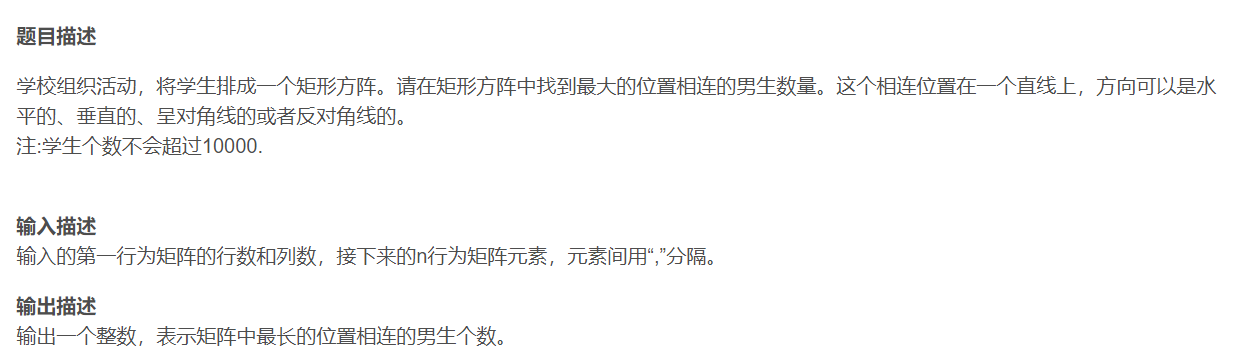
# **E卷-学生方阵[200分]（ Java | Python3 | C++ | C语言 | JsNode | Go）**







import java.util.Scanner;

public class Main {

// 检查坐标是否在矩阵范围内

static boolean ok(int x, int y, int n, int m) {

return x >= 0 && x < n && y >= 0 && y < m;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.



import sys

# input() 用于读取标准输入

def input():

return sys.stdin.readline()

# write() 用于输出标准输出

def write(x):

return sys.stdout.write(str(x) + "\n")

# 读取矩阵的行数和列数

n, m = map(int, input().split(","))

# g 用于存储输入的矩阵，矩阵元素为 1 表示男生，0 表示其他

g = []

for \_ in range(n):

line = list(input().split(","))

a = [0] \* m

for i in range(m):

if line[i].strip() == "M": # 记得strip去掉可能的空格

a[i] = 1 # 'M' 表示男生，用1来表示

g.append(a) # 把当前行加入矩阵

# 四个方向的位移：水平、垂直、对角线、反对角线

d = ((1, 0), (0, 1), (1, 1), (1, -1))

# ok 函数用于检查坐标是否在矩阵的边界内

def ok(x, y):

if x < 0 or x >= n or y < 0 or y >= m:

return 0 # 如果超过边界返回0

return 1 # 在边界内返回1

mx = 0 # 用于存储最长相连男生的数量

# 遍历矩阵的每一个元素

for i in range(n):

for j in range(m):

if not g[i][j]: # 如果当前坐标不是男生，则跳过

continue

# 遍历四个方向

for k in range(4):

di, dj = i - d[k][0], j - d[k][1] # 计算在当前方向的前一个坐标

if ok(di, dj) and g[di][dj]: # 如果前一个坐标在范围内且是男生，则跳过

continue

c = 1 # 初始化当前方向相连男生的数量

di, dj = i + d[k][0], j + d[k][1] # 计算下一个坐标

# 当下一个坐标在范围内且是男生时，继续检查下一个

while ok(di, dj) and g[di][dj]:

c += 1 # 累加相连男生的数量

di, dj = di + d[k][0], dj + d[k][1] # 继续移动到下一个坐标

mx = max(mx, c) # 更新最大相连男生的数量

write(mx) # 输出结果



#include <iostream>

#include <vector>

#include <string>

#include <sstream>

using namespace std;

// 检查坐标是否在矩阵范围内

bool ok(int x, int y, int n, int m) {

return x >= 0 && x < n && y >= 0 && y < m;

}

int main() {

int n, m;

char comma;

cin >> n >> comma >> m; // 读取矩阵的行数和列数

vector<vector<int>> g(n, vector<int>(m, 0)); // 初始化矩阵

string line;

getline(cin, line); // 读取剩余的换行符

// 读取矩阵数据

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

getline(cin, line);

stringstream ss(line);

string cell;

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

getline(ss, cell, ',');

if (cell == "M") {

g[i][j] = 1; // 'M' 表示男生，用1来表示

}

}

}

// 四个方向的位移：水平、垂直、对角线、反对角线

int d[4][2] = {{1, 0}, {0, 1}, {1, 1}, {1, -1}};

int mx = 0; // 用于存储最长相连男生的数量

// 遍历矩阵的每一个元素

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

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

if (!g[i][j]) continue; // 如果当前坐标不是男生，则跳过

for (int k = 0; k < 4; ++k) {

int di = i - d[k][0], dj = j - d[k][1];

if (ok(di, dj, n, m) &&

g[di][dj]) continue; // 如果前一个坐标在范围内且是男生，则跳过

int c = 1; // 初始化当前方向相连男生的数量

di = i + d[k][0], dj = j + d[k][1];

while (ok(di, dj, n, m) && g[di][dj]) {

c++; // 累加相连男生的数量

di += d[k][0], dj += d[k][1]; // 继续移动到下一个坐标

}

mx = max(mx, c); // 更新最大相连男生的数量

}

}

}

cout << mx << endl; // 输出结果

return 0;

}



#include <stdio.h>

#include <string.h>

// 检查坐标是否在矩阵范围内

int ok(int x, int y, int n, int m) {

return x >= 0 && x < n && y >= 0 && y < m;

}

int main() {

int n, m;

char comma;

scanf("%d%c%d", &n, &comma, &m); // 读取矩阵的行数和列数

int g[100][100] = {0}; // 初始化矩阵，假设最大100x100

char line[1000];

// 读取矩阵数据

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

scanf("%s", line);

char\* token = strtok(line, ",");

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

if (strcmp(token, "M") == 0) {

g[i][j] = 1; // 'M' 表示男生，用1来表示

}

token = strtok(NULL, ",");

}

}

// 四个方向的位移：水平、垂直、对角线、反对角线

int d[4][2] = {{1, 0}, {0, 1}, {1, 1}, {1, -1}};

int mx = 0; // 用于存储最长相连男生的数量

// 遍历矩阵的每一个元素

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

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

if (!g[i][j]) continue; // 如果当前坐标不是男生，则跳过

for (int k = 0; k < 4; ++k) {

int di = i - d[k][0], dj = j - d[k][1];

if (ok(di, dj, n, m) &&

g[di][dj]) continue; // 如果前一个坐标在范围内且是男生，则跳过

int c = 1; // 初始化当前方向相连男生的数量

di = i + d[k][0], dj = j + d[k][1];

while (ok(di, dj, n, m) && g[di][dj]) {

c++; // 累加相连男生的数量

di += d[k][0], dj += d[k][1]; // 继续移动到下一个坐标

}

if (c > mx) mx = c; // 更新最大相连男生的数量

}

}

}

printf("%d\n", mx); // 输出结果

return 0;

}



const readline = require('readline');

// 创建接口读取输入

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout

});

let inputLines = [];

rl.on('line', (line) => {

inputLines.push(line);

});

rl.on('close', () => {

const [n, m] = inputLines[0].split(',').map(Number);

const g = Array.from({ length: n }, () => Array(m).fill(0));

// 读取矩阵数据

for (let i = 1; i <= n; i++) {

const line = inputLines[i].split(',');

for (let j = 0; j < m; j++) {

if (line[j].trim() === 'M') {

g[i - 1][j] = 1; // 'M' 表示男生，用1来表示

}

}

}

// 四个方向的位移：水平、垂直、对角线、反对角线

const d = [[1, 0], [0, 1], [1, 1], [1, -1]];

let mx = 0; // 用于存储最长相连男生的数量

// 遍历矩阵的每一个元素

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

for (let j = 0; j < m; j++) {

if (!g[i][j]) continue; // 如果当前坐标不是男生，则跳过

for (let k = 0; k < 4; k++) {

let di = i - d[k][0], dj = j - d[k][1];

if (ok(di, dj, n, m) && g[di][dj]) continue; // 如果前一个坐标在范围内且是男生，则跳过

let c = 1; // 初始化当前方向相连男生的数量

di = i + d[k][0];

dj = j + d[k][1];

while (ok(di, dj, n, m) && g[di][dj]) {

c++; // 累加相连男生的数量

di += d[k][0];

dj += d[k][1]; // 继续移动到下一个坐标

}

mx = Math.max(mx, c); // 更新最大相连男生的数量

}

}

}

console.log(mx); // 输出结果

});

// 检查坐标是否在矩阵范围内

function ok(x, y, n, m) {

return x >= 0 && x < n && y >= 0 && y < m;

}



package main

import (

"bufio"

"fmt"

"os"

"strconv"

"strings"

)

// 检查坐标是否在矩阵范围内

func ok(x, y, n, m int) bool {

return x >= 0 && x < n && y >= 0 && y < m

}

func main() {

reader := bufio.NewReader(os.Stdin)

line, \_ := reader.ReadString('\n')

nm := strings.Split(strings.TrimSpace(line), ",")

n, \_ := strconv.Atoi(nm[0])

m, \_ := strconv.Atoi(nm[1])

g := make([][]int, n)

for i := range g {

g[i] = make([]int, m)

}

// 读取矩阵数据

for i := 0; i < n; i++ {

line, \_ := reader.ReadString('\n')

cells := strings.Split(strings.TrimSpace(line), ",")

for j := 0; j < m; j++ {

if cells[j] == "M" {

g[i][j] = 1 // 'M' 表示男生，用1来表示

}

}

}

// 四个方向的位移：水平、垂直、对角线、反对角线

d := [4][2]int{{1, 0}, {0, 1}, {1, 1}, {1, -1}}

mx := 0 // 用于存储最长相连男生的数量

// 遍历矩阵的每一个元素

for i := 0; i < n; i++ {

for j := 0; j < m; j++ {

if g[i][j] == 0 {

continue // 如果当前坐标不是男生，则跳过

}

for k := 0; k < 4; k++ {

di, dj := i-d[k][0], j-d[k][1]

if ok(di, dj, n, m) && g[di][dj] == 1 {

continue // 如果前一个坐标在范围内且是男生，则跳过

}

c := 1 // 初始化当前方向相连男生的数量

di, dj = i+d[k][0], j+d[k][1]

for ok(di, dj, n, m) && g[di][dj] == 1 {

c++ // 累加相连男生的数量

di += d[k][0]

dj += d[k][1] // 继续移动到下一个坐标

}

if c > mx {

mx = c // 更新最大相连男生的数量

}

}

}

}

fmt.Println(mx) // 输出结果

}