《算法分析与设计》课程

实验指导书

实验一　分治算法和贪婪法

一、实验目的

1. 掌握分治算法的设计思想与分析方法；
2. 能够按题目要求编程实现分治算法；
3. 理解和掌握贪婪算法的基本思想；
4. 使用贪婪算法求解背包问题以及最小花费生成树问题。

二、实验仪器及设备

1. 硬件环境：PC 机一台（1G 以上内存）
2. 软件环境：Windows 环境、C 语言（Visual Studio 2015）

三、实验内容

1. 采用分治算法求 10 个实数{ 9,3,6,2,1,8,4,5,7,23 }序列中的最大元素和最小元素，并分析算法的时间复杂度。
2. 采用分治算法方法实现快速排序算法。要求：输入数组为{9,4,6,2,5,8,4,5,6,22 }；限定数组排序范围；输出显示为递增顺序排序的数组。
3. 利用贪婪法求如下背包问题的最优解：n=5，M=100, 价值 P={20,30,66,40,60}，重量为 w={10,20,30,40,50}。

四、实验原理与分析

1. 分治算法基本原理：将一个难以直接解决的大问题，分解成一些规模较小的相同子问题，各子问题相互独立；递归地解决各子问题，将子问题的解归并成原问题的解。
2. 快速排序的思想方法：把序列就地划分为两个子序列，使第一个子序列的所有元素都小于第二个子序列的所有元素，不断地进行这样的划分，最后

构成 n 个子序列，每个子序列只有一个元素，这时，整个序列就是按递增顺序排序的序列了。

3. 快速排序算法的描述

```
void quick_sort(Type A[ ], int low, int high)
 {
        int k;
        if (low < high) {
              k = split(A, low, high);
              quick_sort(A, low, k-1);
              quick_sort(A, k+1, high);
        }
}
```

4. 贪婪算法基本原理：从问题的某一个初始解出发，在每一个阶段都根据贪心策略来做出当前最优的决策，逐步逼近给定的目标，尽可能快地求得更好的解。当达到算法中的某一步不能再继续前进时，算法终止。

5. 背包问题策略：价值与重量的比值大的优先装

背包问题算法描述：

```
knapsack（v[1..n],w[1..n])
i ←1
while i<=n do
a[i] ← v[i]/w[i]
i ←i+1
sort a
w ←0
p ←0
while i<=n    and w+w[i]<=W do
x[i] ← 1
w ←w+ w[i]
p ←p+ p[i]
i ←i+1
```

```
if i<=n then
x[i]←(W-w)/w[i]
p ←x[i]*p[i]
return    x,p
```

6. 最小花费生成树基本原理：在一个|V|个点的无向连通图中，取|V|−1 条边，并使所有点相连，所得到的子图被称为原图的一棵生成树；在一个带权的无向连通图中各边权之和最小的一棵生成树即为原图的最小生成树。
7. 最小花费生成树算法描述：

```
inline void Prim() {
memset(vis, 0, sizeof vis);
int s = n;
while(s --) {
    minn = INF;
    for (int i = 1;i <= n; ++i) {
        if (!vis[i] && v[i] < minn) {
            minn = v[i], pos = i;
        }
    }
    vis[pos] = 1, ans += v[pos];
    for (int i = 1;i <= n; ++i) {
        if (!vis[i] && v[i] > dis[pos][i]) {
            v[i] = dis[pos][i];
        }
    }
}
}
```

五、实验要求

1. 认真分析题目的条件和要求，复习相关的理论知识，选择适当的解决方

案和算法；

2. 编写上机实验程序，作好上机前的准备工作；
3. 上机调试程序，并试算各种方案，记录计算的结果（包括必要的中间结果）；
4. 分析和解释计算结果；
5. 按照要求书写实验报告。

六、思考题

1. 分析一下快速排序算法的时间复杂度和空间复杂度。
2. 为什么说分治算法中的组合步确定了分治算法的实际性能？
3. 求如下背包问题的最优解：n=7, M=15, 价值 P={10,5,15,7,6,18,3},重量为 w={2,3,5,7,1,4,1}。

实验二　动态规划和回溯法

一、实验目的

1. 理解动态规划的基本思想。
2. 运用动态规划算法解决最短路径问题。
3. 理解回溯法的基本原理，掌握使用回溯法求解实际问题。
4. 运用回溯法解决皇后问题及图着色问题。

二、实验仪器及设备

1. 硬件环境：PC 机一台（1G 以上内存）
2. 软件环境：Windows 环境、C 语言（Visual Studio 2015）

三、实验内容

1. 编写实验程序，实现利用动态规划方法求解图 2.1 中从顶点 0 到顶点 6 的最短路径问题。

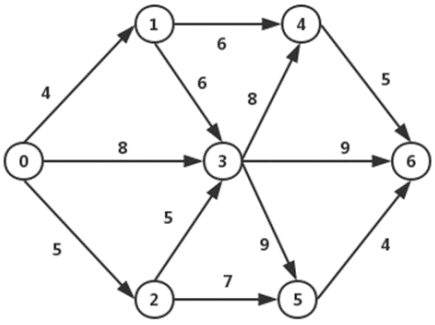

图 2.1

2. 编写程序实现 4 皇后问题的求解；
3. 编写程序实现用 3 种颜色为图 2.2 着色问题。

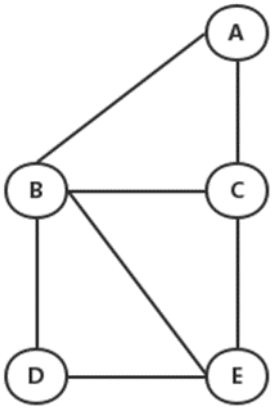

图 2.2

四、实验原理与分析

1. 动态规划问题基本原理：

（1）经分解得到的各个子问题往往不是相互独立的。

比如:A1A2A3 与 A2A3A4 有共同的子问题 A2A3

（2）在求解过程中，将已解决的子问题的最优值进行保存，在需要时可以轻松找出。

（3）通常采用表的形式记录子问题的最优值，即在实际求解过程中，一旦某个子问题被计算过，不管该问题以后是否用得到，都将其计算结果填入该表，需要的时候就从表中找出该子问题的最优值。

（4）根据最优值，记录最优决策，构造最优解。

2. 最短路径问题算法：

核心代码如下：

```
struct NODE {      // 邻接表结点的数据结构
   int          v_num;          // 邻接顶点的编号
   Type       len;          // 邻接顶点与该顶点的费用
   struct NODE *next;       // 下一个邻接顶点
     };
   struct NODE node[n];     // 多段图邻接表头结点
   Type     cost[n];
   int       route[n];
   int       path[n];
 Type fgraph(struct NODE node[ ], int route[ ], int n)
```

```
{     int i;
      struct NODE *pnode;
      int    *path = new int[ n ];
      Type min_cost, *cost = new    Type[ n ];
      for(i=0; i<n; i++) {
            cost[ i ] = MAX_VALUE_TYPE;
            path[ i ] = -1;      rouet[ i ] = 0;
     }
     cost[ n-1 ] = ZERO_VALUE_OF_TYPE;
          for (i=n-2; i>=0; i--) {
      pnode = node[ i ]->next;
    while (pnode != NULL) {
        if (pnode->len + cost[pnode->v_num] < cost[ i ]) {
              cost[ i ] = pnode->len + cost[pnode->v_num];
              path[ i ] = pnode->v_num;
       }
        pnode = pnode->next;
}
}
  i = 0;
while ((route[ i ] != n-1) && (path[ i ] != -1)) {
        i++;
        route[ i ] = path[ route[ i-1 ] ];
}
   min_cost = cost[ 0 ];
  delete path;     delete cost;
  return min_cost;
}
```

3. 回溯算法基本原理：回溯法是在仅给出初始结点、目标结点及产生子结点的条件（一般由问题题意隐含给出）的情况下，构造一个图（隐式图），然后按照深度优先搜索的思想，在有关条件的约束下扩展到目标结点，从而找出问题的解。回溯法是一种“能进则进，进不了则换，换不了则退”的基本搜索方法。
4. 皇后问题的算法：

核心代码如下：

```
 void n_queens(int n,int x[])
{
 int k=1;
 x[1]=0;
 while (k>0){
 x[k]=x[k] +1;//在当前列加 1 的位置开始搜索
 while ((x[k]<=n)&&(!place(x,k)))//当前列位置是否满足条件
 x[k] = x[k] +1;//不满足条件，继续搜索下一列位置
 if (x[k]<=n) { //存在满足条件的列
   if (k==n) break; //是最后一个皇后，完成搜索
   else{
        K=K+L; x[k]=0; //不是，则处理下一个行皇后
        }
    }
 else{      // 已判断完 n 列，均没有满足条件
    x[k]=0;   k=k-1;// 第 k 行复位为 0,回溯到前一行.
     }
   }
 }
```

5. 图着色问题的算法：

核心代码如下：

（1） 数据结构

```
int  n;        // 顶点个数
int  m;        // 最大颜色数
int  k;        // 顶点号码，或搜索深度
int  x[n];          // 顶点的着色
BOOL   c[n][n]; // 布尔值表示的图的邻接矩阵
```

函数 ok：判断顶点着色的有效的性有效返回 TRUE，无效返回 FALSE

```
BOOL ok(int x[ ], int k, BOOL c[ ][ ], int n)
```

```
    {
        int i;
        for (i=0; i<k; i++) {
           if (c[ k ][ i ] && (x[ k ] == x[ i ])
              return FALSE;
        return TRUE;
    }
void m_coloring(int n, int m, int x[ ], BOOL c[ ][ ])
{    int i,k;
   for (i=0;i<n;i++)          x[i] = 0;
     k = 0;
    while (k >= 0) {
        x[ k ] = x[ k ] + 1;
        while ((x[k] <= m) && (!ok(x, k, c, n)))    x[k] = x[k] + 1;
        if (x[k] <= m) {         if (k == n-1) break;
                    else k = k + 1;    }
    else {     x[k] = 0;     k = k – 1;     }
    }
        if (k==n-1) return TRUE;
    else FALSE;
 }
```

五、实验要求

1. 认真分析题目的条件和要求，复习相关的理论知识，选择适当的解决方案和算法；
2. 编写上机实验程序，作好上机前的准备工作；
3. 上机调试程序，并试算各种方案，记录计算的结果（包括必要的中间结果）；
4. 分析和解释计算结果；
5. 按照要求书写实验报告。

六、思考题

1. 用递归函数设计一个求解货郎担问题的动态规划算法，并估计其时间复杂性。

2. 使用回溯算法解八皇后问题时，在最坏情况下，求所生成的搜索树的结点总数。