P140

7、

1. 对于一个包含100万随机数的数组排序，快速排序比插入排序快多少倍？
2. 是非题：对于n>1的n元素数组，是否存在插入排序比快速排序跟快的情形

解：（1）插入排序的平均速率是Cavg=O(n2)，而快速排序的平均速率是Cavg=nlog2n,得插入排序约/快速排序等于50000，所以快速排序比插入排序快了50000倍

1. 存在

P225

1. 为下列问题设计一个动态规划算法。已知小木棍的销售价格pi和长度i相关，i=1,2...,n，如何把长度为n的木棍切割若干根长度为整数的小木棍，使得所能获得的总销售价格最大？该算法的时间效率和空间效率各是多少？

长度为n的最大价值 price(n)=MAX(price(i)+price(n-i))长度为n的价格有两种 第一种：原始长度为n时的价格 第二种：加n分割为个小块 再加起来的价格

设长度1~n长度的木棍价格为p[1…n]

先从最短的长度 1开始找相对应长度可得到的最大价值，因为长度1无法再分，所以maxprice[1] 就为原始长度价格 p[1]

然后长度2的可得到的最大价值maxprice[2]就为maxprice[1] +maxprice[1] 和 p[2]之中最大的那个3的可得到的最大价值maxprice[3]就为 maxprice[1]+maxprice[2] 、maxprice[2]+maxprice[1]和p[3]中最大的那个（为什么没有maxprice[1]+maxprice[1]+maxprice[1]？因为1+1=2，长度为2的最大价格maxprice[2]已知，所以不需要再把2分为1+1了，而1+2=3、2+1=3时的长度3的maxprice[3]的价格才是不知道的。也就是每个长度只需要分为两部分就行了）

因为比当前长度小的所有整数长度的对应的最大价格都是已知的，所以长度为n时只需要找到maxprice[1]+maxprice[n-1]、maxprice[2]+maxprice[n-2]、…、maxprice[i]+maxprice[n-i]、…、maxprice[n-1]+maxprice[1]、p[n]中最大的值，再赋值给maxprice[n]

此算法的时间效率是O(n^2)，空间效率是O (N)

P229

1. 对于背包问题的自底向下动态规划算法，请证明：

它的时间效率属于Ɵ(nW)

它的空间效率属于Ɵ(nW)

从一张填好的动态规划表中求得最优子集的组合所用的时间属于O(n)

P264

1. 写一个程序，为给定的英文文本构造一套哈夫曼编码，生成一套哈夫曼编码。

写一个程序，对一段用哈夫曼编码的英文文本进行解码

做一个实验，测试对包含1000个单词的一段英文文本进行哈夫曼编码时，典型的压缩率位于什么样的区间。

对编码程序做一个实验，测试如果用标准的估计频率代替英文文本中字符的实际出现频率，该程序的压缩率会有怎样的变化。

P331

1. 用回溯法生成{1，2，3，4}的所有排列

代码：

import java.util.\*;

public class pailie{

    static int n;

    static int[] a;//一种排列情况

    static boolean[] b;//判断数字i是否已被使用

    static void dfs(int step){//step表示当前到了第几位

        if(step==n+1){//前n位已经放好，输出这种排列情况

            for(int i=1;i<=n;i++)

                System.out.print(a[i]);

            System.out.println();

            return;

        }

        for(int i=1;i<=n;i++){

            if(!b[i]){//若数字i还未被使用

                a[step]=i;//在当前位上放i

                b[i]=true;//防止重用

                dfs(step+1);//放下一位数字

                b[i]=false;//回溯法，进行另一种尝试

            }

        }

    }

    public static void main(String[] args){

        Scanner in=new Scanner(System.in);

        n=in.nextInt();

        a=new int[n+1];

        b=new boolean[n+1];

        dfs(1);//从第一位开始放

    }

}

结果：

