

上海交通大学试卷 (A 卷)

(2018 至 2019 学年 第 一 学期)

班级号 _____ 学号 _____
课程名称 _____ 数据结构与算法 _____

姓名 _____
成绩 _____

一、选择题 (每题 1.5 分, 共 24 分)

1. 设入栈顺序为 A, C, D, E, B, 则出栈顺序不可能是_____。
A. ACDEB B. AEBDC C. DCEAB D. EBDAC
2. 若以一个大小为 8 的数组来实现循环队列, 当前的 rear 和 front 的位置分别为 2 和 6, 当依次执行一次出队、三次入队和一次出队后, 当前 rear 和 front 的位置分别是_____。
A. 5, 0 B. 4, 1 C. 7, 0 D. 7, 4
3. 图的深度优先搜索类似于树的_____次序遍历。
A. 前序 B. 中序 C. 后序 D. 层次
4. 为了实现图的广度优先遍历, BFS 算法使用的一个辅助数据结构是_____。
A. 栈 B. 队列 C. 二叉树 D. 树
5. 设二叉树的先序遍历序列和后序遍历序列正好相反, 则该二叉树满足的条件是_____。
A. 任一结点只有一个儿子 B. 空或只有一个结点
C. 任一结点无右儿子 D. 任一结点无左儿子
6. 对一组数据(13, 69, 85, 47, 8, 30, 58, 22, 73, 21)进行升序排序, 如果采用堆排序方法, 建立的初始堆为_____。
A. (8, 13, 30, 22, 21, 85, 58, 47, 73, 69)
B. (8, 13, 30, 21, 22, 58, 85, 47, 73, 69)
C. (85, 73, 58, 69, 21, 30, 13, 22, 47, 8)
D. (85, 73, 58, 69, 21, 30, 13, 47, 22, 8)
7. 在一个长度为 n 的顺序表的任意位置插入一个新元素的时间复杂度为_____。
A. $O(n^2)$ B. $O(1)$ C. $O(\log n)$ D. $O(n)$
8. 已知一个线性表中最常用的操作是删除第一个元素和在最后一个元素之后插入一个元素, 则采用_____存储方式最节省运算时间。
A. 仅有头指针的单循环链表 B. 双链表
C. 仅有尾指针的单循环链表 D. 单链表

我承诺，我将严格遵守考试纪律。

承诺人：_____

题号										
得分										
批阅人(流水阅卷教师签名处)										

9. 分别按照下列序列构造二叉查找树，与用其他三个序列构造结果不同的是_____。
- A. (100, 83, 96, 61, 121, 119, 134) B. (100, 121, 119, 134, 83, 61, 96)
- C. (100, 61, 83, 96, 121, 119, 134) D. (100, 83, 61, 96, 121, 134, 119)
10. 对线性表进行二分查找时，要求线性表必须_____。
- A. 以顺序方式存储 B. 以顺序方式存储，且数据元素有序
- C. 以链接方式存储 D. 以链接方式存储，且数据元素有序
11. 比较次数与排序的初始状态无关的排序方法是_____。
- A. 冒泡排序 B. 直接插入排序 C. 快速排序 D. 直接选择排序
12. 设有关键字初始序列{Q, H, C, Y, A, M, S, R, D, F, X}，新序列{F, H, C, D, P, A, M, Q, R, S, Y, X}是采用_____方法对初始序列进行第一趟扫描的结果。
- A. 直接插入排序 B. 以第一元素为标准元素的快速排序
- C. 二路归并排序 D. 冒泡排序
13. 设一组初始记录关键字序列为{ 26, 48, 17, 76, 36, 79, 21, 43, 37, 71 }，其中含有 5 个长度为 2 的有序子表，则用归并排序的方法对该记录关键字序列进行一趟归并后的结果为_____。
- A. (17, 26, 48, 76, 21, 36, 43, 79, 37, 71)
- B. (26, 48, 36, 79, 17, 21, 43, 79, 37, 71)
- C. (17, 26, 48, 76, 21, 36, 37, 43, 71, 79)
- D. (17, 26, 36, 48, 76, 79, 21, 43, 37, 71)
14. 设某哈希表的长度为 150，哈希函数 $H(k) = k \% P$ ，则 P 通常情况下最好选择_____。
- A. 141 B. 139 C. 151 D. 149
15. 后缀表达式 $4\ 7\ 2\ 3\ *\ -\ *\ 8\ 2\ /\ -$ 的结果为_____。
- A. 0 B. 1 C. -2 D. 7
16. 下列说法正确的是_____。
- A. 求最短路径的 Dijkstra 算法中，边的权值可以为负
- B. Floyd 算法中，边的权值可以为负
- C. 深度优先搜索算法不可用于判断有向图中是否存在回路
- D. Dijkstra 算法不允许图中有回路

二、填空题（每空 1.5 分，共 24 分）

1. 设顺序线性表中有 n 个数据元素，则第 i 个位置上插入一个数据元素需要移动表中_____个数据元素；删除第 i 个位置上的数据元素需要移动表中_____个数据元素。
2. 设一组初始记录关键字序列为(49, 38, 65, 97, 76, 13, 27, 50)，则以 $d = 3$ 为增量的一趟希尔排序结束后的结果为_____。
3. 设查找表中有 100 个元素，如果用二分查找方法查找数据元素 x ，则最多需要比较_____次就可以断定数据元素 x 是否在查找表中。
4. 设某二叉树中度为 0 的结点个数为 N_0 ，度为 1 的结点个数为 N_1 ，则该二叉树总的结点总个数为_____。
5. 设有向图 G 的顶点集合为 $V = \{a, b, c, d, e\}$ ，边集合为 $\{ \langle a, b \rangle, \langle c, b \rangle, \langle a, c \rangle, \langle b, d \rangle, \langle c, e \rangle, \langle d, e \rangle \}$ ，那么它的一种拓扑排序为_____。
6. 假设某的电文由 8 个字母组成，其频率分别为 0.07, 0.19, 0.02, 0.06, 0.32, 0.03, 0.21, 0.10，为这 8 个字母设计哈夫曼编码，其中编码长度最大的字母的编码是_____位。
7. 在一棵 m 阶的 B+ 树中，非叶子的树根结点的子树数目最少为_____，非树根结点的分支结点的子树数目最少为_____。
8. 假设有 n 个关键字，它们具有相同的 Hash 函数值，用线性探测方法解决冲突，把这 n 个关键字存入散列表中，至少需要做_____次探测操作。
9. 设有一待排序的初始序列(49,38,65,97,76,13,27,50)，则第 4 趟直接选择排序结束后的结果为_____。
10. 设有向图 G 的存储结构用邻接矩阵 A 表示，则 A 中第 i 行中所有非零元素个数之和等于顶点 i 的_____，第 i 列中所有非零元素个数之和等于顶点 i 的_____。

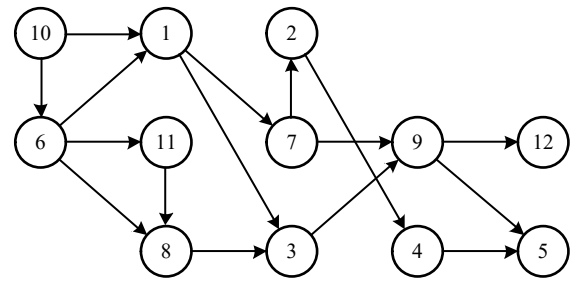
11. 下面程序段用于实现非加权图的最短路径查找，请在下划线处填上正确的语句。

```
template < class TypeOfVer, class TypeOfEdge >
void adjListGraph < TypeOfVer, TypeOfEdge > : : unweightedShortDistance
    (TypeOfVer start, TypeOfEdge noEdge ) const
{
    LinkQueue < int > q;
    TypeOfEdge * distance = new TypeOfEdge[Vers];
    int * prev = new int [Vers];
    int u, sNo;
    edgeNode * p;

    for (int i = 0; i < Vers; ++i ) distance[i] = noEdge;           //初始化
    //寻找起始结点的编号
    for ( sNo = 0; sNo < Vers; ++ sNo )  if ( _____ ) break;
    if ( sNo == Vers ) {
        cout << " 起始结点不存在" << endl;
        return;
    }
    //寻找最短路径
    distance[sNo] = 0;
    prev[sNo] = sNo;
    q. enqueue(sNo);
    while ( ! q. isEmpty( ) ) {
        u = q. dequeue( );
        for ( p = verList[u]. head; p != NULL; p = p->next )
            if ( distance[p->end] == noEdge ) {
                _____
                _____
                q. enqueue( p->end );
            }
    }
}
```

三、简答题（每题 8 分，共 24 分）

- 按照数字从小到大的顺序（从 1 开始），写出该有向图的深度优先搜索和广度优先搜索序列结果，并分别画出所得到的生成树/森林。



- 一颗二叉树的前序、中序和后序遍历的序列如下，其中有些部分未标出，请根据现有信息画出该二叉树。
 前序遍历序列：____，____，C，D，E，____，G，H，I，____，K
 中序遍历序列：C，B，____，____，F，A，____，J，K，I，G
 后序遍历序列：____，E，F，D，B，____，J，I，H，____，A

3. 根据散列函数 $H(\text{key}) = \text{key} \bmod 7$ ，将关键字 7, 4, 16, 5, 6, 18, 13, 2, 8, 9, 12, 3 依次插入到一个规模为 7 的开散列表中，且每次插入元素时插入到单链表的表头。请画出该散列表的存储结构。

四、代码分析题（每题 10 分，共 20 分）

1. 下列程序实现了一种对二叉查找树的操作

```
struct TreeNode {
    int val;
    struct TreeNode *left;
    struct TreeNode *right;
    TreeNode(int x) :
        val(x), left(NULL), right(NULL) {
    }
};

class Solution {
public:
    TreeNode* Function(DataNode* pRoot, unsigned int k)
    {
        if ( pRoot == NULL )
            return NULL;
        TreeNode* ObjNode = NULL;
        Function ( pRoot, &ObjNode, k );
        return ObjNode;
    }
    void Function ( TreeNode* pRoot, TreeNode** ObjNode, unsigned int &k )
    {
        if ( pRoot->left != NULL )
            Function ( pRoot->left, ObjNode, k );
        if ( *ObjNode == NULL )
        {
            if ( k == 1 )
            {
                *ObjNode = pRoot;
            }
            k--;
        }
        if ( *ObjNode == NULL && pRoot->right != NULL )
            Function ( pRoot->right, ObjNode, k );
    }
}
```

请回答以下问题：

- 1) 说明该段程序实现的功能（6 分）
- 2) 写出该操作的时间复杂度（4 分）

2. 下面是某算法的代码实现。

```
template < class TypeOfVer, class TypeOfEdge >
void adjListGraph < TypeOfVer, TypeOfEdge > :: topSort() const
{
    linkQueue < int > q;
    edgeNode * p;
    int current;
    int * inDegree = new int [Vers];
    for ( int i = 0; i < Vers; ++i )    inDegree[i] = 0;
    for ( i = 0; i < Vers; ++i ) {
        for ( p = verList[i]. head; p != NULL; p = p -> next )
            ++ inDegree[p -> end];
    }
    for ( i = 0; i < Vers; ++ i )
        if ( inDegree[i] == 0 )    q. enqueue(i);
    cout << " The result is: " << endl;
    while ( ! q. isEmpty( ) ) {
        current = q. dequeue( );
        cout << verList[current]. ver << '\t';
        for ( p = verList[current]. head; p != NULL; p = p -> next )
            if ( ++ inDegree[p -> next] == 0 )    q. enqueue( p -> next );
    }
    cout << endl;
}
```

请回答以下问题：

- 1) 结合上述代码，分析算法的原理以及实现的功能。（6 分）
- 2) 分析该算法的时间复杂度。（4 分）

五、程序题（共 8 分）

奇偶交换排序算法的思路为：第一趟对所有奇数的 i ，将 $a[i]$ 和 $a[i+1]$ 进行比较，第二趟对所有偶数的 i ，将 $a[i]$ 和 $a[i+1]$ 进行比较，每次比较时若 $a[i] > a[i+1]$ ，将二者交换，以后重复上述两趟过程，直至整个数组有序。

请按照上述思路，完成算法的代码。

```
void oesort ( int *a, int length ) {
```

```
}
```