搜索专题

一、问题引入

1.简明的数学模型揭示问题本质。对于这一类试题，我们尽量用解析法求解。

2.对给定的问题建立数学模型，或即使有一定的数学模型，但采用数学方法解决有一定困难。对于这一类试题，我们只好用模拟或搜索求解。

二、搜索的本质

搜索的本质就是逐步试探，在试探过程中找到问题的解。

三、搜索问题考察的范围

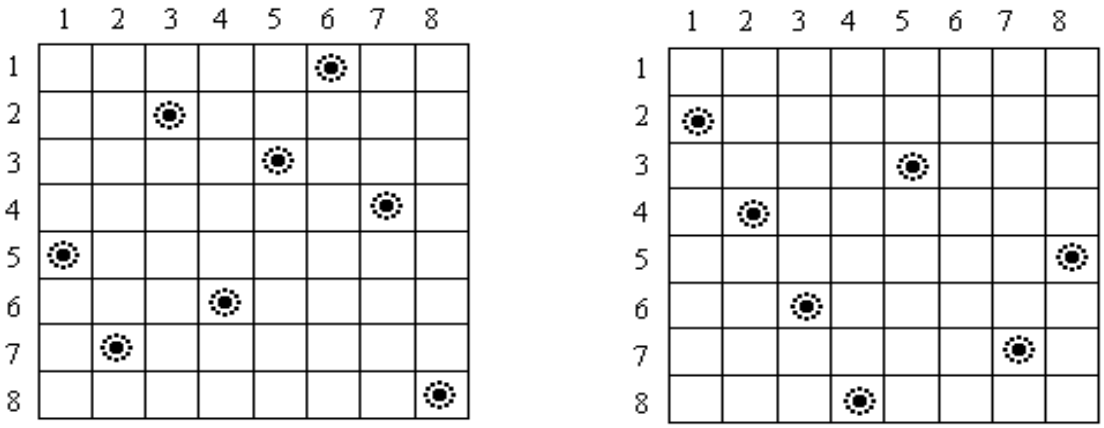
1.算法的实现能力

2.优化算法的能力

四、典型试题分析

【例 1】N 皇后问题

在 N*N 的棋盘上放置 N 个皇后而彼此不受攻击（即在棋盘的任一行，任一列和任一对角线上不能放置 2 个皇后），编程求解所有的摆放方法。



八皇后的两组解

分析：

由于皇后的摆放位置不能通过某种公式来确定，因此对于每个皇后的摆放位置都要进行试探和纠正，这就是“回溯”的思想。

在 N 个皇后未放置完成前，摆放第 i 个皇后和第 i+1 个皇后的试探方法是相同的，因此完全可以采用递归的方法来处理

```
Procedure Try(I:integer);
{搜索第 I 行皇后的位置}
var
```

```
    j:integer;
begin
    if I=n+1 then 输出方案；
    for j:=1 to n do
        if 皇后能放在第 I 行第 J 列的位置 then begin
                放置第 I 个皇后；
               对放置皇后的位置进行标记；
                Try（I+1）
                对放置皇后的位置释放标记；
       end;
end;
```

怎样判断某列放置了皇后

```
    A:array [1..MaxN] of Boolean;
     {竖线被控制标记}
```

怎样判断某对角线上放置了皇后

```
    B:array [2..MaxN * 2] of Boolean;
     {左上到右下斜线被控制标记}
    C:array [1 - MaxN..MaxN - 1] of Boolean;
     {左下到右上斜线被控制标记}
```

【例 2】背包问题

已知一个容量大小为 M 重量的背包和 N 种物品，每种物品的重量为 Wi。若将物品放入背包将得到 Pi 的效益，求怎样选取物品将得到效益最大

分析：

本题可以用递归求解：设当前有 N 个物品，容量为 M；因为这些物品要么选，要么不选，我们假设选的第一个物品编号为 I（1~I-1 号物品不选），问题又可以转化为有 N-I 个物品（即第 I+1~N 号物品），容量为 M-Wi 的子问题……如此反复下去，然后在所有可行解中选一个效益最大的便可。

另外，为了优化程序，我们定义一个函数如下：

F[I]表示选第 I~N 个物品能得到的总效益。不难推出：

F[N]=Pn

F[I]=F[I+1]+Pi　　（I=1…N-1）

假设当前已选到第 I 号物品，如果当前搜索的效益值+F[I+1]的值仍然比当前的最优值小，则没有必要继续搜索下去。

算法框架：

```
Procedure Search(I:Integer; J:Byte);      {递归求解}
Var K                              :Byte;
Begin
  If Now+F[J]<=Ans Then Exit;       {如果没有必要搜索下去}
  If Now>Ans Then Begin         {修改最优解}
    Ans:=Now;
    Out:=Ou;
  End;
  For K:=J To N Do                  {选取物品}
```

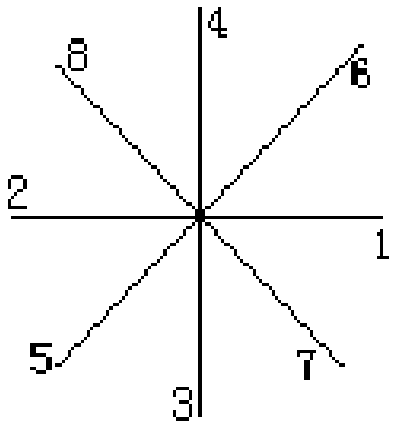
```
    If W[K]<=I Then Begin
      Now:=Now+P[K];
      Ou[K]:=True;
      Search(I-W[K],K+1);
      Now:=Now-P[K];
      Ou[K]:=False;
    End;
End;
```

【例 3】找国都名

给出一个矩阵及一些国都名：

```
o k d u b l i n        dublin
a l p g o c e v        tokyo
r a s m u s m b        london
o s l o n d o n        rome
y i b l g l r c        bonn
k r z u r i c h        paris
o a i b x m u z        oslo
t p q g l a m v        lima
```

要求从这个矩阵中找出这些国都名，并输出它们的起始位置及方向



搜索的方向

算法分析：

将字符矩阵读入到二维数组，然后对每一个国都名进行搜索，首先需要在矩阵中找到国都名的第一个字符，然后沿八个方向进行搜索。直到找到国都名为止。若在矩阵中没有找到，则输出相应的信息。

在搜索过程时，类似八皇后问题，建立一个标志数组，标识已经搜索过的方向，在对八个方向搜索时，可以建立一个方向数组，使得程序更加简洁明了

```
Const
    Fx : Array[1..8,1..2] Of Shortint       {定义八个方向}
      =((0,1),(0,-1),(1,0),(-1,0),(1,-1),(-1,1),(1,1),(-1,-1));
```

```
Procedure Work(T,X,Y:Integer);
{搜索路径，T 为国都名的字符位置，X，Y 为当前搜索的坐标}
Var I : Integer;
Begin
  If T=Length(S)+1 Then begin  {搜索完，打印输出}
     Out;  exit  end;
  For I:=1 To 8 Do    {八个方向进行搜索}
    Begin
      X:=X+Fx[I,1]; Y:=Y+Fx[I,2]; {坐标变化}
      If (A[X,Y]=S[T])And(B[X,Y]) Then    Begin
           W:=W+Chr(I+48);  {记录路径}
           B[X,Y]:=False;    {设置已经搜索}
           Work(T+1,X,Y);    {继续搜索下一个}
           Delete(W,Length(W),1);{恢复原路径}
           B[X,Y]:=True;      {恢复标志}
        End;
      X:=X-Fx[I,1]; Y:=Y-Fx[I,2]; {返回后，坐标恢复}
    End;
End;
```

【例 4】彩票问题

已知：彩票上的数字：1,2,…,M

彩民的选择：A1,A2,…,An，其中 Ai 属于 1,2,…,M

每人只能买一张彩票，每人彩票选择都不同

抽出两个自然数 X 和 Y。

如果 1/A1+2/A2+…+1/An= X/Y，则中奖（获取纪念品）。

输入：

N，M，X，Y

输出：

所需准备的纪念品数量

1≤X, Y≤100，1≤N≤10，1≤M≤50。

输入数据保证输出结果不超过10^5。

分析：对于每个数，有选和不选两种可能性，显然可以建立如下模型：

x1/1 + x2/2 + x3/3 + … + xm/m = X/Y

其中，xi=0 或者 1(1<=i<=m)

x1+x2+x3+…+xm=n

逐个搜索 xi

O(2m)

x1/1 + x2/2 + x3/3 + … + xm/m = X/Y

同时乘以 m!*Y 通分。

令 Ti=m!*Y/i (1<=i<=m), T0=m!*X

则：

T1x1+T2x2+T3x3+…+Tmxm=T0

这就变成了一个 01 背包问题。
每个包裹的体积是 Ti，箱子体积 T0。从 M 个中选 N 个，填满箱子。求方案数。
T1x1+T2x2+T3x3+…+Tmxm=T0
如何剪枝？
f[i, T]表示为了满足 T1x1+T2x2+…+Tmxm=T，最少要让多少个 xi 取 1。
f[i, T]=min{f[i-1,T], f[i-1,T-Ti]+1}
按照 xm, xm-1, xm-2, …, x1 的顺序搜索。
假设 xp~xm 都已经取定，令 S=Tpxp+Tp+1xp+1+…+Tmxm，L=xp+xp+1+…+xm，如果 f[p-1, T-S]+L>N，那么就可以回溯，不必继续搜索了。
T~O(m!)。太大了！
f 数组开不下，时间上也不允许。
T1x1+T2x2+T3x3+…+Tmxm=T0
f[i, T]表示为了满足 T1x1+T2x2+…+Tmxm=T，至少要让多少个 xi 取 1。
f[i, T]=min{f[i-1,T], f[i-1,T-Ti]+1}
动态规划的思想，空间矛盾太大
抓住矛盾：解决空间问题！
T1x1+T2x2+T3x3+…+Tmxm=T0
f[i, T]表示为了满足(T1x1+T2x2+…+TmXm)mod P=T，至少要让多少个 xi 取 1。
f[i, T]=min{f[i-1,T], f[i-1,(T-Ti)mod P]+1}
0<=T<P
按照 xm, xm-1, xm-2, …, x1 的顺序搜索。
假设 xp~xm 都已经取定，令 S=Tpxp+Tp+1xp+1+…+Tmxm，L=xp+xp+1+…+xm，如果 f[p-1, (T-S)mod P]+L>N，那么就可以回溯，不必继续搜索了。剪枝效果有所削弱，但是空间复杂度降到了 O(mP)，这里 P 可以任取。

【例 5】邮票面值设计

给定一个信封，最多只允许粘贴 N 张邮票，计算在给定 K（N+k<=40) 种邮票的情况下（假定所有的邮票数量都足够），如何设计邮票的面值，能得到最大 max ，使得 1－max 之间的每一个邮资值都能得到。

例如，N=3，K＝2，如果面值分别为 1 分、4 分，则在 1 分-6 分之间的每一个邮资值都能得到（当然还有 8 分、9 分和 12 分）：如果面值分别为 1 分、3 分，则在 1 分-7 分之间的每一个邮资值都能得到。可以验证当 N=3，K＝2 时，7 分就是可以得到连续的邮资最大值，所以 MAX=7，面值分别为 1 分、3 分。

样例：

INPUT

N=3 k=2

OUTPUT

1　3

MAX=7

基本算法：

搜索的过程实质上是枚举 K 种邮票的面值，然后计算这 K 种邮票对应的 MAX 值。于是可以得到如下算法：

Procedure Search(m) 搜索第 m 种邮票的面值

1.　If m = K+1 Then [

```
2.      If 当前方案更优 Then 保存当前方案;
3.      Exit;
4.    ]
5.    For I⇓ Am-1+1 to N＊Am-1+1 do [
          {N*Am-1+1 肯定不能有前 K 中邮票构成}
6.      Am := I;
7.      Search(m+1);
8.    ]
```

优化（算法 2）

显然一定有面值为 1 的邮票。可以按递增的次序来搜索各种邮票的面值，关键在于确定每一层搜索的上界。设当前搜索到第 m 层，用不超过 N-1 张前 m-1 种邮票可以得到的 MAX 值记为 MAX(m-1)，则第 m 种邮票面值的上界是 MAX(m-1)+1，否则邮资值 MAX(m-1)+1 就无法用这 m 种邮票贴出来了。

Procedure Search(m) 搜索第 m 种邮票的面值

```
1.    If m = K+1 Then [
2.      If 当前方案更优 Then 保存当前方案;
3.      Exit;
4.    ]
5.    For I⇓ Am-1+1 to Max(M-1) do [
6.      Am := I;
7.      Search(m+1);
8.    ]
```

【例 6】埃及分数

在古埃及，人们使用单位分数的和(形如 1/a 的, a 是自然数)表示一切有理数。如：2/3=1/2+1/6,但不允许 2/3=1/3+1/3,因为加数中有相同的。

对于一个分数 a/b,表示方法有很多种，但是哪种最好呢？首先，加数少的比加数多的好，其次，加数个数相同的，最小的分数越大越好。

如：

19/45=1/3 + 1/12 + 1/180

19/45=1/3 + 1/15 + 1/45

19/45=1/3 + 1/18 + 1/30,

19/45=1/4 + 1/6 + 1/180

19/45=1/5 + 1/6 + 1/18.

最好的是最后一种，因为 1/18 比 1/180,1/45,1/30,1/180 都大。

给出 a,b(0<a<b<1000),编程计算最好的表达方式。

输入：a b

输出：若干个数，自小到大排列，依次是单位分数的分母。

例如：

输入：19 45

输出：5 6 18

分析

1.节点类型

是一个 K 元组（a1,a2,...ak），代表当前解中的分母 a1,a2..ak.

2.节点扩展方法

按照 a1<a2<a3..<ak 的顺序扩展,扩展第 k 层节点的时候，最简单的办法就是从 a[k-1]+1 开始枚举 a[k],一直到预先确定的最大值。

但是这个最大值怎么确定呢？直观的讲，a[k]总不能太大，因为如果 a[k]太大，1/a[k]就很小，1/a[k+1] .. 1/a[k+2].. 就更小，那么，尽管加了很多项，还可能不够 a/b.

例如已经扩展到

19/45=1/5+????

如果第二项是 1/100, 那么由于 19/45-1/5=2/9=0.22222...

那么接下来至少要 0.2222/(1/100)=22 项加起来才足够 2/9, 所以继续　　扩展下去至少还要扩展 22 项,加起来才差不多够 a/b。

可变深度的搜索算法

```
PROCEDURE Search(k,a,b); {决定第 k 个分母 d[k]}
1.      If k=depth+1 then exit                          {depth 需要进行枚举}
2.      else if (a =1) and (b>d[k-1]) then [
                {a 整除 b 的情况}
3.        d[k]:=b;
4.        if not found or (d[k]<answer[k]) then  更新解;
5.      ]
6.      else [
7.      s:=max{ b div a,d[k-1]+1};     {确定 d[k]的上下界 s,t; }
                t:=(depth-k+1)*b div a
8.      for i:=s to t do [
9.       d[k]:=i;
10.             m := gcd(a,b);                          {a,b 的最大公约数为 m}
11.       search(k+1,(i*a-b) div m,(b*i) div m);
12.     ]
13.   ]
```