穷举、复杂穷举

1. **背包问题**(knapsack.cpp)

【问题描述】

小明有一个最多只能装 10 斤重的网袋，现有白菜 5 斤、猪肉 2 斤、鱼 3.5 斤，酱油连瓶重 1.7 斤、白糖 1 斤、土豆 5.1 斤。设计一个程序，使小明的网袋里装的物品的总重最大。

【参考程序一】{普通版}

```
#include <cstdio>
#include <iostream>
using namespace std;

int main()
{
   float w[7], maxv=0, weight;
   int baicai, zhurou, yu, jiangyou, baitang, tudou;

   w[1] = 5; w[2] = 2;  w[3] = 3.5;
   w[4] = 1.7; w[5] = 1; w[6] = 5.1;

   for (baicai=0; baicai<=1; baicai++)
     for(zhurou=0; zhurou<=1; zhurou++)
       for(yu=0; yu<=1; yu++)
         for(jiangyou=0; jiangyou<=1; jiangyou++)
            for(baitang=0; baiting<=1; baiting++)
              for(tudou=0; tudou<=1; tudou++)
              {
                weight=baicai*w[1]+zhurou*w[2]+yu*w[3]+\
                       jiangyou*w[4]+baitang*w[5]+tudou*w[6];
                if (weight <= 10 && weight > maxv)
                     maxv = weight;
              }

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

【参考程序二】{推广版}

背包的重量、物品的件数、各物品的重量都是输入的。

```
#include <cstdio>
#include <iostream>
using namespace std;
```

```
int main()
{
   float beibao;
   int n;

   cin >> beibao >> n;
   float w[n+1], sum, maxv=0;
   int b[n+1], i;

   for (i=1; i<=n; i++)  cin >> w[i];   //10，  4，  2，  3，4，5

   for(i=0; i<=n; i++)    b[i] = 0;

   while (b[0] == 0)
   {
      j = n;
      while ( b[j]==1)
      {
         b[j]=0;
         j--;
      }
      b[j] = 1;
      sum = 0;
      for (i=1; i<=n; i++)
         sum += w[i] * b[i];

      if (sum<=beibao && sum>maxv) maxv = sum;
   }

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

1. 最长不下降子序列(LIS.cpp)

【问题描述】

设有一个正整数序列 a[1]、a[2]、…、a[n]，对于下标 i1< i2 <…< ih，若有 a[i1] < a[i2] <…< a[ih]，则称存在一个长度为 h 的不下降序列。

例如，下列数：

13 7 9 16 38 24 27 38 44 49 21 52 63 15

对于下标 i1=1, i2=4, i3=5, i4=9, i5=13, 满足：

13 < 16 < 38 < 44 < 63

则存在长度为 5 的不下降序列。

但是，我们看到还存在着其他的不下降序列，如 7 < 9 < 16 < 18 < 19 < 21 < 22 < 63，则

存在长度为 8 的不下降序列。

【输入】(LIS.in)

输入的第一行是一个整数 N，表示数列中整数的个数。

接下来输入的是这 N 个数。

【输出】(LIS.out)

一个数字，表示最长不下降子序列的长度。

【样例输入 1】

```
14
13,7,9,16,38,24,27,38,44,49,21,52,63,15
```

【样例输出 1】

```
10      '7  9  16  24  27  38  44  49  52  63
```

【样例输入 2】

```
11
10,23,18,21,45,61,104,71,83,91,87
```

【样例输出 2】

```
8      '10  18  21  45  61  71  83  91
```

```
while (b[0] == 0)
{
      j = n;
      while ( b[j]==1)
      {
         b[j]=0;
          j--;
      }
      b[j] = 1;

      for (i=1; i<=n; i++)
         c[i]= a[i] * b[i];

   int f=0;
   for(i=1; i<n;i++)
      if(a[i]>a[i+1])
      {
            f=1;
            break;
      }

   if (f==0)
     for(i=1; i<=n; i++)
        cout <<c[i]<<" ";
  }
```

2. **刻度尺问题**（ruler.cpp）

【问题描述】

一根 29cm 长的尺子，只允许在上面刻 7 个刻度，要能用它量出 1~29cm 的各种长度。试问这根尺的刻度应该怎样选择？

【说明】

各刻度（之间）不能通过叠加来获得新的刻度，只能通过相减来得到新的刻度。例如，若有了 1cm 这个刻度，就不能通过 1+1=2（cm）来得到 2cm 这一刻度。相反，若有了刻度 1cm，则能得到 29-1=28(cm)。还有，若有了 1cm 和 3cm 这两个刻度，就能够得到 3-1=2(cm)这一新刻度。

【参考程序】

```
#include <cstdio>
#include <iostream>
using namespace std;

int main()
{
   int a[7+1], b[29+1];
   int a2, a3, a4, a5, a6, a7;

   a[1] = 1; b[1] = 1; b[28] = 1;  b[29] = 1;

   for(a2=2;a2<=22; a2++)   //循环变量可以换用 I、J 等，再将其值存入 a(i)中
   {
      a[2]=a2;
      for(a3=a[2]+1; a3<=23; a3++)
      {
         a[3]=a3;
         for(a4=a[3]+1; a4<=24; a4++)
         {
            a[4] = a4;
            for(a5=a[4]+1; a5<=25; a5++)
            {
               a[5] = a5;
               for (a6=a[5]+1; a6<=26; a6++)
               {
                  a[6]=a6;
                  for (a7=a[6]+1; a7<=27; a7++)
                  {
                     a[7]=a7;
                     for (i=2; i<=27; i++)   b[i] = 0;
                     for (i=2; i<=7; i++)
                     {
                        b[a[i]] = 1;
                        b[29-a[i]] = 1;
                     }
```

```
                    for (i=1; i<=6; i++)
                       for (j=i+1; j<=7; j++)
                          b[a[j] - a[i]] = 1;
                    int s = 0;
                    for (i=1; i<=29; i++)  s += b[i];
                    if (s == 29)
                    {
                       for (i=1; i<=7; i++)  cout <<a[i]<<' ';
                       cout << endl;
                    }
                }
              }
            }
          }
        }
      }

   return 0;
}
```

【运行结果】

1 2 14 18 21 24 27
1 4 10 16 22 24 27

1 3 6 13 20 24 28
1 5 9 16 23 26 28

3. **邮票问题**(stamps.cpp)

【问题描述】

邮局发行一套票面有四种不同值的邮票，如果每封信所贴邮票张数不超过三枚，存在整数 r，使得用不超过三枚的邮票，可以贴出连续的整数 1、2、3、……、r 来，找出这四种面值数，使得 r 最大。

【参考程序】

```
#include <cstdio>
#include <iostream>
using namespace std;

int a[4+1], b[121], c[4+1];
int a2, a3, a4;
void check();

int main()
{
   a[1] = 1;
```

```
   for (a2=2; a2<=4; a2++)
   {
      a[2] = a2;
      for (a3=a[2]+1; a3<=a[2]*3+1; a3++)
      {
         a[3] = a3;
         for(a4=a[3]+1; a4<=a[3]*3+1; a4++)
         {
            a[4] = a4;
            check();
         }
      }
   }

   cout << maxv << endl;
   for(i=1; i<=4; i++)  cout << c[i]<<' ';
   return 0;
}

void check()
{
   int s = 0, i, j, k, L;
   for (i=1; i<=121; i++)   b[i] = 0;

   for (i=0; i<=3; i++)
     for (j=0; j<=3-i; j++)
       for (k=0; k<=3-i-j; k++)
          for (L=0; L<=3-i-j-k; L++)
          {
             s = a[1]*i+a[2]*j+a[3]*k+a[4]*L;
             b[s] = 1;
          }

   i = 2
   while (b[i]==1)  i++;

   i--;
   if (i > maxv)
   {
       maxv = i;
       for (j=1; j<=4; j++)
          c[j] = a[j];
   }
}
```

{24, 1 4 7 8}

4. **环绕数**(round.cpp)

【问题描述】

一个环绕数有如下三个特点：

a) 每个数字指示了它下一个数字的位置（自左向右数，数到末尾后，再绕到最左位往右数）；

b) 组成这个环绕数的数字只轮到一次；

c) 当所有数字都轮过一次后，正好回到第一次开始所取到的那个数字。

例：3162 就是一个环绕数：

- 取该数任一数字作为开始，如取 1；
- 由此数字开始向右数 1 位，轮到了数字 6；
- 由 6 向右数，数到 2 时绕回到 3，再向左数共数 6 位，就轮到了数字 3；
- 由 3 向右数 3 位，便轮到了数字 2；
- 由 2 绕回到 3 再向右数，共数 2 位，于是回到 1。

求：以 3 开头的四位数中，共有几个环绕数，分别为多少？

【参考程序】

```
DIM AS INTEGER B(4), F(4)
DIM AS INTEGER digit2, digit3, digit4

b(1) = 3

FOR digit2 = 1 to 9   '也可以穷举所有以 3 开头的四位数
  FOR digit3 = 1 to 9
    FOR digit4 = 1 to 9
      b(2)=digit2 : b(3) = digit3 : b(4) = digit4
      FOR k = 1 TO 4
          f(k) = 0
      NEXT k
      x = b(1) : y = 1   ' y 是下标
      FOR j = 1 TO 4
          y = (y + x - 1) mod 4 + 1
          x = b(y)
          f(y) = 1
      NEXT j
      s = 1
      FOR k = 1 TO 4
          s = s * f(k)
      NEXT k
      IF s = 1 THEN PRINT 3*1000+digit2*100+digit3*10+digit4
    NEXT
  NEXT
```

NEXT

Sleep : end

3122,3126,3162,3166,3333,3337,3373,3377,3522,3526,3562,3566,3733,
3737,3773,3777,3922,3926,3962,3966
共：20 个；