

中国矿业大学计算机学院

2021 级本科生课程设计报告

课程名称 程序设计实验

报告时间 2022 年 9 月

学生姓名 杨学通

学 号 08213129

专 业 人工智能

任课教师 杨小冬

《程序设计综合实践》课程报告评分表

序号	课程教学目标	考查方式与考查点	占比	得分
1	目标 1： 掌握一门计算机高级语言，并能使用特定的软件开发工具，设计、开发、调试及运行应用程序。	使用程序设计集成开发工具设计开发、调试应用程序，考察计算机工程基础知识	10%	
2	目标 2： 针对具体的应用问题，进行功能需求分析，确定设计目标，并能绘制算法流程图。	系统需求、系统流程图、软件界面设计、关键类图及软件扩展描述，考察问题分析能力	40%	
3	目标 3： 在进行需求分析的基础上，设计软件运行界面、关键类、编写代码，调试并正确运行满足需求的应用程序。	软件代码编写、调试、运行演示、系统功能扩展，考察计算机工程实践能力	50%	
总分			100%	

评阅人：

年 月 日

目录

2022-9	1
2022-9-1 如此编码.....	1
1 题目描述	1
2. 问题分析	1
3 代码	1
4 运行调试.....	3
2022-9-2 何以包邮?	3
1 题目描述	3
2. 问题分析	3
3 代码	4
4 运行调试.....	5
2022-9-3 防疫大数据.....	6
1 题目描述	6
2. 问题分析	6
3 代码	7
4 运行调试.....	9
2022-6	9
2022-6-1 归一化处理.....	9
1 题目描述	9
2. 问题分析	9
3 代码	10
4 运行调试.....	11
2022-6-2 滑雪！大冒险！	11
1 题目描述	11
2. 问题分析	11
3 代码	12
4 运行调试.....	14
2022-6-3 角色授权.....	15
1 题目描述	15
2 代码	16
3 运行调试.....	19
提交清单.....	20
总结.....	20

实验四 CSP 测试

2022-9

2022-9-1 如此编码

1 题目描述

题目背景

某次测验后，顿顿老师在黑板上留下了一串数字 23333 便飘然而去。凝望着这个神秘数字，小 P 同学不禁陷入了沉思……

题目描述

已知某次测验包含 n 道单项选择题，其中第 i 题 ($1 \leq i \leq n$) 有 a_i 个选项，正确选项为 b_i ，满足 $a_i \geq 2$ 且 $0 \leq b_i < a_i$ 。比如说， $a_i = 4$ 表示第 i 题有 4 个选项，此时正确选项 b_i 的取值一定是 0、1、2、3 其中之一。

顿顿老师设计了如下方式对正确答案进行编码，使得仅用一个整数 m 便可表示 b_1, b_2, \dots, b_n 。

首先定义一个辅助数组 c_i ，表示数组 a_i 的前缀乘积。当 $1 \leq i \leq n$ 时，满足：

$$c_i = a_1 \times a_2 \times \dots \times a_i$$

特别地，定义 $c_0 = 1$ 。

于是 m 便可按照如下公式算出：

$$\begin{aligned} m &= \sum_{i=1}^n c_{i-1} \times b_i \\ &= c_0 \times b_1 + c_1 \times b_2 + \dots + c_{n-1} \times b_n \end{aligned}$$

易知， $0 \leq m < c_n$ ，最小值和最大值分别当 b_i 全部为 0 和 $b_i = a_i - 1$ 时取得。

试帮助小 P 同学，把测验的正确答案 b_1, b_2, \dots, b_n 从顿顿老师留下的神秘整数 m 中恢复出来。

2. 问题分析

由题目所给的关于 m 的公式可以发现当 m 对 c_1 取余时，由于后面的代数式中都含有 c_1 项，故可以将高位的代数式去掉，只剩 $c_0 \times b_1$ ，即 $m \% c_1 = c_0 \times b_1$ 。

同理

$$(m - c_0 \times b_1) \% c_2 = c_1 \times b_2$$

$$(m - c_1 \times b_2) \% c_3 = c_2 \times b_3$$

.....

由此可以将 b_i 求出

3 代码

```
#include<iostream>
using namespace std;
int main()
{
```

```
int n, m;
int* a, * b, * c;
int k;
cin >> n >> m;
c = new int[n + 1];
c[0] = 1;
b = new int[n + 1];
for (int i = 1; i < n + 1; i++)
{
    cin >> c[i];
}
k = c[1];
for (int i = 1; i < n + 1; i++)
{
    c[i] = k;
    k = k * c[i + 1];
}
for (int i = 1; i < n; i++)
{
    b[i] = (m % c[i]) / c[i - 1];
    m -= b[i] * c[i - 1];
}
b[n] = m / c[n - 1];
for (int i = 1; i < n + 1; i++)
{
    cout << b[i] << ' ';
}
delete[] b;
delete[] c;
return 0;
}
```

4 运行调试

2022-9-2 何以包邮?

1 题目描述

题目描述

新学期伊始, 适逢顿顿书城有购书满 x 元包邮的活动, 小 P 同学欣然前往准备买些参考书。一番浏览后, 小 P 初步筛选出 n 本书加入购物车中, 其中第 i 本 ($1 \leq i \leq n$) 的价格为 a_i 元。考虑到预算有限, 在最终付款前小 P 决定再从购物车中删去几本书 (也可以不删), 使得剩余图书的价格总和 m 在满足包邮条件 ($m \geq x$) 的前提下最小。

试帮助小 P 计算, 最终选购哪些书可以在凑够 x 元包邮的前提下花费最小?

2. 问题分析

本题最先想到的方法是, 将所有可能的情况枚举出来, 通过比较, 或者排序的方法, 将符合条件的最小金额输出。该方法易于理解, 对于数据量比较小的情况也非常实用。当数据量增大时, 任意书本的组合情况将十分复杂, 十分浪费内存空间, 计算时间也比较长。

我们换一种思路:

对于每本书, 都有选和不选两种方案, 可以对应一棵完全二叉树;

遍历所有书的两种方案, 找到最小值, 采用 dfs 算法;

对二叉树进行剪枝;

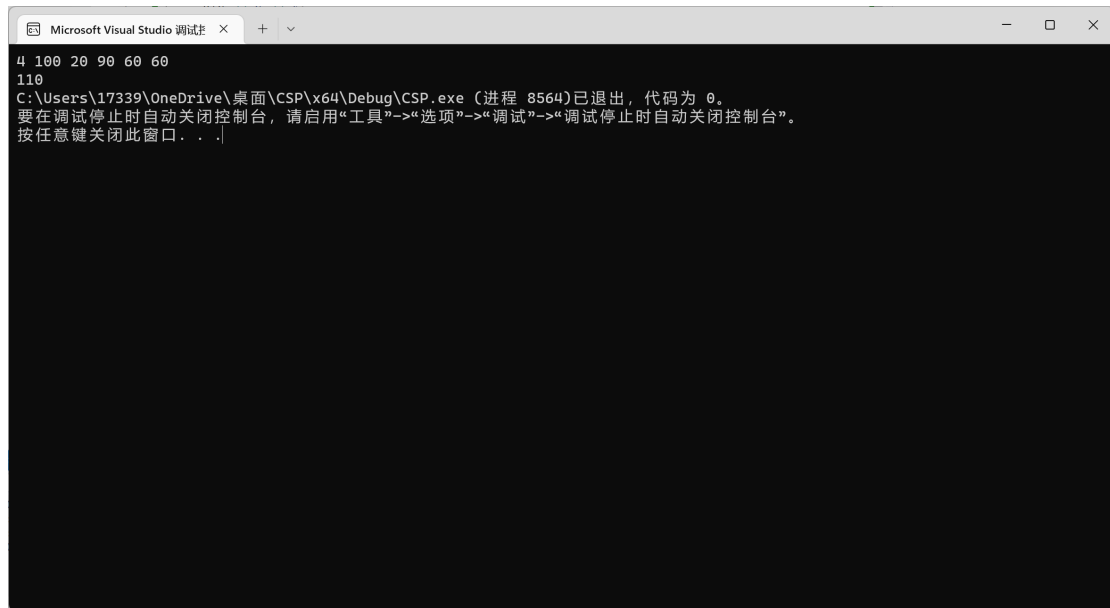
剪枝思路: 当前节点 sum 加上剩余节点的所有值之和已经小于 x 了, 进行剪枝;

求剩余节点的所有值，可以采用前缀和的思想。

3 代码

```
#include<iostream>
#include<algorithm>
using namespace std;
const int N = 50;
int n, x;
int price[N];
int ans = 1e9;
void dfs(int k, int sum)
{
    if (sum >= x)
    {
        ans = min(sum, ans); //更新最小值
    }
    if (k > n)
    {
        return;
    }
    dfs(k+1, sum+price[k]); //选择这本书
    dfs(k+1, sum); //不选择这本书
}
int main()
{
    cin >> n >> x;
    for (int i = 1; i <= n; i++)
    {
        cin >> price[i];
    }
    dfs(1, 0); //从第一本书开始
    cout << ans << endl;
}
```

4 运行调试



The screenshot shows the 'Microsoft Visual Studio 调试' (Microsoft Visual Studio Debug) window. The console output is as follows:

```
4 100 20 90 60 60
110
C:\Users\17339\OneDrive\桌面\CSP\x64\Debug\CSP.exe (进程 8564) 已退出，代码为 0。
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口。 . . .
```


2022-9-3 防疫大数据

1 题目描述

题目描述

为了简化和便于处理问题，我们用连续的整数来表示日期，例如 -1 、 0 、 1 、 2 就是连续的 4 个日期。每日收到的漫游数据分为若干条，每条表示某一个用户在某日到访过某地区，记为： $\langle d, u, r \rangle$ ，其中， u 表示用户， r 表示地区，它们都是一个正整数； d 表示日期，是一个整数。每日收到的漫游数据中，可能会有这样的数据，需要在处理的过程中考虑：

- 延迟数据：即在某日收到的漫游数据中，日期小于当日的日期。例如，在 2 日收到的某条漫游数据，内容是某用户在 1 日到访过某地区；
- 重复数据：即在某日收到的漫游数据中，可能存在两条完全相同的数据；
- 多地访问数据：即在某日收到的漫游数据中，存在两条数据，其日期和用户相同，但地区不同。

每日收到的疫情风险地区的信息，是一个列表，列表中的每个元素表示某个地区在某日被列为疫情风险地区。收到该消息的当日起 7 日内，该地区处于风险状态。即对于地区 r ，我们称 r 在 d 日处于风险状态，当且仅当存在一个日期 $d_0 \in (d-7, d]$ ，在 d_0 日收到地区 r 的风险信息。例如，在 1 日收到地区 1 的风险信息，表示自 1 日（包含）至 8 日（不包含）地区 1 处于风险状态。如果分别在 1 日和 6 日收到地区 1 的风险信息，那么意味着地区 1 自 1 日（包含）至 13 日（不包含）持续处于风险状态。

每日生成的存在风险的手机用户的名单，是一个列表，列表中的每个元素表示某个用户被认为存在疫情风险。我们认为，同时满足下列条件的用户，会被列入当日生成的风险名单：

- 该用户在近 7 日内曾经出现在接收到的漫游数据中，并且近 7 日内有到访某个地区的记录；
- 该用户在近 7 日内到访的地区在到访的那一日处于风险状态；
- 上述存在风险的地区自到访日至生成名单当日持续处于风险状态。

形式化地，在 d 日生成的风险名单中，用户 u 被列入风险名单，当且仅当：

存在一个日期 $d_0 \in (d-7, d]$ ，存在一条 d_0 日收到的漫游数据 $\langle d_1, u, r \rangle$ ，使得

- $d_1 \in (d-7, d]$ ，并且
- 对于任意的 $D \in [d_1, d]$ ，地区 r 在 D 日处于风险状态。

例如，在第 0 日收到下列漫游数据：

$\langle 0, 1, 1 \rangle$ ； $\langle -1, 1, 1 \rangle$ ； $\langle -1, 2, 1 \rangle$ ； $\langle 0, 2, 2 \rangle$

又在第 0 日收到了 1 号地区为风险地区的消息，那么此后第 0 日（包含）至第 7 日（不包含），1 号地区都处于风险状态。

在第 0 日生成的风险名单中，用户 1 被列入风险名单，因为用户 1 在第 0 日到访了 1 号地区。用户 2 不被列入风险名单，因为用户 2 在第 0 日到访了 2 号地区，但是在第 0 日 2 号地区不是风险地区；虽然用户 2 在第 -1 日到访了 1 号地区，但是在第 -1 日 1 号地区不是风险地区。

假设在第 1 日，又收到下列漫游数据：

$\langle 0, 3, 1 \rangle$ ； $\langle 1, 2, 2 \rangle$ ； $\langle 1, 3, 2 \rangle$

同时没有收到新的风险地区的消息，那么在第 1 日生成的风险名单中，用户 1 和 3 被列入风险名单，因为刚收到的信息显示用户 3 在第 0 日到访了 1 号地区，而用户 1 在第 0 日到访了 1 号地区，且在第 1 日 1 号地区仍处于风险状态，虽然第 1 日没有收到用户 1 的漫游数据，但是仍然将用户 1 列入风险名单。

假设后续没有收到更多漫游数据和风险地区信息，直到第七日，收到下列漫游数据：

$\langle 5, 4, 1 \rangle$

同时没有收到新的风险地区的消息，此时，没有用户被列入风险名单。例如对于用户 4，虽然在第 5 日到访了 1 号地区，但是生成风险名单的当日，1 号地区已经不是风险地区。

假设在第 8 日，没有收到更多的漫游数据，但是收到了 1 号地区为风险地区的消息，那么在第 8 日生成的风险名单中，也没有用户被列入风险名单。因为在第 7 日，地区 1 不处于风险状态，用户 4 在第 5 日到访了 1 号地区，虽然第 5 日和本日（第 8 日）地区 1 处于风险状态，但是，由于不满足持续处于风险状态的条件，用户 4 不被列入风险名单。

2. 问题分析

某一日收到的漫游数据中，有 i 个地区划为风险地区，我们记每个地区的编号为 p_i ，由于可能存在重复的数据，我们用一个整型容器组 `set<int>`

fengxianri[]记录当日被划分为风险地区的编号，我们用 fengxianri 的下标作为日期的编号，一旦某地在第 k 天被划为风险地区，则 fengxianri[k]到 fengxianri[k+7]中都会记录这个地区；

定义结构体数组 member，包含三个整型数据成员的 d, u, r 记录某人在某天到了某地；

申请容器 `vector<member> v[50]`，记录所有漫游数据；

分类，将某天进入风险地区的人员编号压入容器rel[i]中(i 表示对应的日期编号)；

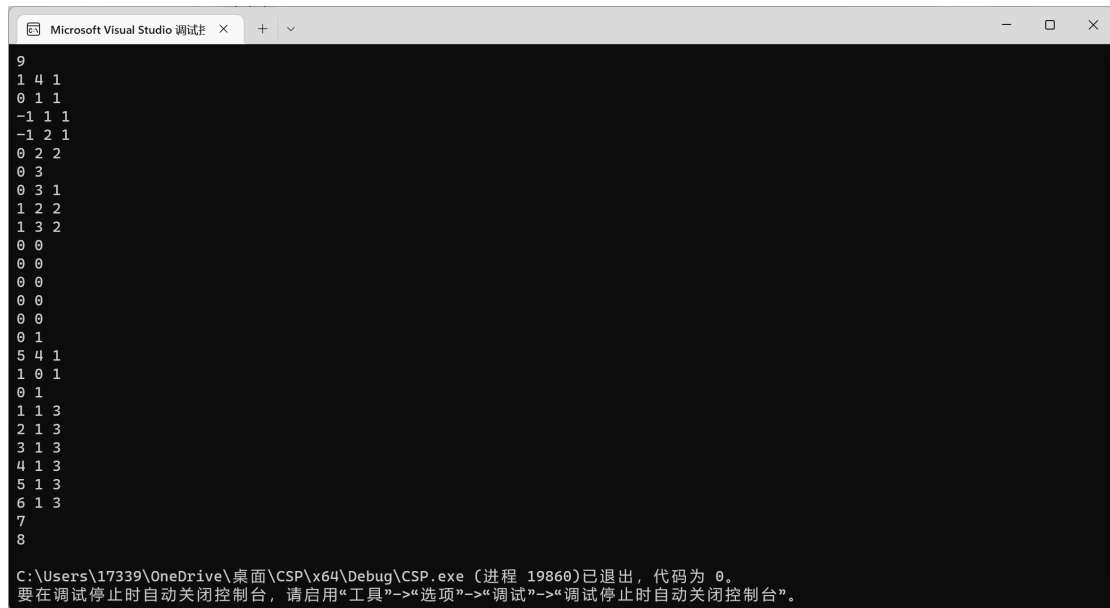
3 代码

```
#include<iostream>
#include<vector>
#include<set>
#include<algorithm>
using namespace std;
set<int> fengxianri[20], rel[20];
struct member
{
    int d, u, r;
}a;
vector<member> v[50];
int n, pi;
int ri[20], mi[20];
int main()
{
    cin >> n;
    for (int i = 0; i < n; i++)
    {
        cin >> ri[i] >> mi[i];
        for (int j = 1; j <= ri[i]; j++)
        {
            cin >> pi;
            for (int k = i; k < i + 7; k++)
            {
                fengxianri[k].insert(pi);
            }
        }
        for (int j = 0; j < mi[i]; j++)
        {
            cin >> a.d >> a.u >> a.r;
            if (a.d < 0)
            {
                continue;
            }
        }
    }
}
```

```
        v[i].push_back(a);
    }
    for (int j = max(0, i - 6); j <= i; j++)
    {
        for (int k = 0; k < v[j].size(); k++)
        {
            int d = v[j][k].d;
            int r = v[j][k].r;
            int u = v[j][k].u;

            if (d < i - 6) {
                continue;
            }
            bool flag = 1;
            for (int day = d; day <= i; day++)
            {
                flag &= fengxianri[day].count(r);
            }
            if (flag) {
                rel[i].insert(u);
            }
        }
    }
}
for (int i = 0; i < n; i++)
{
    cout << i << ' ';
    for (set<int>::iterator it = rel[i].begin(); it != rel[i].end(); it++)
    {
        cout << *it << " ";
    }
    cout << endl;
}
return 0;
}
```

4 运行调试



```

9
1 4 1
0 1 1
-1 1 1
-1 2 1
0 2 2
0 3
0 3 1
1 2 2
1 3 2
0 0
0 0
0 0
0 0
0 0
0 1
5 4 1
1 0 1
0 1
1 1 3
2 1 3
3 1 3
4 1 3
5 1 3
6 1 3
7
8
C:\Users\17339\OneDrive\桌面\CSP\x64\Debug\CSP.exe (进程 19860)已退出，代码为 0。
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。

```

2022-6

2022-6-1 归一化处理

1 题目描述

问题描述

这里假定需要处理的数据为 n 个整数 a_1, a_2, \dots, a_n 。

这组数据的平均值：

$$\bar{a} = \frac{a_1 + a_2 + \dots + a_n}{n}$$

方差：

$$D(a) = \frac{1}{n} \sum_{i=1}^n (a_i - \bar{a})^2$$

使用如下函数处理所有数据，得到的 n 个浮点数 $f(a_1), f(a_2), \dots, f(a_n)$ 即满足平均值为 0 且方差为 1：

$$f(a_i) = \frac{a_i - \bar{a}}{\sqrt{D(a)}}$$

2. 问题分析

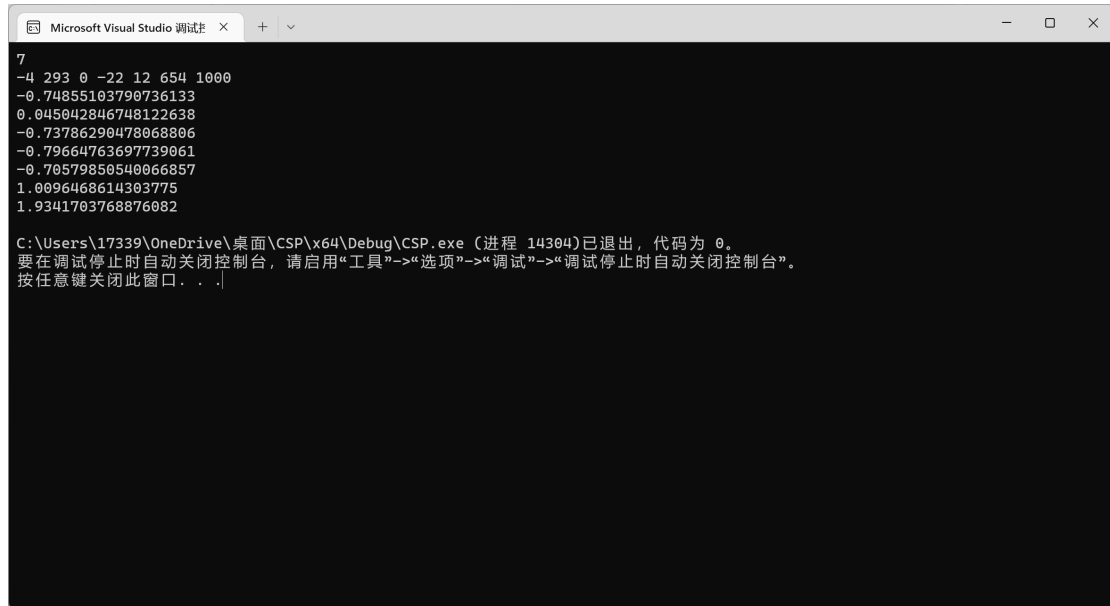
根据公式编写代码即可。

3 代码

```
#include<iostream>
#include<iomanip>
#include<cmath>
using namespace std;
int main() {
    int n;
    double sum = 0;
    double average;
    double fangcha = 0;
    cin >> n;
    int* a = new int[n];
    for (int i = 0; i < n; i++)
    {
        cin >> a[i];
        sum += a[i];
    }
    average= sum / n;

    for (int i = 0; i < n; i++)
    {
        fangcha += (a[i] - average) * (a[i] - average);
    }
    fangcha = sqrt(fangcha / n);
    for (int i = 0; i < n; i++)
    {
        cout << setprecision(17);
        cout << (a[i] - average) / fangcha << endl;
    }
    delete[]a;
    return 0;
}
```

4 运行调试



```
Microsoft Visual Studio 调试
7
-4 293 0 -22 12 654 1000
-0.74855103790736133
0.045042846748122638
-0.73786290478068806
-0.79664763697739061
-0.70579850540066857
1.0096468614303775
1.9341703768876082

C:\Users\17339\OneDrive\桌面\CSP\x64\Debug\CSP.exe (进程 14304) 已退出，代码为 0。
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口。 . . |
```

2022-6-2 滑雪！大冒险！

1 题目描述

问题描述

西西艾弗岛上种有 n 棵树，这些树的具体位置记录在一张绿化图上。
简单地说，西西艾弗岛绿化图可以视作一个大小为 $(L+1) \times (L+1)$ 的 01 矩阵 A ，
地图左下角（坐标 $(0,0)$ ）和右上角（坐标 (L,L) ）分别对应 $A[0][0]$ 和 $A[L][L]$ 。
其中 $A[i][j] = 1$ 表示坐标 (i,j) 处种有一棵树， $A[i][j] = 0$ 则表示坐标 (i,j) 处没有树。
换言之，矩阵 A 中有且仅有的 n 个 1 展示了西西艾弗岛上 n 棵树的具体位置。

传说，大冒险家顿顿的宝藏就埋藏在某棵树下。
并且，顿顿还从西西艾弗岛的绿化图上剪下了一小块，制成藏宝图指示其位置。
具体来说，藏宝图可以看作一个大小为 $(S+1) \times (S+1)$ 的 01 矩阵 B (S 远小于 L)，对应着 A 中的某一部分。
理论上，绿化图 A 中存在着一处坐标 (x,y) ($0 \leq x,y \leq L-S$) 与藏宝图 B 左下角 $(0,0)$ 相对应，即满足：
对 B 上任意一处坐标 (i,j) ($0 \leq i,j \leq S$)，都有 $A[x+i][y+j] = B[i][j]$ 。
当上述条件满足时，我们就认为藏宝图 B 对应着绿化图 A 中左下角为 (x,y) 、右上角为 $(x+S,y+S)$ 的区域。

实际上，考虑到藏宝图仅描绘了很小的一个范围，满足上述条件的坐标 (x,y) 很可能存在多个。
请结合西西艾弗岛绿化图中 n 棵树的位置，以及小 P 手中的藏宝图，判断绿化图中有多少处坐标满足条件。

特别地，藏宝图左下角位置一定是一棵树，即 $A[x][y] = B[0][0] = 1$ ，表示了宝藏埋藏的位置。

2. 问题分析

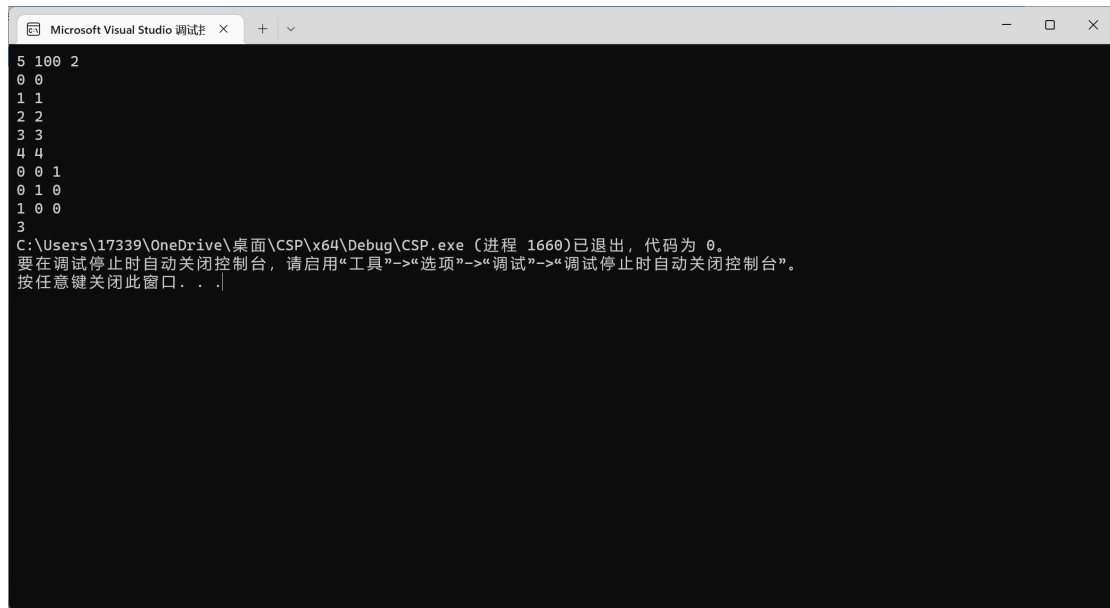
以绿化图的坐标为遍历的起始位置，例如，假设藏宝图为一个 3×3 的矩阵，
以绿化图的 (a,b) 坐标为藏宝图的左上坐标 $(0,0)$ ；
遍历一个藏宝图大小的矩阵，并记录绿化图和藏宝图对应坐标值相同的次数；
如果对应次数等于 9，则该点符合条件；
遍历下一个坐标，直至用藏宝图图覆盖了整个绿化图。

3 代码

```
#include<iostream>
using namespace std;
int green[1005][2];
int treasure[51][51];
int main()
{
    int num = 0;
    int ant = 0;
    int n, l, s;
    cin>>n>>l>>s;
    for (int i = 0; i < n; i++)
    {
        cin>>green[i][0]>>green[i][1];
    }
    for (int i = 0; i < s + 1; i++)
    {
        for (int j = 0; j < s + 1; j++)
        {
            cin>>treasure[s - i][j];
            if (treasure[s - i][j] == 1)
                num++;
        }
    }
    for (int i = 0; i < n; i++)
    {
        if (green[i][0] <= 1 - s && green[i][1] <= 1 - s)
        {
            int x = green[i][0];
            int y = green[i][1];
            int temp = 0;
            for (int j = 0; j < n; j++)
            {
                if (green[j][0] >= x && green[j][0] - s <= x && green[j][1] >=
y && green[j][1] - s <= y)
                {
                    temp++;
                }
            }
            if (temp != num)
            {
                continue;
            }
        }
    }
}
```

```
bool flag = true;
for (int j = 0; j < s + 1; j++)
{
    for (int k = 0; k < s + 1; k++)
    {
        if (treasure[j][k] == 1)
        {
            bool in = false;
            for (int m = 0; m < n; m++)
            {
                if (green[m][0] == j + x && green[m][1] == k + y)
                {
                    in = true;
                    break;
                }
            }
            if (in == false)
            {
                flag = false;
                break;
            }
        }
    }
    if (flag == false)
    {
        break;
    }
}
if (flag == true)
{
    ant++;
}
}
cout<<ant;
return 0;
}
```


4 运行调试



```
Microsoft Visual Studio 调试  x + -  
5 100 2  
0 0  
1 1  
2 2  
3 3  
4 4  
0 0 1  
0 1 0  
1 0 0  
3  
C:\Users\17339\OneDrive\桌面\CSP\x64\Debug\CSP.exe (进程 1660)已退出, 代码为 0。  
要在调试停止时自动关闭控制台, 请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。  
按任意键关闭此窗口. . .|
```

2022-6-3 角色授权

1 题目描述

问题描述

用户表示授权模型中的一个已识别的主体，该识别过程由此前的鉴别过程完成。一个用户具有下列要素：

- 名称：是一个字符串，用于唯一标识一个用户；
- 用户组：是一个数组，包含若干个字符串，表示该用户所属的用户组。

一个待授权的行为，包括下列要素：

- 主体：是一个用户，包括试图进行该行为的用户的名称和该用户所属的用户组；
- 操作：是一个字符串，一般是一个动词，例如 Read、Open、Close 等；
- 资源：表示该行为的操作对象，由资源种类和资源名称描述。资源种类例如 Door、File 等；在一个特定的资源种类中，资源名称唯一确定了一个资源。

需要注意的是，一个待授权的行为的主体信息，即用户名称和所属用户组，是由前一步骤的鉴别过程完成的。因此，每次授权过程中，

每个待授权的行为都会包含主体用户和其关联的用户组的信息。由于鉴权过程中的其它因素，同一个名称的用户在先后两次待授权的行为中所属的用户组可能有区别，

不能存储或记忆此前每个待授权的行为中，用户与用户组的关联情况，而是要按照每次待授权的行为中给出的信息独立判断。

角色是这种授权模型的基本单位，它指明了一个用户可以执行的操作，角色的清单中描述了角色所允许的操作。一个角色包含下列要素：

- 名称，是一个字符串，用于唯一标识一个角色；
- 操作清单，是一个数组，包含一个或多个操作，表示该角色允许执行的操作集合；
- 资源种类清单，是一个数组，包含一个或多个资源种类，表示该角色允许操作的资源的种类集合；
- 资源名称清单，是一个数组，包含若干个资源名称，表示该角色允许操作的资源的名称集合。

判断一个角色能否对某个资源执行某个操作的过程是：

1. 检查该角色的操作清单，如果该角色的操作清单中不包含该操作，且该角色的操作清单中也不包含字符串 *，那么不能执行该操作；
2. 检查该角色的资源种类清单，如果该角色的资源种类清单中不包含该资源的种类，且该角色的资源种类清单中也不包含字符串 *，那么不能执行该操作；
3. 检查该角色的资源名称清单，如果该角色的资源名称清单中不包含该资源的名称，且该角色的资源名称清单不是空数组，那么不能执行该操作；
4. 允许执行该操作。

例如，假设有某个角色 Doorman，其允许执行的操作有 Open 和 Close，其允许操作的资源类型有 Door，其允许操作的资源名称有 FrontDoor 和 BackDoor。

如果某用户与这个角色关联，那么该用户可以对名为 FrontDoor 的 Door 执行 Open 操作，但是不能对 BackDoor 的 Door 执行 Delete 操作。

同时，一个角色能允许进行的操作可以用通配符来表示。例如，另有一个角色 Admin，其允许执行的操作有 *，允许操作的资源类型是 *，其允许操作的资源名称列表为空，那么与该角色关联的所有用户可以执行任何操作。值得注意的是，一个角色的操作清单，只能用允许列表的方式列举该角色允许进行的操作，而不能禁止角色进行某个操作。

角色关联指明了一个用户和一个或多个角色之间的关系。一个角色关联包含下列要素：

- 角色名称，是一个字符串，用于指明一个角色；
- 授权对象清单，是一个数组，包含一个或多个用户名称或者用户组名称，表示该角色关联的用户和用户组的集合。

判断一个用户能否执行某个操作的过程是：

1. 检查所有的角色关联的授权对象清单，如果清单中包含该用户的名称，或者该清单中包含该用户所属的某一个用户组的名称，那么选取该角色关联所关联的角色；
2. 对于所有被选取的角色，判断这些角色是否能否对该资源执行该操作，如果所有角色都不能执行该操作，那么不能执行该操作；
3. 允许执行该操作。

由此可见，一个角色关联可以将一个角色与多个用户或用户组关联起来。例如，如果有一个角色关联，其关联的角色名称为 Doorman，其关联的用户和用户组清单为

用户 foo1、用户 foo2、用户组 bar。那么这些用户会与 Doorman 角色关联：

- 名为 foo1 的用户，属于用户组 bar；
- 名为 foo2 的用户，属于用户组 barz；
- 名为 foo3 的用户，属于用户组 bar 和 barz。

但是，属于用户组 barz 的名为 foo4 的用户不能与 Doorman 的角色关联。

从上述判断规则可以知道，一个用户可能与多个角色相关联，在这种情况下，该用户允许进行的操作是这些角色被允许进行的操作集合的并集。

2 代码

```
#include<iostream>
#include<map>
#include<set>
#include<vector>
#include<queue>
using namespace std;
struct group
{
public:
    set<string> group_role;
};

struct role
{
public:
    set<string> options;
    set<string> resource;
    set<string> r_name;
};

map<string, role> map_role;
map<string, group> map_group;
struct user
{
public:
    set<string> user_group;
    set<string> user_role;

    bool check_empower(string o, string r, string n) {
        set<string>::iterator it = user_role.begin();
        for (; it != user_role.end(); it++) {
            if (map_role[*it].options.count("*") ||
map_role[*it].options.count(o))
                if (map_role[*it].resource.count("*") ||
map_role[*it].resource.count(r))
                    if (map_role[*it].r_name.count(n) ||
map_role[*it].r_name.empty())
                        return true;
        }

        set<string>::iterator it2 = user_group.begin();
        for (; it2 != user_group.end(); it2++) {
            it = map_group[*it2].group_role.begin();
```

```
        for (; it != map_group[*it2].group_role.end(); it++) {
            if (map_role[*it].options.count("*") ||
map_role[*it].options.count(o))
                if (map_role[*it].resource.count("*") ||
map_role[*it].resource.count(r))
                    if (map_role[*it].r_name.count(n) ||
map_role[*it].r_name.empty())
                        return true;
        }
    }
    return false;
}
};
map<string, user> map_user;
```

```
int main() {
    ios::sync_with_stdio(false), cin.tie(0), cout.tie(0); //80-100
    int n, m, q;
    int rel[5];
    cin >> n >> m >> q;
    while (n--)
    {
        string rolename;
        int nv, no, nn;
        cin >> rolename;
        role temprole;
        string str;
        cin >> nv;
        for (int i = 0; i < nv; i++) {
            cin >> str;
            temprole.options.insert(str);
        }
        cin >> no;
        for (int i = 0; i < no; i++) {
            cin >> str;
            temprole.resource.insert(str);
        }
        cin >> nn;
        for (int i = 0; i < nn; i++) {
            cin >> str;
            temprole.r_name.insert(str);
        }
        map_role.insert(pair<string, role>(rolename, temprole));
    }
}
```

```
}

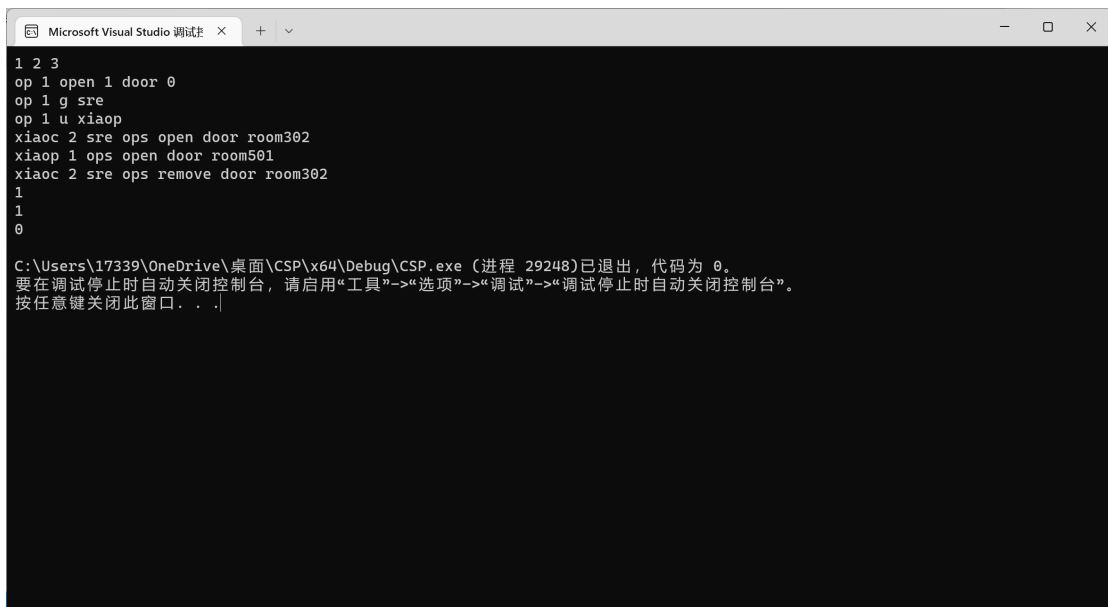
while (m--) {
    string role;
    string str;
    string ugame;
    int ns;
    cin >> role >> ns;
    for (int i = 0; i < 2 * ns; i += 2) {
        cin >> str;
        cin >> ugame;
        if (str == "u") {
            if (map_user.find(ugame) != map_user.end()) {
                map_user[ugame].user_role.insert(role);
            }
            else {
                user tempuser;
                tempuser.user_role.insert(role);
                map_user.insert(pair<string, user>(ugame, tempuser));
            }
        }
        else {
            if (map_group.find(ugame) != map_group.end()) {
                map_group[ugame].group_role.insert(role);
            }
            else {
                group tempgroup;
                tempgroup.group_role.insert(role);
                map_group.insert(pair<string, group>(ugame,
tempgroup));
            }
        }
    }
}

int i = 0;
while (q--) {
    string uname;
    int ng;
    string o, r, n;
    cin >> uname;
    cin >> ng;
    if (map_user.find(uname) == map_user.end()) {
        user u;
        map_user.insert(pair<string, user>(uname, u));
    }
}
```

```
}
map_user[uname].user_group.clear();//30-80

while (ng--)
{
    string group;
    cin >> group;
    map_user[uname].user_group.insert(group);
}
cin >> o >> r >> n;
if (map_user[uname].check_empower(o, r, n))
{
    rel[i]=1; i++;
}
else {
    rel[i] = 0; i++;
}
}
for (int x=0; x<i; x++)
{
    cout << rel[x] << endl;
}
}
```

3 运行调试



The screenshot shows the Microsoft Visual Studio Debug Console window. The title bar reads "Microsoft Visual Studio 调试". The console output is as follows:

```
1 2 3
op 1 open 1 door 0
op 1 g sre
op 1 u xiaop
xiaoc 2 sre ops open door room302
xiaop 1 ops open door room501
xiaoc 2 sre ops remove door room302
1
1
0
```

Below the output, an error message is displayed:

C:\Users\17339\OneDrive\桌面\CSP\x64\Debug\CSP.exe (进程 29248) 已退出，代码为 0。
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口...|

提交清单

[返回试题列表](#)

提交清单										
提交编号	用户名	姓名	试题名称	提交时间	代码长度	编程语言	评测结果	得分	时间使用	空间使用
3315986	杨学通	杨学通	角色授权	11-16 00:16	3.734KB	C++	正确	100	3.828s	242.4MB
3315985	杨学通	杨学通	寻宝！大冒险！	11-16 00:16	3.734KB	C++	错误	0	562ms	3.097MB
3315965	杨学通	杨学通	寻宝！大冒险！	11-16 00:02	1.991KB	C++	正确	100	15ms	2.906MB
3315962	杨学通	杨学通	寻宝！大冒险！	11-16 00:01	45B	C++	编译出错	0	编译出错	编译出错
3315955	杨学通	杨学通	归一化处理	11-15 23:45	525B	C++	正确	100	31ms	3.042MB
3315935	杨学通	杨学通	防疫大数据	11-15 23:26	1.107KB	C++	正确	100	765ms	45.89MB
3315933	杨学通	杨学通	防疫大数据	11-15 23:25	1.102KB	C++	运行错误	60	171ms	11.89MB
3315930	杨学通	杨学通	防疫大数据	11-15 23:24	1.097KB	C++	运行错误	40	15ms	3.609MB
3315781	杨学通	杨学通	何以包邮？	11-15 21:44	836B	C++	正确	100	15ms	7.703MB
3312667	杨学通	杨学通	如此编码	11-13 20:05	422B	C++	正确	100	15ms	2.898MB
3312650	杨学通	杨学通	如此编码	11-13 19:53	397B	C++	正确	100	31ms	2.898MB
3312585	杨学通	杨学通	如此编码	11-13 18:46	540B	C++	错误	80	0ms	2.902MB

总结

通过本次 CSP 模拟考试，我收获很大。利用循环的各种各样的嵌套，数据结构中学到的各种各样的算法，我解决了模拟考试中的许多问题。一个程序包含两个大的部分，一是数据，二是算法。数据包括数据的获取，数据的储存等等，算法即是对已有的数据的加工。通过本次学习，我切实感受到只有语言基础是不够的，需要积累各种各样的算法，来实现程序的各种各样的功能。比如，在 2022-9-1 题目中的编码问题，看似复杂，实际上是对取余运算的一种拓展。当我们想到通过不断的取余，去掉高阶层的数据，通过循环，逐个计算编码时，问题也就不在复杂。算法的背后，是对问题的深刻思考和不断尝试。只有当我们对问题考虑的足够全面，才能使我们的程序更加可靠，更加有效。