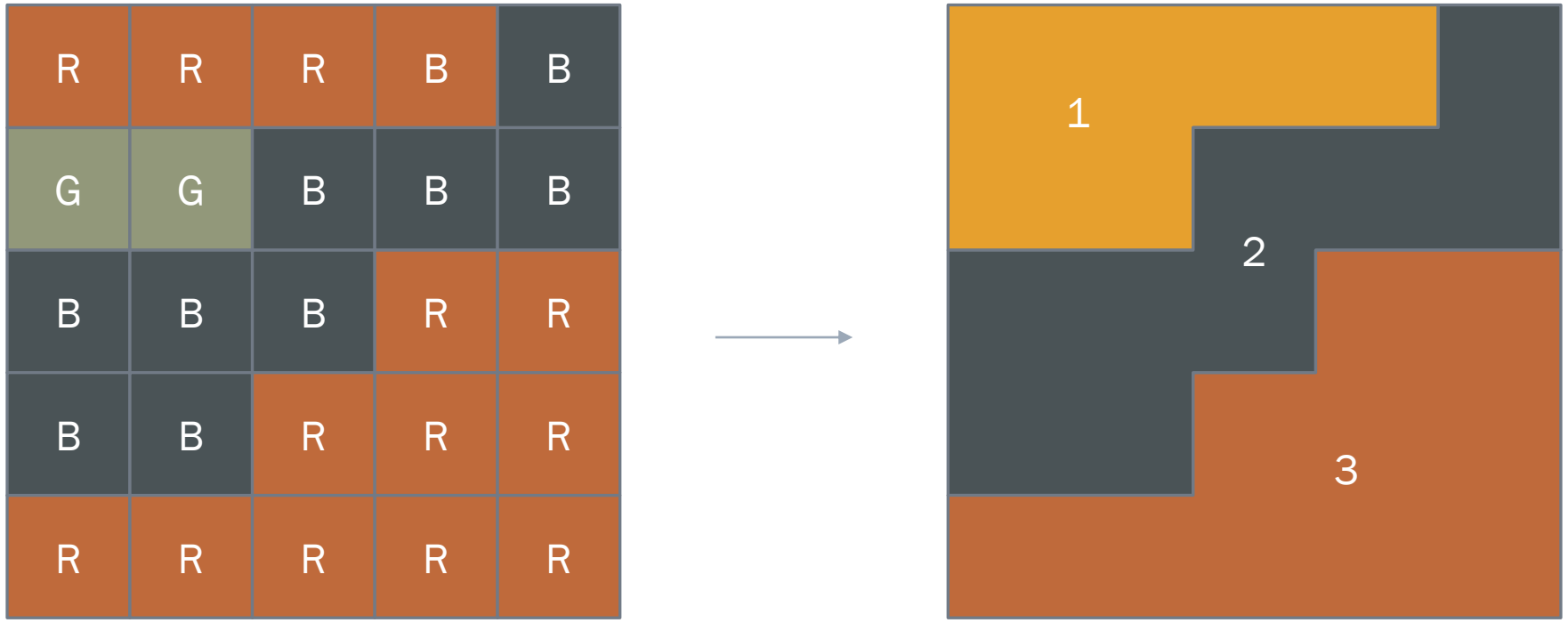
그래프 탐색 문제
비슷한 문제로 섬의 개수를 세는 등의 문제들이 있음

1. 접근



< 적록색약이 아닌 사람의 경우 >

1. 접근



< 적록색약인 사람의 경우 >

2. 구현

적록색약

빨간색(R)과 초록색(G)을 구분하지 못함
즉, 빨간색과 초록색을 같은 색으로 인식

따라서 field의 **G(또는 R)을 R(또는 G)로 통일한**,
적록색약이 보는 **field를 따로 만들기**

헤더파일 및 전역변수 선언

```
typedef pair<int, int> pii;

상하좌우 ----------
int dx[4] = {0, 0, 1, -1};
int dy[4] = {1, -1, 0, 0};


int n;
```

main 함수

```
cin >> n;


vector<vector<char>> field(n, vector<char>(n));
vector<vector<char>> rgbField(n, vector<char>(n));
// input
for(int i=0; i<n; i++) {
    for(int j=0; j<n; j++) {
        cin >> field[i][j];
        적록색약 필드값 통일(G→R)----------
        rgbField[i][j] = field[i][j];
        if (rgbField[i][j] == 'G') {
            rgbField[i][j] = 'R';
        }
    }
}

 각 필드에 대하여 구역 개수 세기----------
int cnt = solve(field);
int rgbCnt = solve(rgbField);

cout << cnt << " " << rgbCnt << endl;

return 0;
```

2. 구현

solve 함수

```
int solve(vector<vector<char>> &field) {

    vector<vector<bool>> visited(n, vector<bool>(n));

    int cnt = 0;
    for(int i=0; i<n; i++) {
        for(int j=0; j<n; j++) {
            if(!visited[i][j]){
                cnt++; 아직 방문하지 않은 필드==새로운 구역
                bfs(i, j, field, visited);
            }
        }
    }

    return cnt;
}
```

bfs 함수

```
void bfs(int x, int y, vector<vector<char>> &field, vector<vector<bool>> &visited) {
    queue<pii> q;
    q.push({x, y});

    while(!q.empty()) {
        pii curr = q.front(); q.pop();

        for(int i=0; i<4; i++) {
            int nx = curr.first + dx[i];
            int ny = curr.second + dy[i];

            if(nx < 0 || nx >= n || ny < 0 || ny >= n) continue;
            아직 방문하지 않은 이웃, 같은 색상 → 같은 구역
            if(!visited[nx][ny] && field[curr.first][curr.second] == field[nx][ny]) {
                visited[nx][ny] = true;
                q.push({nx, ny});
            }
        }
    }
}
```