/\*\*\*\*\*\*\*\* **El Chapo 's longest tunnel'**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

El Chapo an infamous drug lord who was known for his longest tunnel through which he made his escape from the prison.

The tunnel has to be dug fast so his workers decided to divide the work among themselves and in the end, El Chapo should investigate the longest tunnel among them.

Given the path of the tunnel, write a program to help El Chapo in finding the longest tunnel path where a path is indicated by 1 and no path is indicated by 0

The longest path can be found by traveling in horizontal, vertical as well as diagonal.

**Constraint:**

2<=n<=10

arr[i][j] = 1 or 0

0< i,j<10

Input format:

5

1 1 0 0 0

0 1 1 0 0

0 0 1 0 1

1 0 0 0 1

0 1 0 1 1

Output:

5

\*/

#include <bits/stdc++.h>

using namespace std;

int n;

int getVal(vector<vector<int> >v, int i,int j)

{

if(i<0 || i>=n || j<0||j>=n)

return 0;

else

return v[i][j];

}

void findMaxBlock(vector<vector<int>> v, int r, int c, int size,vector<vector<int>> visited, int &maxSize)

{

if(r>= n || c>= n)

return ;

visited[r][c] = 1;

size++;

if( size > maxSize)maxSize =size;

int direction [][2] = {{-1,0},

{-1,-1},

{0,-1},

{1,-1},

{1,0},

{1,1},

{0,1},

{-1,1}};

for(int i=0;i<8;i++)

{

int newi = r + direction[i][0];

int newj = c + direction[i][1];

int val = getVal(v, newi, newj);

if(val > 0 && visited[newi][newj] == 0)

findMaxBlock(v, newi, newj,size,visited,maxSize);

}

visited[r][c] = 0;

}

int getMaxOnes(vector<vector<int>> v)

{

int maxSize = 0;

int size =0;

vector<vector<int>> visited(n,vector<int>(n,0));

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

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

// visited[i][j] = 0;

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

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

if(v[i][j] == 1)

findMaxBlock(v,i,j,size,visited,maxSize);

return maxSize;

}

int main(int argc, char const \*argv[]) {

cin>>n;

vector<vector<int>> v(n,vector<int>(n,0));

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

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

cin>>v[i][j];

cout<<getMaxOnes(v)<<endl;

return 0;

}

/\*

testcase 0:

5

1 1 0 0 0

0 1 1 0 0

0 0 1 0 1

1 0 0 0 1

0 1 0 1 1

output: 5

testcase 1:

6

1 0 1 0 1 0

0 1 0 1 0 1

1 0 1 0 1 0

0 1 0 1 0 1

1 0 1 0 1 0

0 1 0 1 0 1

output : 16

testcase 2:

5

1 1 0 0 1

0 0 0 0 0

1 1 1 1 1

1 1 0 1 0

1 0 0 1 0

output: 10

testcase 3:

6

1 1 0 0 1 0

0 0 0 0 1 1

1 1 1 1 1 0

1 0 1 0 0 1

0 0 1 0 0 1

1 0 1 0 1 0

output: 13

\*/