算法课期中考试试题

1.将以下函数按增长率从小到大排序： （11分）

(1) ; (2) ; (3) ; (4) ;

(5) 。

1.解：题目中的函数按增长率从小到大排序如下：

(1) ; (2) ; (3) ; (4) ;

(5) 。

2.分析以下程序最坏情况下的时间复杂性。 （11分）

PROCEDURE Insertion\_Sort (A);

BEGIN

1. A[0].key := ;
2. FOR i := 2 TO n DO

BEGIN

1. j := i ;
2. WHILE A[ j ].key < A[ j-1 ].key DO

BEGIN

1. swap (A[ j ], A[ j-1]) ;
2. j := j-1 ;

END

END

END;

2.解：分析如下：

第(1)行 ：O(1)

第(3)行 ：O(1)

第(5), (6)行：O(1)+O(1)=O(max(1, 1))=O(1)

第(4)(6)行，至多循环i次，每次循环O(1)时间，

故(4)(6)行需O()=O()时间。

第(3)(6)行需O(1)+O()=O(max(1,))=O()时间。

第(2)(6)行：需O()=O()=O()=O()

时间。

整个程序(1)(6)行需O(1)+O()=O(max(1, ))=O()时间。

3.求解以下递推关系式中的T(n)。 （12分）

(1) 。

(2) 。

3.解：

(1) 有，

故。

(2) 有，

故。

4.设输入关键字：5, 28, 19, 15, 20, 33, 12, 17, 10，用链地址法构造Hash表。设该Hash表有9个表项A[0], A[1], …, A[8]，Hash函数为

。试给出构造好的Hash表。 （11分）

A

0

1 28 19 10

2 20

3 12

4

5 5

6 15 33

7

8 17

5. 给出一个二叉查找树A (A为指向该树树根结点的指针)，设为该二叉查找树中保存的一个集合元素。给出一个算法SUCCESSOR，求树A中元素的后继元素。 （11分）

5.解：算法如下：

PROCEDURE SUCCESSOR (x, A);

VAR

B : 结点指针；

BEGIN

B := A;

WHILE x A.key DO

IF x A.key THEN

BEGIN

B := A;

A := A.leftchild;

END

ELSE IF x A.key THEN

A := A.rightchild;

/\* if we reach here, x A.key \*/

If A.rightchild NIL THEN

BEGIN

B := A.rightchild;

WHILE B.leftchild NIL DO

B := B.leftchild;

END

ELSE IF A.key B.key THEN B := NIL;

RETURN(B);

END;

6.设A[1], A[2], …, A[10]中存放元素：23, 17, 14, 6, 13, 10, 1, 5, 7, 12。问数组A是否一个(父结点大于子结点的)堆？如果不是，用堆排序算法中建堆的方法，将A构造成一个堆。 （11分）

6.解：A[1], A[2], …, A[10]构成二叉树如下：

1

2 3

4 5 6 7

8 9 10

不构成堆，因为A[4]=6<A[9]=7。

用堆排序建堆的算法，调整如上图得到堆如下图。

1

2 3

4 5 6 7

8 9 10

7.设计一个O(n)时间的算法，找出一棵树的中心。 (11分)

7.解：算法如下：

(1)输入一棵树T;

(2)找出T的所有叶子的集合;

(3);

(4)FOR i := 1 TO k DO

BEGIN

(5) 在T中删除，并将与相邻的点的度数减1；

(6) 若的度数为1，则将加入;

END;

(7)若T中只剩一个顶点，或两个相邻的顶点和，则找到T的中心或，算法终止；

(8)否则，，返回步骤(2)。

8.给定一个无权的连通图G，设计一个O(MAX(n, e))的算法，求G中从顶点u到顶点v的最短路径。 (11分)

8.解：用宽度优先搜索来找顶点u和v的最短路。

算法：

PROCEDURE BFS (u, v, adjlist: 图的邻接表；VAR visit: 访问标志数组)；

VAR

queue : ARRAY [1max] OF integer;

prior : ARRAY [1max] OF integer;

j, hp, tp : integer;

p : 顶点指针；

BEGIN

FOR j := 1 TO max DO

BEGIN

visit[ j ] := false;

prior[ j ] := 0;

END;

hp := 1; tp := 1;

visit[ u ] := true;

queue[hp] := u;

REPEAT

p := adjlist[queue[hp]];

WHILE p NIL DO

BEGIN

j := p.vertex;

IF NOT visit[ j ] THEN

BEGIN

visit[ j ] := true;

prior[ j ] := queue[hp];

tp := tp + 1;

queue[tp] := j;

END;

p := p.link;

END;

hp := hp + 1;

UNTIL (hp > tp) OR visit[v];

j := v;

WRITE( j );

WHILE j u DO

BEGIN

j := prior[ j ];

WRITE( j );

END;

END;

9.写出二分查找的递归算法，并分析其最坏情况下的时间复杂性。

(11分)

9.解：二分查找的递归算法如下：

PROCEDURE Binary\_Search (l, h, x; VAR i);

VAR

m : integer;

BEGIN

IF l h THEN

BEGIN

m := (l + h) DIV 2;

IF x.key < F[m].key THEN

Binary\_Search(l, m1, x, i)

ELSE IF x.key > F[m].key THEN

Binary\_Search(m+1, h, x, i)

ELSE /\* x.key = F[m].key \*/

BEGIN

i := m;

l := h+1;

END;

END;

END;

主算法：Binary\_Search(1, n, x, i)，i的初值为0。

设为该算法最坏情况下的运行时间，则

设, 有。

。