16-17 学年《算法设计与分析》期中参考答案和评分标准

一、单选题（每题 2 分，共 30 分）

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
B	B	A	A	A	C	D	B	D	D	A	C	C	B	C

二、填空题（每题 1 分，共 10 分）

1．10^3　$n\log n$　n^2　2^n
2．出口条件(或者边界条件)
3．$O(n^{\log_2 7})$ 或者 $O(n^{2.81})$
4．23
5．(1, 0.5, 1)
6．限界函数
7．最少
8．优先队列式
9．274
10．*output*(*a*, *n* – 1)

三、算法应用题（每题 8 分，共 40 分）

1．(1) 加工次序(1, 7, 6, 4, 5, 8, 3, 2)，最短加工时间 73 小时 (各 2 分)
(2) (共 4 分)

产品编号	M_1 开始	M_1 结束	M_2 开始	M_2 结束	M_2 空闲
1	0	2	2	9	2
7	2	5	9	22	
6	5	22	22	32	
4	22	33	33	40	1
5	33	41	41	47	1
8	41	51	51	55	4
3	51	67	67	69	12
2	67	72	72	73	3

2．覆盖方案：其中阴影方格为特殊方格（标注出 L 型骨牌序号，从 1 号开始）。

3	3	4	4	8	8	9	9
3	2	2	4	8	7	7	9
6	2	5	5	11		7	10
6	6	5	1	11	11	10	10
18	18	19	1	1	13	14	14
18	17	19	19	13	13	12	14
21	17	17	20	16	12	12	15
21	21	20	20	16	16	15	15

3．根据递归关系式，求 $A_1 A_2 A_3 A_4$ 最少数乘次数为求 *m*[1][4] (2 分)

第一阶段 —1 个矩阵：

m[1][1] = *m*[2][2] = *m*[3][3] = *m*[4][4] = 0

第二阶段 —2 个矩阵：

m[1][2] = *m*[1][1] + *m*[2][2] + $p_0p_1p_2$ = 0 + 0 + 50 ´ 10 ´ 40 = 20000

m[2][3] = *m*[2][2] + *m*[3][3] + $p_1p_2p_3$ = 0 + 0 + 10 ´ 40 ´ 30 = 12000

m[3][4] = *m*[3][3] + *m*[4][4] + $p_2p_3p_4$ = 0 + 0 + 40 ´ 30 ´ 5 = 6000

第三阶段 —3 个矩阵：

m[1][3] = min(*m*[1][2] + *m*[3][3] + $p_0p_2p_3$, *m*[1][1] + *m*[2][3] + $p_0p_1p_3$) = min(20000 + 0 + 50 ´ 40 ´ 30, 0 + 12000 + 50 ´ 10 ´ 30) = 27000

m[2][4] = min(*m*[2][3] + *m*[4][4] + $p_1p_3p_4$, *m*[2][2] + *m*[3][4] + $p_1p_2p_4$) = min(12000 + 0 + 10 ´ 30 ´ 5, 0 + 6000 + 10 ´ 40 ´ 5) = 8000

第四阶段 —4 个矩阵：

m[1][1] + *m*[2][4] + $p_0p_1p_4$ = 0 + 8000 + 50 ´ 10 ´ 5 = 10500

m[1][2] + *m*[3][4] + $p_0p_2p_4$ = 20000 + 6000 + 50 ´ 40 ´ 5 = 36000

m[1][3] + *m*[4][4] + $p_0p_3p_4$ = 27000 + 0 + 50 ´ 30 ´ 5 = 34500

m[1][4] = min(10500, 36000, 34500) = 10500

每种划分表达式及乘法次数计算正确得 2 分。

4．(1) 将这 10 位客户的申请按照结束时间 *f*(*i*) 递增排序，如下表：

i	1	2	3	4	5	6	7	8	9	10
s(*i*)	1	3	0	5	3	5	6	8	8	11
f(*i*)	4	5	6	7	8	9	10	11	12	13

1) 选择申请 1(1, 4)　　　　　　　　　　　　　　　　(3 分)

2) 依次检查后续客户申请，只要与已选择的申请相容不冲突，则选择该申请，直到所有申请检查完毕：

可以依次选择申请 4(5, 7)、申请 8(8, 11)、申请 10(11, 13)。(3 分)

因此可以满足：申请 1(1, 4)、申请 4(5, 7)、申请 8(8, 11)、申请 10(11, 13) 共 4 个客户申请，为可以满足的最大客户人数。　　(2 分)

(2) 算法的时间复杂度为 O(*n*log*n*)。因为在贪心选择之前，需要按申请的结束时间递增排序　　　　　　　　　　　　(2 分)

5．(1) 设 *f*(*i*) 表示：从左向右扫描过来直到以 *a*[*i*] 元素结尾的序列，获得的最长上升子序列的长度，且子序列包含 *a*[*i*] 元素(1 £ *i* £ *n*)。

$$f(i) = \begin{cases} 1 & i = 1 \\ \max\{f(j) + 1\text{：当 } a[i] > a[j];\ 1 £ j < i\} & i > 1 \\ 1 & i > 1;\ 1 £ j < i,\ a[i] £ a[j] \end{cases}$$

即，*f*(*i*)是从 *f*(1), *f*(2), ...到 *f*(*i* – 1) 中找最大的一个值，再加 1。或者就是 1。主要是看 *a*[*i*] 这个元素能否加入到之前已经获得的最长上升子序列，如果能加入，是之前已获得的最长上升子序列长度加 1；如果不能加入，就取这最后一个元素作为一个单独子序列，长度为 1。　　(4 分)

最后，整个序列最长上升子序列长度为 max{*f*(*i*): 1 £ *i* £ *n*}　　(1 分)

(2) 对于序列{4, 2, 6, 3, 1, 5, 2, 11, 14, 12}

最长上升子序列的长度为 5　　　　　　　　　　　　　(3 分)

四、算法设计题（使用 C 或 C++或 Java 语言实现）（每题 10 分，共 20 分）

1．输入一个超长正整数 *s* 和一个正整数 *c*，　求 *s* × *c* 的精确值。

```
#include <stdio.h>
#include <string.h>
int main( )
{
    char s[256];
    int c, b, d;
    int a[256], i, j, n;                        //  1 分
    scanf("%s", &s);
    scanf("%d", &c);
    n = strlen(s);
    d = 0;                                      //  2 分
    for (i = 0, j = n – 1; i < n; i ++, j --)
    {
        b = (s[j] – 48) * c + d;
        a[i] = b % 10;
        d = b / 10;
    }                                           //  4 分
    while (d != 0)
    {
        a[n] = d % 10;
        d = d / 10;
        n ++;
    }                                           //  2 分
    for (i = n – 1; i >= 0; i --)
        printf("%d", a[i]);                     //  1 分
    return 0;
}
```

2．时间复杂度为 *O*(*n*) 的求最大子段和的算法

```
int MaxSum(int a[ ], int n)
{
    int sum = 0, b = 0;                         //  2 分
    for (int i = 1; i <= n; i ++)
    {            //2 分
        if (b > 0)
            b += a[i];
        else
            b = a[i];                           //  3 分
        if (b > sum)
            sum = b;                            //  2 分
    }
    return sum;                                 //  1 分
}
```

注：如果算法正确，时间复杂度大于 *O*(*n*)，扣 4 分。