暴力求解法之枚举算法

“暴力求解法”：把问题所有的可能情况都列举出来，然后根据要求逐一判断，最后找到问题的解。是一种“没有办法的办法”，看上去很“笨”，但在问题范围不是很大的情况下往往很有效，而且准确率也很高，有些题目使用这种“暴力求解”方法，能得部分分。

一．简单枚举算法

一一列举问题所有可能的情况，找出问题的解，往往使用循环嵌套+选择结构实现，所以比较简单。

1.基本思想

枚举法的基本思想是根据提出的问题枚举所有可能状态，并用问题给定的条件检验哪些是需要的，哪些是不需要的。能使命题成立，即为其解。

一般简单的枚举结构：循环+判断语句。

2 枚举法的条件

虽然枚举法本质上属于搜索策略，但是它与后面讲的回溯法有所不同。因为适用枚举法求解的问题必须满足两个条件：

⑴可预先确定状态的元素个数 n；

⑵状态元素 a_1，a_2，…，a_n 的可能值为一个连续的值域。

3 枚举法的框架结构

设 a_{i1}—状态元素 a_i 的最小值；a_{ik}—状态元素 a_i 的最大值(1≤i≤n)，即 $a_{11}≤a_1≤a_{1k}$，$a_{21}≤a_2≤a_{2k}$，$a_{i1}≤a_i≤a_{ik}$，……，$a_{n1}≤a_n≤a_{nk}$。

结构格式：

```
for a1←a11 to a1k do
    for a2←a21 to a2k do
        ……
            for ai←ai1 to aik do
                ……
                    for an←an1 to ank do
                        if 状态(a1，…，ai，…，an)满足检验条件
```

then 输出问题的解；

4 枚举法的优缺点

枚举法的优点：

⑴由于枚举算法一般是现实生活中问题的“直译”，因此**比较直观，易于理解**；

⑵由于枚举算法建立在考察大量状态、甚至是穷举所有状态的基础上，所以**算法的正确性比较容易证明**

枚举法的缺点:

枚举算法的效率取决于枚举状态的数量以及单个状态枚举的代价，因此**效率比较低**。

直译”枚举：直接根据题意设定枚举对象、范围和约束条件。

注意认真审题，不要疏漏任何条件

5.枚举算法的优化：

（1）减少枚举的状态（减少循环层数）。

（2）减少状态的枚举范围（缩小状态的上下界）。

（3）选择合适的枚举对象和顺序。

1．7650 不定方程求解

http://noi.openjudge.cn/ch0201/7650/

【描述】

给定正整数 a，b，c。求不定方程 ax+by=c 关于未知数 x 和 y 的所有非负整数解组数。

【输入】

一行，包含三个正整数 a，b，c，两个整数之间用单个空格隔开。每个数均不大于 1000。

【输出】

一个整数，即不定方程的非负整数解组数。

【样例输入】

2 3 18

【样例输出】

4

【参考代码:】

```
#include<cstdio>
#include<iostream>
using namespace std;
int main(){
    int a,b,c,s=0;
    cin>>a>>b>>c;
    for(int x=0;x<=1000;x++)
            for(int y=0;y<=1000;y++)
                    if(a*x+b*y==c)s++;
    cout<<s<<endl;
    return 0;
}
```

如果每个数都不大于 100000？1000000?

输入：20 30 1000000

看运行时间

改进 1：

```
#include<cstdio>
#include<iostream>
using namespace std;
int main(){
    int a,b,c,s=0;
    cin>>a>>b>>c;
    for(int x=0;x<=c/a;x++){
            for(int y=0;y<=(c-a*x)/b;y++){
                    if(a*x+b*y==c)s++;
            }
    }
    cout<<s<<endl;
    return 0;
}
```

改进 2：

```
#include<cstdio>
#include<iostream>
using namespace std;
int main(){
    int a,b,c,s=0;
    cin>>a>>b>>c;
    for(int x=0;x<=c/a;x++){
            if((c-a*x)%b==0)s++;
    }
    cout<<s<<endl;
    return 0;
}
```

提供枚举效率的方法：

（1）减少变量的枚举范围；

（2）减少循环层数，减少枚举的变量。

附：

3526:最简真分数

http://noi.openjudge.cn/ch0201/3526/

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

参考代码：

```
#include<cstdio>
#include<iostream>
#include<algorithm>
using namespace std;
int a[610];
int f[1001][1001];
int n;
int gcd(int a,int b){
	return b==0?a:gcd(b,a%b);
}
int main(){
	cin>>n;
	for(int i=0;i<n;i++)cin>>a[i];
	sort(a,a+n);
	int s=0;
	for(int i=0;i<n-1;i++)
		for(int j=i+1;j<n;j++)
			if(gcd(a[i],a[j])==1)s++;
	cout<<s<<endl;
	return 0;
}
```

（3）选择合适的枚举顺序，减少枚举范围：排序后再枚举。

扩展：

能生成多少个不同的分数值（最简后）？

2．1749:数字方格

http://noi.openjudge.cn/ch0201/1749/

【描述】

如上图，有 3 个方格，每个方格里面都有一个整数 a1，a2，a3。已知 0 <= a1, a2, a3 <= n，而且 a1 + a2 是 2 的倍数，a2 + a3 是 3 的倍数， a1 + a2 + a3 是 5 的倍数。

你的任务是找到一组 a1，a2，a3，使得 a1 + a2 + a3 最大。

【输入】

一行，包含一个整数 n (0 <= n <= 100)。

【输出】

一个整数，即 a1 + a2 + a3 的最大值。

【样例输入】

3

【样例输出】

5

【参考代码】

```
#include<cstdio>
#include<iostream>
using namespace std;
int n,ans=0;
int main(){
    cin>>n;
    for(int a1=0;a1<=n;a1++)
        for(int a2=0;a2<=n;a2++)
           for(int a3=0;a3<=n;a3++)
                if((a1+a2)%2==0&&(a2+a3)%3==0&&(a1+a2+a3)%5==0)
                    ans=max(ans,a1+a2+a3);
    cout<<ans<<endl;
    return 0;
}
```

3. 1812:完美立方

http://noi.openjudge.cn/ch0201/1812/

【描述】

形如 $a^3= b^3 + c^3 + d^3$ 的等式被称为完美立方等式。例如 $12^3= 6^3 + 8^3 + 10^3$ 。

编写一个程序，对任给的正整数 N (N≤100)，寻找所有的四元组(a, b, c, d)，使得 $a^3 = b^3 + c^3 + d^3$，其中 a,b,c,d 大于 1, 小于等于 N，且 b<=c<=d。

【输入】

一个正整数 N (N≤100)。

【输出】

每行输出一个完美立方。输出格式为：

Cube = a, Triple = (b,c,d)

其中 a,b,c,d 所在位置分别用实际求出四元组值代入。

请按照 a 的值，从小到大依次输出。当两个完美立方等式中 a 的值相同，则 b 值小的优先输出、仍相同则 c 值小的优先输出、再相同则 d 值小的先输出。

【样例输入】

24

【样例输出】

Cube = 6, Triple = (3,4,5)

Cube = 12, Triple = (6,8,10)

```
Cube = 18, Triple = (2,12,16)
Cube = 18, Triple = (9,12,15)
Cube = 19, Triple = (3,10,18)
Cube = 20, Triple = (7,14,17)
Cube = 24, Triple = (12,16,20)
```

【参考代码】

```
//1812:完美立方：枚举顺序
#include<cstdio>
#include<cmath>
#include<iostream>
using namespace std;
int main(){
    int n;
    cin>>n;
    for(int a=2;a<=n;a++)
            for(int b=2;b<a;b++)
                    for(int c=b;c<a;c++)
                            for(int d=c;d<a;d++){
                                    int a3=d*d*d+b*b*b+c*c*c;
                                    if(a*a*a==a3) printf("Cube = %d, Triple = (%d,%d,%d)\n",a,b,c,d);
                            }
    return 0;
}
```

4．砝码称重

http://noi.openjudge.cn/ch0201/8755/

【问题描述】设有 1g、2g、3g、5g、10g、20g 的砝码各若干枚（其总重<=1000），求用这些砝码能称出不同的重量个数。

【文件输入】输入 1g、2g、3g、5g、10g、20g 的砝码个数。
【文件输出】输出能称出不同重量的个数。
【样例输入】1 1 0 0 0 0
【样例输出】3

参考代码：

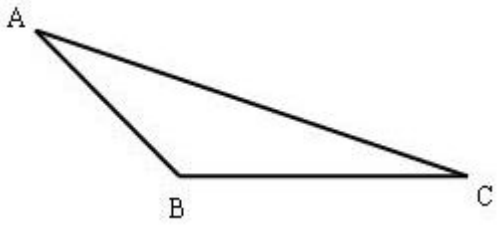
```
#include<iostream>
using namespace std;
int f[1001];
```

```
int a[7],c[7];
int main(){
    for(int i=1;i<=6;i++)cin>>a[i];
    for(int i1=0;i1<=a[1];i1++)
        for(int i2=0;i2<=a[2];i2++)
            for(int i3=0;i3<=a[3];i3++)
                for(int i4=0;i4<=a[4];i4++)
                    for(int i5=0;i5<=a[5];i5++)
                        for(int i6=0;i6<=a[6];i6++)
                            f[i1+i2*2+i3*3+i4*5+i5*10+i6*20]=1;
    int ans=0;
    for(int i=1;i<=1000;i++)ans=ans+f[i];
    cout<<"Total="<<ans<<endl;
    return 0;
}
```

思考：有无其他算法？将来多重背包问题。

5．周长最大三角形

有 n(<=100)根木棍，已知他们的长度(<=10000)，现在从中选出 3 根木棍组成周长尽可能长的三角形。



请计算出最大周长，如果无法组成三角形输出“no”。
输入第一行：n
第二行:n 根木棍的长度。
如：
输入：
6
2 3 10 4 1 6
输出：
13
（选 3 4 6）

分析：
朴素的方法：分别枚举 3 条边判断能否组成三角形，如果能则更新最大值。
参考代码：

```
#include<cstdio>
#include<iostream>
```

```
using namespace std;
int a[101];
int main(){
    int n,s=0;
    cin>>n;
    for(int i=1;i<=n;i++)cin>>a[i];
    for(int i=1;i<=n-2;i++)
        for(int j=i+1;j<=n-1;j++)
            for(int k=j+1;k<=n;k++)
                if (a[i]+a[j]>a[k]&&a[i]+a[k]>a[j]&&a[j]+a[k]>a[i])
                    s=max(s,a[i]+a[j]+a[k]);
    if(s)cout<<s<<endl;
    else cout<<"no"<<endl;
    return 0;
}
```

思考：

是否还有效率更高（循环次数少）的算法？当 n=100000。

6．换钱问题

要将一张 100 元的大钞票，换成等值的 10 元、5 元、2 元、1 元一张的小钞票，每次换成 40 张小钞票，每种至少 1 张。如，有一种换法：

10 元：　1 张
5 元：　5 张
2 元：　31 张
1 元：　3 张

问：一共有多少种换法。

参考代码：

```
#include<cstdio>
#include<iostream>
using namespace std;
int main(){
    int s=0;
    for(int a=1;a<=10;a++)
        for(int b=1;b<=20;b++)
```

```
            for(int c=1;c<=50;c++)
                for(int d=1;d<=100;d++)
                    if(a+b+c+d==40 && 10*a+5*b+2*c+d==100)
                        s=s+1;
    cout<<s<<endl;
    return 0;
}
```

扩展：

问题 1: 10 张 100 元的换成 400 张？

```
#include<cstdio>
#include<iostream>
using namespace std;
int main(){
    int s=0;
    for(int a=1;a<=100;a++)
        for(int b=1;b<=200;b++)
            for(int c=1;c<=500;c++){
                int d=400-a-b-c;
                if(d>0&&10*a+5*b+2*c+d==1000)s=s+1;
            }
    cout<<s<<endl;
    return 0;
}
```

问题 2:

50 张 100 元的 换成 2000 张？

方法 1：减少变量的枚举范围

```
#include<cstdio>
#include<iostream>
using namespace std;
int main(){
    int s=0;
    for(int a=1;a<=500;a++)
        for(int b=1;b<=(5000-10*a)/5;b++)
            for(int c=1;c<=(5000-10*a-5*b)/2;c++){
                int d=2000-a-b-c;
                if(d>0 && 10*a+5*b+2*c+d==5000)s=s+1;
            }
    cout<<s<<endl;
    return 0;
}
```

方法 2：减少枚举的变量

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

```
    int s=0;
    for(int a=1;a<=500;a++)
        for(int b=1;b<=(5000-10*a)/5;b++){
               int c=3000-9*a-4*b;
               int d=8*a+3*b-1000;
               if(c>0&&d>0)s=s+1;
        }
    cout<<s<<endl;
    return 0;
}
```

注意枚举顺序：先枚举 10 元的和先枚举 1 元的哪个效果更好？

7．排列（三个三位数）

http://noi.openjudge.cn/ch0201/8757/

用 1~9 组成 3 个三位数 abc,def,ghi，每个数字恰好使用一次，要求 abc:def:ghi=1:2:3。请按“abc:def:ghi=1:2:3”的格式输出所有的解。

【分析】

只需枚举 abc,范围：123 到 329（ghi 最大 987，然后除以 3=329）。

可求出 def 和 ghi，然后验证即可。

参考代码 1：

```
#include<cstdio>
#include<cstring>
#include<iostream>
using namespace std;
int f[10];
int main(){
    for (int n=111;n<=329;n++){
        memset(f,0,sizeof(f));
        int x=n,y=2*n,z=3*n;
        f[x/100]=1;f[x/10%10]=1;f[x%10]=1;
        f[y/100]=1;f[y/10%10]=1;f[y%10]=1;
        f[z/100]=1;f[z/10%10]=1;f[z%10]=1;
        int s=0;
        for (int i=1;i<=9;i++) s=s+f[i];
        if (s==9) cout<<n<<":"<<2*n<<":"<<3*n<<"=1:2:3"<<endl;
    }
    return 0;
}
```

参考代码 2：

```
#include<cstdio>
#include<cstring>
#include<iostream>
```

```
#include<algorithm>
using namespace std;
char s[11];
int f[11];
int main(){
    for (int n=111;n<=333;n++){
        sprintf(s,"%d%d%d",n,2*n,3*n);
        memset(f,0,sizeof(f));
        for(int i=0;i<=8;i++) f[s[i]-48]=1;
        int sum=0;
        for(int i=1;i<=9;i++) sum+=f[i];
        if (sum==9) cout<<n<<":"<<2*n<<":"<<3*n<<"=1:2:3"<<endl;
    }
    return 0;
}
```

参考代码 3：

```
#include<cstdio>
#include<cstring>
#include<iostream>
#include<algorithm>
using namespace std;
char s[11];
int main(){
    for (int n=111;n<=333;n++){
        sprintf(s,"%d%d%d",n,2*n,3*n);
        sort(s,s+9);
        int i;
        for(i=0;i<=8&&s[i]==49+i;i++);
        if (i==9) cout<<n<<":"<<2*n<<":"<<3*n<<"=1:2:3"<<endl;
    }
    return 0;
}
```

知识点：

判断 a[1]~a[9]正好是 1~9 的一个全排列的方法：

（1）f[1~9]=0,然后 f[a[1]~a[9]]=1 之后，看 f[1~9]的和是否为 9。

（2）把 a[1~9]从小到大排序，判断是否满足 a[i]=i（i：1~9）。

另外，可以利用 springf 把数值转为为字符数组，减少分离每一位数。

8. 除法 uva 725

给你一个数 n(2 <= n <= 79)。

输出所有形如 abcde/fghij=n 的表达式，期中 a~j 是 0~9 的一个排列（可以包含前导 0，如 02345 也算）。0 表示输入结束。

In ase there are no pairs of n umerals satisfying the ondition, y ou m ust write \There are no solutions for N .". Separate the output for t w o dieren t v alues of N b y a blank line.

Sample Input

61

62

0

Sample Output

There are no solutions for 61.

79546 / 01283 = 62

94736 / 01528 = 62

【分析】

只需要枚举 fghij 就可以计算出 abcde。fghij 的范围 01234~49876，计算出 abcde(<=98765)，然后分别求出 abcdefghij，判断是否为 0~9 的一个排列。

一种简单的方法是：

设置一个数组 f[0]~f[9]=0，令 f[a]~f[j]=1，然后求 f[0]+…+f[9]的和是否为 10。

具体实现是可以设置数组 a[10],用 a[0]~a[9]分别表示 a~j，这样容易实现。

另一个方法是把得到的 a[0]~a[9]排序，然后根据 i=0~9,是否满足 a[i]=i?

```
#include<cstdio>
#include<iostream>
#include<cstring>
#define LL long long
using namespace std;
int f[11];
int main(){
    int n,T=0;
    while(cin>>n&&n){
        if (T++) cout<<endl;
        int cnt=0;
        for(int y=1234;y<=49876;y++){
            int x=n*y;
            if (x>98765) break;
            LL z=(LL)x*100000+y;
            memset(f,0,sizeof(f));
            for(int i=0;i<=9;i++){
                f[z%10]=1;
                z=z/10;
            }
            int s=0;
            for(int i=0;i<=9;i++) s+=f[i];
            if(s==10){
                cnt++;
                printf("%05d / %05d = %d\n", x, y, n);
            }
```

```
        }
        if(cnt==0) printf("There are no solutions for %d.\n",n);
    }
    return 0;
}
```

9. 分数拆分 uva 10976

给你一个数 k，求所有使得 1/k = 1/x + 1/y 成立的 x≥y 的整数对。0 < k ≤ 10000。

Sample Input

2

12

sample Output

2

1/2 = 1/6 + 1/3

1/2 = 1/4 + 1/4

8

1/12 = 1/156 + 1/13

1/12 = 1/84 + 1/14

1/12 = 1/60 + 1/15

1/12 = 1/48 + 1/16

1/12 = 1/36 + 1/18

1/12 = 1/30 + 1/20

1/12 = 1/28 + 1/21

1/12 = 1/24 + 1/24

【分析】

关键是如何枚举 x 和 y 的范围，这里不明显。根据要求，首先 y>k；又 x>=y，所有 1/k<=2/y ，即 y<=2k，所以 y 的枚举区间[k+1，2k]，然后求 x，当 1/k - 1/y 的结果分子为 1 即为一组解。

参考代码：

```
cin>>k;
int cnt=0;
for(int i=k+1;i<=2*k;i++){
    if(k*i%(i-k)==0){
        cnt++;
        y[cnt]=i;
        x[cnt]=k*i/(i-k);
    }
}
cout<<cnt<<endl;;
for(int i=1;i<=cnt;i++){
    cout<<1<<"/"<<k<<" = "<<1<<"/"<<x[i]<<" + "<<1<<"/"<<y[i]<<endl;
}
```

10．火柴棒等式[NOIP2008]

【问题描述】

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

【输入】

输入文件 matches.in 共一行，又一个整数 n（n<=24）。

【输出】

输出文件 matches.out 共一行，表示能拼成的不同等式的数目。

【输入输出样例 1】

matches.in	matches.out
14	2

【输入输出样例 1 解释】

2 个等式为 0+1=1 和 1+0=1。

【输入输出样例 2】

matches.in	matches.out
18	9

【输入输出样例 2 解释】

9 个等式为：

0+4=4

0+11=11

1+10=11

2+2=4

2+7=9

4+0=4

7+2=9

10+1=11

11+0=11

分析：

本题最多 24 根火柴，等号和加号共用 4 根火柴，所以 A,B,C 这 3 个数字需用 20 根火柴。我们考查 A 和 B 的最大的取值可能：0~9 这 10 个数字所用的火柴数为 6,2,5,5,4,5,6,3,7,6，很明显数字 1 用的火柴棒最少只要 2 根，不妨让 B 为 1，那么 A 和 C 最多可以使用 18 根火柴，而 C>=A，满足条件的 A 的最大取值为 1111。所以枚举 A 和 B 的范围是从 0~1111，求出 c 然后计算需要的火柴数量是否为 n-4。

为了加快速度，可以将 0 到 2222 的所有整数需要的火柴棒数目提前算好保存在数组中。

参考代码 1：

```
#include<iostream>
#include<cstdio>
using namespace std;
int d[2223]={6,2,5,5,4,5,6,3,7,6};
int n;
int main(){
    for(int i=10;i<=2222;i++){
           d[i]=0;
           int k=i;
           while(k>0){
              d[i]+=d[k%10];
              k/=10;
           }
    }
    cin>>n;
    n=n-4;
    int ans=0;
    for(int a=0;a<=1111;a++)
       for(int b=0;b<=1111;b++){
           int c=a+b;
           if(d[a]+d[b]+d[c]==n)ans++;
       }
    cout<<ans<<endl;
    return 0;
}
```

参考代码 2：

```
#include<iostream>
#include<cstdio>
using namespace std;
int d[2223]={6,2,5,5,4,5,6,3,7,6};
int n;
int main(){
    for(int i=10;i<=2222;i++)d[i]=d[i/10]+d[i%10];
    cin>>n;
    n=n-4;
    int ans=0;
    for(int a=0;a<=1111;a++)
       for(int b=0;b<=1111;b++)
           if(d[a]+d[b]+d[a+b]==n)ans++;
    cout<<ans<<endl;
    return 0;
}
```

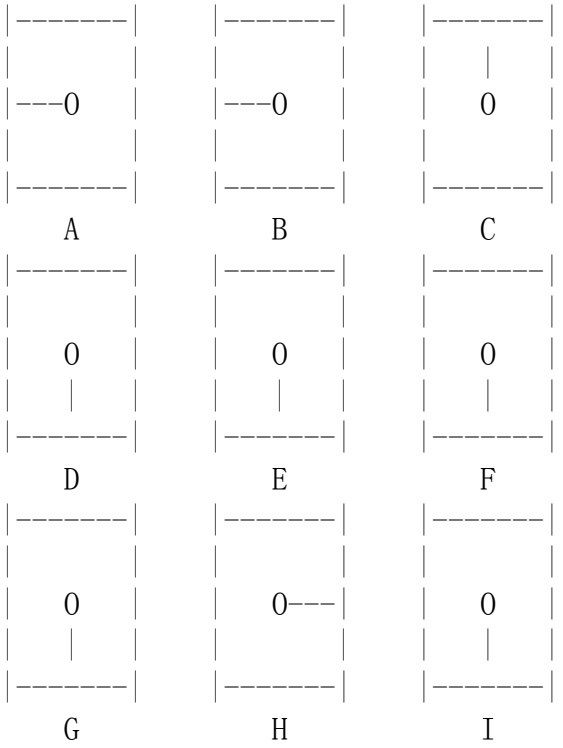
11．拨钟问题（POJ1166）

http://noi.openjudge.cn/ch0201/1816/

描述

有 9 个时钟，排成一个 3*3 的矩阵。

现在需要用最少的移动，将 9 个时钟的指针都拨到 12 点的位置。共允许有 9 种不同的移动。如下表所示，每个移动会将若干个时钟的指针沿顺时针方向拨动 90 度。



移动	影响的时钟
1	ABDE
2	ABC
3	BCEF
4	ADG
5	BDEFH
6	CFI
7	DEGH
8	GHI
9	EFHI

输入

9 个整数，表示各时钟指针的起始位置，相邻两个整数之间用单个空格隔开。其中，0=12 点、1=3 点、2=6 点、3=9 点。

输出

输出一个最短的移动序列，使得 9 个时钟的指针都指向 12 点。按照移动的序号从小到大输出结果。相邻两个整数之间用单个空格隔开。

样例输入 1

3 3 0
2 2 2
2 1 2
样例输出 1
4 5 8 9
样例输入 2
1 1 1
2 2 2
3 3 3
样例输出 2：
1 1 2 3 3 5 5 7 7 8 8 8 9 9
分析：
每个时钟最多拨动 3 次，第 4 次就回到初始位置了;所以每次移动次数范围：0,1,2,3 。一共 9 种移动，依次枚举每种移动需要移动几次。
//x 拨动 i 次的位置是（x+i）%4。

```
#include<iostream>
#include<cstdio>
using namespace std;
int a[10],b[10],c[10],d[10];
int ans=28;
int main(){
    for(int i=1;i<=9;i++)cin>>a[i];
    for(c[1]=0;c[1]<=3;c[1]++)
        for(c[2]=0;c[2]<=3;c[2]++)
            for(c[3]=0;c[3]<=3;c[3]++)
                for(c[4]=0;c[4]<=3;c[4]++)
                    for(c[5]=0;c[5]<=3;c[5]++)
                        for(c[6]=0;c[6]<=3;c[6]++)
                            for(c[7]=0;c[7]<=3;c[7]++)
                                for(c[8]=0;c[8]<=3;c[8]++)
                                    for(c[9]=0;c[9]<=3;c[9]++){
                                        b[1]=(a[1]+c[1]+c[2]+c[4])%4;
                                        b[2]=(a[2]+c[1]+c[2]+c[3]+c[5])%4;
                                        b[3]=(a[3]+c[2]+c[3]+c[6])%4;
                                        b[4]=(a[4]+c[1]+c[4]+c[5]+c[7])%4;
                                        b[5]=(a[5]+c[1]+c[3]+c[5]+c[7]+c[9])%4;
                                        b[6]=(a[6]+c[3]+c[5]+c[6]+c[9])%4;
                                        b[7]=(a[7]+c[4]+c[7]+c[8])%4;
                                        b[8]=(a[8]+c[5]+c[7]+c[8]+c[9])%4;
                                        b[9]=(a[9]+c[6]+c[8]+c[9])%4;
                                        int s=0;
                                        for(int i=1;i<=9;i++)s+=b[i];
                                        if(s==0){
                                            int sum=0;
                                            for(int i=1;i<=9;i++)sum+=c[i];
```

```
                                    if(sum<ans){
                                        ans=sum;
                                        for(int i=1;i<=9;i++)d[i]=c[i];
                                    }

                                }
                            }
    for(int i=1;i<=9;i++)
        while(d[i]--)cout<<i<<" ";
    return 0;
}
```

12. 四大湖问题

上地理课时，四个学生回答我国四个淡水湖大小时说：
A 学生：鄱阳湖第 3,洞庭湖第 1，洪泽湖第 4.
B 学生：洞庭湖第 4，洪泽湖第 1，鄱阳湖第 2，太湖第 3
C 学生：洪泽湖第 4，洞庭湖第 3
D 学生：太湖第 4，鄱阳湖第 1，洪泽湖第 2，洞庭第 3
对于湖的大小，每个学生仅答对一个，请编程判断四个湖的大小。
依次输出：鄱阳湖，洞庭湖，洪泽湖，太湖的名次（1,2,3,4 表示）。
分析：
如果用 abcd 分别表示四个湖的名次，那么：
怎样描述每个同学答对 1 个；
怎样描述 abcd 互不相同，正好是 1234 中的一个。

```
#include<cstdio>
#include<iostream>
using namespace std;
int f[5];
int main(){
    for(int a=1;a<=4;a++)
        for(int b=1;b<=4;b++)
            for(int c=1;c<=4;c++)
                for(int d=1;d<=4;d++){
                    int a1=(a==3)+(b==1)+(c==4);
                    int a2=(b==4)+(c==1)+(a==2)+(d==3);
                    int a3=(c==4)+(b==3);
                    int a4=(d==4)+(a==1)+(c==2)+(b==3);
                    f[1]=f[2]=f[3]=f[4]=0;
                    f[a]=f[b]=f[c]=f[d]=1;
                    if(f[1]+f[2]+f[3]+f[4]==4&&a1*a2*a3*a4==1)
                        cout<<a<<b<<c<<d<<endl;
                }
    return 0;
```

```
}
```