//1월 9일 까지 숙제

//연구소 과제

#include <iostream>

#include <stack>

#include <vector>

#include <algorithm>

#define endl "\n"

#define MAX 8

using namespace std;

int row,col;

int DataArray[MAX][MAX], Data\_cp[MAX][MAX];

int max\_safe=0;

typedef struct {

    int i;

    int j;

}point;

typedef struct {

    point point1;

    point point2;

    point point3;

}points;

point moveXY[4]={{0,1},{1,0},{0,-1},{-1,0}};

stack<point> s;

vector<point> wall\_list;

vector<point> virus;

vector<points> wall\_comb;

void Input()

{

    cin>>row>>col;

    for(int i=0;i<row;i++){

        for(int j=0;j<col;j++){

            cin>>DataArray[i][j];

            if(DataArray[i][j]==0) {wall\_list.push\_back({i,j});}

            if(DataArray[i][j]==2) {virus.push\_back({i,j});}

        }

    }

}

void Copy()

{

    for(int i=0;i<row;i++){

        for(int j=0;j<col;j++){

            Data\_cp[i][j]=DataArray[i][j];

        }

    }

}

void checkWall()

{

    for (int i=0;i<wall\_list.size();i++){

        for (int j=i+1;j<wall\_list.size();j++){

            for(int k=j+1;k<wall\_list.size();k++){

                wall\_comb.push\_back({{wall\_list.at(i)},{wall\_list.at(j)},{wall\_list.at(k)}});

            }

        }

    }

}

void MakeWall( points point)

{

    Data\_cp[point.point1.i][point.point1.j]=1;

    Data\_cp[point.point2.i][point.point2.j]=1;

    Data\_cp[point.point3.i][point.point3.j]=1;

}

int DFS(int i,int j)

{

    Data\_cp[i][j]=2;

    for(int k=0;k<4;k++){

        int nexti=i+moveXY[k].i;

        int nextj=j+moveXY[k].j;

        if (0<=nexti and nexti<row and 0<=nextj and nextj<col){

            if (Data\_cp[nexti][nextj]==0)

            {

                 Data\_cp[nexti][nextj]=2;

                s.push({nexti,nextj});

            }

        }

    }

    while(!s.empty())

    {

        point data;

        data=s.top();

        s.pop();

        return DFS(data.i,data.j);

    }

    return 0;

}

void Emergency()

{

    for (int i=0;i<virus.size();i++){

        DFS(virus.at(i).i,virus.at(i).j);

    }

}

int CheckSafe()

{

    int count=0;

     for(int i=0;i<row;i++){

        for(int j=0;j<col;j++){

            if (Data\_cp[i][j]==0) count++;

        }

    }

    return count;

}

int Solution(){

   points data, max\_data;

   Copy();

   checkWall();

    for (int i=0;i<wall\_comb.size();i++)

    {

        data=wall\_comb.at(i);

        Copy();

        MakeWall(data);

        Emergency();

        int safe=CheckSafe();

        max\_safe=max(max\_safe, safe);

    }

    wall\_comb.clear();

    return max\_safe;

}

int main()

{

    Input();

    int result=Solution();

    cout<<result<<endl;

    return 0;

}