武汉科技大学

2019 年全国硕士研究生招生考试初试自命题试题

科目名称：数据结构（C 语言版）（☑A 卷□B 卷）科目代码：856

考试时间：3 小时　满分 150 分

可使用的常用工具：☑无　□计算器　□直尺　□圆规（请在使用工具前打√）

注意：所有答题内容必须写在答题纸上，写在试题或草稿纸上的一律无效；考完后试题随答题纸交回。

一、选择题(共 15 小题，每小题 2 分，共 30 分)

1. 计算算法的时间复杂度是属于一种（　　）的方法。

A）事前统计　B）事前分析估算　C）事后统计　D）事后分析估算

2. 数据的逻辑结构可以分为（　　）。

A）静态结构和动态结构　B）物理结构和存储结构

C）线性结构和非线性结构　D）虚拟结构和抽象结构

3. 线性表若采用链式存储结构时，要求内存中可用存储单元的地址（　　）。

A）必须是连续的　B）部分地址必须是连续的

C）一定是不连续的　D）连续不连续都可以

4. 线性表既可以用带头结点的链表表示，也可以用不带头结点的链表表示，前者最主要的好处是（　　）。

A）使空表和非空表的处理统一　B）可以加快对表的遍历

C）节省存储空间　D）可以提高存取表元素的速度

5. 若用一个大小为 6 的数组来实现循环队列，且当前 rear 和 front 的值分别为 0 和 3。当从队列中删除一个元素，再加入两个元素后，rear 和 front 的值分别为（　　）。

A）1 和 5　B）2 和 4　C）4 和 2　D）5 和 1

6. 对二叉树 T 中的某个结点 x，它在先根序列、中根序列、后根序列中的序号分别为 pre(x),in(x)、post(x),a 和 b 是 T 中的任意两个结点,下列选项一定错误的是(　　)。

A）a 是 b 的后代且 pre（a）<pre（b）　B）a 是 b 的祖先且 post（a）>post（b）

C）a 是 b 的后代且 in（a）<in（b）　D）a 在 b 的左边且 in（a）<in（b）

7. 若二叉树的前序序列和后序序列正好相反，则该二叉树一定是（　　）的二叉树。

A）空或只有一个结点　B）任一结点无左子树

C）任一结点无右子树　D）高度等于其结点数

8. 下面几个符号串编码集合中，不是前缀编码的是（　　）。

A）{0,10,110,1111}　B）{11,10,001,101,0001}

C）{00,010,0110,1000}　D）{b,c,aa,ac,aba,abb,abc}

准考证号码：　密封线内不要写题　报考专业：　姓名：

9. 一个 n 个顶点的连通无向图,其边数至少为（　　）。

A）n-1　　B）n　　C）n+1　　D）n*logn

10. 下面（　　）方法可以判断出一个有向图中是否有环（回路）?

A）深度优先遍历　　B）求最短路径　　C）拓朴排序　　D）求关键路径

11. 下列关于无向连通图特性的叙述中，正确的是（　　）。

（1）所有顶点的度数之和为偶数。

（2）边数比顶点个数减 1 要大。

（3）至少有 1 个顶点的度为 1。

A）只有（1）　　B）只有（2）　　C）（1）和（2）　　D）（1）和（3）

12. 静态查找表与动态查找表二者的根本差别在于（　　）。

A）它们的逻辑结构不一样　　B）施加在其上的操作不同

C）包含的数据元素的类型不一样　　D）存储实现不一样

13. 设有100个结点,用二分法查找时,最大比较次数是（　　）。

A）25　　B）50　　C）10　　D）7

14. 对初始数据序列{8，3，9，11，2，1，4，7，5，10，6}进行希尔排序。若第一趟排序结果为{1，3，7，5，2，6，4，9，11，10，8}，第二趟排序结果为{1，2，6，4，3，7，5，8，11，10，9}，则两趟排序采用的增量分别是（　　）。

A）3，1　　B）3，2　　C）5，2　　D）5，3

15. 下列排序算法中，（　　）算法可能会出现下面情况：初始数据有序时，花费时间反而更多。

A）堆排序　　B）冒泡排序　　C）快速排序　　D）希尔排序

二、填空题(共 10 小题，每小题 2 分，共 20 分)

1. 将两个各有 n 个元素的有序表归并成一个有序表，其最少比较次数是（　　）次。
2. 在无表头结点的单链表 L 的表头插入 s 结点的语句序列是（　　）。
3. 循环队列存储在数组 A[0..m]中，尾指针为 rear，则数据元素 x 入队时,首先将 x 放到队尾所在位置，然后队尾后移，其中队尾后移的操作语句为（　　）。
4. 由 5 个结点可以构造出（　　）种不同的树。
5. 已知一棵完全二叉树的第 6 层(设根为第 1 层)有 8 个叶结点,则完全二叉树的结点个数最多是（　　）。
6. 设森林 F 中有 3 棵树，三棵树的结点个数依次是 n1，n2 和 n3，则与森林 F 相对应二叉树的根结点的右子树上的结点个数是（　　）个。
7. 有 n 个结点，e 条边的无向图采取邻接表存储，求最小生成树的 Kruskal 算法的时间复杂度是（　　）。
8. 已知一无向图 G=（V，E），其中 V={a,b,c,d,e } E={(a,b),(a,d),(a,c),(d,c),(b,e)} 现用某一种图遍历方法从顶点 a 开始遍历图，得到的序列为 abecd，则采用的是（　　）遍历方法。

9. 有序表包含 16 个数据，顺序组织。若采用二分查找方法，则在等概率情况下，查找成功时的 ASL 值是（　　）。

10. 一组记录的排序码为（46，79，56，38，40，84），则利用堆排序的方法建立的初始堆（大顶堆）中第 2，3，4 个排序码分别为（　　）。

三、判断题(对的答√错的答×，共 10 小题，每小题 2 分，共 20 分)

1. 数据元素是数据的最小单位。

2. 一般认为，随问题规模 n 的增大，算法执行时间的增长速度较快的算法最优。

3. 在一个长度为 n 的线性表的第 i 个位置（1≤i≤n+1）插入一个元素时，需要向后移动 n+1-i 个元素。

4. 用链接方式存储的队列，在进行出队运算时仅需修改头指针。

5. 对于任何一颗二叉树，如果其叶子结点数为 n0，则度为 2 的结点数为 n0－1。

6. 如果给定二叉树的先序遍历序列和后序遍历序列，则该二叉树是唯一的。

7. 一颗有 n 个顶点的生成树有且仅有 n－1 条边。

8. 对 AOV 网进行拓扑排序时，结束后如果还有顶点没有被输出，且这些顶点的入度均＞0，则该网必定有环存在。

9. 采用线性探测法处理冲突，在查找成功的情况下，可能要探测的这些位置上的关键字一定都是同义词。

10. 堆排序不是稳定的排序方法。

四、综合应用题(共 5 小题，每小题各 8 分，共 40 分)

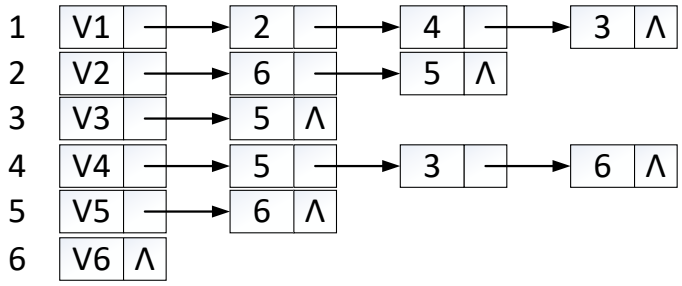
1. 将如下图所示矩阵的非零元素逐行存放于数组 B 中（下标从 0 开始存放，即 A_{11} 存放在 B[0]中），使得 B[k] =A[i，j]，求：

（1）用 i，j 表示 k 的变换公式。

（2）用 k 表示 i，j 的变换公式。

$$\begin{bmatrix} a_{1,1} & a_{1,2} & & & & \\ a_{2,1} & a_{2,2} & & & 0 & \\ & & a_{3,3} & a_{3,4} & & \\ & & a_{4,3} & a_{4,4} & & \\ & 0 & & \cdots\cdots & & \\ & & & & a_{2m-1,2m-1} & a_{2m-1,2m-1} \\ & & & & a_{2m,2m-1} & a_{2m,2m} \end{bmatrix}$$

2. 已知 AOV 网的邻接表如下图所示，要求：



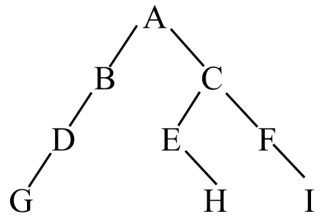
（1）画出该 AOV 网。

（2）给出基于该 AOV 网邻接表的从顶点 V1 出发的 DFS 序列。

（3）给出基于该 AOV 网邻接表的从顶点 V1 出发的 BFS 序列。

（4）写出对该 AOV 网按照上述邻接表进行拓扑排序得到的拓扑序列。

3. 根据下列二叉树，要求：



（1）写出其先序遍历序列、中序遍历序列和后序遍历序列。

（2）顺序存储该二叉树，画出其存储示意图。

4. 如果一颗非空 k 叉树 T（k≥2）中每个非叶子结点都有 k 个孩子。请回答下列问题并给出推导过程。

（1）若 T 有 m 个非叶子结点，则 T 的叶子节点有多少个？

（2）若 T 的高度为 h，则 T 的结点数最多是多少个？最少是多少个？

5. 散列表长度是 13，散列函数 H(K)=k % 13，解决冲突用线性探测再散列法。给定的关键字序列为{19,14,23,1,68,20,84,27,55,11,10,79}，要求：

（1）画出哈希表。

（2）求出等概率下查找成功的平均查找长度。

（3）求出等概率下查找失败的平均查找长度。

五、算法设计题(共 4 小题，每小题 10 分，共 40 分)

1. 用带头结点的双向循环链表（结点结构为（Left，Data，Right））表示的线性表为 L=（a1,a2, …an)。试设计如下算法 Fun 实现将 L 改造成：L=（a1,a3,…an,…a4,a2)。

```
void Fun(DbLinkList &L);
```

2. 若表达式以字符形式已存入数组 s 中，‘#’为表达式的结束符，试设计如下算法 Match 判断表达式中括号（‘([{})’）是否配对，如果配对，则返回 1，否则返回 0。

```
int Match(char *s);
```

3. 树采用孩子兄弟链表作为存储结构（结点结构 CSTree 为（firstchild，data，nextsibling)），试设计如下非递归算法 Leaf 计算树的叶子节点数。

```
int Leaf(CSTree *T); //函数返回值就是树的叶子节点数
```

4. 已知无向图采用邻接表存储方式，试设计如下算法 DeletEdge 删除图中(i,j)边。

```
void DeletEdge(AdjList g,int i,int j);
```

武汉科技大学

2019 年全国硕士研究生招生考试初试自命题试题答案

科目名称：数据结构（C 语言版）（☑A 卷□B 卷）科目代码：856

考试时间：3 小时　　满分 150 分

可使用的常用工具：☑无　□计算器　□直尺　□圆规（请在使用工具前打√）

注意：所有答题内容必须写在答题纸上，写在试题或草稿纸上的一律无效；考完后试题随答题纸交回。

一、选择题(共 15 小题，每小题 2 分，共 30 分)

BCDAB ADBAC ABDDC

二、填空题(共 10 小题，每小题 2 分，共 20 分)

1. n
2. s->next=L; L=s;
3. rear=(rear+1)%(m+1)
4. 9
5. 111
6. n2+n3
7. O(eloge)
8. 深度优先
9. 54/16
10. 79，56，38

三、判断题(对的答√错的答×，共 10 小题，每小题 2 分，共 20 分)

××√×√　　×√√××√

四、综合应用题(共 5 小题，每小题各 8 分，共 40 分)

1.

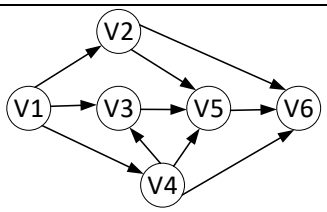
（1）(4 分) k=2(i-1)+(j+1)%2

（2）(2 分) i=k/2+1

(2 分) j=k/2+k%2+1-k/2/2

2.

（1）(2 分)AOV 网



（2）(2 分)DFS 序列：V1，V2，V6，V5，V4，V3

（3）(2 分)BFS 序列：V1，V2，V4，V3，V6，V5

（4）(2 分)拓扑序列：V1，V2，V4，V3，V5，V6

3.

（1） (1 分)先序：ABDGCEHFI

(1 分)中序：GDBAEHCFI

(1 分)后序：GDBHEIFCA

（2） (5 分)顺序存储示意图

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	B	C	D	^	E	F	G	^	^	^	^	H	^	I

4.

（1）(4 分)m(k-1)+1

因为 T 中只存在度为 0 和 k 的结点。

N=n0+nk=B+1=k*nk+1----→ n0=(k-1)nk+1 (nk 就是 m)

（2）(2 分)最多：$(k^h-1)/(k-1)$

除第 h 层外，第 1 到 h-1 层的每个结点的度都是 k，即满 k 叉树。

$N=k^0+k^1+k^2+\cdots+k^{h-1}=(k^h-1)/(k-1)$

(2 分)最少：k(h-1)+1

除第 1 层外，每层都有 k 个结点，其中 1 个分支节点和 k-1 个叶子

即：N=（h-1）k+1

5.

（1）(4 分)画出哈希表

0	1	2	3	4	5	6	7	8	9	10	11	12
	14	1	68	27	55	19	20	84	79	23	11	10
	1	2	1	4	3	1	1	3	9	1	1	3

（2）(2 分)成功时的平均查找长度：

（1+2+1+4+3+1+1+3+9+1+1+3）/8=30/12=5/2

（3）(2 分)失败时的平均查找长度

（1+2+3+4+5+6+7+8+9+10+11+12+13）/13=91/13=7

五、算法设计题(共 4 小题，每小题 10 分，共 40 分)

1.

```
void Fun(DbLinkList &L)
{
    Tail=L->Left;
    p=L->Right;
    i=1;
```

```
    while(p&&p!=Tail)
    {
        if(i%2==0)
        {   q=p;   //删除结点 p
            p=p->next;
            p->Left=q->Left;
            q->Left->Right=p;
            q->Right=Tail->Right;  //插入到 Tail 之后
            q->Left=Tail;
            Tail->Right->Left=q;
            T->Right=q;
        }
        else p=p->Right;
        i++;
    }
}
```

2.

```
int Match(char *s)
{
     char s[maxsize];
     int top=0,i=0;
     while(s[i]!=' #' )
     {
        switch(s[i])
        {
            case  '( ':
            case  '[ ':
            case  '{ ': s[top++]=s[i]; i++;  break;  //入栈
            case  ')' : if(s[top-1]==' ( ') {  top--;  i++;  break; }
                        else { printf("不匹配"); return 0;  }
            case  ']' : if(s[top-1]==' [ ') {  top--;  i++;  break; }
                        else { printf("不匹配"); return 0;  }
            case  '}' : if(s[top-1]==' { ') {  top--;  i++;  break; }
                        else { printf("不匹配"); return 0;  }
            case  '#' : if(top==0) { printf("匹配成功"); return 1; }
                        else { printf("不匹配"); return 0;  }
            default: i++;  //读入其它字符，不作处理
        }
     }
}
```

3.

```
int Leaf(CSTree *T)
{
    if(T==NULL) return 0;
    int front=1,rear=1;  //front,rear 是队头队尾元素的指针
```

```
    int last=1;          //last 指向树中同层结点中最后一个结点
    int temp=0;          //叶子结点数
    Q[rear]=T;           //Q 是以树中结点为元素的队列
    while(front<=last)
    {
        p=Q[front++];      //队头出队列
        if(p->firstchild==NULL) temp++;
        while(p!=NULL)       //层次遍历
       {
           if(p->firstchild) Q[++rear]=p->firstchild; //第一子女入队
           p=p->nextsibling; //同层兄弟指针后移
        }
        if(front>last){ last=rear; } //本层结束,last 移到指向下层最右结点处
     }
     return temp;
}
```

4.

```
void DeletEdge(AdjList g,int i,int j)
{
     p=g[i].firstarc;
     pre=NULL; //删顶点 i 的边结点(i,j),pre 是前驱指针
     while (p)
     {
        if(p->adjvex==j)
        {   if(pre==NULL)  g[i].firstarc=p->next;
            else   pre->next=p->next;
            free(p);
            break; //释放结点空间，退出循环
         }
        else {  pre=p;   p=p->next;    }   //沿链表继续查找
     }
     p=g[j].firstarc;
     pre=NULL;    //删顶点 j 的边结点(j,i)
     while (p)
     {
         if (p->adjvex==i)
         {   if(pre==NULL)  g[j].firstarc=p->next;
             else    pre->next=p->next;
             free(p);
             break;  //释放结点空间，退出循环
         }
         else {   pre=p;   p=p->next;   }   //沿链表继续查找
     }
}
```