```
/*
    Time complexity: O(N^2)
    Space complexity: O(N^2)

    where N by N are the dimensions of the cake
*/

// Consider the cake to be connected graph with every vertex connected to its nearby 4 vertices
int dfs(vector<vector<int>> &cake, vector<vector<bool>> &visited, int x, int y, int n) {
    if (visited[x][y]) {
        return 0;
    }

    visited[x][y] = true;
    int count = 1;

    int dx[] = {0, 0, -1, 1};  // Change in x while taking four moves
    int dy[] = {1, -1, 0, 0};  // Change in y while taking four moves

    for (int i = 0; i < 4; ++i) {
        int X = x + dx[i];
        int Y = y + dy[i];
        if (X >= 0 && X < n && Y >= 0 && Y < n && cake[X][Y] == 1) {
            count += dfs(cake, visited, X, Y, n);
        }
    }

    return count;
}

int getBiggestPieceSize(vector<vector<int>> &cake, int n) {
    int biggestPieceSize = 0;
    vector<vector<bool>> visited(n, vector<bool>(n, false));

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if (cake[i][j] == 1 && !visited[i][j]) {
                biggestPieceSize = max(biggestPieceSize, dfs(cake, visited, i, j, n));
            }
        }
    }

    return biggestPieceSize;
}
```