#include<bits/stdc++.h>

using namespace std;

vector<vector<pair<int,int> > > graph;

int min\_ = INT\_MAX;

int max\_ = INT\_MIN;

int cmax\_ = INT\_MAX;

int fmin\_ = INT\_MIN;

string smin,smax,cstr,fstr,kstr;

typedef pair<int, string> pi;

priority\_queue<pi,vector<pi>,greater<pi> > pq;

void addEdge(int x, int y, int w){

graph[x].push\_back(make\_pair(y,w));

graph[y].push\_back(make\_pair(x,w));

}

void display(){

for(int i=0;i<graph.size();i++){

cout<<i<<"->";

for(auto it: graph[i]){

cout<<it.first<<" "<<it.second<<", ";

}

cout<<endl;

}

}

void allPath(int src, int dest, vector<bool> visited, string ans){

if(src==dest){

cout<<ans<<endl;

return;

}

visited[src] = true;

for(auto it: graph[src]){

if(visited[it.first]==false){

string str = to\_string(it.first);

allPath(it.first,dest,visited,ans+str);

}

}

visited[src] = false;

}

void island(int i, int j, vector<vector<bool>> &visited, vector<vector<int> > &v){

if(i<0 || i>=visited.size() || j<0 || j>=visited[0].size() || visited[i][j]==true || v[i][j]==1){

return;

}

visited[i][j] = true;

island(i-1,j,visited,v);

island(i+1,j,visited,v);

island(i,j-1,visited,v);

island(i,j+1,visited,v);

}

int main(){

int n,m;

cin>>n>>m;

vector<vector<int> > v(n,vector<int>(m));

for(int i=0;i<n;i++){

for(int j=0;j<n;j++){

cin>>v[i][j];

}

}

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

int count = 0;

for(int i=0;i<n;i++){

for(int j=0;j<n;j++){

if(v[i][j]==0 && visited[i][j]==false){

island(i,j,visited,v);

count++;

}

}

}

cout<<count;

}