江苏大学

硕士研究生入学考试样题

科目代码： 851

科目名称： 数据结构　　满分： 150 分

一、单项选择题(每小题 1 分，共 10 分，下列每小题给出的四个选项中，只有一项符合题目要求)

1. 当输入非法错误时，一个“好”的算法能够进行适当的处理，而不会产生难以理解的输出结果。这称为算法的(_____)。

   (A) 正确性　(B) 可行性　(C) 健壮性　(D) 有穷性

2. 若线性表最常用的操作是存取第 i 个元素及其前驱和后继元素的值，为了节省时间应采用以下哪一种存储方式最合适？(_____)

   (A) 顺序表　(B) 单链表　(C) 单循环链表　(D) 双向循环链表

3. 为了增加内存空间的利用率和减少溢出的可能性，由两个栈共享一片连续的内存空间时，应将两栈的栈底分别设在这片内存空间的两端，这样，当(_____)时，才产生上溢。

   (A) 两个栈的栈顶同时到达栈空间的中心点

   (B) 其中一个栈的栈顶到达栈空间的中心点

   (C) 两个栈的栈顶在栈空间的某一位置相遇

   (D) 两个栈均不空，且一个栈的栈顶到达另一个栈的栈底

4. 深度为 h 的满 m 叉树的第 k 层有(_____)个结点($1 \le k \le h$)。

   (A) m^{k-1}　(B) m^k-1　(C) m^{h-1}　(D) m^h-1

5. 要连通具有 n 个顶点的有向图，至少需要(_____)条边。

   (A) n-l　(B) n　(C) n+l　(D) 2n

6. 折半查找的时间复杂度为(_____)

   (A) O（n^2）　(B) O（n）　(C) O（$n\log_2 n$）　(D) O（$\log_2 n$）

7. 对序列{25，19，17，18，30，-10，14，28}用希尔排序方法排序，经一趟后序列变为{25，-10，14，18，30，19，17，28}，则该次采用的增量是(_____)。

   (A) l　(B) 4　(C) 3　(D) 2

8. 删除堆顶元素重建堆的时间复杂度是(_____)。

   (A) $O(n\log_2 n)$　(B) O(n)　(C) $O(\log_2 n)$　(D) O(1)

9. 下列排序算法中，(_____)算法可能会出现下面情况：在最后一趟开始之前，所有元素都不在其最终的位置上。

   (A) 堆排序　(B) 冒泡排序　(C) 快速排序　(D) 插入排序

10. 在排序表的数据元素的关键字分布随机情况下，就排序算法所用的辅助空间大小而言，堆排序、快速排序、归并排序的关系是(_____)。

(A) 堆排序<快速排序<归并排序

(B) 堆排序<归并排序<快速排序

(C) 堆排序>归并排序>快速排序

(D) 堆排序>快速排序>归并排序

二、填空题(每小题 2 分，共 10 分)

1. 下面程序段中带下划线的语句的执行次数的数量级是_____。

   for(i=1;i<=n;i++) { j=1; while (j<i) {x=x+1; j=j+1;}}

2. 在一个长度为 n 的顺序表中的第 i 个元素（1≤i≤n）之前插入一个元素时，需向后移动_____个元素。

3. 设用下标从 0 到 maxsize-1 的数组存放循环队列中元素，分别以 rear 和 length 指示循环队列中队尾位置和队列中包含的元素个数，则队首元素位置应该为_____。

4. 二维数组 A 的每个元素是由 6 个字符组成的串，其行下标 i=0，1，…，8，列下标 j=1，2，…，9。若 A 按行先存储，元素 A[8，5]的起始地址与当 A 按列先存储时的元素 A[_____]的起始地址相同。设每个字符占一个字节。

5. 广义表的表尾是指除第一个元素之外，_____。

三、应用题(共 80 分)

1. (14 分)已知一棵树的孩子——兄弟存储如表 1 所示，data 代表数据，firstchild 代表指向第一个孩子的存储位置，nextsibling 代表指向下一个兄弟的存储位置，假设存储位置从 1 开始。要求：

（1） 画出该树。

（2） 写出该树的先序遍历序列和后序遍历序列。

（3） 画出该树对应的二叉树的后序后继线索二叉树。

表 1 树的孩子——兄弟存储

存储位置	1	2	3	4	5	6	7	8	9	10	11	12
data	W	A	B	C	D	E	F	G	H	I	J	K
firstchild	2	4	0	0	8	7	0	9	0	0	12	0
nextsibling	0	3	5	6	0	0	0	10	0	11	0	0

2. (5 分) 设要传输的一段电文是 aadcbadbabcdababad，请设计一种能唯一译码且编码长度最小的编码方案，以便对传输的电文进行加密传输。要求给出编码方案的具体设计过程和各字符的编码。

3. (8 分)已知一个有向图的顶点集 V 和边集 E 分别为：V={1，2，3，4，5，6，7}，E={<2，1>，<3，2>，<3，6>，<4，3>，<4，5>，<4，6>，<5，1>，<5，7>，<6，1>，<6，2>，<6，5>}。要求：

（1） 若采用邻接表存储该图，并且每个顶点邻接表中的弧结点都是按照邻接点序号从小到大的次序链接的，试给出基于这样存储的邻接表的拓扑有序序列。

（2） 仍然以（1）中的邻接表存储该图，请给出基于这样存储的邻接表的从顶点 4 出发的 DFS 序列以及 DFS 生成树。

4. (15 分)已知某有向网的逆邻接表如图 1 所示。要求：

（1） 画出该有向网。

（2） 用 dijkstra 算法，求从源点 V1 出发到其它各终点的最短路径以及长度，请给出

求最短路径以及长度的过程。

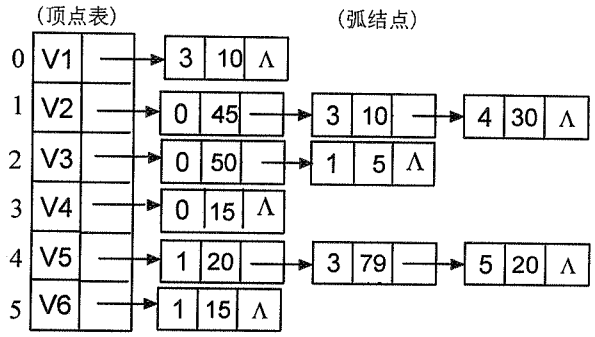

图 1　有向网的逆邻接表

5. (22 分) 已知整数数组 A 的 7 个元素为{26，33，35，29，19，12，22}，用以下排序方法进行由小到大排序，要求：

（1）　以第一个元素 26 为基准元素，写出第一趟快速排序过程以及结果；

（2）　要想不进行关键字的比较而完成排序，应选用什么样的排序方法，写出排序过程。

（3）　试给出折半插入排序的每一趟排序结果，并按发生的关键字比较的先后次序给出最后一趟的关键字的比较情况。

6. (3 分)请用加括号、删除括号的方法把中缀表达式(A + B) * D + E / (F + A * D) 转换为后缀表达式，请给出转换过程。

7. (13 分) 有一结点的关键字序列 F={10, 24, 32, 17, 31, 30, 46, 47, 40, 63, 49}，假设装填因子（负载系数）α=0.8，哈希函数为：H (K)=K % 13，K 为关键字，用线性探测法处理冲突，要求：

（1）　画出相应的哈希表。

（2）　求出等概率下查找成功时的平均查找长度 ASL（给出计算的详细式子）。

（3）　请描述在闭散列表中查找给定值的算法步骤（或算法流程）。

四、简答题(共 30 分)

1. （15 分）设 LA、LB 分别是两个带头结点的有序（从小到大）单链表的头指针，其类型定义为 linklist，单链表结点由数据域 data 和指针域 next 构成，数据域 data 为整型。又 LA 表中的元素值各不相同，LB 表中的元素值也各不相同。请仔细阅读如下算法，并回答问题：

（1）　该算法的功能是什么？

（2）　算法返回的 x，y，z 中值的含义是什么？

（3）　若执行 exam 前 LA 表中的结点值依次是：6，10，16，20，25，30，36，40，LB 表中结点值依次是：8，11，16，25，29，30，52，请问执行 exam 后，LA 表的结点值和 LB 表的结点值依次分别什么？

```
void exam(linklist   &LA, linklist   &LB, int &x, int &y, int &z)
{   /*      pa,pb,p,q 是指向单链表结点的指针      */
    pa=LA->next; LA->next=NULL;q=LB;pb=LB->next; x=0; y=0;z=0;
    while   (pa!=NULL && pb!=NULL)
    {
```

```
            if (pa->data<pb->data) {p=pa; pa=pa->next; delete p ; y=y+1;   z=z+1;}
            else if (pa->data>pb->data) {q=pb;   pb=pb->next; }
            else if (pa->data == pb->data)
                {
                    p=pa;   pa=pa->next;   p->next= LA->next;   LA->next=p;
                    q->next=pb->next;   delete pb;   pb= q->next;   x=x+1;   z=z+1;
                }
        }
        while  (  pa!=NULL  )  { p=pa; pa=pa->next; delete p; y=y+1;z=z+1; }
    }//exam
```

2. (7 分)如图 2 所示，已对主表建立了完全索引表和二级索引表，现要查找关键字为 3060 的记录的“其他信息”。请给出一种较优的查找方案，描述查找的过程以及发生的关键字比较情况。

二级索引表

关键字	地址
1100	1
2100	5
3100	9
4100	12

完全索引表

	关键字	地址
1	1010	12
2	1003	2
3	1050	5
4	1028	9
5	2071	3
6	2016	4
7	2034	11
8	2029	15
9	3017	6
10	3060	7
11	3046	14
12	4078	1
13	4005	13
14	4031	10
15	4024	8

主表

	关键字	其他信息
1	4078	
2	1003	
3	2071	
4	2016	
5	1050	
6	3017	
7	3060	
8	4024	
9	1028	
10	4031	
11	2034	
12	1010	
13	4005	
14	3046	
15	2029	

图 2　主表、完全索引表和二级索引表

3. (8 分)请描述在二叉排序树中删除一个数据元素的算法思想。

五、算法设计题(共 20 分)

1. (10 分) 有向图以邻接表为存储结构，试编写在有向图中插入一条弧<v1,v2,w>的算法，若该弧已经在邻接表中，则不插入。其中 v1 是弧尾顶点，v2 是弧头顶点，v1、v2 是顶点序号，w 是<v1,v2>弧上的权值（假设顶点的序号为顶点在顶点表 VertexesTable 数组中的下标，图中顶点从下标 0 开始存储）。
2. (10 分)以二叉链表为二叉树的存储结构，试利用栈的基本操作写出先序遍历的非递归算法。

注：（1）采用类 C 语言或 C 语言或 C++语言描述你的算法，关键之处请给出简要注释。

（2）算法中可使用下面给出的存储结构。若算法中还使用到其他的存储结构或运算，请给出其定义或实现。

```
//图的邻接表存储结构
const int MaxVertexes = 20;   //最大的顶点数
template <class vertexType, class arcType> class Graph;
template < class arcType> struct ArcNode {// 定义边(弧)结点
      friend class Graph <class vertexType, class arcType>;
      int adjvex;   //和边(或弧)相关联的另一个顶点序号
      arcType weight; //边(或弧)上的信息(权)
      ArcNode<arcType> *nextarc ; //指向下一条边(弧)结点的指针
      ArcNode( ) { }   //构造函数
      ArcNode( int v , arcType w ) : adjvex( v ) , weight( w ) , next( NULL ){ }   //构造函数
};
template < class arcType , class vertexType > struct VertexNode {
// 定义顶点结点
      friend class Graph <class vertexType, class arcType>;
      vertexType data; //顶点的信息
      ArcNode<arcType> *firstarc ; //指向依附该顶点的边(弧)链表
};
template <class vertexType, class arcType> Graph{//定义图
private:
      VertexNode < arcType , vertexType > * VertexesTable;//顶点表
      int CurrentNumVertexes; //当前的顶点数
      int CurrentNumArcs; //当前的边(或弧)数
public:
      Graph:CurrentNumVertexes (0),CurrentNumArcs(0){};//构造函数
      Graph ( vertexType v[ ] , int num = MaxVertexes ); //构造函数
      ~Graph ( );//析构函数
      int InsertArc( int v1, int v2, arcType w );
      //在有向图中插入一条从顶点 v1 到 v2 的弧，弧的权值是 w。插入成功返回 1，否则
返回 0
};
//二叉树的二叉链表存储结构
template <class Type> class BinaryTree;   //二叉链表类前视声明,以便使用友元
template <class Type> class BinTreeNode { //结点类
     friend class BinaryTree<Type>;
private:
     BinTreeNode<Type> *leftChild,   *rightChild;      //结点的左、右孩子指针域
     Type data;              //结点的数据域
public:
     BinTreeNode ( ) : leftChild (NULL), rightChild (NULL) { } //构造函数,构造一个空结
```

```
点
    BinTreeNode (Type d, BinTreeNode<Type> *lp = NULL, BinTreeNode<Type> *rp
=NULL ) : data (d),  leftChild (lp), rightChild (rp) { }//构造函数,构造一个数据域的值为 d
的结点
    Type GetData ( ) const { return data; }//取结点数据值
    BinTreeNode<Type> *GetLeftChild ( ) const { return leftChild; } //取结点的左孩子的
指针值
    BinTreeNode<Type> *GetRightChild ( ) const { return rightChild; }//取结点的右孩子
的指针值
};
template <class Type> class BinaryTree { //二叉链表类
private:
    BinTreeNode <Type> *root;    //二叉树根结点指针
public:
    BinaryTree ( ) : root (NULL) { }    //构造函数
    ~BinaryTree ( ) { destroy ( root ); }      //析构函数
};
//顺序栈的存储结构
template <class Type> class seqstack  {
private:
    int top;                 //栈顶指针
    type *stacka;           //数组名
    int maxsize;            //栈最大可容纳元素个数
public:
    seqstack( int size ); //构造函数
    ~seqstack( ) { delete [ ] stacka;} //析构函数
    void push (const Type & item); //元素 item 进栈
    Type pop(   );  //栈顶元素出栈，并返回栈顶元素
    Type gettop(   );  //读取栈顶元素，返回栈顶元素
    int empty (   ) const {return top = = -1;}//栈空返回 1,否则返回 0
    int full (   ) const {return top = = maxsize-1;}   //如果栈中元素个数等于 maxsize,则返
    回 1,否则返回 0
    void clear(   ) {top = -1;}   //清空栈
};
```