# 程序设计实训

南京大学智能科学与技术学院 史桀绮

# 课程形式

1-4周，6-9周每周二7-8节

南雍楼

每周二课上布置本周练习题，周六截止

# 联系方式

jayceesjq@gmail.com

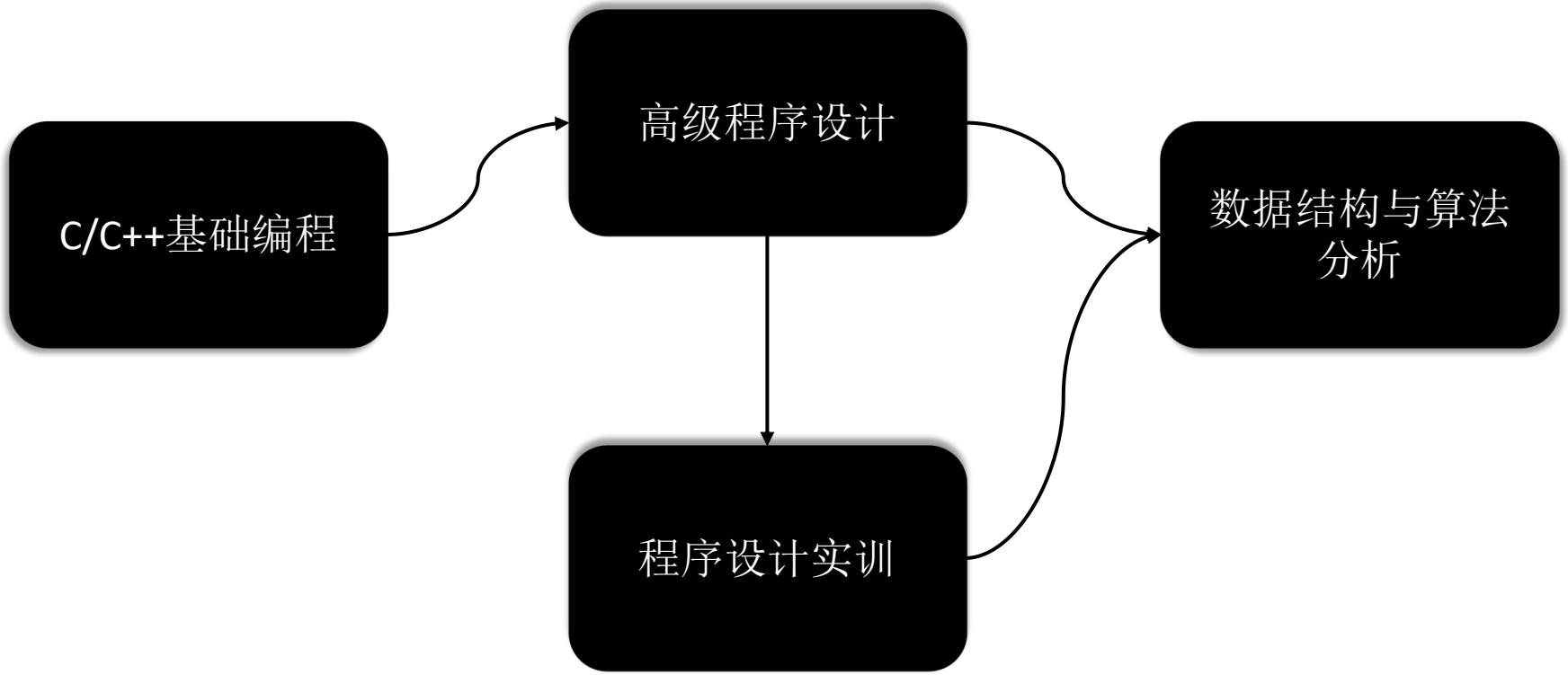
南雍楼

# 分数设置

课上随机签到，课后完成上机习题，结课提交大作业

签到(20%),平时练习(40%)，结课大作业(40%)

# 课程定位



# 课程目的

回顾面向对象的编程

复习基础算法结构

# 参考书目


重点大学计算机专业系列教材
新标准C++程序设计教程
郭炜 编著
清华大学出版社


普通高等教育"十一五"国家级规划教材
21世纪大学本科计算机专业系列教材
李文新
郭 炜 编著
余华山
李晓明 主审
程序设计导引及在线实践（第2版）
http://www.tup.com.cn
根据教育部"高等学校计算机科学与技术专业规范"组织编写
与美国 ACM 和 IEEE CS Computing Curricula 最新进展同步
国家精品课程教材
清华大学出版社


"十二五"普通高等教育本科国家级规划教材
普通高等教育"十一五"国家级规划教材
国家精品课程教材
清华大学 计算机系列教材
郑莉 编著
C++语言程序设计
（第5版）
学生用书
清华大学出版社

# 大作业

- 五人一组完成，结课提交源代码、实验报告和一份6-8分钟的介绍视频，主要解释实现的重要功能，展示完整的编译-运行、试玩流程。需要时可以展示重要代码片段。

- 完成的题目选题(可作为参考):
  - 扫雷小游戏
  - 贪吃蛇小游戏
  - 2048
  - 自选

- 电脑端程序，使用C++完成，需要运用面向对象的编程方法

- 最后二-三节课会在课上给大家6分钟时间讲解工作，每组附3分钟提问-答疑时间，并在教学平台上互相提交评分，作为总得分的35%。剩余5%为老师进行代码复查。

# 课程平台

- https://cslab-cms.nju.edu.cn/classrooms/7whbqffr/announcement
  - 校内平台
  - 用于发布课程通知、调查问卷、互动讨论
  - 最终大作业视频和互相打分也提交在这个平台

- http://njuszoj.openjudge.cn/
  - 校外平台，openjudge
  - 用于进行编程练习
  - 每节课会发布5道必须完成题目，附加n道(n >= 0)附加题目，感兴趣的同学可以进行练手。附加题不计入最终得分。
  - 所有必答题目都会在课堂上进行讲解，附加题推荐同学们在互动讨论中讨论。

准备好了吗！

准备好了吗！

# 算法专题练习-枚举法

# 枚举法的要点

1. 给出解空间

建立简洁的数学模型，想清楚需要枚举的要素

# 枚举法的要点

1. 给出解空间

建立简洁的数学模型，想清楚需要枚举的要素

2. 减少枚举的空间

枚举的范围是什么？是所有的内容都需要枚举吗？

# 枚举法的要点

1. 枚举的内容

建立简洁的数学模型，想清楚需要枚举的要素

2. 枚举的范围

枚举的范围是什么？是所有的内容都需要枚举吗？

3. 枚举的顺序

根据题目判断。比如例题中要求的是最大的符合条件的素数，那么从大到小枚举比较合适。

# 求特殊自然数

一个十进制自然数,它的七进制与九进制表示都是三位数，且七进制与九进制的三位数码表示顺序正好相反。编程求此自然数,并输出显示。

**输入**

无。

**输出**

三行:

第一行是此自然数的十进制表示;

第二行是此自然数的七进制表示;

第三行是此自然数的九进制表示。

# 求特殊自然数

一个十进制自然数,它的七进制与九进制表示都是三位数，且七进制与九进制的三位数码表示顺序正好相反。编程求此自然数,并输出显示。

**输入**

无。

**输出**

三行:

第一行是此自然数的十进制表示;

第二行是此自然数的七进制表示;

第三行是此自然数的九进制表示。

# 求特殊自然数

本题说明了7和9进制都是三位数，因此可以推断这个数字很小，可以使用枚举法直接遍历完成。

1. 需要枚举的内容：十进制数字的7进制和9进制

# 求特殊自然数

本题说明了7和9进制都是三位数，因此可以推断这个数字很小，可以使用枚举法直接遍历完成。

1. 需要枚举的内容：十进制数字的7进制和9进制

2. 枚举的空间： 最大值为(8 + 9*8 + 9*9*8 = 728)，最小为(7*7*1)=49(可以更大一点)

# 求特殊自然数

本题说明了7和9进制都是三位数，因此可以推断这个数字很小，可以使用枚举法直接遍历完成。

1. 需要枚举的内容：十进制数字的7进制和9进制

2. 枚举的空间： 最大值为(8 + 9*8 + 9*9*8 = 728)，最小为(7*7*1)=49(可以更大一点)

3. 选择枚举顺序：从小到大

# 最简真分数

给出n个正整数，任取两个数分别作为分子和分母组成最简真分数，编程求共有几个这样的组合。

**输入**
第一行是一个正整数n（n<=600）。
第二行是n个不同的整数，相邻两个整数之间用单个空格隔开。整数大于1且小于等于1000。
**输出**
一个整数，即最简真分数组合的个数。

**样例输入**
7
3 5 7 9 11 13 15

**样例输出**
17

# 最简真分数

说明中提示n最大为600，数据量较小，可以考虑直接枚举完成。

1. 需要枚举的内容：两个数字是否是最简真分数

# 最简真分数

说明中提示n最大为600，数据量较小，可以考虑直接枚举完成。

1. 需要枚举的内容：两个数字是否是最简真分数

2. 枚举的空间： n个不同的数的两两组合

# 最简真分数

说明中提示n最大为600，数据量较小，可以考虑直接枚举完成。

1. 需要枚举的内容：两个数字是否是最简真分数

2. 枚举的空间： n个不同的数的两两组合

3. 选择枚举顺序：按照输入顺序从左到右

# 最简真分数

两个数字是否是最简真分数->求两个数字是否有公约数->辗转相除法

# 最简真分数

两个数字是否是最简真分数->求两个数字是否有公约数->辗转相除法

两个整数的最大公约数等于其中较小的数和两数相除余数的最大公约数。

# 最简真分数

两个数字是否是最简真分数->求两个数字是否有公约数->辗转相除法

两个整数的最大公约数等于其中较小的数和两数相除余数的最大公约数。

# 最简真分数

两个数字是否是最简真分数->求两个数字是否有公约数->辗转相除法

两个整数的最大公约数等于其中较小的数和两数相除余数的最大公约数。

```
int gcd(int a, int b){
    if (b == 0) return a;
    return gcd(b, a % b);
}
```

没有公约数->返回值是1。

# 最简真分数

```
int gcd(int a, int b){
    if (b == 0) return a;
    return gcd(b, a % b);
}

for ( int i = 1; i <= n; i++)
    for (int j = i+1; j <= n; j++)
        if (gcd(a[i], a[j]) == 1)
            totalnum++;
```

# 猴子吃桃

海滩上有一堆桃子，N只猴子来分。第一只猴子把这堆桃子平均分为N份，多了一个，这只猴子把多的一个扔入海中，拿走了一份。第二只猴子接着把剩下的桃子平均分成N份，又多了一个，它同样把多的一个扔入海中，拿走了一份。第三、第四、……，第N只猴子仍是最终剩下的桃子分成N份，扔掉多了的一个，并拿走一份。编写程序，输入猴子的数量N，输出海滩上最少的桃子数，使得每只猴子都可吃到桃子。

输入
一个整数N。
输出
输出当猴子数量为N时海滩上最少的桃子数。结果保证在int型范围内。

| 样例输入 | 样例输出 |
| --- | --- |
| 2 | 7 |

# 猴子吃桃

海滩上有一堆桃子，N只猴子来分。第一只猴子把这堆桃子平均分为N份，多了一个，这只猴子把多的一个扔入海中，拿走了一份。第二只猴子接着把剩下的桃子平均分成N份，又多了一个，它同样把多的一个扔入海中，拿走了一份。第三、第四、……，第N只猴子仍是最终剩下的桃子分成N份，扔掉多了的一个，并拿走一份。编写程序，输入猴子的数量N，输出海滩上最少的桃子数，使得每只猴子都可吃到桃子。

输入
一个整数N。
输出
输出当猴子数量为N时海滩上最少的桃子数。结果保证在int型范围内。

| 样例输入 | 样例输出 |
| --- | --- |
| 2 | 7 |

# 猴子吃桃

直觉想到:

$X = N * a_1 + 1 = N * a_2 + a_1 + 2 = \ ....$

太复杂了!

# 猴子吃桃

直觉想到:

$X = N * a_1 + 1 = N * a_2 + a_1 + 2 = \ ....$

太复杂了!

# 猴子吃桃

换个思路，从后往前进行枚举。

假设倒数第i个猴子面对的桃子个数为$b_i$，$b_1 = a_n = m * n + 1$

# 猴子吃桃

换个思路，从后往前进行枚举。

假设倒数第i个猴子面对的桃子个数为$b_i$，$b_1 = a_n = m * n + 1$

那么，$b_{i-1} = (b_i - 1) * (n-1) / n$，$b_{i-1} \% (n - 1) = 0$

需要求的是第一只猴子面对的桃子数量bn = a1

# 猴子吃桃

换个思路，从后往前进行枚举。

假设倒数第i个猴子面对的桃子个数为$b_i$，$b_1 = a_n = m * n + 1$

倒数第i-1个猴子，就是倒数第i个猴子后面的猴子，在倒数第i个拿完之后去拿桃子

那么，$b_{i-1} = (b_i - 1) * (n-1) / n$，$b_{i-1} \% (n - 1) = 0$

需要求的是第一只猴子面对的桃子数量$b_n = a_1$

# 猴子吃桃

换个思路，从后往前进行枚举。

假设倒数第i个猴子面对的桃子个数为$b_i$，$b_1 = a_n = m * n + 1$

倒数第i-1个猴子，就是倒数第i个猴子后面的猴子，在倒数第i个拿完之后去拿桃子

那么，$b_{i-1} = (b_i - 1) * (n-1) / n$，$b_{i-1} \% (n - 1) = 0$

倒数第i个猴子扔掉了一个桃子，并拿走了1/n

需要求的是第一只猴子面对的桃子数量$b_n = a_1$

# 猴子吃桃

需要求的是第一只猴子面对的桃子数量$b_n = a_1$

1. 需要枚举的内容：整数m，即最后一只猴子拿走的桃子数量

# 猴子吃桃

需要求的是第一只猴子面对的桃子数量$b_n = a_1$

1. 需要枚举的内容：整数m，即最后一只猴子拿走的桃子数量

2. 枚举的空间：m从1开始一直到找到结果为止，在过程中任意一个b不满足整除n-1就停止

# 猴子吃桃

需要求的是第一只猴子面对的桃子数量$b_n = a_1$

1. 需要枚举的内容：整数m，即最后一只猴子拿走的桃子数量

2. 枚举的空间：m从1开始一直到找到结果为止，在过程中任意一个b不满足整除n-1就停止

3. 选择枚举顺序：从小到大

# 猴子吃桃

还有另外一种解法！

# 猴子吃桃：脑筋急转弯

换个思路，假设给这一堆桃子补上N-1个，就正好可以分给N个猴子。

# 猴子吃桃：脑筋急转弯

换个思路，假设给这一堆桃子补上N-1个，就正好可以分给N个猴子。

假设补完之后桃子总数为X，则第一个猴子分走了总数的1/N，剩下X*(N-1)/N个桃子。

# 猴子吃桃：脑筋急转弯

换个思路，假设给这一堆桃子补上N-1个，就正好可以分给N个猴子。

假设补完之后桃子总数为X，则第一个猴子分走了总数的1/N，剩下X * (N-1)/N个桃子。

第一个猴子实际上拿走了比补之前多一个的桃子，但是没有丢掉桃子，相互抵消了。因此第二只猴子也相当于补上了N-1个桃子。假设原先第二个猴子应该面对$X_2$个桃子，实际第二个猴子面对$X_2$+N-1个桃子，也正好够分给N只猴子。因此，第二只猴子分走X * (N-1)/(N * N)只桃子，剩下X * (N-1) * (N-1) / (N * N)只桃子。

# 猴子吃桃：脑筋急转弯

换个思路，假设给这一堆桃子补上N-1个，就正好可以分给N个猴子。

假设补完之后桃子总数为X，则第一个猴子分走了总数的1/N，剩下X * (N-1)/N个桃子。

第一个猴子实际上拿走了比补之前多一个的桃子，但是没有丢掉仍和桃子。假设原先第二个猴子应该面对$X_2$个桃子，实际第二个猴子面对$X_2$+N-1个桃子，也正好够分给N只猴子。因此，第二只猴子分走X * (N-1)/(N * N)只桃子，剩下X * (N-1) * (N-1) / (N * N)只桃子。

最后一只猴子会取走X * ((N-1)/N)$^{N-1}$ / N只桃子，并且全部拿走。因此X最小需要是$N^N$

# 猴子吃桃：脑筋急转弯

换个思路，假设给这一堆桃子补上N-1个，就正好可以分给N个猴子。

假设补完之后桃子总数为X，则第一个猴子分走了总数的1/N，剩下X * (N-1)/N个桃子。

第一个猴子实际上拿走了比补之前多一个的桃子，但是没有丢掉仍和桃子。假设原先第二个猴子应该面对$X_2$个桃子，实际第二个猴子面对$X_2$+N-1个桃子，也正好够分给N只猴子。因此，第二只猴子分走X * (N-1)/(N * N)只桃子，剩下X * (N-1) * (N-1) / (N * N)只桃子。

最后一只猴子会取走X * ((N-1)/N)$^{N-1}$ / N只桃子，并且全部拿走。因此X最小需要是$N^N$

会出现错误！有没有忘记考虑的边界条件？

# 猴子吃桃：脑筋急转弯

换个思路，假设给这一堆桃子补上N-1个，就正好可以分给N个猴子。

假设补完之后桃子总数为X，则第一个猴子分走了总数的1/N，剩下X * (N-1)/N个桃子。

第一个猴子实际上拿走了比补之前多一个的桃子，但是没有丢掉仍和桃子。假设原先第二个猴子应该面对$X_2$个桃子，实际第二个猴子面对$X_2$+N-1个桃子，也正好够分给N只猴子。因此，第二只猴子分走X * (N-1)/(N * N)只桃子，剩下X * (N-1) * (N-1) / (N * N)只桃子。

最后一只猴子会取走X * ((N-1)/N)$^{N-1}$ / N只桃子，并且全部拿走。考虑n > 2时n-1和n互质，因此X最小需要是$N^N$

# 特殊密码锁

有一种特殊的二进制密码锁，由n个相连的按钮组成（n<30），按钮有凹/凸两种状态，用手按按钮会改变其状态。

然而让人头疼的是，当你按一个按钮时，跟它相邻的两个按钮状态也会反转。当然，如果你按的是最左或者最右边的按钮，该按钮只会影响到跟它相邻的一个按钮。

当前密码锁状态已知，需要解决的问题是，你至少需要按多少次按钮，才能将密码锁转变为所期望的目标状态。

# 特殊密码锁

**输入**

两行，给出两个由0、1组成的等长字符串，表示当前/目标密码锁状态，其中0代表凹，1代表凸。

**输出**

至少需要进行的按按钮操作次数，如果无法实现转变，则输出impossible。

**样例输入**

011

000

**样例输出**

1

# 特殊密码锁

1. 每个灯按两下就会抵消按一下的效果，相当于没有按过。因此，一个灯只有按和不按两个状态。

# 特殊密码锁

1. 每个灯按两下就会抵消按一下的效果，相当于没有按过。因此，一个灯只有按和不按两个状态。

2. 只需要考虑是否需要按下第一个灯。第一个灯状态固定后，第二个灯也随之决定状态，同理后续所有灯。

# 特殊密码锁

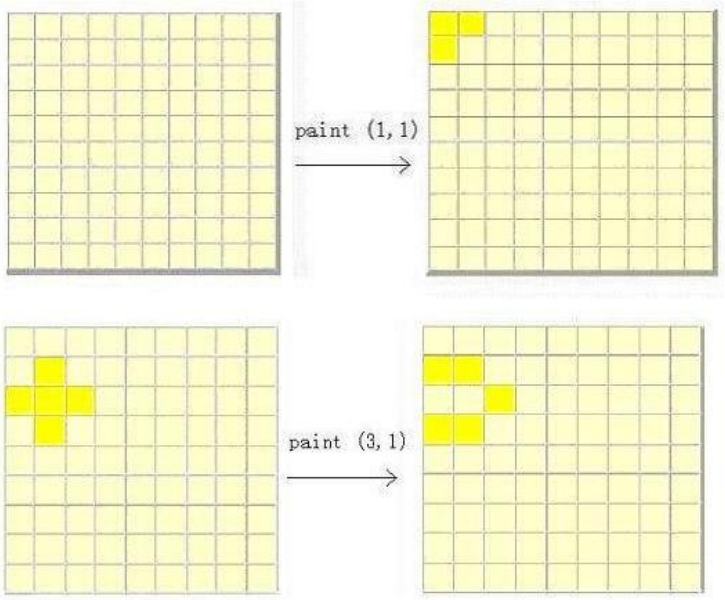
1. 每个灯按两下就会抵消按一下的效果，相当于没有按过。因此，一个灯只有按和不按两个状态。

2. 只需要考虑是否需要按下第一个灯。第一个灯状态固定后，第二个灯也随之决定状态，同理后续所有灯。

因此，只需要枚举第一个灯的状态(两种)，就可以依次推出后一个灯是否要按。枚举第一个灯的两种状态，直接输出操作次数少的。

# 进阶题目：画家问题

有一个正方形的墙，由N*N个正方形的砖组成，其中一些砖是白色的，另外一些砖是黄色的。Bob是个画家，想把全部的砖都涂成黄色。但他的画笔不好使。当他用画笔涂画第(i, j)个位置的砖时， 位置(i-1, j)、 (i+1, j)、 (i, j-1)、 (i, j+1)上的砖都会改变颜色。请你帮助Bob计算出最少需要涂画多少块砖，才能使所有砖的颜色都变成黄色。



# 进阶题目：画家问题

**输入**

第一行是一个整数n (1≤n ≤15)，表示墙的大小。接下来的n行表示墙的初始状态。每一行包含n个字符。第i行的第j个字符表示位于位置(i,j)上的砖的颜色。“w”表示白砖，“y”表示黄砖。

**输出**

如果能够将所有的砖都涂成黄色，则输出最少需要涂画的砖数，否则输出“inf”。

**样例输入**

5

wwwww

wwwww

wwwww

wwwww

wwwww

**样例输出**

15

# 画家问题

1. 同样，一块砖涂画两下就会抵消一下的效果，相当于没有染色过。因此，砖只有白色和黄色两个状态。枚举所有$2^{N*N}$砖头状态同样会导致TE。

# 画家问题

1. 同样，一块砖涂画两下就会抵消一下的效果，相当于没有染色过。因此，砖只有白色和黄色两个状态。枚举所有$2^{N*N}$砖头状态同样会导致TE。

2. 每块砖头的状态会受上下左右四个位置影响。对于第一行砖头，相当于只受到同一排和下一排对应位置砖头影响。

# 画家问题

1. 同样，一块砖涂画两下就会抵消一下的效果，相当于没有染色过。因此，砖只有白色和黄色两个状态。枚举所有$2^{N*N}$砖头状态同样会导致TE。

2. 每块砖头的状态会受上下左右四个位置影响。对于第一行砖头，相当于只受到同一排和下一排对应位置砖头影响。

3. 假如第一排染色情况确定，那么第二排状态也确定了。第一排还是白色的位置第二排一定要染色，否则一定不要染色。

# 画家问题

1. 同样，一块砖涂画两下就会抵消一下的效果，相当于没有染色过。因此，砖只有白色和黄色两个状态。枚举所有$2^{N*N}$砖头状态同样会导致TE。

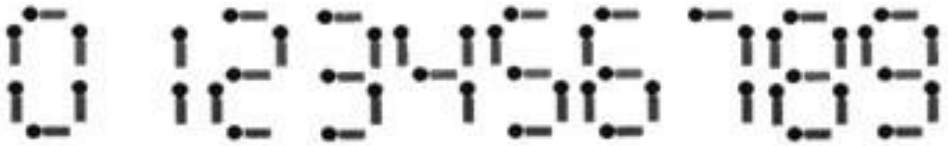
2. 每块砖头的状态会受上下左右四个位置影响。对于第一行砖头，相当于只受到同一排和下一排对应位置砖头影响。

3. 假如第一排染色情况确定，那么第二排状态也确定了。第一排还是白色的位置第二排一定要染色，否则一定不要染色。

因此，只需要枚举第一排砖块的状态，就可以确定后续所有砖头的染色情况，枚举数量改为$2^N$。

# 火柴棒等式

给你n根火柴棍，你可以拼出多少个形如“A+B=C”的等式？等式中的A、B、C是用火柴棍拼出的整数（若该数非零，则最高位不能是0）。用火柴棍拼数字0-9的拼法如图所示：



注意：

1. 加号与等号各自需要两根火柴棍

2. 如果A≠B，则A+B=C与B+A=C视为不同的等式（A、B、C≥0）

3. n根火柴棍必须全部用上

# 火柴棒等式

## 输入
输入一个整数n(n <= 24)。

## 输出
输出能拼成的不同等式的数目。

## 样例输入
5

## 样例输出
0

# 火柴棒等式

显然每个数字只有一种拼法，因此可以提前用数组num[10]存下每个数字需要的火柴棒数量。

# 火柴棒等式

显然每个数字只有一种拼法，因此可以提前用数组num[10]存下每个数字需要的火柴棒数量。

假如枚举A, B, C的值，如何确定枚举范围，即数字的最大值?

# 火柴棒等式

显然每个数字只有一种拼法，因此可以提前用数组num[10]存下每个数字需要的火柴棒数量。

假如枚举A, B, C的值，如何确定枚举范围，即数字的最大值?

注意题目中给定火柴棒数量n=24，可以通过火柴棒数量来进行限制。

# 火柴棒等式

显然每个数字只有一种拼法，因此可以提前用数组num[10]存下每个数字需要的火柴棒数量。

假如枚举A, B, C的值，如何确定枚举范围，即数字的最大值?

注意题目中给定火柴棒数量n=24，可以通过火柴棒数量来进行限制。

1是使用火柴棒最少的(2根)，考虑1111+1=1112使用了25根火柴，因此可以认为两个加数都小于1111。

# 火柴棒等式

显然每个数字只有一种拼法，因此可以提前用数组num[10]存下每个数字需要的火柴棒数量。

假如枚举A, B, C的值，如何确定枚举范围，即数字的最大值?

注意题目中给定火柴棒数量n=24，可以通过火柴棒数量来进行限制。

1是使用火柴棒最少的(2根)，考虑1111+1=1112使用了25根火柴，因此可以认为两个加数都小于1111。

A和B都从0到1111进行枚举，看有多少种满足和等于n的解法。

# 火车上的人数

火车从始发站（称为第1站）开出，在始发站上车的人数为a，然后到达第2站，在第2站有人上、下车，但上、下车的人数相同，因此在第2站开出时（即在到达第3站之前）车上的人数保持为a人。

从第3站起（包括第3站）上、下车的人数有一定规律：上车的人数都是上两站上车人数之和，而下车人数等于上一站上车人数，一直到终点站的前一站（第n-1站），都满足此规律。

现给出的条件是：共有n个车站，始发站上车的人数为a，最后一站下车的人数是m（全部下车）。试问x站开出时车上的人数是多少？

# 火车上的人数

**输入**

一行，包含四个整数a，n，m和x，相邻两个整数之间用单个空格隔开。0 <= a <= 10, 3 <= x < n <= 15, 0 <= m <= 10000。

题目保证数据有唯一解。允许有人在同一站上下车。

**输出**

一个整数，为从x站开出时车上的人数。

**样例输入**

5 7 32 4

**样例输出**

13

# 火车上的人数

寻找规律：

| 站点 | 上车人数 | 下车人数 | 车上人数 |
| --- | --- | --- | --- |
| 1 | a | 0 | a |
| | | | |
| | | | |
| | | | |
| | | | |

# 火车上的人数

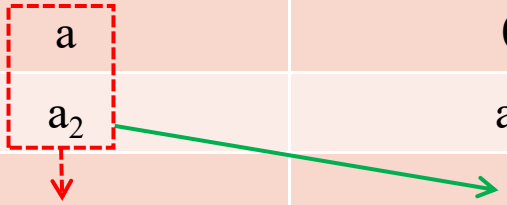
寻找规律：

| 站点 | 上车人数 | 下车人数 | 车上人数 |
| --- | --- | --- | --- |
| 1 | a | 0 | a |
| 2 | $a_2$ | $a_2$ | a |
| | | | |
| | | | |
| | | | |

# 火车上的人数

寻找规律：

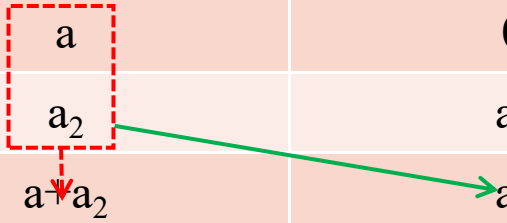
| 站点 | 上车人数 | 下车人数 | 车上人数 |
| --- | --- | --- | --- |
| 1 | a | 0 | a |
| 2 | $a_2$ | $a_2$ | a |
| | | | |
| | | | |
| | | | |



# 火车上的人数

寻找规律：

| 站点 | 上车人数 | 下车人数 | 车上人数 |
| --- | --- | --- | --- |
| 1 | a | 0 | a |
| 2 | $a_2$ | $a_2$ | a |
| 3 | $a+a_2$ | $a_2$ | 2a |
| | | | |
| | | | |



# 火车上的人数

寻找规律：

| 站点 | 上车人数 | 下车人数 | 车上人数 |
| --- | --- | --- | --- |
| 1 | a | 0 | a |
| 2 | $a_2$ | $a_2$ | a |
| 3 | $a+a_2$ | $a_2$ | 2a |
| 4 | $a+2a_2$ | $a+a_2$ | $2a+a_2$ |
| | | | |

# 火车上的人数

寻找规律：

| 站点 | 上车人数 | 下车人数 | 车上人数 |
| --- | --- | --- | --- |
| 1 | a | 0 | a |
| 2 | $a_2$ | $a_2$ | a |
| 3 | $a+a_2$ | $a_2$ | 2a |
| 4 | $a+2a_2$ | $a+a_2$ | $2a+a_2$ |
| 5 | $2a+3a_2$ | $a+2a_2$ | $3a+2a_2$ |

# 火车上的人数

寻找规律：

| 站点 | 上车人数 | 下车人数 | 车上人数 |
| --- | --- | --- | --- |
| 1 | a | 0 | a |
| 2 | $a_2$ | $a_2$ | a |
| 3 | $a+a_2$ | $a_2$ | 2a |
| 4 | $a+2a_2$ | $a+a_2$ | $2a+a_2$ |
| 5 | $2a+3a_2$ | $a+2a_2$ | $3a+2a_2$ |

假设第i站上车$a_i$人，下车$b_i$人，车上人数为$c_i$，那么i>=3时有规律：

$a_i = a_{i-1} + a_{i-2}$

$b_i = a_{i-1}$

$c_i = c_{i-1} + a_i - b_i = c_{i-1} + a_{i-2}$

# 火车上的人数

假设第i站上车$a_i$人，下车$b_i$人，车上人数为$c_i$，那么i>=3时有规律：

$a_i = a_{i-1} + a_{i-2}$

$b_i = a_{i-1}$

$c_i = c_{i-1} + a_i - b_i$

已知最后一站下车的人数是$c_{n-1} = m$（就是n-1站车上的人数），求$c_x$

# 火车上的人数

假设第i站上车ai人，下车bi人，车上人数为ci，那么i>=3时有规律：

$ai = a_{i-1} + a_{i-2}$

$bi = a_{i-1}$

$ci = c_{i-1} + a_i - b_i$

已知最后一站下车的人数是$c_{i-1} = m$（n-1站车上的人数），求$c_x$

按照表格中的假设，第一站上车a人，$a_1=a$，$b_1=0$, $c_1=a$

第二站上车$a_2$人，$a_2=a_2$, $b_2=a_2$, $c_2=a$

其中a为输入数据，可以推导以$a_2$为参数的方程，直接代入$a_2$计算

# 火车上的人数

假设第i站上车ai人，下车bi人，车上人数为ci，那么i>=3时有规律：

$ai = a_{i-1} + a_{i-2}$

$bi = a_{i-1}$

$ci = c_{i-1} + a_i - b_i$

已知最后一站下车的人数是$c_{i-1} = m$（n-1站车上的人数），求$c_x$

按照表格中的假设，第一站上车a人，$a_1=a$，$b_1=0$, $c_1=a$

第二站上车$a_2$人，$a_2=a_2$, $b_2=a_2$, $c_2=a$

其中a为输入数据，可以推导以$a_2$为参数的方程，直接代入$a_2$计算

更方便的解法：不推方程，直接枚举$a_2$

# 火车上的人数

1. 需要枚举的内容：第二站上车人数

# 火车上的人数

1. 需要枚举的内容：第二站上车人数

2. 枚举的空间：

   $n = 4$时，$m = c3 = 2a$, 和$a_2$无关，$a_2$在任意范围内

   $n > 4$时，$m = c_{n-1} > a_2$，因此可以把$a_2$的最大值设置为m-1，枚举[0, m-1]

# 火车上的人数

1. 需要枚举的内容：第二站上车人数

2. 枚举的空间：

   $n = 4$时，$m = c3 = 2a$, 和$a_2$无关，$a_2$在任意范围内

   $n > 4$时，$m = c_{n-1} > a_2$，因此可以把$a_2$的最大值设置为m-1，枚举[0, m-1]

3. 选择枚举顺序：从小到大

递推求出$c_{n-1}$, 看什么情况下满足$c_{n-1} = m$；如果找到，就代表存在符合条件的$a_2$，并计算输出$c_x$

# 再次提醒：大作业

- 五人一组完成，结课提交源代码、实验报告和一份6-8分钟的介绍视频，主要解释实现的重要功能，展示完整的编译-运行、试玩流程。需要时可以展示重要代码片段。

- 完成的题目选题(可作为参考):
  - 扫雷小游戏
  - 贪吃蛇小游戏
  - 2048
  - 自选

- 电脑端程序，使用C++完成，需要运用面向对象的编程方法

- 最后二-三节课会在课上给大家6分钟时间讲解工作，每组附3分钟提问-答疑时间，并在教学平台上互相提交评分，作为总得分的35%。剩余5%为老师进行代码复查。