习题参考答案

习题

12.1 将装载问题改进的递归算法改写为迭代回溯算法。

```
Backtrack()
1   bestw←0, i←1,flag1[1]←0,flag2[1]←0, cw←0
2   while i≥1 and i≤n+1 do
3           if  i=n+1  then
4                   if cw>bestw then bestw←cw ; i←i-1
5           else
6                   if c(i)≤W and flag1[i]=0 and flag2[i]=0 then
7                           cw←cw+w[i];flag1[i]←1
8                           flag1[i+1]←0;flag2[i+1]←0;i←i+1
9                   else if c(i)≤W and flag1[i]=1 and flag2[i]=0 then
10                          cw←cw-w[i];flag2[i]←1;flag1[i+1]←0
11                          flag2[i+1]←0;i←i+1
12                  else if c(i)≤W and flag1[i]=1 and flag2[i]=1 then
13                          i←i-1
14                  else if c(i)>W and flag1[i]=0 and flag2[i]=0 then
15                          flag1[i]←1;flag2[i]←1;flag1[i+1]←0
16                          flag2[i+1]←0;i←i+1
17                  else if c(i)>W and flag1[i]=1 and flag2[i]=1 then
18                          i←i-1
```

12.2

证明：先在这个棋盘上做一个与棋盘重合的坐标系，则棋盘上的皇后等同于坐标系上的点。在这个坐标系中的两个点在同一对角线当且仅当它们的连线的斜率为 1 或-1。先有两个皇后i, j，它们的坐标分别为$(i, x_i), (j, x_j)$，则这两点的斜率为$(x_i - x_j)/(i - j)$，要使这两皇后在同一对角线上，当且仅当它们连线的斜率为 1 或-1，即$(x_i - x_j)/(i - j)$ 等于 1 或-1，展开，所证成立。

12.3

```
 8queen(k)
1   t←false
2   for i←1 to 8 do
3           x[k] ←i
4           if place(k) then //  第 k 个皇后能放在 i 位置上
5                   if k=8 then
6                       t←true; output x
7                   else
8                           t1←8queen(k+1)
9                           if t1 then t←true
10  return t
```

12.4

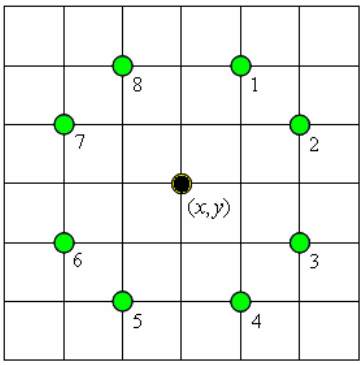
```
 Nqueens(k)
1   x[1]←0
2    m←1
3    num←0
3    while m>0 do
4           flag[m]←0
5           while x[m]≤n-1 d0
6                   x[m]← x[m] +1
7                   if place(m)=true then
8                           flag[m]←1; num←num+1
9                           if m=n then
10                                  if num≥k then return true
11                          else    m← m+1
12                                  x[m] ← 0;   flag[m] ←0
13          if flag[m]=1 then   num←num-1
14          m ← m-1
15   return false
```

12.5

```
 Backtrack(i)
1   if i>n then
2           for j←1 to n do
3                   if x[j]=1 then output j
4   else
5           x[i]←1
6           Backtrack(i+1)
7           x[i]←0
8           Backtrack(i+1)
```

12.6

首先将起点作为当前位置，按照象棋马的移动规则，搜索有没有可以移动的相邻位置；如果有可以移动的相邻位置，则移动到其中的一个相邻位置，并将这个相邻位置作为新的当前位置，按同样的方法继续搜索通往终点的路径；如果搜索不成功，则换另外一个相邻位置，并以它作为新的当前位置继续搜索通往终点的路径。马走日字，当马一开始在黑点(x, y)时，它下一步可以到达的点有八个，分别是方向 1：$(x+1, y+2)$，方向 2：$(x+2, y+1)$，方向 3：(x+2, y-1)，…，如图所示。引进增量数组，dx[]= (1, 2, 2,…)，dy[] = (2, 1, -1,…)，现在只需要知道方向 k，下一步的位置就是$(x+dx[k], y+dy[k])$。



令 flag[0..8;0..8]表示棋盘，并初始化为 0，表示这些位置均未跳过。令(x0,y0)为起始位置，当前位置点为(x, y)。route 数组纪录访问路线。

```
JumpHorse(i)
1   for k←1 to 8 do
2     if x + dx[k]≥0 and x + dx[k] < 8 and y + dy[k]≥0 and y + dy[k] <8 then
3           route[i]←k
4           x ← x + dx[k];   y← y + dy[k]
5           if x=x0 and y=y0 then
6                 output(i); return
7           else
8                 if flag[x, y]=0 then
9                       flag[x, y]←1
10                      JumpHorse(i+1)
11          x←x - dx[k];   y←y - dy[k]
```

12.7

```
3SAT(i)
1   if i>m then output
2   else
3         for k=0 to 1 do
4               x[i]←k; t←false
5               for j=1 to n do
6                     if c[j]=1 then t←true return
7               if   t=false   then   3SAT(i+1)
```

还可以利用约束函数，若某个子句中的变元均已经取值，但是某个子句仍为假，则可以剪支。

12.8

```
Hamiton(i)
1   if i = n then
2         if   w[x[n-1],x[n]]≠∞  and w[x[n],1]≠∞  then
3                     output x[i]
4   else   for   j ← i   to   n-1   do
5         if   w[x[i-1],x[j]]≠∞  then
6                           x[i] ↔x[j]
```

```
7                               Hamiton(i +1)
8                               x[i] ↔x[j]
```

12.9

```
TSP()
1   while i>0 do
2         j←i+1
3         while   j≤n do
4               if w[x[i],x[j]]≠∞  and cw+ w[x[i], x[j]] < bestw then
5                     x[i+1] ↔x[j];       cw ←cw+ w[x[i],x[i+1]]
6                     i←i+1;   j←i+1
7                     if i=n then
8                           if     w[x[n],1]≠∞  then
9                                 if cw + w[x[n],1] < bestw then
10                                                bestw←cw+ w[x[n],1]
11                                                for k ← 1 to n do
12                                                      bestx[k] ← x[k]
13                    else   j←j+1
14        cw←cw-w[x[i-1],x[i]];   w[x[i-1],x[i]] ← ∞
15        i←i-1
```

12.10

令 $x[i]$表示雇员 i 做做第 $x[i]$工作，显然解空间可以构造成一棵排列树。

```
BacktrackPerm(i)
1   if   i>n   then
2        sum←0
3        for j←1 to n do
4              sum←sum+c[j, x[j]]
5        if sum<bestc then
6              bestc←sum
7              for j←1 to n do
8                    bestx[j] ←x[j]
9   else
10       for j←i to n do
11             x[i] ↔ x[ j]
12             BacktrackPerm(i+1)
13             x[i] ↔ x[ j]
```

12.11　用回溯法求解并行机调度问题（见第 11 章）。

令 $x[i]$表示将第 $i(1\le i\le n)$个任务分配给第 $x[i]$台机器。解空间为一棵 m 叉树(总共 m 台机器)。

```
BacktrackMachine(i)
1  if   i>n   then
2       temp←0
3       for j←1 to m do
4            if length[j]>temp then temp←length[i]
5       if temp<bestc then
6            bestc←temp
7            for j←1 to n do
8                 bestx[j] ←x[j]
9  else
10      for j←1 to m do
11           length[j] ←length[j]+t[i]
12           x[i] ←j+1
13           if length[j]<bestc then
14                BacktrackMachine(i+1)
15           length[j] ←length[j]-t[i]
```

12.12

```
Backtrackjob(i)
1     if   i > n   then
2               if   t<bestcthen
3                    bestc←t
4                    for   j ← 1 to n do
5                         bestx[j] ← x[j]
6     else   for   j ← i   to   n   do
7               if t≥r[x[j]]   then
8                    t←t+p[x[j]]+q[x[j]]
9                    if t< bestc   then
10                        x[i] ↔ x[j]
11                        Backtrackjob(i +1)
12                        x[i] ↔x[j]
13                   t←t-p[x[j]]-q[x[j]]
```

12.13

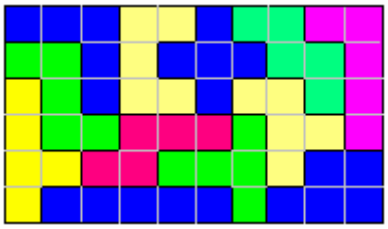


图 12.18

本题实现起来比较繁琐，可以不做。

12.14

```
BacktrackPerm(i)
```

```
1  if   i>n   then
2       sum←0
3       for j←1 to n do
4            sum←sum+P[j, x[j]] * Q[x[j], j]
5       if bestc<sum then
6            bestc←sum
7            for j←1 to n do
8                 bestx[j] ←x[j]
9  else
10        for j←i to n do
11             x[i] ↔ x[ j]
12             if place(i) then
13                  BacktrackPerm(i+1)
14             x[i] ↔ x[ j]
```

其中 $x[i]$表示男队员 i 和女队员 $x[i]$配对。place(i)测试如果不同男队员和同一个女队员 x[i]配对，则违反约束条件，返回 false。

12.15

令锯后共 n 段，每段的长度为 $x[i]$，总长度为 S

```
Findmin(S,n,x[n])
1     for i←1 to S do
2          if S% i =0 then
3               if Ping(S,i,1)=true then return i
Ping(s,i,k)
1  if s=0 then return true
2  else
3       j←k; ss←0
4       for j←k to n do
5            if ss+x[j]≤i then
6                 x[k] ↔x[j]
7                 ss←ss+x[k]
8                 if ss=i then t←Ping(s-i, i, k+1)
9                      if t then return true
10                     else     ss←ss-x[k]
11                              x[k] ↔x[j]
12 return   false
```

12.16　请为零件切割问题的回溯算法设计一个考虑浪费的剪支函数或者改进切割的过程。有不同的办法，请搜索相关文献。

实验题

12.17　对旅行商问题，分别用动态规划法和回溯算法求解，用实验分析方法分析哪个算法更有效。

12.18　完成 XOJ 如下题目：1008，1027，1039，1063，1064。

12.19　完成 POJ 如下题目：1010，1011，1020，1062，1167，1190，1085，1753，2078，2488，2677。