放开递推思想

——由 Ural 1036 想到的

【关键字】：递推　容斥原理　方程整数解的个数

【前　言】：

做 DP/递推的题目多了，看到一道要求“个数”的题目，会很自然地想到递推。但递推一定是唯一或者最好的方法吗？本文从 Ural 1036(Lucky Tickets)谈起，简述容斥原理的一类应用。

【正　文】：

Ural 1036 题目大意（原文见附录）：给出 n 和 s，求“前 n 位的和＝后 n 位和＝s/2”的 2n 位数的个数。每个数的最高位可以为 0。

这题见得最多的方法是递推，时间复杂度为 O(n*s*k)(k 为进制，本题中取 10)。因为此算法与本文关系不大，故不再赘述，有兴趣的可以参见附录中的程序（感谢 Amber）。

我们来分析一下这个题目(因 s 为奇数时答案为 0，本文假设 s 为偶数)。

题目说明是 2n 位数，而且前 n 位和=后 n 位和=s/2，则这个数的前 n 位可以表示如下：

$$x_1+x_2+x_3+\cdots+x_n=s/2$$

因为要求满足条件的数的个数，由乘法原理知，只要用前 n 位可能的方法数×后 n 位的方法数即可。又因为题目说最高位可以为 0，则前 n 位的方法数＝后 n 位的方法数，我们只需要求任一个即可。本

文中求前 n 位的方法数。

由分析知，我们只需要求得以上方程的非负整数解的个数再平方即可。但怎么求以上方程的非负整数解的个数呢？

应用组合学公式(定理)，我们可以知道：

方程$x_1+x_2+x_3+\cdots+x_k=n$的非负整数解的个数为$C(n+k-1,n)$(*)

证明从略。

既然有了公式，我们是否就可以直接计算答案了呢？答案是否定的。当 n=2, s=20 时，用以上公式算出的结果会比正确答案大，原因在于它忽略了题意：十进制。如果不考虑进制，则每一位都可能取到 0 至 20，但在十进制中，不可能有一位上的数大于 9。

于是我们知道，我们需要求的是*$x_1+x_2+x_3+\cdots+x_n=s/2$ 的非负整数解的个数，其中：$x_i<=9$*

加上这个要求，问题似乎就没有简单的组合公式可用了。

【容斥原理】:

集合 S 中不具有性质 P1,P2,...,Pm 的元素的个数由下式给出(Ai 表示满足 Pi 的元素集合)：

$$|\overline{A1}\cap\overline{A2}\cap\ldots\cap\overline{Am}|=|S|-\Sigma|Ai|+\Sigma|Ai\cap Aj|-\Sigma|Ai\cap Aj\cap Ak|+\ldots+(-1)^m|A1\cap A2\cap\ldots\cap Am|$$

这个有什么用呢？请看：

令性质Pi（1<=i<=n）为x_i>=10。

应用原理，|S|为$x_1+x_2+x_3+\cdots+x_n=s/2$ 的非负整数解的个数。由(*)式知，|S|=*$C(s/2+n-1,s/2)$；*

因为集合A1 由满足性质P1 的所有元素组成，即由所有满足 $X_1>=10$ 的解组成。作变量代换：$y_1=x_1-10, y_2=x_2, \cdots, y_n=x_n$，我们可以知道$|A_1|$与$y_1+y_2+\cdots+y_n=s/2-10$ 的非负整数解的个数相同，即$C(s/2-10+n-1, s/2-10)$

同理，$|A_2|=|A_3|=\cdots=|An|=C(s/2-10+n-1, s/2-10)=|A_1|$；即这种情况出现C(n,1)次。

集合$A_1 \cap A_2$由所有满足$X_1, X_2>=10$ 的解组成。进行变量代换：$z_1=x_1-10, z_2=x_2-10, z_3=x_3, \cdots, z_n=x_n$，可知$|A_1 \cap A_2|$与$z_1+z_2+\cdots+z_n=s/2-10*2$ 的非负整数解的个数相同，即$C(s/2-10*2+n-1, s/2-10*2)$。由容斥原理知：这种情况出现C(n,2)次。

集合$A_1 \cap A_2 \cap A_3$由所有满足$X_1, X_2, X_3>=10$ 的解组成…C(n,3)

……

最后应用容斥原理，即可得到不满足性质Pi的解数，即所求的：$x_1+x_2+x_3+\cdots+x_n=s/2$ 的非负整数解的个数，其中：$x_i<=9$。

最后分析一下复杂度。因为计算容斥原理时，只要出现$|A_1 \cap A_2 \cap A_3 \cdots A_P|=0$，即可确定以后的计算结果均为 0，可以停止计算，我们可以期望只需要进行很少次的计算就会有$|A_1 \cap A_2 \cap A_3 \cdots A_P|=0$，停止计算。事实上，在题目要求的范围内，计算的次数一般不会超过 10。而每次计算组合时是以近乎常数的时间复杂度进行的，所以最后的程序效率是非常高的。经试验大数据，使用此方法的程序效率比AMBER

大牛的程序高。

由此我们得出结论：递推/DP 不是万金油，我们也没有必要强迫自己具有递推/DP 的思维方式。因为很多时候不必使用递推/DP 就可以解决问题，甚至可以解决得更完美。

【鸣谢】（排名不分先后^_^）：

我自己、所有搞 OI 的朋友。

【参考书目】：

《Introductory Combinatorics 3rd Edition》机械工业出版社 译版

附录：

Amber 大牛的 Ural 1036 AC 程序：

```
(*
Same to 1044 without HighP
f(i,S)=sum{f(i,S-k)|0<=k<=9}
*)
program Ural_1036(Input, Output);
const
    MaxSum = 1000;
    MaxLen = 100;
type
    TIndex = Longint;
    THP = record
        Len: TIndex;
        D: array[1..MaxLen] of Shortint;
    end;
    TDP = array[0..MaxSum div 2] of THP;
var
    N, S: TIndex;

procedure Add(A, B: THP; var C: THP);
var
    i: TIndex;
    tmp: THP;
begin
    if A.Len = 0 then A.Len := 1;
    if B.Len = 0 then B.Len := 1;
    if A.Len < B.Len then
    begin
        tmp := A;
        A := B;
        B := tmp;
    end;
    FillChar(C, SizeOf(C), 0);
    C.Len := A.Len + 1;
    for i := 1 to A.Len do
        Inc(C.D[i], A.D[i] + B.D[i]);
    for i := 1 to A.Len do
    begin
        Inc(C.D[i + 1], C.D[i] div 10);
        C.D[i] := C.D[i] mod 10;
    end;
```

```
    while (C.D[C.Len] = 0) and (C.Len > 1) do
        Dec(C.Len);
end;

procedure Multiply(A, B: THP; var C: THP);
var
    i, j: TIndex;
begin
    if A.Len = 0 then A.Len := 1;
    if B.Len = 0 then B.Len := 1;
    FillChar(C, SizeOf(C), 0);
    for i := 1 to A.Len do
        for j := 1 to B.Len do
        begin
            Inc(C.D[i + j - 1], A.D[i] * B.D[j]);
            Inc(C.D[i + j], C.D[i + j - 1] div 10);
            C.D[i + j - 1] := C.D[i + j - 1] mod 10;
        end;
    C.Len := A.Len + B.Len;
    while (C.Len > 1) and (C.D[C.Len] = 0) do
        Dec(C.Len);
end;

procedure Print(A: THP);
var
    i: TIndex;
begin
    for i := A.Len downto 1 do
        Write(A.D[i]);
    Writeln;
end;

procedure Main;
var
    F1, F2: TDP;
    Sum: THP;
    i, j, k: TIndex;
begin
    FillChar(F1, SizeOf(F1), 0);
    Readln(N, S);
    if Odd(S) then
    begin
        Writeln('0');
        Exit;
```

```
    end;
    F1[0].D[1] := 1;
    for i := 1 to N do
    begin
        FillChar(F2, SizeOf(F2), 0);
        for j := 0 to S div 2 do
            for k := 0 to 9 do
                if j - k >= 0 then
                    Add(F2[j], F1[j - k], F2[j]);
        F1 := F2;
    end;
    Multiply(F1[S div 2], F1[S div 2], Sum);
    Print(Sum);
end;
begin
   { Assign(Input, 'i.txt');
    Reset(Input);
    Assign(Output, 'o.txt');
    Rewrite(Output);  }
    Main;
  {  Close(Input);
    Close(Output);  }
end.
```

使用文中方法的程序（时间问题没使用高精度）：

```
var
  n,s   :integer;
function c(a,b:longint):qword;
var
  i,x,y :longint;
begin
  if b>a then exit(0);
  if (a=b)or(a*b=0) then exit(1);
  if b>a-b then
    begin
      x:=b+1;
      y:=a-b;
    end
  else
    begin
      x:=a-b+1;
      y:=b;
```

```
    end;
  c:=1;
  for i:=x to a do c:=c*i;
  for i:=2 to y do c:=c div i;
end;

function calc(k:longint):qword;
var
  tmp,num       :qword;
begin
  tmp:=c(s-k*10+n-1,n-1);
  num:=c(n,k);
  calc:=tmp*num;
end;

procedure main;
var
  i     :longint;
  ans   :qword;
  curr  :int64;
begin
  if odd(s) then
    begin
      writeln(0);
      halt;
    end;

  s:=s shr 1;
  ans:=0;
  i:=0;
  repeat
    curr:=calc(i);
    inc(ans,(-1)**i *curr);
    inc(i);
  until curr=0;
  writeln(ans*ans);
end;
{========main========}
begin
  readln(n,s);
  main;
end.
```

Lucky tickets

Time Limit: 2.0 second
Memory Limit: 1 000 KB

You are given a number 1<=N<=50. Every ticket has its 2N-digit number. We call a ticket lucky, if the sum of its first N digits is equal to the sum of its last N digits. You are also given the sum of ALL digits in the number. Your task is to count an amount of lucky numbers, having the specified sum of ALL digits.

Input

Two space-separated numbers: N and S. Here S is the sum of all digits. Assume that S<=1000.

Output

The amount of lucky tickets.

Sample Input

2 2

Sample Output

4