

上海交通大学试卷(B卷)

(2011至2012学年第一学期)

班级号_____ 学号_____ 姓名_____
课程名称 《数据结构(A类)》 成绩_____

一、单项选择题(每格1.5分,共30分)

1. 当一棵有n个结点的二叉树按层次从上到下,同层次从左到右将数据存放在一维数组a[1..n]中时,数组中第i个结点的左孩子为()。
A. a[2i] (当 $2i \leq n$) B. a[i/2]
C. a[2i+1] (当 $2i+1 \leq n$) D. 无法确定
2. 散列文件使用散列函数将记录的关键字值计算转化为记录的存放地址,因为散列函数是一对一的关系,则选择好的()方法是散列文件的关键。
A. 散列函数 B. 散列函数和冲突处理
C. 冲突处理 D. 除余法中的质数
3. 将长度为n的单链表链接在长度为m的单链表之后的算法的时间复杂度为()。
A. O(1) B. O(n) C. O(m) D. O(m+n)
4. 若某线性表最常用的操作是存取任一指定序号的元素和在最后进行插入和删除运算,则利用()存储方式最节省时间。
A. 顺序表 B. 双链表 C. 单循环链表 D. 带头结点的双循环链表
5. 一个栈的输入序列为12345,则下列序列中不可能是栈的输出序列的是()。
A. 23415 B. 54321 C. 23145 D. 15423
6. 有关平衡二叉树的说法()是不对的。
A. 左右子树高度之差绝对值不超过1 B. 左右子树都是平衡二叉树
C. 有时左子树略高,有时右子树略高 D. 平衡二叉树是一个完全二叉树
7. 下列排序算法中,其中()是稳定的。
A. 插入排序、希尔排序与起泡排序 B. 归并排序、起泡排序与插入排序
C. 选择排序、归并排序与堆排序 D. 插入排序、快速排序与堆排序

我承诺，我将严格遵守考试纪律。

承诺人：_____

题号	一	二	三	四	五	总分
得分						
批阅人(流水阅) 卷教师签名处)						

8. 无向图 $G = (V, E)$, 对该图进行深度优先遍历, 得到的顶点序列正确的是 ()。其中 $V = \{a, b, c, d, e, f\}$, $E = \{(a, b), (a, e), (a, c), (b, e), (c, f), (f, d), (e, d)\}$ 。
A. a, b, e, c, d, f B. a, c, f, e, b, d C. a, e, b, c, f, d D. a, e, d, f, c, b

9. 程序段:

```
for (i = n-1; i >= 0; i--)  
    for (j = 1; j <= i; j++)  
        if (a[j] > a[j+1]) {  
            temp = a[j];  
            a[j] = a[j+1];  
            a[j+1] = temp;  
        }
```

其中 n 为正整数, 则语句 $temp = a[j]$ 在最坏情况下执行频度是 ()。

- A. $O(n)$ B. $O(n \log n)$ C. $O(n^3)$ D. $O(n^2)$

10. 下面描述不正确的是 ()。

- 1) 求从指定源点到其余各顶点的 Dijkstra 最短路径算法中弧上权不能为负的原因是在实际应用中无意义。
2) 利用 Dijkstra 求每一对不同顶点之间的最短路径的算法时间是 $O(n^3)$; (图用邻接矩阵表示)
3) Floyd 求每对不同顶点对的算法中允许弧上的权为负, 但不能有权和为负的回路。
A. (1), (3) B. (1) C. (1), (2), (3) D. (2), (3)

11. 已知关键序列 5, 8, 12, 19, 28, 20, 15, 22 是小根堆 (最小化堆), 插入关键字 3, 调整后得到的小根堆是 ()。

- A. 3, 5, 12, 8, 28, 20, 15, 22, 19 B. 3, 5, 12, 19, 20, 15, 22, 8, 28
C. 3, 8, 12, 5, 20, 15, 22, 28, 19 D. 3, 12, 5, 8, 28, 20, 15, 22, 19

12. 已知某二叉树的后序遍历序列是 dabec, 中序遍历序列是 debac, 它的前序遍历是 ()。

- A. acbed B. decab C. deabc D. cedba

13. 如果图用邻接表结构存储，则常见操作的算法时间复杂度（ ）。
- A. 仅与顶点的个数有关 B. 仅与边的条数有关
C. 与顶点的个数、边的条数都可能有关 D. 与两者都无关
14. 在一棵 m 阶的 B+ 树中，每个非叶结点的儿子数 S 应满足（ ）。
- A. $\left\lfloor \frac{m+1}{2} \right\rfloor \leq S \leq m$ B. $\left\lfloor \frac{m}{2} \right\rfloor \leq S \leq m$
C. $1 \leq S \leq \left\lfloor \frac{m}{2} \right\rfloor$ D. $1 \leq S \leq \left\lfloor \frac{m+1}{2} \right\rfloor$
15. 已知有向图 $G = (V, E)$ ， G 的拓扑序列是（ ）。其中 $V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$ ， $E = \{\langle v_1, v_2 \rangle, \langle v_1, v_3 \rangle, \langle v_1, v_4 \rangle, \langle v_2, v_5 \rangle, \langle v_2, v_6 \rangle, \langle v_3, v_5 \rangle, \langle v_3, v_6 \rangle, \langle v_4, v_6 \rangle, \langle v_5, v_7 \rangle, \langle v_6, v_7 \rangle\}$ 。
- A. $v_1, v_3, v_4, v_6, v_2, v_5, v_7$ B. $v_1, v_3, v_2, v_6, v_4, v_5, v_7$
C. $v_1, v_3, v_4, v_5, v_2, v_6, v_7$ D. $v_1, v_2, v_5, v_3, v_4, v_6, v_7$
16. 在含有 n 个关键字的最小化堆中，关键字最大的记录有可能存储在（ ）位置上。
- A. $\lfloor n/2 \rfloor$ B. $\lfloor n/2 \rfloor + 2$ C. 1 D. $\lfloor n/2 \rfloor - 1$
17. 利用孩子兄弟链表存储树，则根结点的右指针是（ ）。
- A. 指向最左孩子 B. 指向最右孩子 C. 空 D. 不确定
18. 当采用分块查找时，数据的组织方式为（ ）。
- A. 数据分成若干块，每块内数据有序
B. 数据分成若干块，每块内数据不必有序，但块间必须有序，每块内最大（或最小）的数据组成索引块
C. 数据分成若干块，每块内数据有序，但块间必须有序，每块内最大（或最小）的数据组成索引块
D. 数据分成若干块，每块（除最后一块外）中数据个数需相同
19. 下面给出的四种排序方法中，排序过程中的比较次数与元素初始排列顺序无关的是（ ）。
- A. 插入排序法 B. 选择排序法 C. 快速排序法 D. 堆排序法
20. 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

二. 程序填充题 (每题 1.5 分, 共 30 分)

1. 在一个双向链表中, 定义 fence 指针指向下一个插入的位置, next 和 prev 分别表示一个元素向前与向后的指针, 用以下代码实现插入一个新的结点。

```
fence->next = new Link<Element> (item, _____, _____);  
if (_____ != NULL)  
    _____ = fence->next;
```

2. 请在下列算法的横线上填入适当的语句。

```
// 以 hA 和 hB 为头指针的单链表分别表示有序表 A 和 B,  
// 本算法判别表 A 是否包含在表 B 内, 若是, 则返回“true”, 否则返回“false”,  
// 要求: 为递归算法实现  
bool inclusion(node *hA, node *hB) {  
    pA = hA->next; pB = hB->next;  
    if (pA == NULL) _____;  
    while (_____)  
        if (pA->data == pB->data) _____  
        else _____;  
        _____;  
    }  
}
```

3. 已给如下关于单链表的类型说明:

```
struct node {  
    int data;  
    node *next;  
};
```

以下程序采用链表合并的方法, 将两个已排序的单链表合并成一个链表而不改变其排序性(升序), 这里两链表的头指针分别为 p 和 q.

```
void mergeLink(node *p, node *q) {  
    node *h, *r;  
    h = new node;  
    h->next = NULL; r = h;  
    while ((p != NULL) && (q != NULL)) {  
        if (p->data <= q->data) {  
            _____; r = p; p = p->next;  
        }  
        else {  
            _____; r = q; q = q->next;  
        }  
    }  
}
```

```

if (p == NULL) r->next = q;
_____;
p = h->next; delete(h);
}

```

4. 下面是对不带头结点的单链表 L 进行就地逆置的算法，试在空缺处填入适当的语句。

```

void reverse(linkList &L) {
    p = NULL; q = L;
    while (q != NULL) {
        _____;
        q->next = p; p = q;
        _____;
    }
    _____;
}

```

5. 将二叉树 bt 中每一个结点的左右子树互换的算法如下，其中 enQueue(Q, bt)，deQueue(Q)，isEmpty(Q) 分别为进队，出队和判别队列是否为空的函数，请填写算法中得空白处，完成其功能。

```

struct btNode {
    int data;
    struct btNode *leftChild, *rightChild;};
void exchange(btNode *bt) {
    btNode *p, *q;
    if (bt) {
        enQueue(Q, bt);
        while (!isEmpty(Q)) {
            p = deQueue(Q); q = _____;
            p->rightChild = _____;
            _____ = q;
            if (p->leftChild) _____;
            if (p->rightChild) _____;
        }
    }
}

```

三. 计算简答题 (15 分)

1. 已知一组关键字 $T = (12, 2, 16, 30, 8, 28, 4, 10, 20, 6, 18)$, 执行下列算法从小到大进行排序, 请分别给出下列排序算法第一趟和第二趟排序结束时的结果:

- 1) 希尔排序 (第一趟排序的增量为 5, 第二趟排序的增量为 2)。(5 分)
- 2) 快速排序 (选第一个记录为界点)。(5 分)

2. 已知字母使用频率 (见下表), 求这 8 个字符的 Huffman 编码值。要求写出 Huffman 树的建立过程, 约定在子树合并时, 根的权值小的子树为左子树, 大的为右子树。

字母	C1	C2	C3	C4	C5	C6	C7	C8
使用频率 (%)	5	25	3	6	9	12	36	4

四. 程序分析题 (10 分)

请分析下列 dijkstra 算法中带“//”标记的 5 个 for 语句及 1 个赋值语句加以注释，说明其功能，并分析算法的时间复杂度和空间复杂度。

图的抽象类及邻接表类的定义如下：

```
template <class TypeOfEdge>
class graph {
public:
    virtual bool insert(int u, int v, TypeOfEdge w) = 0;
    virtual bool remove(int u, int v) = 0;
    virtual bool exist(int u, int v) const = 0;
    virtual numOfVer() const {return vers;}
    virtual numOfEdge() const {return edges;}
protected:
    int vers, edges;
};

template <class TypeOfVer, class TypeOfEdge>
class adjListGraph:public graph<TypeOfEdge> {
public:
    adjListGraph(int vSize, const TypeOfVer d[]);
    bool insert(int u, int v, TypeOfEdge w);
    bool remove(int u, int v);
    bool exist(int u, int v) const;
    ~adjListGraph();
private:
    struct edgeNode { // 邻接表中存储边的结点类
        int end; // 终点编号
        TypeOfEdge weight; // 边的权值
        edgeNode *next;
    };
    struct verNode{ // 保存顶点的数据元素类型
        TypeOfVer ver; // 顶点值
        edgeNode *head; // 对应的单链表的头指针
        verNode(edgeNode *h = NULL) {head = h;}
    };
    verNode *verList;
};
```

dijkstra 算法的实现如下：

```
template <class TypeOfVer, class TypeOfEdge>
void adjListGraph <TypeOfVer, TypeOfEdge>::dijkstra(TypeOfVer start,
                                                     TypeOfEdge noEdge) const {
    TypeOfEdge *distance = new TypeOfEdge[vers];
    int *prev = new int[vers];
    bool *known = new bool[vers];
    int u, sNo, i, j;
    edgeNode *p;
    TypeOfEdge min;
    for (i = 0; i < vers; ++i) { // 
        known[i] = false;
        distance[i] = noEdge;
    }
    for (sNo = 0; sNo < vers; ++sNo) // 
        if (verList[sNo].ver == start) break;
    if (sNo == vers){
        cout << "起始结点不存在" << endl;
        return;
    }
    distance[sNo] = 0;
    prev[sNo] = sNo;
    for (i = 1; i < vers; ++i) {
        min = noEdge;
        for (j = 0; j < vers; ++j) // 
            if (!known[j] && distance[j] < min) {
                min = distance[j];
                u = j;
            }
        known[u] = true; // 
        for (p = verList[u].head; p != NULL; p = p->next) // 
            if (!known[p->end] && distance[p->end] > min+p->weight) {
                distance[p->end] = min+p->weight;
                prev[p->end] = u;
            }
    }
    for (i = 0; i < vers; ++i) // 
```

```
    cout << "从" << start << "到" << verList[i].ver << "的路径为: " << endl;
    printPath (sNo, i, prev);
    cout << "\n 长度为: " << distance[i] << endl;
}
}
```

五. 程序设计题 (15 分)

假设以带头结点的循环单链表存储一个优先队列，并且只设一个指向队尾元素的指针，没有头指针，元素入队时总是插入在队尾。优先队列类定义如下：

```
template <class Type>
class Queue <Type> {
    struct node {
        Type data;
        int priority;
        node *next;
        node(){};
        node(Type x, node *n = NULL): data(x), next(n){}
    };
    node *rear;
    Queue();
    bool isEmpty();
    void enQueue(Type x);
    Type deQueue();
    ~Queue();
}
```

请编写相应的出队程序，并注释之。

```
template <class Type>
Type Queue <Type>::deQueue()
```