二维数组2

人员

韩承睿、刘嘉航、辛帅辰、李翰如、方俊喆、崔吉诺、刘祺、刘峰烁、秦显森、齐振玮 到课

牟茗、高健桓 线上

徐浩然 未到

作业

```
下节课要求默写 noi 1.6 07:有趣的跳跃
必做：  1997 - 孤独的素数
选做：  1197 - 拐角II
```

课堂表现

大部分同学二维数组掌握的不是很好，说明之前的内容没有完全消化理解。

同学们课后一定要好好复习上节课的知识。

课堂内容

求最大值、最小值

```cpp
#include <iostream>
#include <algorithm>
using namespace std;
int main()
{
    int a, b;
    cin >> a >> b;
    int t = max(a, b);  // 求最大值
    int k = min(a, b);  // 求最小值
    cout << t;

    int c,d;
    cin >> c >> d;
    int s = max(max(a,b),max(c,d));  // s 代表 a b c d 中的最大值
    return 0;
}
```

1996 - 每个小组的最大年龄

```cpp
1,2,3,...,n

for (int i = 1; i <= n; i++) {
    // 处理第 i 行
    // 找 a[i][1], a[i][2], a[i][3], ... , a[i][m] 中的最大值
    int maxx = -1;
    for (int j = 1; j <= m; j++) {

        maxx = max(maxx, a[i][j]);  // 拓展，取最大

        if (a[i][j] > maxx) {
            maxx = a[i][j];
        }
    }

    // 此时maxx 代表 a[i][1] ~ a[i][m] 里的最大值
    cout << maxx << endl;
}
```

```cpp
#include <bits/stdc++.h>

using namespace std;

const int maxn = 100 + 5;
int w[maxn][maxn];

int main()
{
  int n, m; cin >> n >> m;
  for (int i = 1; i <= n; ++i) {
    for (int j = 1; j <= m; ++j) cin >> w[i][j];
  }

  for (int i = 1; i <= n; ++i) {
    int maxx = 0;
    for (int j = 1; j <= m; ++j) {
      maxx = max(maxx, w[i][j]);
    }
    cout << maxx << endl;
  }
  return 0;
}
```

1998 - 找朋友

```
小T在 (x,y) 这个位置

接下来要看第 x 行有多少跟 a[x][y] 相同的数  ->  sum1
a[x][1], a[x][2], a[x][3], ..., a[x][m]
纵坐标: 从1~m
for (int j = 1; j <= m; j++) {  // j: 1~m
    if (a[x][j] == a[x][y]) {
        sum1++;
    }
}

接下来要看第 y 列有多少跟 a[x][y] 相同的数  ->  sum2
a[1][y], a[2][y], ..., a[n][y]
横坐标: 从1~n
for (int i = 1; i <= n; i++) {  // i: 1~n
    if (a[i][y] == a[x][y]) {
        sum2++;
    }
}

最后结果 = sum1 + sum2 - 2
```

```cpp
#include <bits/stdc++.h>

using namespace std;

const int maxn = 200 + 5;
int w[maxn][maxn];

int main()
{
  int n, m; cin >> n >> m;
  for (int i = 1; i <= n; ++i) {
    for (int j = 1; j <= m; ++j) cin >> w[i][j];
  }
  int x, y; cin >> x >> y;

  int sum = 0;
  for (int i = 1; i <= n; ++i) {
    if (w[i][y]==w[x][y]) sum++;
  }
  for (int j = 1; j <= m; ++j) {
    if (w[x][j]==w[x][y]) sum++;
  }
  cout << sum-2 << endl;
  return 0;
}
```

1997 - 孤独的素数

1. 输入a数组
2. 求f数组
   f[i][j]:
      如果a[i][j]是质数, f[i][j]=1
      否则, f[i][j]=0
3. 如何构造f数组呢?
   本质就是判断质数(判断a[i][j]是否是质数即可)
4. 如何判断a[i][j]是否是质数?
   找a[i][j]有几个因数即可
   如果a[i][j]只有2个因数 -> a[i][j] 是质数
   否则 -> a[i][j] 不是质数
      a[i][j]的因数一定比a[i][j]小，所以因数可能取值为1~a[i][j]
      因此只需要统计 1~a[i][j] 中有几个数是 a[i][j] 的因数
5. 有了f数组之后
   第一步需要找到f数组中所有的1
   第二步需要判断周围8个方向存不存在1
      如果周围8个方向不存在1 -> 说明是孤独的素数

```cpp
#include <iostream>

using namespace std;

int a[55][55], f[55][55];
int dx[] = {-1, -1, -1, 0, 0, 1, 1, 1};
int dy[] = {-1, 0, 1, -1, 1, -1, 0, 1};

int main() {
    int n, m;
    cin >> n >> m;
    for (int i = 1; i <= n; ++i) {
        for (int j = 1; j <= m; ++j) {
            cin >> a[i][j];
        }
    }

    // 接下来 a 数组  ->  f 数组
    // a[i][j]是质数 -> f[i][j]=1
    // a[i][j]不是质数 -> f[i][j]=0
    for (int i = 1; i <= n; ++i) {
        for (int j = 1; j <= m; ++j) {
            // 判断a[i][j]是否是质数
            // 在1~a[i][j]之间找因数数量，判断是否等于2
            int sum = 0;
            for (int k = 1; k <= a[i][j]; ++k) {
                // 判断 k 是不是 a[i][j] 的因数
                if (a[i][j]%k == 0) {
                    sum++;
                }
            }

            if (sum == 2) {  // a[i][j]是质数
                f[i][j] = 1;
            } else {
                f[i][j] = 0;
            }
        }
    }

    // 根据 f 数组，来找孤独的素数，统计数量
    int res = 0;
    for (int i = 1; i <= n; ++i) {
        for (int j = 1; j <= m; ++j) {
            if (f[i][j] == 1) {  // 1. 只找 f[i][j]==1 的位置
                // 2. 要判断相邻的 8 个位置里面有几个 1
                int sum2 = 0;
                for (int k = 0; k < 8; ++k) {
                    int ni = i+dx[k], nj = j+dy[k];  // 相邻位置坐标
                    // 还需要判断 (ni,nj) 这个点在不在 二维矩阵 中
```

```
                    if (ni>=1&&ni<=n&&nj>=1&&nj<=m&&f[ni][nj] == 1) {
                        sum2++;
                    }
                }

                // 只有 sum2==0 时，说明8个相邻位置都是合数
                // (i,j) 这个位置是孤独的素数
                if (sum2==0) {
                    res++;
                }
            }
        }
    }

    cout << res << endl;
    return 0;
}
```