数据结构中评价算法的两个重要指标是 时间复杂度和空间复杂度 。

数据的运算最常用的有5种，它们分别是 索引、插入、删除、修改和排序 。

对数据存储无外乎是 顺序存储、链式存储、索引存储、散列存储 等四种。

以下程序的功能是实现带附加头结点的单链表数据结点逆序连接，请填空完善之。

void reverse(\*head)

/\* head为附加头结点指针/

linklist \*p,\*q;

p=h->next; h->next=NULL;

While（p!=NULL)

{q=p; p=p->next; q->next=h->next; h->next=(2) q }

}

下列程序判断字符串s 是否对称，对称则返回1，否则返回0；如 f("abba")返回1，f("abab")返回0,完善之

int f (char \*s)

{ int i=0,j=0;

while (s[j]) ++j ;

for(j--; i<j && s[i]==s[j]; i++,j--);

Return i<j?0:1;

}

对广义表执行以下操作的结果：GetHead【GetTail【GetHead【((a,b),(c,d))】】】 b 。

已知单链表L，(\*p)结点即不是开始结点，也不是尾结点，(\*q)结点为（\*p）结点的直接前趋结点，则删除(\*p)结点的语句组为（共两句）： q->next=p->next;free(p) 。

分析以下程序段的时间复杂度为 o(log3n) 。

k=1;

While (k<=n)

k = k\*3;

假设循环队列设front和rear指示队头、队尾元素位置，maxsize为循环队列最大容量，则循环队列长度为： (maxsize+rear-front)%maxsize 。

计算一棵二叉树的高度，系统采用栈来完成。请将下列算法填空完整。

int Height(Btree \*t)

{ int h1,h2;

if(t==NULL)

return 0;

else if ( t->lchild==NULL&&t->rchild==NULL )

return 1;

else

{ h1= Heigh(t->lchild) ;

h2= Heigh(t->rchild) ;

Return (h1>h2?h1:h2)+1 ;

}

}

已知q是指向中序线索二叉树上某个结点的指针，本函数返回中序下\*q的后继的指针，请完善之

BiTree InSucc(BiTree q){

r=q->rchild;

while(!r->ltag) r=r->lchild;

Return r ;

}//ISucc

设t是给定的一棵二叉树，下面的递归程序count(t)用于求得:二叉树t中具有非空的左,右两个儿子的结点个数N2;只有非空左儿子的个数NL;只有非空右儿子的结点个数NR和叶子结点个数N0。N2、NL、NR、N0都是全局量，且在调用count(t)之前都置为0.

typedef struct node

{int data; struct node \*lchild,\*rchild;}node;

int N2,NL,NR,N0;

void count(node \*t)

{if (t->lchild!=NULL) if (t->rchild!=NULL) N2++; else NL++;

else if (t->rchild!=NULL) NR++; else N0++ ;

if(t->lchild!=NULL) count(t->lchild) ; if (t->rchild!=NULL) count(t->rchild) ;

}

设要将序列（Q, E, C, Y, P, A, M, B, R, D, Z, X）中的关键码按字母序的升序重新排列，

则：冒泡排序一趟扫描的结果是 E,C,Q,P,A,M,B,R,D,Y,X,Z ；

初始步长为4的希尔（shell）排序一趟的结果是 P,A,C,B,Q,D,M,X,R,E,Z,Y ；

二路归并排序一趟扫描的结果是 E,Q,C,Y,A,P,B,M,D,R,X,Z ；

快速排序一趟扫描的结果是(采用初始数据为支点，两头交替逼近的实现方法) D,E,C,B,P,A,M,Q,R,Y,Z,X ；

给定关键字序列11，36，10，1，3，9，19，13，21试分别用顺序查找、二分查找（假设已排序）、二叉排序树查找、散列查找(开地址，用线性探查法（模取11的HASH函数）和拉链法)来实现查找的平均查找长度ASL值分别是

顺序查找 5 ;二分查找 25/9 ;二叉排序树查找 28/9 ;线性探查法 16/9 ;拉链法 11/9 .