NOIp2002 普及组解题报告

湖南 黄艺海

题一：级数求和

[问题描述]：

已知：Sn=1+1/2+1/3+…+1/n。显然对于任意一个整数 K，当 n 足够大的时候，Sn 大于 K。现给出一个整数 K（1<=K<=15），要求计算出一个最小的 n，使得 Sn>K

[问题分析]：

这道题目非常简单，题目的意思已经把该题的算法描述得再清楚不过了，初始时 Sn=0,n=0,然后每次循环 n←n+1,Sn←Sn+1/n,，直到 Sn 大于 K，最后输出 K。另外实型(Real 是最慢的，建议用 Extended)的运算速度不是很快，而 K 为 1~15 之间的整数，所以最后可以交一张表（常量数组），以达到最好的效果

题二：选数

[问题描述]：

已知 n （1<=n<=20）个整数 x1,x2,…,xn （1<=xi<=5000000），以及一个整数 k（k<n）。从 n 个整数中任选 k 个整数相加，可分别得到一系列的和。现在，要求你计算出和为素数共有多少种。

[问题分析]：

本题动态规划无从下手，也无数学公式可寻，看来只能搜索（组合的生成算法），其实 1<=n<=20 这个约束条件也暗示我们本题搜索是有希望的，组合的生成可用简单的 DFS 来实现，既搜索这 k 个整数在原数列中的位置，由于组合不同于排列，与这 k 个数的排列顺序无关，所以我们可以令 **a[I]<a[I+1]**（a[I]表示第 I 个数在原数列中的位置），这个组合生成算法的复杂度大约为 **C(n,k)**，下面给出递归搜索算法的框架：

```
Proc  Search(dep)
Begin
   for i <- a[dep - 1] + 1 to N - (M - dep) do
      1：a[dep] <- i
      2：S <- S + x[i]
      3：if dep < k then Search(dep + 1)  else 判断素数
      4：S <- S - x[i]
End
```

接下来的问题就是判断素数，**判断一个整数 P(P>1)是否为素数最简单的方法就是看是否存在一个素数 a(a<=sqrt(P))是 P 的约数**，如果不存在，该数就为素数，由于在此题中 1<=xi<=5000000,n<=20,所以要判断的数 P 不会超过 100000000，sqrt(p)<=10000，因此，为了加快速度，我们可以用筛选法将 2…10000 之间的素数保存到一个数组里（共 1229 个），这样速度估计将提高 5~6 倍。

特别注意：本题是要求使和为素数的情况有多少种，并不是求有多少种素数，比赛时就有很多同学胡乱判重而丢了 12 分；还有 1 不是素数，在判素数时要对 1 做特殊处理。

题三：产生数

[问题描述]：

给出一个整数 n(n<10^30)和 k 个变换规则（k<=15）。

规则：

1 个数字可以变换成另一个数字

规则的右部不能为零。

问题：

给出一个整数 n 和 k 个规则

求出：

经过任意次的变换（0 次或多次），能产生出多少个不同的整数。

[问题分析]：

认真分析题目之后发现，本题搜索显然是不行的，而且对于只需计数而不需求具体方案的题目，一般都不会用搜索解决，其实本题不难看出，可以用乘法原理直接进行计数用 Fi 表示数字 i 包括本身可以变成的数字总个数（这里的变成可以是直接变成也可以是间接变成，比如 3->5,5->7,那么 3->7），那么对于一个数 a（用数组存,长度为 n），根据乘法原理它能产生出 **F[a[1]]*F[a[2]]*F[a[3]]*…F[a[n]]**个不同整数，相信这一点大家不难理解。那么现在的关键就是如何求 Fi，由于这些变换规则都是反应的数字与数字之间的关系，这很容易让我们想到用图来表示这种关系：

1： 建立一个有向图 G，初始化 g[i, j] ← False

2： 如果数字 i 能直接变成数字 j，那么 g[i, j] ← True

容易知如果数字 i 能变成数字 j，那么 i 到 j 必须存在路径，否则 i 是不可能变成 j 的，这样一来，Fi 的求解就显得非常简单了，求一个顶点 v 包括本身能到达的顶点数的方法相当多，可以用 BFS,DFS,Dijkstra,Floyd，这里介绍一种类似 Floyd 的有向图的传递闭包算法，该算法实现简单 ，在解决这类问题时比 Floyd 效率更高，所谓有向图的传递闭包就是指可达性矩阵 A=[a[i, j]],其中

a[i, j] = True 从 i 到 j 存在通路

a[i, j] = False 从 i 到 j 不存在通路

所以有向图传递闭包算法只需将 floyd 算法中的算术运算符操作'+'用相应的逻辑运算符'and'和'or'代替就可以了，其算法如下：

```
for k ← 1 to n do
  for i ← 1 to n do
  for j ← 1 to n do
    a[i, j] = a[i, j] or (a[i, k] and a[k, j])
```

最后值得注意的是当 n 很大时输出可能会超过 Comp 类型的范围，所以要使用高精度乘法，由于高精度算法是信息学竞赛中的基础，这里就不在详述。

[问题描述]：

棋盘上 A 点有一个过河卒，需要走到目标 B 点。卒行走的规则：可以向下、或者向右。

同时在棋盘上 C 点有一个对方的马，该马所在的点和所有跳跃一步可达的点称为对方马的控制点。

棋盘用坐标表示，A 点(0,0)、B 点(n, m) (n,m 为不超过 20 的整数),同样马的位置坐标是需要给出的。现在要求你计算出卒从 A 点能够到达 B 点的路径的条数

[问题分析]：

这是一道老得不能再老的题目了，很多书上都有类似的题目，NOIp97 普及组的最后一题就和本题几乎一模一样。有些同学由于没见过与之类似的题目，在比赛时用了搜索，当 n 到 14，15 左右就会超时，其实，本题稍加分析，就能发现：要到达棋盘上的一个点，只能从左边过来或是从上面下来，所以根据加法原理，到达某一点的路径数目，等于到达其相邻上，左两点的路径数目之和，因此我们可以使用逐列（或逐行）递推的方法来求出从起始顶点到重点的路径数目，即使有障碍（我们将马的控制点称为障碍），这一方法也完全适用，只要将到达该点的路径数目置为 0 即可，用 F[i,j]表示到达点(i,j)的路径数目，g[i,j]表示点(i, j)有无障碍，递推方程如下：

$$
\begin{aligned}
&F[0,0] = 1\\
&F[i,j] = 0 \qquad \{ g[x,y] = 1 \}\\
&F[i,0] = F[i-1,0] \qquad \{i > 0, g[x,y] = 0\}\\
&F[0,j] = F[0,j-1] \qquad \{j > 0, g[x,y] = 0\}\\
&F[i,j] = F[i-1,j] + F[i,j-1] \ \{i > 0, j > 0, g[x, y] = 0\}
\end{aligned}
$$

本题与第三题一样，也要考虑精度问题，当 n,m 都很大时，可能会超过 MaxLongInt,所以要使用 Comp 类型计数(Comp 类型已经足够了，即使 n=20,m=20，没有任何障碍的情况下的结果也只有 14，5 位的样子)。

总结:

四道题目其实都很容易,要想到正确可行的方法并不难,考察的是大家的编程基础,一些基本算法的简单应用,并不需要什么优化技巧,关键是看大家对这些基本算法是否已熟练掌握,只有熟练掌握这些算法,在考试中才能在较短的时间内做好每道题,我们一定要重视基础!