



中國人民大學

RENMIN UNIVERSITY OF CHINA

信息学院

SCHOOL OF INFORMATION

程序设计荣誉课程

5. 算法1——穷举法

授课教师：游伟 副教授、孙亚辉 副教授

授课时间：周二14:00 – 15:30, 周四10:00 – 11:30 (教学三楼3304)

上机时间：周二18:00 – 21:00 (理工配楼二层机房)

课程主页：<https://www.youwei.site/course/programming>

引子：21级图灵班选拔卷第一、二题

一、设有限集合 $A = \{a_1, a_2, \dots, a_n\}$, $\sum_{i=1}^n a_i$ 叫做集合 A 的和, 记作 S_A .

- 若集合 $P = \{x | x = 2n - 1, n \text{ 是正整数}, n \leq 4\}$, P 的含有3个元素的全体子集分别为 P_1, P_2, \dots, P_k , 求 $\sum_{i=1}^k S_{P_i}$.
- 若集合 $Q = \{x | x = 2n - 1, n \text{ 是正整数}, n \leq 20\}$, 定义 Q^{20} 为 Q 的所有非空子集中满足集合的和不大于20的子集构成的集合 (即 $Q^{20} = \{X | X \subseteq Q \text{ and } X \neq \emptyset \text{ and } S_X \leq 20\}$)。例如: $\{1, 5, 9\} \in Q^{20}$, 而 $\{1, 5, 7, 9\} \notin Q^{20}$ 。求 $|Q^{20}|$ (即 Q^{20} 的元素个数)。

二、Rachel is a deceitful child. She has a funny pattern of telling lies. She tells lies six days a week and will always tell the truth on a certain day of a week.

Rachel made the following statements on three consecutive days:

- Day 1: "I lie on Monday and Tuesday."
- Day 2: "Today is either Thursday, Saturday, or Sunday."
- Day 3: "I lie on Wednesday and Friday."

Which day does Rachel tell the truth?

目录

1. 用穷举法解决逻辑问题
2. 用穷举法解决数值问题

穷举法

- 基本思想：将问题的所有可能的答案一一列举，然后根据条件判断此答案是否合适，保留合适的，丢弃不合适的
- 使用穷举算法解题的基本思路：
 - 确定穷举变量、穷举范围和判定条件
 - 逐一枚举可能的解，验证每个解是否是问题的解
- 穷举算法一般按照如下3个步骤进行
 - 确定题解的可能范围，不能遗漏任何一个真正解，也要避免重复
 - 判断是否是真正解的方法
 - 使可能解的范围降至最小，以便提高解决问题的效率

4.1 用穷举法解决逻辑问题

- 示例1：谁做的好事
- 示例2：破案

示例1：谁做的好事

图灵班有四位同学中的一位做了好事，不留名，表扬信来了之后，校长问这四位是谁做的好事。

A说：不是我。

B说：是C。

C说：是D。

D说：他胡说。

已知三个人说的是真话，一个人说的是假话。现在要根据这些信息，找出做了好事的人。

输出可行解，**如果无解**则输出“No Solution”。（特判是否有解）

示例1：谁做的好事

■ 将四个人说的四句话写成关系表达式

- 在声明变量时，我们让 `thisman` 表示要寻找的做了好事的人，定义它是字符变量，其可能的取值范围为{'A', 'B', 'C', 'D'}，分别代表四个人
- 让 `"=="` 的含义为“是”，让 `"!="` 的含义为“不是”

`char thisman = ""`; // 定义字符变量并初始化为空

说话人	说的话	写成关系表达式
A	“不是我”	<code>thisman != 'A'</code>
B	“是C”	<code>thisman == 'C'</code>
C	“是D”	<code>thisman == 'D'</code>
D	“他胡说”	<code>thisman != 'D'</code>

示例1：谁做的好事

■ 穷举所有可能的状态

- A、B、C、D四个人，只有一位是做好事者。令做好事者为1，未做好事者为0，可以有如下4种状态（情况）
- 第一种状态是假定A是做好事者，第二种状态是假定B是做好事者，...
- 所谓穷举是按照这四种假定**逐一去测试**四个人的话有几句是真话，如果不满足三句为真，就否定掉这一假定，换下一个状态再试

状态	A	B	C	D	赋值表达式
1	1	0	0	0	thisman='A'
2	0	1	0	0	thisman='B'
3	0	0	1	0	thisman='C'
4	0	0	0	1	thisman='D'

示例1：谁做的好事

- 情况1：假定让thisman='A'代入四句话中

说话人	说的话	关系表达式	值
A	thisman!='A';	'A'!='A'	0
B	thisman=='C';	'A'=='C'	0
C	thisman=='D';	'A'=='D'	0
D	thisman!='D';	'A'!='D'	1

四个关系表达式的值的和为1，不满足3句话为真，**假设不成立**，因此显然不是'A'做的好事。

示例1：谁做的好事

- 情况2：假定让thisman='B'代入四句话中

说话人	说的话	关系表达式	值
A	thisman!='A';	'B'!='A'	1
B	thisman=='C';	'B'=='C'	0
C	thisman=='D';	'B'=='D'	0
D	thisman!='D';	'B'!='D'	1

四个关系表达式的值的和为2，不满足3句话为真，**假设不成立**，因此显然不是'B'做的好事。

示例1：谁做的好事

- 情况3：假定让thisman='C'代入四句话中

说话人	说的话	关系表达式	值
A	thisman!='A';	'C'!='A'	1
B	thisman=='C';	'C'=='C'	1
C	thisman=='D';	'C'=='D'	0
D	thisman!='D';	'C'!='D'	1

四个关系表达式的值的和为3，**假设成立**，因此就是'C'做的好事。

示例1：谁做的好事

- 情况4：假定让thisman='D'代入四句话中

说话人	说的话	关系表达式	值
A	thisman!='A';	'D'!='A'	1
B	thisman=='C';	'D'=='C'	0
C	thisman=='D';	'D'=='D'	1
D	thisman!='D';	'D'!='D'	0

四个关系表达式的值的和为2，不满足3句话为真，**假设不成立**，因此显然不是'D'做的好事。

示例1：谁做的好事

- 用流程图表示

第一块
循环结构

for (k=1; k<=4; k=k+1)

被试者 **thisman = 64+k;**
sum = (被试者 thisman != 'A')+
(被试者 thisman == 'C')+
(被试者 thisman == 'D')+
(被试者 thisman != 'D');

真

sum == 3

假

输出该被试者;
有解标志 **flag=1;**

第二块
分支结构

真

flag != 1

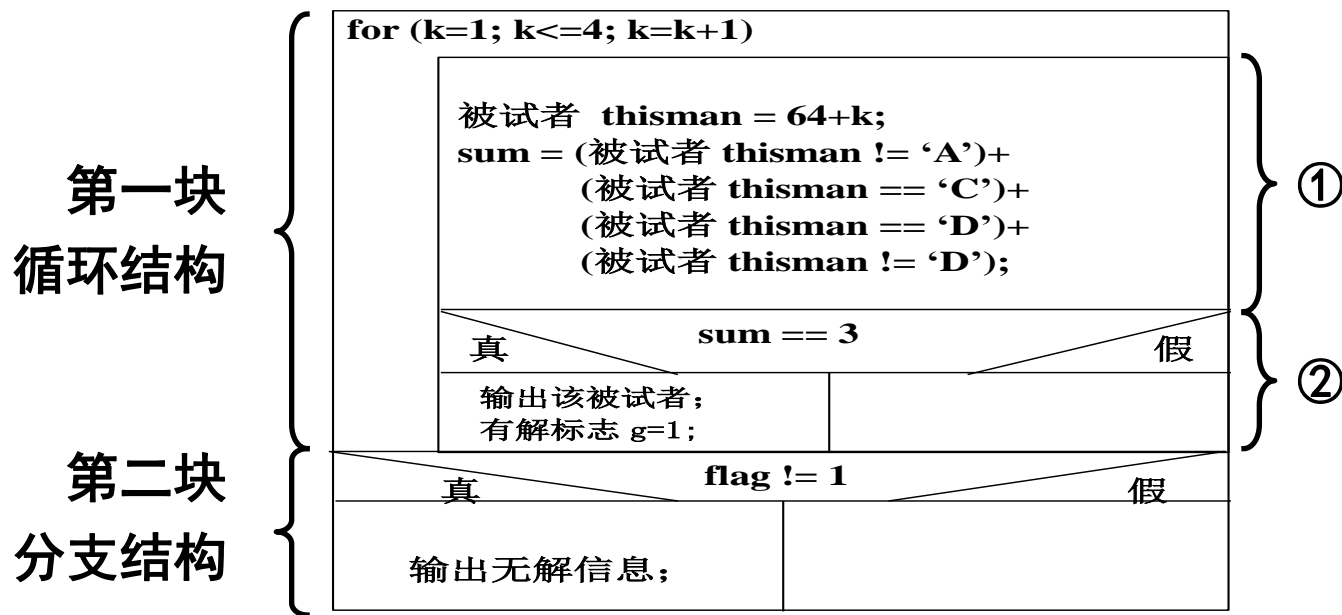
假

输出无解信息;

示例1：谁做的好事

■ 用流程图表示

- 第一块是**循环结构**，功能是产生被试对象，依次为A、B、C、D。并测试四句话有多少句为真，如有3句为真，则可确定做好事者，同时置有解标志为1（①中含两条赋值语句，②中含一条分支语句）
- 第二块为**分支结构**，功能是判断是否无解，如为真，则输出无解信息



示例1：谁做的好事

■ 程序源代码

```
1. #include <stdio.h>
2. int main(int argc, char **argv) {
3.     int k=0, sum=0, flag=0;
4.     char thisman = ' ';
5.     for (k=1; k<=4; k++) {
6.         thisman = 64+k; //A的ASCII码是65
7.         sum = (thisman!='A') + (thisman=='C') + (thisman=='D') + (thisman!='D');
8.         if (sum == 3) {
9.             printf("%c did the good thing.\n", thisman);
10.            flag = 1;
11.        }
12.    }
13.    if (flag != 1) printf("Can't find who did the good thing.\n");
14.}
```

思考：可否通过穷举谁说的假话来解题？

代码的问题：B、C说谎的时候，
没有穷举所有可能的情况

谁做的好事 (穷举撒谎者)

```
1. int main() {
2.     char liar = ' ', thisman = ' ';
3.     int a, b, c, d, sum, flag = 0;
4.     for (liar = 'A'; liar <= 'D'; liar++) {
5.         switch (liar) {
6.             case 'A':
7.                 thisman = 'A';
8.                 a = (thisman != 'A');
9.                 b = (thisman == 'B');
10.                c = (thisman == 'C');
11.                d = (thisman != 'D');
12.                sum = a + b + c + d;
13.                if (sum == 3) {
14.                    printf("A did the good thing");
15.                    flag = 1;
16.                }
17.                break;
18.            case 'B':
19.                thisman = 'D';
20.                a = (thisman != 'A');
21.                b = (thisman == 'B');
22.                c = (thisman == 'C');
23.                d = (thisman != 'D');
24.                sum = a + b + c + d;
25.                if (sum == 3) {
26.                    printf("D did the good thing");
27.                    flag = 1;
28.                }
29.                break;
```

```
30.        case 'C':
31.            thisman = 'C';
32.            a = (thisman != 'A');
33.            b = (thisman == 'B');
34.            c = (thisman == 'C');
35.            d = (thisman != 'D');
36.            sum = a + b + c + d;
37.            if (sum == 3) {
38.                printf("C did the good thing");
39.                flag = 1;
40.            }
41.            break;
42.        case 'D':
43.            thisman = 'D';
44.            a = (thisman != 'A');
45.            b = (thisman == 'B');
46.            c = (thisman == 'C');
47.            d = (thisman != 'D');
48.            sum = a + b + c + d;
49.            if (sum == 3) {
50.                printf("D did the good thing");
51.                flag = 1;
52.            }
53.            break;
54.        }
55.    }
56.    if (flag == 0) printf("no solution");
57.    return 0;
58.}
```

by 武思豪

谁做的好事（枚举撒谎者）

```
1. #include <stdio.h>
2. #include <string.h>
3. int ans[4] = {0, 0, 0, 0}; //做好事情况, 1表示做好事
4. void jd(int a, int b) { //judge,按照ppt表格内的逻辑编写
5.     if (a == 0) { //a是枚举第几个人, b是他的说话状态
6.         if (b == 0) {
7.             ans[0] = 1;
8.             return;
9.         }
10.    }
11.    if (a == 1) {
12.        if (b == 1) {
13.            ans[2] = 1;
14.            return;
15.        }
16.    }
17.    if (a == 2) {
18.        if (b == 1) {
19.            ans[3] = 1;
20.            return;
21.        }
22.    }
23.    if (a == 3) {
24.        if (b == 0) {
25.            ans[3] = 1;
26.            return;
27.        }
28.    }
29. }
```

```
30. int main()
31. {
32.     int truee[4] = {1, 1, 1, 1}; //撒谎情况, 0表示撒谎
33.
34.     for (int i = 0; i <= 3; i++)
35.     {
36.         //第一次没加这句错了, 因为每一次循环都要重新开始累加
37.         memset(ans, 0, sizeof(ans));
38.         truee[i] = 0; // 0代表说假话, for循环枚举
39.         jd(0, truee[0]); // 对每个人的真假状态进行judge
40.         jd(1, truee[1]); // 用jd函数对ans数组进行赋值
41.         jd(2, truee[2]);
42.         jd(3, truee[3]);
43.         if ((ans[0] + ans[1] + ans[2] + ans[3]) == 1)
44.         {
45.             // 结果确定, 只有一个人做了这件事情
46.             printf("%c", 'A' + i);
47.             return 0;
48.         }
49.         truee[i] = 1; // 恢复当年的模样
50.     }
51.     printf("not found");
52.     return 0;
53. }
```

谁做的好事（枚举撒谎者）

- 将每个人的陈述转换成表达式：
 - 数组statements记录每个人的陈述，statements[i][j]代表第i个人对第j个人是不是做好事的描述
 - statements[i][j] == 1: 第i个人认为第j个人做了好事
 - statements[i][j] == -1: 第i个人认为第j个人没有做好事
 - statements[i][j] == 0: 第i个人对第j个人是否做好事没有描述
- 如果i撒谎，则i对每个人的描述取反，statements[i][j] *= -1
- 判断是否可行：
 - 每个i对每个j的描述不产生冲突
 - 做好事的人数只能为1

谁做的好事 (枚举撒谎者)

```
1. #include <stdio.h>
2. #include <math.h>

3. int main()
4. {
5.     int liar, i, j, sum, abs_sum;
6.     int contradiction; //是否冲突 (abs_sum > abs(sum))
7.     int cnt_good; //做好事的人数, 只能为1
8.     int solution = 0;
9.     int statements[4][4] = {
10.        {-1, 0, 0, 0}, //A说: 不是我
11.        {0, 0, 1, 0},  //B说: 是C
12.        {0, 0, 0, 1},  //C说: 是D
13.        {0, 0, 0, -1}, //D说: 他胡说
14.    };
15.
16.    for (liar = 0; liar <= 3; liar++)
17.    {
18.        contradiction = 0;

19.        for (j = 0; j <= 3; j++) //liar撒谎则他的描述取反
20.            statements[liar][j] *= -1;
21.
22.        cnt_good = 0;
23.        for (j = 0; j <= 3; j++) {
24.            for (sum = 0, i = 0; i <= 3; i++)
25.                sum += statements[i][j];
26.            for (abs_sum = 0, i = 0; i <= 3; i++)
27.                abs_sum += abs(statements[i][j]);
```

```
28.            if (abs_sum > abs(sum)) { //几个人的描述产生矛盾
29.                contradiction = 1;
30.                break;
31.            }

32.            if (!contradiction && sum > 0) cnt_good++;
33.        }

34.        if (cnt_good == 1) { //做好事的人数只能为1
35.            printf("%c lies.", 'A'+liar);
36.            solution = 1;
37.        }
38.
39.        for (j = 0; j <= 3; j++) //恢复原样
40.            statements[liar][j] *= -1;
41.    }
42.
43.    if (!solution) printf("No solution\n");
44.    return 0;
45.}
```

示例2：破案

某地刑侦大队对涉及六个嫌疑人的一桩疑案进行分析：

- A、B 至少有一人作案；
- A、E、F 三人中至少有两人参与作案；
- A、D 不可能是同案犯；
- B、C 或同时作案，或与本案无关；
- C、D 中有且仅有一人作案；
- 如果 D 没有参与作案，则 E 也不可能参与作案

输出一组可行解。

示例2：破案

- 将案情的每一条写成逻辑表达式
 - CC1: A和B至少有一人作案
 - CC2: A和D不可能是同案犯
 - CC3: A、E、F 中至少有两人涉嫌作案
 - CC4: B和C或同时作案，或都与本案无关
 - CC5: C、D中有且仅有一人作案
 - CC6: 如果D没有参与作案，则E也不可能参与作案
- 将案情分析的6条归纳成一个破案综合判断条件CC

CC = CC1 && CC2 && CC3 && CC4 && CC5 && CC6

示例2：破案

- CC1: A和B至少有一人作案

- 令 A 变量表示 A 作案, B 变量表示 B 作案

- 显然这是或的关系, 有 $CC1 = (A \parallel B)$

A	B	CC1
0	0	0
1	0	1
0	1	1
1	1	1

示例2：破案

- CC2: A和D不可能是同案犯
 - $CC2 = \neg (A \ \&\& \ D)$

A	D	A&&D	CC2
1	0	0	1
1	1	1	0
0	0	0	1
0	1	0	1

示例2：破案

- CC3：A、E、F 中至少有两人涉嫌作案
 - 有3种不同的情况
 - 情况1：A 和 E 作案， $(A \ \&\& \ E)$
 - 情况2：A 和 F 作案， $(A \ \&\& \ F)$
 - 情况3：E和 F 作案， $(E \ \&\& \ F)$
 - 这三种可能性是 或 的关系， $CC3 = (A \ \&\& \ E) \ || \ (A \ \&\& \ F) \ || \ (E \ \&\& \ F)$
- CC4：B和C或同时作案，或都与本案无关
 - 有2种不同的情况
 - 情况1：同时作案， $(B \ \&\& \ C)$
 - 情况2：都与本案无关， $(!B \ \&\& \ !C)$
 - 这两种可能性是 或 的关系， $CC4 = (B \ \&\& \ C) \ || \ (!B \ \&\& \ !C)$
- CC5：C、D中有且仅有一人作案
 - $CC5 = (C \ \&\& \ !D) \ || \ (!C \ \&\& \ D)$

示例2：破案

■ CC6：如果D没有参与作案，则E也不可能参与作案

■ 分析这一条比较麻烦一些，可以列出真值表再归纳

■ $CC6 = D \parallel !E$ （实际上是蕴含关系： $!D \rightarrow !E$ 等价于 $E \rightarrow D$ 等价于 $D \parallel !E$ ）

D	E	!E	CC6	含义	
1	1	0	1	D作案，E也作案	可能
1	0	1	1	D作案，E不作案	可能
0	0	1	1	D不作案，E也不可能作案	可能
0	1	0	0	D不作案，E却作案	不可能

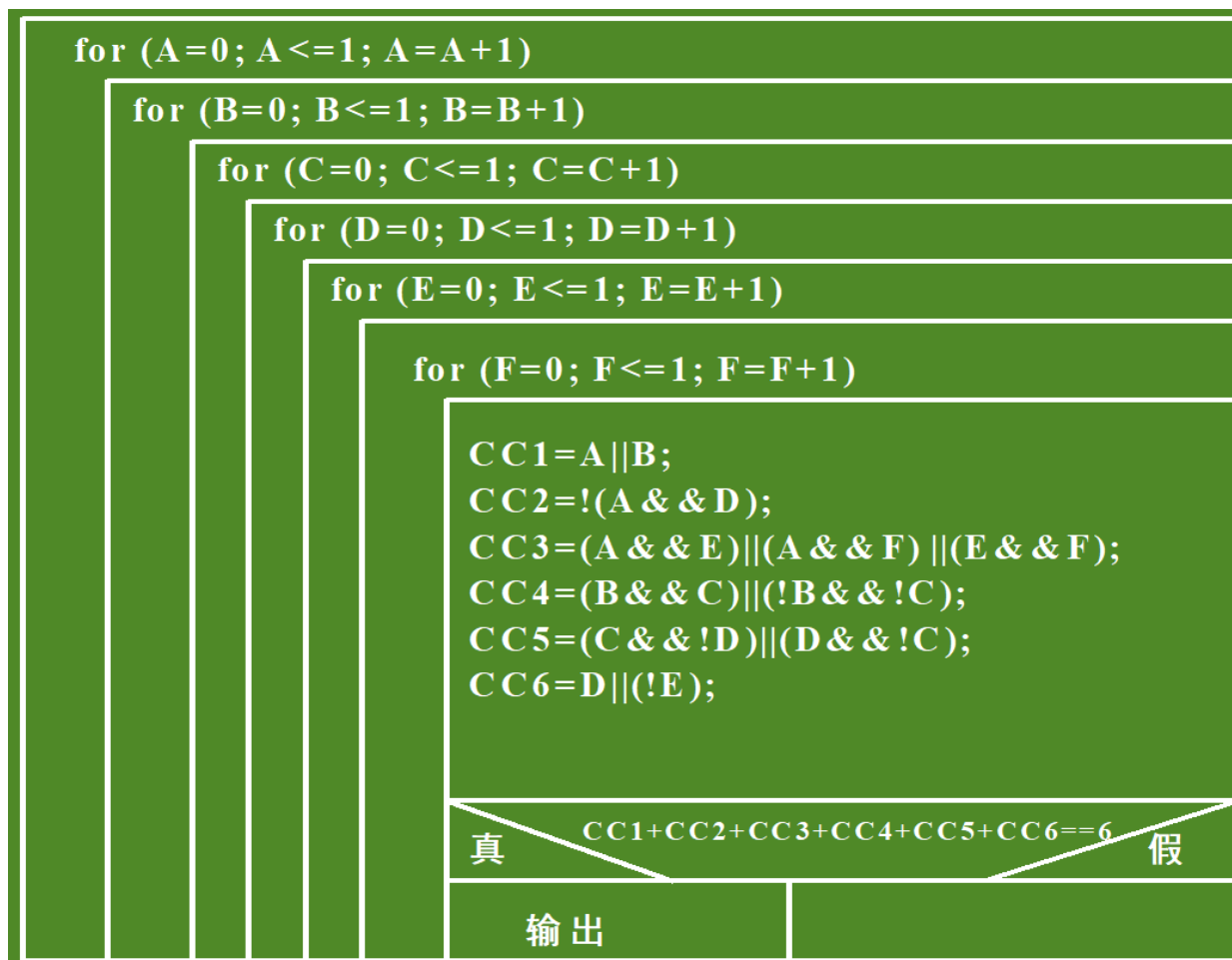
示例2：破案

- 采取穷举方法，穷举什么呢？穷举组合
 - 6个人每个人都有作案或不作案两种可能，故有 2^6 种组合，从这些组合中挑出符合6条分析的作案者
 - 定义6个整数变量，分别表示6个人A, B, C, D, E, F
 - 枚举每个人的可能性：让0表示不是罪犯，让1表示是罪犯

A	B	C	D	E	F
0	0	0	0	0	0
0	0	0	0	0	1
0	0	0	0	1	0
.....
1	1	1	1	1	1

示例2：破案

■ 用流程图表示



示例2：破案

■ 程序源代码

这段程序有什么问题？

```
1. #include <stdio.h>
2. int main(int argc, char **argv) {
3.     int cc1, cc2, cc3, cc4, cc5, cc6;
4.     int A, B, C, D, E, F;
5.     for (A=0; A<=1; A++)
6.         for (B=0; B<=1; B++)
7.             for (C=0; C<=1; C++)
8.                 for (D=0; D<=1; D++)
9.                     for (E=0; E<=1; E++)
10.                        for (F=0; F<=1; F++)
11.                            {
12.                                cc1 = A||B; cc2 = !(A&&D); cc3 = (A&&E)|| (A&&F)|| (E&&F);
13.                                cc4 = (B&&C)|| (!B&&!C); cc5 = (C&&!D)|| (!C&&D); cc6 = D||!E;
14.                                if (cc1+cc2+cc3+cc4+cc5+cc6 == 6) {
15.                                    printf("A:%d B:%d C:%d D:%d E:%d F:%d\n", A, B, C, D, E, F);
16.                                    break;
17.                                }
18.                            }
19. }
```

A:1 B:1 C:1 D:0 E:0 F:1

思考：用穷举法求解选拔题二

二、 Rachel is a deceitful child. She has a funny pattern of telling lies. She tells lies six days a week and will always tell the truth on a certain day of a week.

Rachel made the following statements on three consecutive days:

- Day 1: "I lie on Monday and Tuesday."
- Day 2: "Today is either Thursday, Saturday, or Sunday."
- Day 3: "I lie on Wednesday and Friday."

Which day does Rachel tell the truth?

求解选拔题二

■ 穷举变量:

- truth: 代表Rachel周几说真话
- day: 代表Day1是周几

■ 穷举范围: 0~6, 分别代表周日/一/二/三/四/五/六 (为什么这么表示?)

■ 判定条件分四种情况:

- 情况1: Day1说真话 $\Leftrightarrow \text{day} == \text{truth}$
- 情况2: Day2说真话 $\Leftrightarrow (\text{day}+1)\%7 == \text{truth}$
- 情况3: Day3说真话 $\Leftrightarrow (\text{day}+2)\%7 == \text{truth}$
- 情况4: 其它 $\Leftrightarrow \text{truth} != \text{day} \ \&\& \ \text{truth} != (\text{day}+1)\%7 \ \&\& \ \text{truth} != (\text{day}+2)\%7$

Day 1: "I lie on Monday and Tuesday."
Day 2: "Today is either Thursday, Saturday, or Sunday."
Day 3: "I lie on Wednesday and Friday."

情况1: Day1说真话

- `day == truth` ←情况1对应的逻辑表达式
- `truth != 1 && truth != 2` ←Day1对应的逻辑表达式 (T)
- `!((day+1)%7 == 4 || (day+1)%7 == 6 || (day+1)%7 == 0)` ←Day2对应的逻辑表达式 (F)

化简后得: `(day+1)%7 != 4 && (day+1)%7 != 6 && (day+1)%7 != 0`

- `!(truth != 3 && truth != 5)` ←Day3对应的逻辑表达式 (F)

化简后得: `truth == 3 || truth == 5`

- 综合上述表达式:

```
if (day == truth) flag =
```

```
    (truth != 1 && truth != 2) &&
```

```
    ((day+1)%7 != 4 && (day+1)%7 != 6 && (day+1)%7 != 0) &&
```

```
    (truth == 3 || truth == 5);
```

Day 1: "I lie on Monday and Tuesday."
Day 2: "Today is either Thursday, Saturday, or Sunday."
Day 3: "I lie on Wednesday and Friday."

情况2: Day2说真话

- $(\text{day}+1)\%7 == \text{truth}$

←情况2对应的逻辑表达式

- $!(\text{truth} != 1 \ \&\& \ \text{truth} != 2)$

←Day1对应的逻辑表达式 (F)

化简后得: $\text{truth} == 1 \ || \ \text{truth} == 2$

- $(\text{day}+1)\%7 == 4 \ || \ (\text{day}+1)\%7 == 6 \ || \ (\text{day}+1)\%7 == 0$

←Day2对应的逻辑表达式 (T)

- $!(\text{truth} != 3 \ \&\& \ \text{truth} != 5)$

←Day3对应的逻辑表达式 (F)

化简后得: $\text{truth} == 3 \ || \ \text{truth} == 5$

- 综合上述表达式:

```
if ((day+1)%7 == truth) flag =
```

```
    (truth == 1 || truth == 2) &&
```

```
    ((day+1)%7 == 4 || (day+1)%7 == 6 || (day+1)%7 == 0) &&
```

```
    (truth == 3 || truth == 5);
```

Day 1: "I lie on Monday and Tuesday."
Day 2: "Today is either Thursday, Saturday, or Sunday."
Day 3: "I lie on Wednesday and Friday."

情况3: Day3说真话

- $(\text{day}+2)\%7 == \text{truth}$ ←情况3对应的逻辑表达式
- $!(\text{truth} != 1 \ \&\& \ \text{truth} != 2)$ ←Day1对应的逻辑表达式 (F)
化简后得: $\text{truth} == 1 \ || \ \text{truth} == 2$
- $!((\text{day}+1)\%7 == 4 \ || \ (\text{day}+1)\%7 == 6 \ || \ (\text{day}+1)\%7 == 0)$ ←Day2对应的逻辑表达式 (F)
化简后得: $(\text{day}+1)\%7 != 4 \ \&\& \ (\text{day}+1)\%7 != 6 \ \&\& \ (\text{day}+1)\%7 != 0$
- $\text{truth} != 3 \ \&\& \ \text{truth} != 5$ ←Day3对应的逻辑表达式 (T)
- 综合上述表达式:

```
if ((day+2)%7 == truth) flag =  
    (truth == 1 || truth == 2) &&  
    ((day+1)%7 != 4 && (day+1)%7 != 6 && (day+1)%7 != 0) &&  
    (truth != 3 && truth != 5);
```

Day 1: "I lie on Monday and Tuesday."
Day 2: "Today is either Thursday, Saturday, or Sunday."
Day 3: "I lie on Wednesday and Friday."

情况4：其它

- `day != truth && (day+1)%7 != truth && (day+2)%7 != truth` ←情况4对应的逻辑表达式
- `!(truth != 1 && truth != 2)` ←Day1对应的逻辑表达式 (F)
化简后得: `truth == 1 || truth == 2`
- `!((day+1)%7 == 4 || (day+1)%7 == 6 || (day+1)%7 == 0)` ←Day2对应的逻辑表达式 (F)
化简后得: `(day+1)%7 != 4 && (day+1)%7 != 6 && (day+1)%7 != 0`
- `!(truth != 3 && truth != 5)` ←Day3对应的逻辑表达式 (F)
化简后得: `truth == 3 || truth == 5`
- 综合上述表达式:
`if (day != truth && (day+1)%7 != truth && (day+2)%7 != truth) flag =`
 `(truth == 1 || truth == 2) &&`
 `((day+1)%7 != 4 && (day+1)%7 != 6 && (day+1)%7 != 0) &&`
 `(truth == 3 && truth == 5)`

Day 1: "I lie on Monday and Tuesday."
Day 2: "Today is either Thursday, Saturday, or Sunday."
Day 3: "I lie on Wednesday and Friday."

求解选拔题二

■ 程序源代码

```
1. int main(void) {
2.     int truth, day, flag = 0;
3.     for (truth = 0; truth < 7; truth++) {
4.         for (day = 0; day < 7; day++) {
5.             if (day == truth) flag = (truth != 1 && truth != 2) &&
6.                 ((day+1)%7 != 4 && (day+1)%7 != 6 && (day+1)%7 != 0) &&
7.                 (truth == 3 || truth == 5);
8.             else if ((day+1)%7 == truth) flag = (truth == 1 || truth == 2) &&
9.                 ((day+1)%7 == 4 || (day+1)%7 == 6 || (day+1)%7 == 0) &&
10.                (truth == 3 || truth == 5);
11.            else if ((day+2)%7 == truth) flag = (truth == 1 || truth == 2) &&
12.                ((day+1)%7 != 4 && (day+1)%7 != 6 && (day+1)%7 != 0) &&
13.                (truth != 3 && truth != 5);
14.            else flag = (truth == 1 || truth == 2) &&
15.                ((day+1)%7 != 4 && (day+1)%7 != 6 && (day+1)%7 != 0) &&
16.                (truth == 3 && truth == 5);
17.            if (flag) { flag = 1; break; }
18.        }
19.        if (flag) break;
20.    }
21.    printf("truth: %d day: %d", truth, day);
22.}
```

4.2 用穷举法解决数值问题

- 示例3：百钱买百鸡
- 示例4：同构数

示例3：百钱买百鸡

公元前五世纪，我国古代数学家张丘建在《算经》一书中提出了“百鸡问题”：鸡翁一值钱五，鸡母一值钱三，鸡雏三值钱一。百钱买百鸡，问鸡翁、鸡母、鸡雏各几何？

输出所有可行解。

一般解法

- 穷举变量: n_cocks , n_hens , n_chicks 分别表示鸡翁、鸡母、鸡雏数量
- 穷举范围:
 - $0 \leq n_cocks \leq 100$
 - $0 \leq n_hens \leq 100$
 - $0 \leq n_chicks \leq 100$
- 判定条件:
 - $n_cocks + n_hens + n_chicks = 100$ (百鸡)
 - $5 * n_cocks + 3 * n_hens + n_chicks / 3 = 100$ (百钱)

一般解法

总共执行了 $101^3=1030301$ 次判定

■ 程序源代码

```
1. #include <stdio.h>
2. int main(int argc, char **argv)
3. {
4.     int n_cocks, n_hens, n_chicks, cnt = 0;
5.     for (n_cocks = 0; n_cocks <= 100; n_cocks++)
6.         for (n_hens = 0; n_hens <= 100; n_hens++)
7.             for (n_chicks = 0; n_chicks <= 100; n_chicks+=3)
8.                 {
9.                     if (n_cocks + n_hens + n_chicks == 100 && 5 * n_cocks + 3 * n_hens + n_chicks / 3 == 100)
10.                        printf("n_cocks: %d, n_hens: %d, n_chicks: %d\n", n_cocks, n_hens, n_chicks);
11.                        cnt++;
12.                }
13.     printf("cnt: %d\n", cnt);
14. }
```

```
n_cocks: 0, n_hens: 25, n_chicks: 75
n_cocks: 3, n_hens: 20, n_chicks: 77
n_cocks: 4, n_hens: 18, n_chicks: 78
n_cocks: 7, n_hens: 13, n_chicks: 80
n_cocks: 8, n_hens: 11, n_chicks: 81
n_cocks: 11, n_hens: 6, n_chicks: 83
n_cocks: 12, n_hens: 4, n_chicks: 84
cnt: 1030301
```

这段程序有什么问题？

优化解法

- 穷举变量: n_cocks , n_hens , n_chicks 分别表示鸡翁、鸡母、鸡雏数量
- 穷举范围:
 - $0 \leq n_cocks \leq \min(100, 100/5)$,
 - $0 \leq n_hens \leq \min(100-n_cocks, (100-5*n_cocks)/3)$
 - $0 \leq n_chicks \leq \min(100-n_cocks-n_hens, (100-5*n_cocks-3*n_hens)*3)$
- 判定条件:
 - $n_cocks + n_hens + n_chicks = 100$ (百鸡)
 - $5 * n_cocks + 3 * n_hens + n_chicks / 3 = 100$ (百钱)

优化解法

总共执行了8133次判定

■ 程序源代码

```
1. #include <stdio.h>
2. int min(int x, int y)
3. {
4.     return (x < y ? x : y);
5. }
6. int main(int argc, char **argv)
7. {
8.     int n_cocks, n_hens, n_chicks, cnt = 0;
9.     for (n_cocks = 0; n_cocks <= min(100, 100/5); n_cocks++)
10.        for (n_hens = 0; n_hens <= min(100-n_cocks, (100-5*n_cocks)/3); n_hens++)
11.            for (n_chicks = 0; n_chicks <= min(100-n_cocks-n_hens, (100-5*n_cocks-3*n_hens)*3); n_chicks += 3)
12.                {
13.                    if (n_cocks + n_hens + n_chicks == 100 && 5 * n_cocks + 3 * n_hens + n_chicks / 3 == 100)
14.                        printf("n_cocks: %d, n_hens: %d, n_chicks: %d\n", n_cocks, n_hens, n_chicks);
15.                    cnt++;
16.                }
17.     printf("cnt: %d\n", cnt);
18. }
```

```
n_cocks: 0, n_hens: 25, n_chicks: 75
n_cocks: 4, n_hens: 18, n_chicks: 78
n_cocks: 8, n_hens: 11, n_chicks: 81
n_cocks: 12, n_hens: 4, n_chicks: 84
cnt: 8133
```

示例4：同构数

穷举 $[a, b]$ 之间的同构数。所谓“同构数”，是指一个正整数 n 是它平方的尾部，则称 n 为同构数。例如6的平方数是36，6出现在36的右端，6就是同构数；76的平方数是5776，76也是同构数。

输出数位和**最大的一组解**。一个数的“数位和”是指这个数的每个数位的和。

示例4：同构数

- 穷举变量及其范围： $i \in [a, b]$
- 判断条件： i 是 $i*i$ 的尾部
- 难点：
 - 如何判断一个数是另一个数的尾部？
 - 如何求数位和？
 - 如何记录最大的解？

示例4：同构数

■ 如何判断一个数是另一个数的尾部

```
1. int is_suffix(int x, int y) { //判断x是否为y的尾部
2.     int flag = 1;
3.     while (x > 0 && y > 0) {
4.         if (x % 10 != y %10) { flag = 0; break; }
5.         x /= 10, y /= 10;
6.     }
7.     return flag;
8. }
```

■ 如何求数位和

```
1. int sum_of_digits(int i) { //求i的数位和
2.     int sum = 0;
3.     while (i > 0) {
4.         sum += (i%10);
5.         i /= 10;
6.     }
7.     return sum;
8. }
```

■ 如何记录最大的解

```
1. int max_sum = 0; //max_sum记录目前为止最大的数位和，赋初值为0，因为正数求和的最小值为0
2. int max_i = 0; //max_i记录目前为止数位和最大的同构数

3. int sod = sum_of_digits(i);
4. if (sod > max_sum) {
5.     max_sum = sod;
6.     max_i = i;
7. }
```

示例4：同构数

■ 程序源代码

```
1. int sum_of_digits(int i) { //求i的数位和
2.     int sum = 0;
3.     while (i > 0) { sum += (i%10); i /= 10; }
4.     return sum;
5. }
6. int is_suffix(int x, int y) { //判断x是否为y的尾部
7.     int flag = 1;
8.     while (x > 0 && y > 0) {
9.         if (x % 10 != y % 10) { flag = 0; break; }
10.        x /= 10, y /= 10;
11.    }
12.    return flag;
13.}
14.int main() {
15.    int a, b, i, sod;
16.    int max_sum = 0, max_i = 0;
17.    scanf("%d %d", &a, &b);
18.    for (i = a; i <= b; i++) {
19.        if (i%10 != 1 && i%10 != 5 && i%10 != 6) continue; //优化：个位数必须是1或5或6，才有可能同构数
20.        if (!is_suffix(i, i*i)) continue;
21.        sod = sum_of_digits(i);
22.        if (sod > max_sum) { max_sum = sod; max_i = i; }
23.    }
24.    printf("max_i: %d\n", max_i);
25.}
```

思考：用穷举法求解选拔题一

一、设有限集合 $A = \{a_1, a_2, \dots, a_n\}$, $\sum_{i=1}^n a_i$ 叫做集合 A 的和, 记作 S_A .

- 若集合 $P = \{x | x = 2n - 1, n \text{ 是正整数}, n \leq 4\}$, P 的含有3个元素的全体子集分别为 P_1, P_2, \dots, P_k , 求 $\sum_{i=1}^k S_{P_i}$.
- 若集合 $Q = \{x | x = 2n - 1, n \text{ 是正整数}, n \leq 20\}$, 定义 Q^{20} 为 Q 的所有非空子集中满足集合的和不超过20的子集构成的集合 (即 $Q^{20} = \{X | X \subseteq Q \text{ and } X \neq \emptyset \text{ and } S_X \leq 20\}$)。例如: $\{1, 5, 9\} \in Q^{20}$, 而 $\{1, 5, 7, 9\} \notin Q^{20}$ 。求 $|Q^{20}|$ (即 Q^{20} 的元素个数)。

求解选拔题一

- $Q = \{x | x = 2n - 1, n \text{ 是正整数}, n \leq 20\}$
 $= \{1, 3, 5, 7, 9, 11, 13, 15, 17, 19\}$
- 穷举变量：
 - 穷举Q的1个元素的子集，1个穷举变量
 - 穷举Q的2个元素的子集，2个穷举变量
 - 穷举Q的3个元素的子集，3个穷举变量
 - 穷举Q的4个元素的子集，4个穷举变量
 - 无需穷举Q的5+个元素的子集，因为5+个元素必然大于20
- 判定条件：各穷举变量的和小等于20

求解选拔题一

■ 程序源代码

```
1. #include <stdio.h>
2. int main(int argc, char **argv) {
3.     int a, b, c, d;
4.     int cnt1=0, cnt2=0, cnt3=0, cnt4=0;
5.     //穷举Q的1个元素的子集
6.     for (a=1; a<=19; a+=2) cnt1++;
7.     printf("cnt1: %d\n", cnt1);
8.     //穷举Q的2个元素的子集
9.     for (a=1; a<=19; a+=2)
10.         for (b=a+2; b<=19; b+=2)
11.             if (a+b <= 20) cnt2++;
12.     printf("cnt2: %d\n", cnt2);
```

```
11.     //穷举Q的3个元素的子集
12.     for (a=1; a<=19; a+=2)
13.         for (b=a+2; b<=19; b+=2)
14.             for (c=b+2; c<=19; c+=2)
15.                 if (a+b+c <= 20) cnt3++;
16.     printf("cnt3: %d\n", cnt3);
17.     //穷举Q的4个元素的子集
18.     for (a=1; a<=19; a+=2)
19.         for (b=a+2; b<=19; b+=2)
20.             for (c=b+2; c<=19; c+=2)
21.                 for (d=c+2; d<=19; d+=2)
22.                     if (a+b+c+d <= 20) cnt4++;
23.     printf("cnt4: %d\n", cnt4);
24.
25.     printf("total: %d", cnt1+cnt2+cnt3+cnt4);
26. }
```

```
cnt1: 10
cnt2: 25
cnt3: 16
cnt4: 4
total: 55
```

思考：假如 $Q = \{x | x = 2n - 1, n \text{ 是正整数}, n \leq m\}$ ， m 由用户输入，那么如何进行穷举？