人工智能学院本科生 2020—2021 学年第 1 学期
《数据结构基础》课程期末考试试卷（A 卷）

专业：_____ 年级：_____ 学号：_____ 姓名：_____ 成绩:

得 分

一 、填空题（本题共 20 分，每空 1 分）

1. 数据结构中，与所使用的计算机无关的是数据的_____。
2. 设计算法时，需要考虑的目标有（至少举出三个）_____、_____、_____等。
3. 使用线性结构、树结构及图结构分别研究数据元素之间_____关系、_____关系和_____关系。
4. 对长度为 N 的数组中进行顺序查找,在最坏情况下所需要的比较次数为_____。
5. 在队列操作中，删除操作在_____进行，插入操作在_____进行。
6. 对于静态顺序队列，设指向队列头的指标是 front，指向队列尾的指标叫做 rear，则判断该队列为空的条件是_____。
7. 由权值分别为 8,3,4,6,5,5 的叶子结点生成一棵赫夫曼树，它的带权路径长度为_____。 树的层数*叶子节点值
8. 已知二叉树后序遍历序列是 dabec，中序遍历序列是 debac，它的前序遍历序列是_____。
9. 在一个无向图中，所有顶点的度数之和等于所有边数的_____倍。在一个具有 n 个顶点的完全无向图中，包含有_____ 条边，在一个具有 n 个顶点的有向完全图中，包含有_____条边。
10. 在深度为 5 的满二叉树中,叶子结点的个数为_____个。

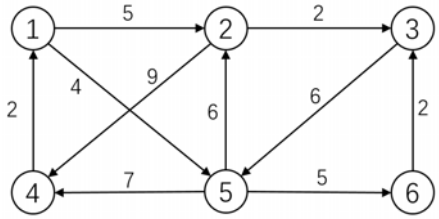
8.后序遍历，最后的是根，即c；中序遍历，根（c）左边是左子树，右边是右子树，因此deba都是左子树，e是左子树的根，再看中序序列，e左边的d是左子树，右边的ba是右子树，后序序列找到b，是右子树的根，再看中序序列，a在b右边，a是b的右子树。思路就这样

```
    c
  e
d   b
      a
```


第八题核心：后序看根（最后一位是根），中序分左右（左右子树）

11. 使用迪杰斯特拉（Dijkstra）算法求下图中从顶点 1 到其他个顶点的最短路径，依次得到的各最短路径的目标顶点是_____。

画*号的顺序



12. Prim 算法适用于求点稀疏边稠密图的最小生成树；Kruskal 适用于求_____图的最小生成树。

得 分

二 、简答题（本题共 20 分，每题 4 分）

1. 请解释以下名词。

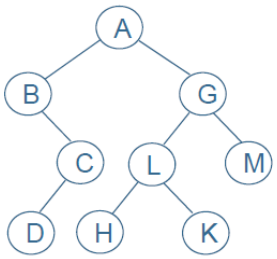
队列

栈

完全二叉树

生成森林

2. 将下面的二叉树转换成森林。



3. 画出下列广义表的图形表示。

D(A(c), B(e), C(a, L(b, c, d)))

原子是正方形

4. 什么是二叉树的线索化？并简要说明如何实现。

5. n 个顶点的无向图的邻接表最多有几个表结点。如果 n=6，当有几条边时能确保它是一个连通图。

n（n-1）

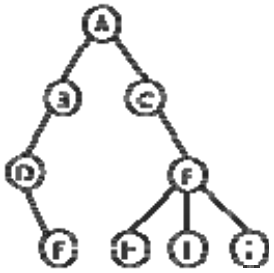
得 分

三 、应用题（本题共 30 分，每题 6 分）

1. 已知模式串 t='abcaabbcabcaabdab'，请简述 KMP 算法，并依据算法计算出 next 数组。

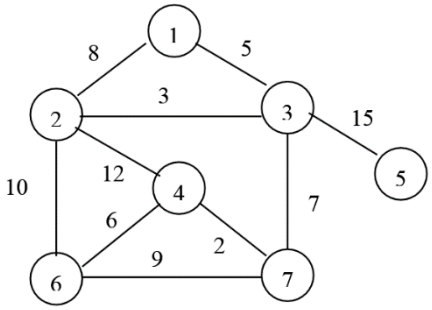
2. 某系统在通信联络中只可能出现 6 种字符{a,b,c,d,e,f}，对应的权重集合是{8, 3, 4, 6, 5, 5}
试设计赫夫曼编码

3. 画出下面这棵树的双亲表示存储结构



Prim

4. 将给定的图简化为最小的生成树，要求从顶点 1 出发。写出结果，并描述你所采用的算法过程。



5. 设一棵二叉排序树的先序遍历序列为 25，16，23，48，35，40，36，79，72，82，请画出该二叉排序树，并简要描述思路。

得 分

四、算法题（本题共 30 分，每题 6 分）

1. 假设循环单链表不为空，且既无表头结点也无表头指针，指针 p 指向链表中某结点。请设计一个算法，将 p 所指结点的前驱结点变为 p 所指结点的后继结点。

```
Function RelocatePredecessor(p):
// 找q 是 p 的前驱，pre_q 是 q 的前驱
    q = p
    While q.next != p:
        pre_q = q       // 记录 q 的前一个
        q = q.next      // q 向后移

      // 原始状态: pre_q -> q -> p -> p.next

    // 2. 开始变形
    pre_q.next = p      // 让前前驱直接指向 p
    q.next = p.next     // 让 q 接管 p 原来的后继
    p.next = q          // 让 p 指向 q

    // 最终状态: pre_q -> p -> q -> p.next
```

2. 在一棵以二叉链表表示的二叉树上，试写出用按层次顺序遍历二叉树的方法，统计树中具有度为 1 的结点数据的算法。首先描述算法思想，之后写出具体算法代码，关键之处给出注释。

```
Function CountDegreeOne(Tree root):
    // 1. 处理空树的情况
    If root == NULL:
        Return 0

    // 2. 初始化变量
    Queue Q                 // 定义一个队列 Q
    Integer count = 0       // 计数器初始化
    Node p                  // 临时指针

    Enqueue(Q, root)        // 根节点入队

    // 3. 开始层次遍历
    While (Q is NOT empty):

        p = Dequeue(Q)      // 队头元素出队

        // 【关键】判断度是否为 1：(有左无右) OR (无左有右
        If ( (p.lchild != NULL AND p.rchild == NULL) OR
             (p.lchild == NULL AND p.rchild != NULL) ):
            count = count + 1

        // 4. 将下一层节点入队
        If p.lchild != NULL:
            Enqueue(Q, p.lchild)   // 左孩子入队
        If p.rchild != NULL:
            Enqueue(Q, p.rchild)   // 右孩子入队

    End While

    Return count
```

3. 编写一个函数实现非递归折半查找（二分查找）算法。

4. 请简述如何利用栈来实现队列，并给出相应伪代码。注：如需要，可用多个栈。

```
Class MyQueue:
    // 定义两个栈
    Stack StackIn    // 输入栈
    Stack StackOut   // 输出栈

    // 1. 入队操作
    Function Enqueue(element):
        StackIn.Push(element)  // 直接放入输入栈

    // 2. 出队操作
    Function Dequeue():
          If StackOut.IsEmpty():

           // 如果输入栈也没数据，说明队列是空的
           If StackIn.IsEmpty():
              Return Error("Queue is empty")

              While StackIn is NOT Empty:
              temp = StackIn.Pop()
              StackOut.Push(temp)

        // 现在输出栈顶就是最早进来的元素
        Return StackOut.Pop()

    // 3. 判断队列是否为空
    Function IsEmpty():
        // 两个栈都空，队列才算空
        Return StackIn.IsEmpty() AND StackOut.Is
```

5. 班上有 N 名学生。其中有些人是朋友，有些则不是。他们的“友谊关系”具有传递性，即若 A 是 B 的朋友，B 是 C 的朋友，那么认为 A 也是 C 的朋友。所有朋友的集合成为朋友圈。请设计算法计算班级里朋友圈总数。重点阐述算法思路及所用到的数据结构和算法，请给出代码或伪代码。

深度优先搜索

```
Function GetFriendCircleCount(M, N):
    // 1. 初始化
    count = 0               // 朋友圈计数器
    visited = [False] * N     // 标记数组，初始全为 False


    For i from 0 to N-1:


        If visited[i] == False:

            count = count + 1  // 发现新圈子，计数 +1
            DFS(M, visited, i, N) // 核心：把跟 i 有关的人全部标记

    Return count

// 辅助函数：深度优先搜索
Function DFS(M, visited, u, N):
    visited[u] = True // 标记当前学生已访问

    // 检查其他所有学生 v
    For v from 0 to N-1:
        // 如果 u 和 v 是朋友，且 v 还没被访问
        If M[u][v] == 1 AND visited[v] == False:
            DFS(M, visited, v, N) // 递归继续找 v 的朋友
```

草稿区

草稿区