学院 软件学院 专业 软件工程 班 年级 学号 姓名

2009～2010 学年第 1 学期期末考试试卷

《 算法分析》（A 卷 共 3 页）

（考试时间：2009 年 11 月 29 日）

题号	一	二	三	四	五	六	七	八	成绩	核分人签字
得分										

一、算法分析（20 分）

1、应用 Master 方法求解以下递归方程

（1）$T(n)=7T(n/2)+cn^2$ (5 分)

解：a=7, b=2, $\log_b a \approx 2.81$, $f(n)=cn^2=O(n^{2.81-\varepsilon})$, 取 $0<\varepsilon<0.5$. 根据 Master 定理有

$$T(n)=\Theta(n^{2.81})$$

（2）$T(n)=4T(n/2)+n^2$ (5 分)

解：a=4, b=2, $\log_b a=2$, $f(n)=cn^2=\Theta(n^2)$, 根据 Master 定理有

$$T(n)=\Theta(n^2\log n)$$

2、假定 T(n)满足以下递归式

$T(n)=c \quad n=1,$

$T(n)=4T(n/2)+n \quad n>1$

式中 c 为正的常数。用归纳法证明 $T(n)\le c_1n^2-c_2n$, c_1 和 c_2 为待定常数.（5 分）

证明:

假设当小于 n 时归纳假设成立，如果取 c2>1，则有

$T(n)=4T(n/2)+n$

$\le 4[c_1(n/2)^2-c_2(n/2)]+n$

$=c_1n^2-c_2n-(c_2n-n)$

$\le c_1n^2-c_2n$

为使归纳假设当 n=1 时也成立，取 c_1 使得 $c_1>c+c_2$ 即可.

3、使用步计数法分析下面算法在最好和最坏情形的渐近时间复杂度（5 分）

```
template <class T>
bool MinMax(T a[], int n, int & Min, int & Max)
{// Locate min and max elements in a[0:n-1].
 // Return false if less than one element.
  if (n < 1) return false;
  Min = Max = 0; // initial guess
  for (int i = 1; i < n; i++)
    if (a[Min] > a[i]) Min = i;
    else if (a[Max] < a[i]) Max = i;
  return true;
}
```

解：

算法中第一行最好/最坏情形均为 Θ(1)；第二行也是 Θ(1)；第三行最好/最坏情形均为 Θ(n)；第四行最好/最坏情形也都是 Θ(n)；第五行最坏情形均为 Θ(n)，最好情形为 Ω(0)；第六行最好/最坏情形均为 Θ(1)。所以，最坏情形时间复杂度为 Θ(1)+ Θ(1)+ Θ(n)+ Θ(n)+ Θ(1)= Θ(n)，而最好情形复杂度为 Θ(1)+ Θ(1)+ Θ(n)+ Θ(1)= Θ(n)。

二、分治法（20分）：

1、在快速排序算法中，如果我们先调用最坏情形时间复杂度O(n)的选择算法找出第n/2小元素并以此为支点（pivot）对要排序的数组进行分划，则改进后的快速排序算法的最坏情形时间复杂度T(n)是什么？假定$n=2^k$，列出T(n)满足的递归方程并分析.（10分）

解：

算法的时间复杂度 T(n)满足以下方程

$$T(n)=2T(n/2)+\Theta(n)$$

其中 Θ(n)为调用选择算法找出第 n/2 小元素的时间加上对数组进行分划的时间。应用 Master 方法求解该递归方程得到 T(n)= Θ(nlogn)，所以改进后的快速排序算法最坏情形时间复杂度为 Θ(nlogn)。

2、以下伪代码是用分治法设计的求解最大-最小问题的算法：

```
Max-Min(A[1,n],max,min)
  if n=1 max←min←a[1],return;
  if n=2 比较 a[1]和 a[2]得到 max 和 min
       else m←n/2
            Max-Min(A[1,m],max1,min1),
            Max-Min(A[m+1,n],max2,min2),
            max←max(max1,max2),
            min←min(min1,min2),
            return.
```

试分析在 $n=2^k$ 时算法所用的关键字比较次数. （10 分）

解：

设T(n)为使用上述分治法算法所需要的比较次数,假设$n=2^k$，则有:

T(1)=0, T(2)=1;

T(n)=2T(n/2)+2, n ＞ 2 .

迭代展开得:$T(n)= 2^{k-1}T(2)+2+\cdots+2^{k-1}=2^{k-1}+ 2^k-2=(3n/2)-2$

三、贪心法（25 分）

1、（1）用 1－优化法求解以下 0/1 背包问题的实例：

n = 5, w = [16, 20, 5, 15,10]，p = [32, 20, 15, 30,15], c = 40.要求写出计算过程.

（2）分析 1-优化法的时间复杂度.（15 分）

解：

（1）物品的密度分别为[2, 1, 3, 2, 1. 5]，排序后物品顺序为[3, 1, 4, 5, 2]. 排序后物品重量为 w’ = [5, 16, 15, 10, 20]，效益为 p’ = [15, 32, 30, 15, 20]. 以下在运行算法时，按排序后的顺序对物品编号。

按1-优化方法,不预先装物品时算法得到的计算结果为x=(1, 1, 1, 0, 0),效益值为77。先放物品1、2、3，得到的结果同于不预先装物品时的结果，即贪心解为x =(1, 1, 1, 0, 0)，效益值为77；先放物品4，得到x =(1, 1, 0, 1, 0),效益值为62；先放物品5，得到x =(1, 0, 1, 0, 1),效益值为65；这些贪心解中，x=(1, 1, 1, 0, 0)为最好。

所以，1-优化方法的结果为x=(1, 1, 1, 0, 0)，效益值为77。回到原来的编号，1－优化算法得到的解为x=(1, 0, 1, 1, 0)，效益值为77。

（2）

算法的时间复杂度分析：

按密度从大到小排序需 $\Theta(n\log n)$;不预先放任何物品执行一遍贪心算法需 $\Theta(n)$时间;每次预先装一物品在对其余物品执行一遍贪心算法需 $\Theta(n)$时间; 共计执行 n 次；所以总的时间为 $\Theta(n^2)$。以上 3 项相加得到 1-优化法的时间复杂度为 $\Theta(n^2+n+n\log n)=\Theta(n^2)$。

2、给定 n 个任务，1,2,…,n，假定任务 i 要求的执行时间是 t_i。如果按顺序 1,…,n 执行这些任务，任务 i 的完成时间 c_i 为 $t_1+t_2+\ldots+t_i$。定义一个任务顺序的平均完成时间 ACT 为 $(c_1+c_2+\ldots+c_n)/n$，不同的顺序可能有不同的平均完成时间。试证明：在 n!个可能的任务顺序中，按最小执行时间优先的贪心策略得到的任务顺序有最小的平均完成时间。（10 分）

证明：

按最小执行时间优先的贪心策略得到的任务顺序等同于按任务执行时间从小到大的排列得到的任务顺序。下面证明在此顺序下 ACT 值最小。

设在某任务顺序中, 顺序号 i>j,但 $t_i<t_j$, 则交换作业 i、j 的顺序,得到一个新的任务顺序. 设原顺序的平均完成时间为 ACT，改变后的平均完成时间为 ACT’, 下面证明 ACT’<ACT。

因为 $nACT=(nt_1+ \cdots+(n-j+1)t_j+\ldots +(n-i+1)t_i+\ldots)$

$nACT'=(nt_1+ \cdots+(n-j+1)t_i+\ldots +(n-i+1)t_j+\ldots)$

$nACT-nACT'= (i-j)t_j-(i-j)t_i=(i-j)(t_j-t_i)>0$，所以, ACT’<ACT，也就是每消除一个逆序 ACT 值减小。所以当无逆序时,即任务按执行时间从小到大排列时，ACT 值最小。

证明 2：通过逐一比较任何一个可行解和贪心解的完成时间进行分析也可以证明贪心解的优化性，此处从略。

四．设 s_i，i=1,2,…,n，和 M 为任意给定的正数。设 J 是下标{1,2,…，n}的一个子集，$S_J=\sum_{j\in J}s_j$ 为子集 J 的和数。最大子集和数问题指：求满足 $S_J\le M$ 且使 S_J 最大的子集 J。

试用动态规划法设计求解最大子集和数问题的算法，要求：

（1）列出动态规划的递归关系

（2）就以下实例执行算法：

n=5，M=12，s_1=8，s_2=2，s_3=7，s_4=9，s_5=5，

并写出回溯（traceback）求解过程。(15 分)

解：

（1）设 f(i , y)为集合{s_i,…,s_n}的约束为 y 的最大子集和数,则以下递归式成立

$$f(n,y)=\begin{cases}s_n, & y\ge s_n\\ 0, & y<s_n\end{cases}$$

$$f(i,y)=\begin{cases}\max\{f(i+1,y),f(i+1,y-s_i)+s_i\}, & y\ge s_i\\ f(i+1,y), & 0\le y<s_i\end{cases}$$

（2）针对题（2）的实例，以下用元组法上述递归：

P(5)=(0,5), Q=(9); 合并 P(5)和 Q 得到

P(4)=(0,5,9)，Q=(7,12); 合并 P(4)和 Q 得到

P(3)=(0,5,7,9,12)，Q=(2,7,9,11)；合并 P 和 Q 得到

P(2)=(0,2,5,7,9,11,12)。利用 P(2)计算 f(1,12)得到 f(1,12)=12。

令 xi=1 表示子集中包含元素 s_i，否则不包含 s_i 元素，则回溯（traceback）求解过程如下：

f(1,12)= f(2,12)=12=〉 x1=0；

f(2,12)= f(3,12)=12=〉 x2=0；

f(3,12)≠f(4,12)=〉 x3=1；

f(4,5)= f(5,5) =〉 x4=0;

最后得到 x5=1。 所以，不超过 12 的最大和数子集为(0,0,1,01)，对应子集{ s_3,s_5}。

答案 2：设 g(i , y)为集合{s_1,…,s_i}的约束为 y 的最大子集和数，则反向列动态规划的递归关系如下：

$$g(i,y)=\begin{cases}\max\{g(i-1,y),g(i-1,y-s_i)+s_i\}, & y\ge s_i\\ g(i-1,y), & 0\le y<s_i\end{cases}$$

求解过程类似，此处从略。

五、已知船的载重量为 c，货箱的重量为 w_i，i=1,…,n。分别用回溯法和分支-限界法找出使装载量最大的装船方案。要求：

1、给出在两种方法中使用的限界条件；

2、就以下实例画出算法展开的部分状态空间树：

n= 4，w = [10, 13 , 7 , 9]，c= 20 。 (20 分)

解：

1、无论是回溯法还是分支-限界法都使用以下限界条件：

(1) 设 X 为状态空间树的一个节点，cw 为在节点 X 处已装的重量, w_{i+1} 为下一个要考虑的货箱的重量,如果 cw+ w_{i+1}>c，则限界 X 的左子节点。

(2) 设 bestw 为当前最优装箱重量, cw 为在节点 X 处已装的重量, r 为未装的货箱的重量之和, 另一个限界条件为:

cw+r<=bestw,

2、