

使程序效率火箭般提速的数学构造(起步版)

V 1.0pascal

2009-4

学习过一些基本算法的同学都应该知道,在分析一个问题的算法时,我们除了算法的正确性应当考虑以外,另一个问题是此算法的时间与空间复杂度,而这个往往是区分一个算法整体质量,甚至在比赛中能否取得理想成绩的关键,这里我们重点研究一下时效优化.

算法执行的时间复杂度一般取决于算法整体的执行次数,状态数目当然还受数据结构的直接影响.我们忽略数据结构等问题,通过一些 USACO 的简单基本题目来说明一下执行次数,以及状态数的优化.

这就不得不说数学构造,它往往能使问题化繁为简,必要时使用数学构造可以令问题复杂度大大降低,避免重复运算,还可压缩状态.同时,使用数学构造的程序通常精巧短小,而且还好想,那么我们例举几个例子具体讨论这个问题:

一、离散化构造连续数列法

例1: 农民的牛奶

三个农民每天清晨5点起床,然后去牛棚给3头牛挤奶.第一个农民在300时刻(从5点开始计时,秒为单位)给他的牛挤奶,一直到1000时刻.第二个农民在700时刻开始,在1200时刻结束.第三个农民在1500时刻开始2100时刻结束.期间最长的至少有一个农民在挤奶的连续时间为900秒(从300时刻到1200时刻),而最长的无人挤奶的连续时间(从挤奶开始一直到挤奶结束)为300时刻(从1200时刻到1500时刻).你的任务是编一个程序,读入一个有N个农民($1 \leq N \leq 5000$)挤N头牛的工作时间列表,计算以下两点(均以秒为单位):

最长至少有一人在挤奶的时间段。

最长的无人挤奶的时间段。

SAMPLE INPUT

```
3
300 1000
700 1200
1500 2100
```

SAMPLE OUTPUT

```
900 300
```

题解:

如果直接模拟复杂度是很大的,所以离散化:按照开始时间升序排序,然后从左到右扫一遍,记录一个当前区间,如果下一个区间是当前区间的子区间,跳过.如果下一个区间和当前区间相交,就合并两个区间放入当前区间,否则就将新区间作为当前区间,然后继续往右扫.(同时记录无人挤奶时间)

参考主程序:

```
procedure main;//排序部分略
var i:longint;
    you,no:longint;
```


目标要找一个最小的移动顺序次将所有的指针指向 12 点。下面原表格列出了 9 种不同的旋转指针的方法，每一种方法都叫一次移动。选择 1 到 9 号移动方法，将会使在表格中对应的时钟的指针顺时针旋转 90 度。

移动方法 受影响的时钟

| | |
|---|-------|
| 1 | ABDE |
| 2 | ABC |
| 3 | BCEF |
| 4 | ADG |
| 5 | BDEFH |
| 6 | CFI |
| 7 | DEGH |
| 8 | GHI |
| 9 | EFHI |

Example

```
9 9 12          9 12 12          9 12 12          12 12 12          12 12 12
6 6 6  5 ->    9  9  9  8->    9  9  9  4 ->    12  9  9  9-> 12 12 12
6 3 6          6  6  6          9  9  9          12  9  9          12 12
12
```

[但这可能不是正确的方法，请看下面]

格式

PROGRAM NAME: clocks

INPUT FORMAT:

第 1-3 行：三个空格分开的数字，每个数字表示一个时钟的初始时间，3, 6, 9, 12。数字的含意和上面第一个例子一样。

OUTPUT FORMAT:

单独的一行包括一个用空格分开的将所有指针指向 12:00 的最短移动顺序的列表。

如果有多种方案，输出那种使的连接起来数字最小的方案。（举例来说 5 2 4 6 < 9 3 1 1）。

SAMPLE INPUT

```
9 9 12
6 6 6
```

SAMPLE OUTPUT

4 5 8 9

题解：这个数学方法是大家惯用的一种，在这里我做简要说明，对于第一只表，若只把它转90度，则要做的操作是111, 222, 333, 444, 555, 66, 777, 88也就构成了数列 (3, 3, 3, 3, 3, 2, 3, 2, 0)，也就是说执行本操作后只有第一只表转了90度，同理直接构造出其他表状态的矩阵：

```
change:array[1..9,1..9]of integer=
((3, 3, 3, 3, 3, 2, 3, 2, 0),
(2, 3, 2, 3, 2, 3, 1, 0, 1),
(3, 3, 3, 2, 3, 3, 0, 2, 3),
(2, 3, 1, 3, 2, 0, 2, 3, 1),
(2, 3, 2, 3, 1, 3, 2, 3, 2),
(1, 3, 2, 0, 2, 3, 1, 3, 2),
(3, 2, 0, 3, 3, 2, 3, 3, 3),
(1, 0, 1, 3, 2, 3, 2, 3, 2),
(0, 2, 3, 2, 3, 3, 3, 3, 3));
```

因为任何一只表转过4次等于没转，所以所有操作执行后相加mod 4即可。结果本身就是最小的。

(程序略)

三、状态压缩数学构造

例3：特殊的质数肋骨

农民约翰的母牛总是生产出最好的肋骨。你能通过农民约翰和美国农业部标记在每根肋骨上的数字认出它们。

农民约翰确定他卖给买方的是真正的质数肋骨，是因为从右边开始切下肋骨，每次还剩下的肋骨上的数字都组成一个质数，举例来说：

7 3 3 1

全部肋骨上的数字 7331 是质数；三根肋骨 733 是质数；二根肋骨 73 是质数；当然，最后一根肋骨 7 也是质数。

7331 被叫做长度 4 的特殊质数。

写一个程序对给定的肋骨的数目 N ($1 \leq N \leq 8$)，求出所有的特殊质数。数字 1 不被看作一个质数。

格式

PROGRAM NAME: sprime

INPUT FORMAT:

单独的一行包含 N。

OUTPUT FORMAT:

按顺序输出长度为 N 的特殊质数, 每行一个。

SAMPLE INPUT

4

SAMPLE OUTPUT

2333

2339

2393

2399

2939

3119

3137

3733

3739

3793

3797

5939

7193

7331

7333

7393

题解:

本题使用DFS, 通过数学知识我们能知道, 从首位开始构造, 1, 4, 6, 8, 9均不能成立, 仅剩下2, 3, 5, 7;

同理间位 (除首末两位后, 中间部分) 仅剩1, 3, 7, 9;

证明: 2, 4, 6, 8为2的倍数, 接在任何数后面均可被2整除, 同理5也被排除; 在这里需要考虑的是7接在3后面为37是质数, 而接在7后为77则不是, 编程时应注意这点;

末尾则可由1, 3, 7, 9组成 (证略)。

下面是程序:

```
program sprime;
```

```
var n:integer;
```

```
sw:array[1..4]of integer=(2, 3, 5, 7);
```

```

jw:array[1..4]of integer=(1,3,7,9);
mw:array[1..4]of integer=(1,3,7,9);
function pd(a:longint):boolean;
var j:cardinal;
begin
  for j:=2 to trunc(sqrt(a)) do
    if (a mod j) =0 then
      exit(false);
    exit(true);
  end;
function cf(i:integer):longint;//10的i次方运算
var j:integer;
begin
  cf:=1;
  for j:=1 to i-1 do
    cf:=cf*10;
  end;
procedure main(wz,t:longint);
var i,i2,i3,t1:longint;
begin
  case wz of
  1:for i:=1 to 4 do
    begin
      t:=sw[i]*cf(n);
      if (wz+1)<=n then begin main(wz+1,t);end
      else writeln(t);
    end;
  else
    if wz=n then
      begin
        t1:=t;
        for i2:=1 to 4 do
          begin
            t1:=t1+mw[i2];
            if pd(t1) then writeln(t1);//检验,输出
            t1:=t;
          end
        end
      else
        begin
          t1:=t;
          for i3:=1 to 4 do
            begin
              t1:=t1+jw[i3]*cf(n-wz+1);

```


所求的等差数列将不会多于 10,000 个。

SAMPLE INPUT

5
7

SAMPLE OUTPUT

1 4
37 4
2 8
29 8
1 12
5 12
13 12
17 12
5 20
2 24

题解：

这道题就是暴力搜索，时限是 5s，方法是很简单的：枚举所有的可能解，没有剪枝的。

但是应注意一些细节，而此题主要时间浪费在判断上，预处理，计算出所有的可能平方数，用 boolean 数组记录，这样在检验时，时间复杂度为 $O(1)$ ；同时数列首位及公差也可以进行构造，请参考程序：

预处理：

```
for j:=0 to m do
  for k:=0 to j do
    IF p[j*j+k*k]<>true then
      begin
        inc(t);
        q2[t]:=j*j+k*k;
        p[q2[t]]:=true;
      end; {先把数字构造出来,再用 boolean 数组记录,这样在判断时,时间复杂度将为
0(1)}
```

接着把平方数排序

主程序:

```
procedure main;
var i, il, j, max, gongcha:cardinal;
    pd2:boolean;
begin
    total:=0;
    max:=2*(m*m);
    for i:=1 to t2 do
    begin
        gongcha:=(max-q2[i]) div (n-1); //构造公差
        for il:=1 to gongcha do
        begin
            pd2:=true;
            for j:=n-1 downto 1 do
                if not p[q2[i]+il*j] then begin pd2:=false;break;end; {这个部分我原来是用函数编写的, 虽然方便但会超时, 所以在编程中应慎用函数, 越是整合的程序越省时间}
            if pd2 then
            begin
                inc(total);
                a[total]:=q2[i];
                b[total]:=il;
            end;
        end;
    end;
end;
```

后计:

此文仅为起步版, 献给OI道路上的初学者, 由于作者水平有限, 如有遗漏, 错误等, 请指正, 联系邮箱 (acmnoi@163.com) ——实验中学OI队