

北京大学信息科学技术学院试卷

考试科目： 数据结构与算法 B 姓名： _____ 学号： _____

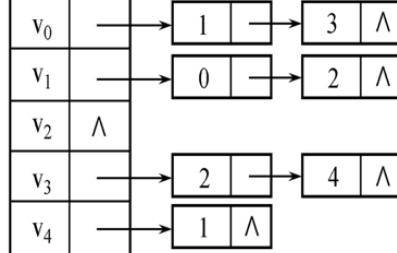
任课教师： _____ 考试时间： 2022 年 6 月 21 日（下午）

请将答案撰写在答题纸上！

校内考试结束后，请将试卷和答题纸一并上交！

一. 选择题（30 分，每小题 2 分）

1. 双向链表中的每个结点有两个指针域，llink 和 rlink，分别指向当前结点的前驱与后继，设 p 指向链表中的一个结点，q 指向一待插入结点，现要求在 p 前插入 q，则正确的插入操作为（ ）。
A: $p \rightarrow llink = q; q \rightarrow rlink = p; p \rightarrow llink \rightarrow rlink = q; q \rightarrow llink = p \rightarrow llink;$
B: $q \rightarrow llink = p \rightarrow llink; p \rightarrow llink \rightarrow rlink = q; q \rightarrow rlink = p; p \rightarrow llink = q \rightarrow rlink;$
C: $q \rightarrow rlink = p; p \rightarrow rlink = q; p \rightarrow llink \rightarrow rlink = q; q \rightarrow rlink = p;$
D: $p \rightarrow llink \rightarrow rlink = q; q \rightarrow rlink = p; q \rightarrow llink = p \rightarrow llink; p \rightarrow llink = q;$
2. 给定一个 N 个相异元素构成的有序数列，设计一个递归算法实现数列的二分查找，考察递归过程中栈的使用情况，请问这样一个递归工作栈的最小容量应为（ ）。
A: N B: $N/2$ C: $\lceil \log_2^N \rceil$ D: $\lceil \log_2^{(N+1)} \rceil$
3. 数据结构有三个基本要素：逻辑结构、存储结构以及基于结构定义的行为（运算）。下列概念中（ ）属于存储结构。
A: 线性表 B: 链表 C: 字符串 D: 二叉树
4. 为了实现一个循环队列（或称环形队列），采用数组 Q[0..m-1]作为存储结构，其中变量 rear 表示这个循环队列中队尾元素的实际位置，添加结点时按 $rear = (rear + 1) \bmod m$ 进行指针移动，变量 length 表示当前队列中的元素个数，请问这个循环队列的队列首位元素的实际位置是（ ）。
A: $rear - length$ B: $(1 + rear + m - length) \bmod m$
C: $(rear - length + m) \bmod m$ D: $m - length$
5. 给定一个二叉树，若其前序遍历序列与中序遍历序列相同，则该二叉树是（ ）。
A: 根结点无左子树的二叉树
B: 根结点无右子树的二叉树
C: 只有根结点的二叉树或非叶子结点只有左子树的二叉树
D: 只有根结点的二叉树或非叶子结点只有右子树的二叉树

6. 用 Huffman 算法构造一个最优二叉编码树，待编码的字符权值分别为{3, 4, 5, 6, 8, 9, 11, 12}，
请问该最优二叉编码树的带权外部路径长度为（ ）。(补充说明：树的带权外部路径长度定义为树中所有叶子结点的带权路径长度之和，结点的带权路径长度定义为该结点到树根之间的路径
长度与该结点权值的乘积)
- A: 58 B: 169 C: 72 D: 182
7. 假设需要对存储开销 1GB (GigaBytes)的数据进行排序，但主存储器（内存）当前可用的存储空间只有 100MB (MegaBytes)。针对这种情况，（ ）排序算法是最适合的。
- A: 堆排序 B: 归并排序 C: 快速排序 D: 插入排序
8. 已知一个无向图 G 含有 18 条边，其中度数为 4 的顶点个数为 3，度数为 3 的顶点个数为 4，其他顶点的度数均小于 3，请问图 G 所含的顶点个数至少是（ ）。
- A: 10 B: 11 C: 13 D: 15
9. 给定一个无向图 G，从顶点 V0 出发进行无向图 G 的深度优先遍历，访问的边集合为：{(V0,V1),
(V0,V4), (V1,V2), (V1,V3), (V4,V5), (V5,V6)}，则下面哪条边（ ）不能出现在 G 中？
- A: (V0, V2) B: (V4, V6)
C: (V4, V3) D: (V0, V6)
10. 已知一个有向图 G 的邻接表（或称逆邻接表）如下图所示，从顶点 v0 出发对该图 G 进行
深度优先周游，得到的深度优先周游结点序列为（ ）。
- A: V₀V₁V₄V₃V₂ B: V₀V₁V₂V₃V₄ C: V₀V₁V₃V₂V₄ D: V₀V₂V₁V₃V₄
- 
11. 若按照排序的稳定性和不稳定性对排序算法进行分类，则（ ）是不稳定排序。
- A: 冒泡排序 B: 归并排序 C: 直接插入排序 D: 希尔排序
12. 若算法的时间复杂度用大 O 表示法表示，以下（ ）分组中的两个排序算法在最坏情况下的时间复杂度相同。
- A: 快速排序和堆排序 B: 归并排序和插入排序
C: 快速排序和选择排序 D: 堆排序和冒泡排序
13. 设结点 X 和 Y 是二叉树中任意的两个结点，在该二叉树的先根周游遍历序列中 X 出现在 Y 之前，而在其后根周游序列中 X 出现在 Y 之后，则 X 和 Y 的关系是（ ）。
- A: X 是 Y 的左兄弟 B: X 是 Y 的右兄弟
C: X 是 Y 的祖先 D: X 是 Y 的后裔

14. 考虑一个森林 F，其中每个结点的子结点个数均不超过 2。如果森林 F 中叶子结点的总个数为 L，度数为 2 结点（子结点个数为 2）的总个数为 N，那么当前森林 F 中树的个数为（ ）。

- A: L-N-1 B: 无法确定 C: L-N D: N-L

15. 回溯法是一类广泛使用的算法，以下叙述中不正确的是（ ）。

- A: 回溯法可以系统地搜索一个问题的所有解或者任意解
B: 回溯法是一种既具备系统性又具备跳跃性的搜索算法
C: 回溯算法需要借助队列数据结构来保存从根结点到当前扩展结点的路径
D: 回溯算法在生成解空间的任一结点时，先判断当前结点是否可能包含问题的有效解，如果肯定不包含，则跳过对该结点为根的子树的搜索，逐层向祖先结点回溯

二. 判断 (10 分, 每小题 1 分; 对填写“Y”，错填写“N”)

1. () 对任意一个连通的、无环的无向图，从图中移除任何一条边得到的图均不连通。
2. () 给定一棵二叉树，前序周游序列和中序周游序列分别是 HGEDBFCA 和 EGBDHFAC 时，其后序周游序列必是 EBDGACFH。
3. () 假设一棵二叉搜索树的结点数值在 1 到 1000 之间，现在查找数值为 363 的结点。以下三个序列皆有可能是查过的序列：A). 2, 252, 401, 398, 330, 344, 397, 363; B). 925, 202, 911, 240, 912, 245, 363; C). 935, 278, 347, 621, 299, 392, 358, 363。
4. () 构建一个含 N 个结点的（二叉）最小值堆，建堆的时间复杂度大 O 表示为 $O(N \log_2 N)$ 。
5. () 队列是动态集合，其定义的出队列操作所移除的元素总是在集合中存在时间最长的元素。
6. () 任一有向图的拓扑序列既可以通过深度优先搜索求解，也可以通过宽度优先搜索求解。
7. () 对任一连通无向图 G，其中 E 是权值最小的边，那么 E 必然属于任何一个最小生成树。
8. () 对一个包含负权值边的图，迪杰斯特拉(Dijkstra)算法能够给出最短路径问题的正确答案。
9. () 分治算法通常将原问题分解为几个规模较小但类似于原问题的子问题，并要求算法实现写成某种递归形式，递归地求解这些子问题，然后再合并这些子问题的解来建立原问题的解。
10. () 动态规划算法把大问题分解为若干较小的子问题进行求解，而且子问题可能相互包含。

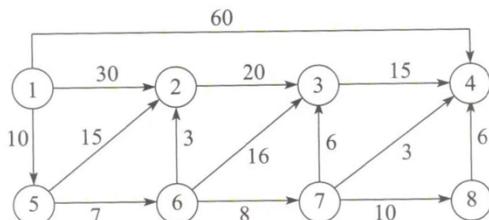
三. 填空 (20 分, 每题 2 分)

1. 目标串长是 n，模式串长是 m，朴素模式匹配算法思想为：从目标串第一个字符开始，依次与模式串字符匹配；若匹配失败，则尝试匹配的目标串起始字符位置往后移一位，重新开始依次和模式串字符匹配；……；直到匹配成功或遍历完整个目标串为止。则该算法中字符的最多比较次数是_____（使用大 O 表示法）。
2. 在一棵含有 n 个结点的树中，只有度（树结点的度指子结点数量）为 k 的分支结点和度为 0 的终端（叶子）结点，则该树中含有的终端（叶子）结点的数目为：_____。
3. 对一组记录进行非递减排序，其关键码为 [46, 70, 56, 38, 40, 80]，则利用快速排序的方法，以第一个记录为基准得到的第一次划分结果为_____。

4. 对长度为 3 的顺序表进行查找，若查找第一个元素的概率为 $1/2$ ，查找第二个元素的概率为 $1/4$ ，查找第三个元素的概率为 $1/8$ ，则执行任意查找需要比较元素的平均个数为_____。
5. 设有一组记录的关键字为 {19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79}，用链地址法（拉链法）构造散列表，散列函数为 $H(key) = key \text{ MOD } 13$ ，散列地址为 1 的链中有_____个记录。
6. 删一个长度为 n 的顺序表的第 i 个数据元素需要移动表中的_____个数据元素。 $(1 \leq i \leq n)$
7. 已知以数组表示的小根堆为 [8, 15, 10, 21, 34, 16, 12]，删除关键字 8 之后需要重新建堆，在此过程中，关键字的比较次数是_____。
8. 在广度优先遍历、拓扑排序、求最短路径三种算法中，可以判断出一个有向图是否有环（回路）的是_____。
9. 有 n ($n \geq 2$) 个顶点的有向强连通图最少有_____条边。
10. 若栈 S1 中保存整数，栈 S2 中保存运算符，函数 F() 依次执行下述各步操作：
 ①从 S1 中依次弹出两个操作数 a 和 b (先弹出 a ，再弹出 b);
 ②从 S2 中弹出一个运算符 op ;
 ③执行相应的运算 $b op a$;
 ④将运算结果压入 S1 中。
 假定 S1 中的操作数依次是 5, 8, 3, 2 (2 在栈顶)，S2 中的运算符依次是 *, -, / (/ 在栈顶)。调用三次 F() 后，S1 栈顶保存的值是_____。

四. 简答 (3 题, 共 20 分)

1. (7 分) 试用 Dijkstra 算法求出下图中顶点 1 到其余各顶点的最短路径，写出算法执行过程中各步的状态，填入下表。



所选顶点	U (已确定最短路径的顶点集合)	T (未确定最短路径的顶点集合)	顶点 1 到其他顶点的最短路径长度						
			2	3	4	5	6	7	8
初态	{1}	{2, 3, 4, 5, 6, 7, 8}	30	∞	60	10	∞	∞	∞

2. (6 分) 给定一组记录的关键码集合为{18, 73, 5, 10, 68, 99, 27}, 回答下列 3 个问题:
- 画出按照记录顺序构建形成的二叉排序(搜索)树(2分);
 - 画出删除关键码为 73 后的二叉排序树(2分)。
 - 画出令原关键码集合(未删除 73 前)查询效率最高的最优二叉排序树(仅需考虑关键码查询成功时的效率,且集合内每个关键码被查询概率相等)(2分)。
3. (7 分) 奇偶交换排序如下所述: 对于原始记录序列 { $a_1, a_2, a_3, \dots, a_n$ }, 第一趟对所有奇数 i , 将 a_i 和 a_{i+1} 进行比较, 若 $a_i > a_{i+1}$, 则将二者交换; 第二趟对所有偶数 i ; 第三趟对所有奇数 i ; 第四趟对所有偶数 i, \dots , 依次类推直到整个记录序列有序为止。伪代码如下:

```

ExSort (a[1..n]) // a[1..n] 为待排序记录, n 为记录数目;
{
    change1 = TRUE, change2 = TRUE;      // 标志变量, bool 型;
    if (n <= 0) return Error;
    while (change1 || change2)
    {
        change1 = FALSE;           // 奇数, ⇔ 为交换运算
        for (i = 1; i < n; i += 2)
            if (a[i] > a[i+1]) { a[i] ⇔ a[i+1]; change1 = TRUE; }
        if (!change1 && !change2) break;
        change2 = FALSE;           // 偶数, ⇔ 为交换运算
        for (i = 2; i < n; i += 2)
            if (a[i] > a[i+1]) { a[i] ⇔ a[i+1]; change2 = TRUE; }
    }
}

```

- 请写出序列{18, 73, 5, 10, 68, 99, 27, 10}在前 4 趟排序中每趟排序后的结果(2分)。
- 奇偶交换排序是否是稳定的排序(1分)。
- 在序列为初始状态为“正序”和“逆序”两种情况下, 试给出序列长度为 n 的情况下, 排序过程所需进行的关键码比较次数和记录的交换次数(4分)。

五. 算法填空(2题, 共 20 分)

1. 填空完成下列程序: 读入一个整数序列, 用单链表存储之, 然后将该单链表颠倒后输出该单链表内容。算法输入的第一行是整数 n, 第二行是 n 个整数, 即要存入单链表的整数序列。

样例输入

5

1 2 3 4 5

样例输出

5 4 3 2 1

```
#include <stdio.h>
#include <malloc.h>
```

```

struct Node
{
    int data;
    struct Node * next;
};

void printList (struct Node * head)
{
    struct Node * p = head;
    while (p) { printf("%d ",p->data); p = p->next; }
    printf("\n");
}

struct Node * reverseList(struct Node * head)
{
    if (head)
    {
        struct Node * p = head->next;
        head->next = _____①_____ ; // (1 分)
        while (p)
        {
            struct Node * q = p;
            p = _____②_____ ; // (1 分)
            q->next = _____③_____ ; // (1 分)
            _____④_____ ; // (2 分)
        }
    }
    return head;
}

int main()
{
    int n;
    scanf("%d",&n);
    //下面读入数据到链表
    struct Node * head = (struct Node *) malloc (sizeof (struct Node));
    head->next = _____⑤_____ ; // (1 分)
    scanf("%d",& (head->data));
    struct Node * p = head;
    for(int i = 1;i < n; ++i)
    {
        p->next = _____⑥_____ ; // (2 分)
        _____⑦_____ ; // (2 分)
        p->next = NULL;
        scanf("%d",& (p->data));
    }
}

```

```

//上面读入数据到单链表，表头指针 head
head = reverseList(head); //颠倒链表
printList(head);
return 0;
}

```

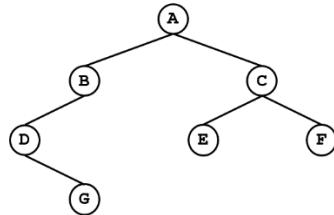
2. 填空完成下列程序：输入一棵二叉树的扩充二叉树的先根周游（前序遍历）序列，构建该二叉树，并输出它的中根周游（中序遍历）序列。这里定义一棵扩充二叉树是指将原二叉树中的所有空指针增加一个表示为@的叶结点。譬如下图所示的一棵二叉树，

输入样例：

ABD@G@@@CE@@@F@@

输出样例：

DGBAECF



```

/* 二叉树的定义 */
struct BinTreeNode;
typedef struct BinTreeNode *PBinTreeNode;
typedef PBinTreeNode *PBinTree;
struct BinTreeNode
{
    char info;
    PBinTreeNode llink;
    PBinTreeNode rlink;
};

/*递归创建从根开始的二叉树*/
PBinTreeNode creatRest_BTree()
{
    PBinTreeNode pbnode;
    char ch;
    scanf("%c", &ch);
    if(ch == '@') _____①_____;// (2 分)
    else
    {
        pbnode = (PBinTreeNode)malloc( sizeof (struct BinTreeNode) );
        if(_____②_____) // (2 分)
        {
            printf("Out of space!\n");
            return pbnode ;
        }
        pbnode->info=ch;
        _____③_____ ; // (2 分) 构造左子树
        _____④_____ ; // (2 分) 构造右子树
    }
    return pbnode;
}

```

```
}

/* 二叉树中根周游 */
void inOrder( PBinTreeNode t )
{
    if (t==NULL)  return;
    inOrder(_____⑤_____); // (1 分)
    printf(" %c ", t->info);
    inOrder(_____⑥_____ ); // (1 分)
}

/*创建完整的二叉树并输出中根序列*/
int main()
{
    PBinTree pbtree;
    pbtree= (PBinTree) malloc ( sizeof ( PBinTreeNode ) );
    if (pbtree==NULL)  return (0);
    *pbtree=creatRest_BTree(); /*递归创建从根开始的二叉树*/

    inOrder(*pbtree);
    return(1);
}
```