Baekjoon / 그림 (1926번)

문제

어떤 큰 도화지에 그림이 그려져 있을 때, 그 그림의 개수와, 그 그림 중 넓이가 가장 넓은 것의 넓이를 출력하여라. 단, 그림이라는 것은 1로 연결된 것을 한 그림이라고 정의하자. 가로나 세로로 연결된 것은 연결이 된 것이고 대각선으로 연결이 된 것은 떨어진 그림이다. 그림의 넓이란 그림에 포함된 1의 개수이다.

입력

첫째 줄에 도화지의 세로 크기 n ($1 \le n \le 500$)과 가로 크기 m ($1 \le m \le 500$)이 차례로 주어진다. 두 번째 줄부터 $n+1$줄 까지 그림의 정보가 주어진다. (단 그림의 정보는 0과 1이 공백을 두고 주어지며, 0은 색칠이 안된 부분, 1은 색칠이 된 부분을 의미한다)

출력

첫째 줄에는 그림의 개수, 둘째 줄에는 그 중 가장 넓은 그림의 넓이를 출력하여라. 단, 그림이 하나도 없는 경우에는 가장 넓은 그림의 넓이는 0이다.

예제 입력 1

```
6 5
1 1 0 1 1
0 1 1 0 0
0 0 0 0 0
1 0 1 1 1
0 0 1 1 1
0 0 1 1 1
```

예제 출력 1

```
4
9
```

문제 접근 방법

1. 첫줄에 입력 받은 두 수로 2차원 배열 만들기
2. 만든 배열에 그림 정보 입력하기
3. 그림의 첫번째부터 시작하여 상하좌우에 있는 그림 파악하기
4. 방문한 그림은 방문하였다고 기록하기
5. 방문하였다고 기록된 그림은 지나치기
6. 파악된 그림정보들을 보고 그림의 크기와 그림의 개수 파악하기

코드

// Baekjoon / 그림(1926) / 72ms, 1.5h

```
#include <iostream>
#include <vector>
#include <queue>
#include <algorithm>

using namespace std;

int bfs(int x, int y, vector<vector<int>>& paper, vector<vector<bool>>& visited, int n, int m) {
    queue<pair<int, int>> q;
    q.push({x, y});
    visited[x][y] = true;

    int area = 1;
    while (!q.empty()) {
        auto [cx, cy] = q.front();
        q.pop();
        int dx[4] = { -1, 1, 0, 0 }, dy[4] = { 0, 0, -1, 1 };
        for (int dir = 0; dir < 4; dir++) {
            int nx = cx + dx[dir];
            int ny = cy + dy[dir];
            if (nx >= 0 && ny >= 0 && nx < n && ny < m) {
                if (paper[nx][ny] == 1 && !visited[nx][ny]) {
                    visited[nx][ny] = true;
                    q.push({nx, ny});
                    area++;
                }
            }
        }
    }

    return area;
}

int main() {
    int n, m;
    cin >> n >> m;
    vector<vector<int>> paper(n, vector<int>(m));
    vector<vector<bool>> visited(n, vector<bool>(m, false));
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < m; ++j)
            cin >> paper[i][j];
    int pictureCount = 0;
    int maxArea = 0;
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < m; ++j) {
            if (!visited[i][j] && paper[i][j] == 1) {
                int area = bfs(i, j, paper, visited, n, m);
                pictureCount++;
                maxArea = max(maxArea, area);
            }
        }
    }
    cout << pictureCount << "\n" << maxArea << "\n";
    return 0;
}
```