**2013科学硕士研究生入学考试参考答案**

**数据结构试题（75分）**

一、简单问题（每小题5分，共15分）

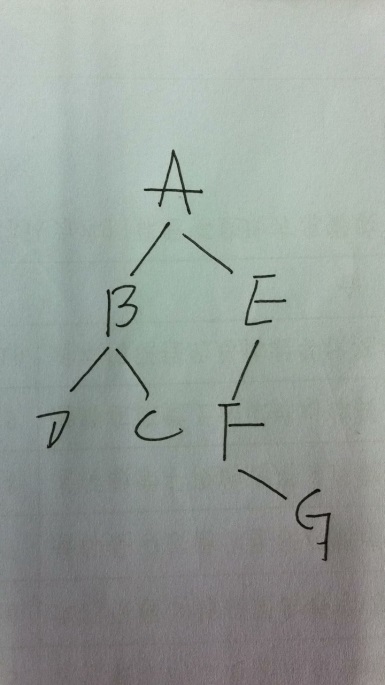
1．抽象数据类型定义了一个数据对象、该数据对象中各元素间的逻辑关系以及一组基本操作。抽象数据类型使我们可以忽略细节，而将精力放到解决问题本质上来。

2. 栈和队列中的元素都是线性结构，只是它们的操作位置受限制。其中栈的操作限制在一端进行，具有LIFO特性；队列的操作一端进行插入，一端进行删除，具有FIFO特性。栈和队列属于限定性线性表。

3. 由于图中的每个结点的前驱和后继的个数都不唯一，遍历过程中，需要设置访问标志数组，其作用有两个：防止重复访问；防止遗漏访问。

二、按要求写结果。

1.



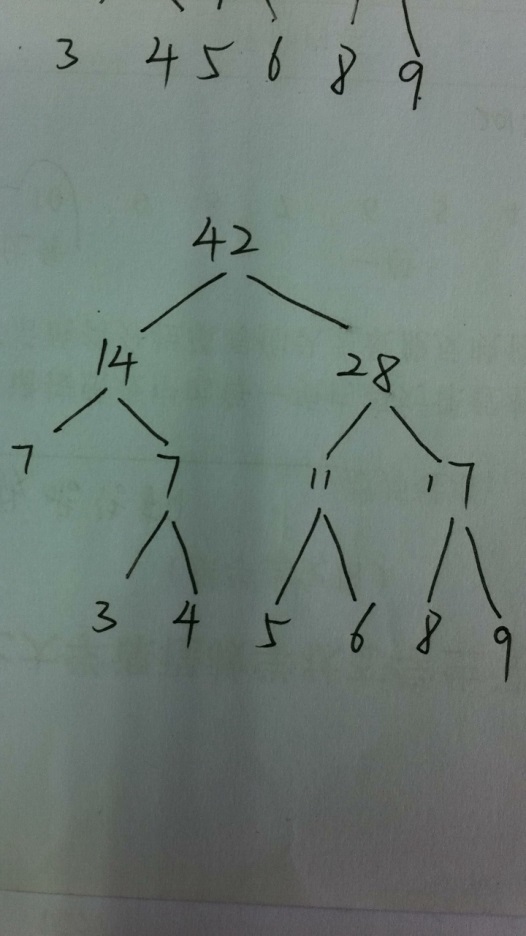
2.最好情况：元素杂乱无序，每次经中轴元素划分，将列表刚好划分为长度均匀的两个字表。 时间复杂度为O(nlogn)；最坏情况：待排序记录按关键字有序排列，此时，算法的时间复杂度为O(n2)。

3.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 下标 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 元素 | 27 |  |  | 52 | 88 | 75 | 33 | 41 | 12 | 66 |
| 比较次数 | ⑤ |  |  | ① | ① | ① | ② | ② | ④ | ⑦ |

ASLsucc=(1+1+1+2+2+4+5+7)/8=23/8 ASLunsucc=(2+1+1+9+8+7+6)/7=34/7

4.构建的哈夫曼树为：



WPL=（3+4+5+6+8+9）\*3+7\*2=119

三、编写程序。

#define N 10

int BinSrch(ElemType A[], KeyType key)//折半查找

{

int low,high,mid;

low=1;

high=l.length;/\*置区间初值\*/

while( low <= high)

{

mid=(low+high) / 2;

if (key==A[mid])

return (mid);/\*找到待查元素\*/

else

if (key<A[mid])

high=mid-1;/\*未找到，则继续在前半区间进行查找\*/

else

low=mid+1;/\*继续在后半区间进行查找\*/

}

return (0);

}

void main()

{

int i,k;

int A[N+1];

printf(“按递增顺序输入%d个元素”,N);

for(i=1;i<=N;i++)

scanf("%d",&A[i]);

printf("输入要查找的元素");

scanf("%d",&k);

i=BinSrch(A,k);

if(i<1) printf("输入的元素不在列表中");

else printf("输入的元素在列表的%d位置",i);

}

四、编写算法。

逐层输出，即需要用广度优先遍历完成。

int visited[MAX\_VERTEX\_NUM]={False};

void TraverseGraph(AdjList g)

{ for(vi=0;vi<g.vexnum;vi++)

if (!visited[vi])

{ BFS(g,vi);

printf("\n\n");

}

}

void BFS(AdjList \*G, int v0)

{ visit(v0); visited[v0]=True;

InitQueue(&Q); EnterQueue(&Q, v0);

while ( ! Empty(Q))

{

DeleteQueue(&Q, &v);

p=G->vertex[v].firsarc;

while(p!=NULL)

{

if(!visited[p->adjvex])

{

printf("(%c,%c) ",G->vertex[v0],G->vertex[v]);

visited[p->adjvex]=1;

EnterQueue(&Q,p->adjvex);

}

p=p->nextarc;

}

}

}

五、编写算法。

1.输入扩展先序，建立一棵二叉树。

void CreateBiTree(BiTree \*bt)

{

char ch;

ch = getchar();

if(ch=='.') \*bt=NULL;

else

{

\*bt=(BiTree)malloc(sizeof(BiTNode)); //生成一个新结点

(\*bt)->data=ch;

CreateBiTree(&((\*bt)->LChild)); //生成左子树

CreateBiTree(&((\*bt)->RChild)); //生成右子树

}

}

2. 可按中序遍历二叉树，如果发现有逆序元素，即不是二叉排序树。

BSTree pre=NULL;//指向刚刚被访问过的结点

int InOrder(BSTree bst)

{

if(bst==NULL) return 1;

else{

left=InOrder(bst->LChild);

if(pre!=NULL)

{

if(pre->key < bst->key) tag=1

else tag=0;

}

right=InOrder(bst->RChild);

return left && right && tag;

}

}