北京大学信息学院

2007 年《数据结构与算法》期中考试答案

——张铭、赵海燕、王腾蛟、宋国杰

一、 填空题（30 分，每空 2 分）

1、 C

2、 指数量级正确，给一半分。按 e 为底计算给一半分。

算法复杂度	nlog n	n^2	$100n^2$	n^3	2n
数据规模	**(e)1.29×10^{12}** **(10)2.889×10^{12}**	**6×10^6**	**6×10^5**	**33019** **或 3.3×10^4**	**1.8×10^{13}**

3、 A, D

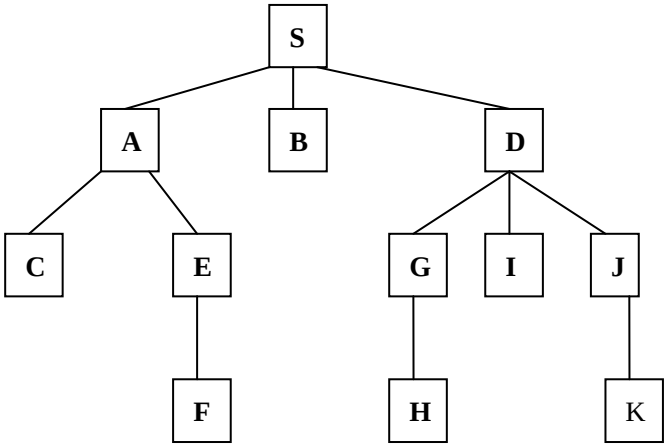
4、 后序

5、 60

6、 11

7、 (n+1)/2 (O(n), n/2 也都给了一半分)

8、



9、 4 或 7（答对一个给了一半分）

10、 k^h， $(k^{h+1}-1)/(k-1)$

二、 简答题（15 分，每题 5 分）

1、 证明: n 个结点的 m 叉树共有 n * m 个指针域, 已使用的指针域为 n-1, 所以空指针的个数为 n(m-1)+1

2、解答：Next = {-1, 0, 0, 0, 0, 1, 1, 1, 2} 或精简版{-1, 0, 0 ,0 ,-1, 1, 1, 0, 0}

匹配返回19，其过程如下图

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
B	A	A	A	B	B	B	C	D	D	D	C	C	H	H	H	H	B	B	B	A	A	A	B	B	B	A	A	D	D
B	A	A	A	B	B	B	A	i=7，j=7, i =next[7]=0																					
							×																						
							B	i=0, j =7, i = next[i] = -1																					
							×																						
								B	i=0, j =8, i = next[i] = -1																				
								…	…																				
														…….															
													×			B	i=0, j = 16, i = next[i] = -1,												
																×													
																	B	A	i=1, j = 17, i = next[1] = 0,									
																		×											
																		B	i=0, j = 18, i = next[0] = -1,									
																		×											
																			B	A	A	A	B	B	B	A	A	成功	

评分标准:

Next 数组3分，错3个扣1分，若给出的为精简版的，整体扣1分

匹配过程2分，

2、 说明边界处理方便得 1 分，下面 4 个点各得一分。说其他的理由，有道理酌情给分。

答: 带头结点的好处，可以方便一些边界处理，例如（答的点有 4 个以上就可以):

（1）单链表为空时的插入;

（2）单链表被删除为空表时的处理;

（3）插入作为表的第一个结点;

（4）删除表的第一个结点;

如果答: 维持当前结点的前驱，这样比较有利于对当前结点的删除，或在当前结点前插入。

这其实包含前面 1-4 点。如果学生还明显地说明了"有利于对空表处理"，则可以给满分。

如果没有这句话，只说明"方便一些边界处理"、"维持当前结点的前驱"，

则扣 1 分，也就是可以给 4 分。

三、 程序填空题（25，前 5 空，每空 2 分，后 3 空，每空 5 分）

(1) info=＊ppos;

(2) rpos=inpos;

(3) rpos-ipos;

(4) ipos;

(5) ppos+1

(6) while ((c_temp=(int)pop()) != '(')　　// 3 分

str_temp[str_temp_pos++]=c_temp;

(7) {　// 比较操作符的优先级进行 push 或者 pop

stack_top_operator=S.pop();　　// 2 分

while (priority(stack_top_operator)>=priority(c_temp)) {// 5 分

str_temp[str_temp_pos++]=stack_top_operator;

stack_top_operator=pop();

}

S.push(stack_top_operator);　　// 1 分

S.push((double)(c_temp));　　// 1 分

}

(8) str_temp[str_temp_pos++]=stack_top_operator;　　// 3 分

四、 算法设计与实现（30，每题 15 分）

1、 评分标准:（按 15 分计）

1、 算法思想:（5 分）

1、 对于串 s 的每个字符在串 t 中搜索一次（可以直接从头至尾搜索或调用 strchr 函数），有无出现，若无则将该字符存储于串 r 中;

2、 消除 r 中的重复字符; 优化版，先把 s 中的重复字符消除

3、 通过定位运算记录 r 中各字符在 s 中的第 1 次出现位置（也可以在第 1 步时保留出 s 中不出现于 t 中字符的位置）

2、算法注释　（2 分）

3、参考代码　（8 分）

```
typedef struct {
        int p;
        char c;
} newType;
int main(int argc, char *argv[])
    char s[30], t[30];
    newType r[30];
    int i, j, k = s0;

    cin >> s;
    cin >> t;
        int lenS = strlen(s);
    int lenT = strlen(t);

    //  消除 s 中的重复字符
    for (i=0; i < lenS; i++) {
        for (j=i-1; j>=0; j--)                  //  当前字符之前的字符逐一比较，
            if (s[j]== s[i]) {                  //  若相等则把后续的字符均左移一个位置
                for (int m=i; m<lenS; m++)
                    s[m]=s[m+1];
                lenS--;
                j=i-1;
            }
    }
        //  断 s 中的字符是否出现在 t 中
    for ( i=0; i < lenS; i++) {
    s       for (j= 0; j<lenT; j++)
            if (s[i]==t[j]) break;              //  出现的话，则跳过
        if (j ==lenT) {                         //  与 t 的所有字符比较且不等，记录在 r 中
            r[k].p = i;
            r[k].c = s[i];
            k++;
        }
    }
 }
```

2、　答案:

```
void CountTreeLeaves(TreeNode* root , int& count )      //得到树的树叶个数
{
    while(NULL!=root)
    {
        if ( root->pChild == NULL )
```

```
            count++;
        CountTreeLeaves( root->LeftMostChild() , count);     //访问头一棵树树根的子树
        root=root->RightSibling();                           //周游其他的树
    }
}

int   Tree<T>::CountTreeLeaves(TreeNode<T> * root )
{
    int count = 0;
    const TreeNode<T> * p = root;
    while(p != NULL)
    {
        if(p->pChild == NULL)
            count += 1;
        else
            count += CountTreeLeaves(p->pChild);
        p = p->pSibling;
    }
    return count;
}
```