2018.10

# Data Structures

**First edition**

数据结构




By Jose & Austin

----------资料目录----------





欢迎扫码加编者微信：

扫一扫上面的二维码图案，加我微信

有志者事竟成

# 前言

在几个月前，只剩下三周时间去复习数据结构。全班非常紧张，害怕考试不能得到高分，于是全班一起去复印社团购了《数据结构》历年真题。我们买来后却大跌眼镜——这质量简直差的不能再差！卖给同学简直坑钱！可是我们也只能将就用了，毕竟时间已经非常紧迫了。最后考完这门课的时候我心想是不是需要花点时间做份《数据结构》复习试题给大家用，让大家享受高质量的试卷？在一系列的事情后我决定开始做这件事（其中包含着我一些承诺和某些“典故”）。我联系了身边各位大神并得到一些人的支持，在此真的很感谢他们，没有他们真的这份试题不能完成。然后经历一两个月后这个计划完成了。又想到在做之初别人告诉我没必要多管闲事，复印社印刷的资料还是可以将就用的。作为一个追求完美的人我是不能容忍这现象的，我还是硬着头皮做了，希望真的可以让读者获得考试 buff，人人上 90 拿满绩点！

本资料收纳了五套试题和一套复习题，供计算机学院、软件学院的同学复习参考。本着以最少的题目得最高分数的原则，我尽量做到把考试 90%以上的知识点融入这份资料里面。资料的每个题目望读者用心研究，学明白了何尝不能达到 88 分，甚至 99 分呢（Richard 的分数就是 99 分）？在这份资料中尤其需要关注 09 级试题（当年大约 45%的人不及格）以及计算机唐敖庆班考试真题（也同时是软件卓越工程师班的考题，考查的知识范围很广）。当然有几个试题不是真题（试题出现了重复，换成其他一些类型的题目了），希望读者明白。

值得一提的是——本资料在制作后期有点仓促，答案没有经过三人校对，所以会出现某些试题答案不对和没给一些算法题目代码。实属抱歉，编者也是学生，也需要学习其他科目，真的做出这份资料费了很多时间。Jose & Austin 制作这份资料完全出自那份初心，所以请读者谅解此事。

在这里真诚感谢 Austin & Alex & Richard & Chen 四人。他们完成了大部分答案、凭着对考题的印象打出了 2017 级计算机唐敖庆班数据结构试题（为他点 100 个赞）、给出了唐敖庆班考题的答案，以及一些算法题解题思路。感谢你们付出，没有你们就没有这份资料！

最后我希望大家用上这份资料能够为你的数据结构期末考试保驾启航。

编者

使用此资料注意：

1. 不保证做的答案全部是正确的，该资料不适合只相信答案的人使用；
2. 部分题目修改过，可能不会与原题一模一样；
3. 如果发现错误请加 QQ 群：783124749 并告诉管理员；也欢迎大家入这个群交流学习；
4. Jose & Austin 拥有资料所有版权。未经许可翻版必追查责任！
5. 该资料是免费共享的，严禁拿该资料获利！

# 2015 年数据结构考试试题 A

一、 选择题（5*2 分）

1. 设栈的输入队列是（a,b,c,d）,则_____不可能是其出栈序列。

A.abcd　　B.bacd　　C.adcb　　D.dcab

2.若一棵二叉树有 10 个度为 2 的结点，则该二叉树的叶结点的个数是_____。

A.9　　B.11　　C.10　　D.12

3.在图形结构中，每个结点的前驱结点数和后继结点数可以有___个。

A.1　　B.2　　C.任意多　　D.0

4.下面的排序算法中不稳定的是___。

A.合并排序　　B.希尔排序　　C.直接插入排序　　D.冒泡排序

5.中缀表达式{A+B*C-[D+E/F*(G+H)]}对应的后缀表达式是____。

A.+ABC*-DEF/*+GH　　B. ABC*+-DEF/GH+*+

C.ABC+*-DEF/*+GH　　D. ABC*+DEF/GH+*+-

二、简答题（55 分）

1. 简述数据结构包含哪三个方面的内容？数据的储存方法有几种，并对每一种存储方法的特点进行简单的描述。（5 分）



2. 给出如下稀疏矩阵的三元组表示（行列从 0 开始编号）:（5 分）

$$A=\begin{bmatrix} 50 & 0 & 0 & 0 \\ 10 & 0 & 20 & 0 \\ 0 & 0 & 0 & 0 \\ -30 & 0 & -60 & 5 \end{bmatrix}$$

3. 构造权值为{5，13，21，7，18，30，41}的哈夫曼树，并给出哈夫曼编码。（6 分）

4. 数组 A 中，每个元素 A[i][j]的长度均为 2 个字节，行下标从 1 到 5，列下标从 1 到 10，从首地址 S 开始连续存放于主存储器中。（8 分）求：

（1）存放该数组共需要多少字节？

（2）存放数组第四列所有元素至少需要多少个单元？

（3）数据按照行存放时，元素 A[2][4]的起始地址是多少？

（4）数据按照列存放时，元素 A[4][7]的起始地址是多少？

5.已知一棵二叉树的后根和中根遍历序列如下（6 分）:

后根遍历序列　　C E F D B H G A

中根遍历序列　　C B E D F A G H

<1>画出这棵二叉树的逻辑结构；

<2>给出这棵二叉树的先根遍历序列。

6. 设散列表长度为 11，散列函数 h(x)=x%11，给定的关键字序列为：1，13，12，34，38，33，27，22。试画出用线性探查法解决冲突时所构建的散列表。（8 分）

7. 待排序的文件为（6，25，7，11，51，17，39，85，8，9），请给出堆排序过程。（10 分）

8. 给定顺序表（5，12，17，19，23，25，30，36），请画出此顺序表对半查找时对应的二叉判定树。当采用对半查找算法查找关键字 30 时，进行多少次比较后查找成功？算出该顺序表成功查找和失败查找的平均比较次数（各个关键字等概率）。（7 分）

三、算法题（35 分）

注意：可选择适用 C++或 ADL 语言，建议使用 ADL 语言；算法开始处必须通过注释说明算法主要思想，关键操作步骤须有注释；书写算法时要有必要的缩进。

1. 设计一个算法，给定整数 K，重排大小为 n 的整型数组 A[1:n]，将 A 中所有小于 K 的关键字排在所有大于 K 的关键字之前。
   <1>给出算法的基本设计的思想（4 分）；
   <2>用算法描述语言描述算法，并要求对算法中的关键步骤给出注释（6 分）。



2. 假定以单链表保存学生成绩信息，data 域值为考试成绩。设计一算法，计算单链表中所有学生成绩的平时分。
   <1>给出算法的基本设计思想（4 分）；
   <2>用算法描述语言描述算法，并要求对算法中的关键步骤给出注释（6 分）。

3. 一棵二叉树以链表形式存储，结点结构为（Left，Data，Right），设计一算法，求二叉树中从根结点到叶结点的一条路径长度等于树的高度的路径，若这样的路径存在多条，则输出路径终点（叶结点）“最左“的一条。
   <1>给出算法的基本设计思想（5 分）；
   <2>用算法描述语言描述算法，并要求对算法中的关键步骤给出注释（10 分）。

# 2014 级数据结构考试试题 A 卷

一、 判断题（10*1 分）

1. 环状队列是一种物理结构。
2. 若完全二叉树的某结点无左孩子，则它必是叶结点。
3. N 个顶点的无向连通图至少有 N+1 条边。
4. 任何 AOV 网的拓扑序列都是唯一的。
5. 无向图的邻接矩阵是对称的，有向图的邻接矩阵一定不对称。
6. 关键路径是 AOE 网中从源点到汇点的最短路径。
7. 如果序列已经有序，直接插入排序的时间复杂性为 O(n)。
8. 对两棵具有相同关键字集合而形状不同的二叉查找树，按中序遍历它们得到的序列是一样的。
9. AVL 树是一棵二叉树，该树上任一结点的平衡因子（平衡系数）的绝对值不大于 1。
10. 散列法存储的基本思想是由关键词的值决定数据的存储地址。

二、 选择题（10*2 分）

1. 设栈的输入序列是（1、2、3、4），则____不可能是其出栈序列。

A．1234 B.2134 C.1432 D.4312

2.求字符串 T 在字符串 S 中首次出现的位置的操作称为____。

A.串的模式匹配 B.求字串 C.求串的长度 D.串的连接

3.下面关于哈夫曼树的说法，不正确的是____。

A. 哈夫曼树没有度为 1 的结点

B.哈夫曼树具有最小的带权路径长度

C.对应一组权值构造出的哈夫曼树一般不是唯一的

D.哈夫曼树除了度为 1 的结点外，还有度为 2 的结点和叶结点

4.任何一棵非空二叉树的叶结点在先根遍历、中根遍历和后根遍历中相对位置____。

A．都会发生改变 B.不会发生改变

C．有可能发生改变 D.部分会发生改变

5.具有 2000 个结点的非空二叉树的最小深度为____。

A. 9 B. 10 C. 11 D. 12

6.若一棵满二叉树有 1024 个叶结点，则该二叉树结点的个数为____。

A. 2045 C. 2046 D. 2047 D. 2048

7.下面关于图的存储的叙述中，____是正确的。

A.用邻接矩阵存储图，占用的存储空间的大小只与图中顶点个数有关，而与边数无关

B. 用邻接矩阵存储图，占用的存储空间的大小只与图中边数有关，而与顶点个数无关

C. 用邻接表存储图，占用的存储空间的大小只与图中顶点个数有关，而与边数无关

D. 用邻接表存储图，占用的存储空间的大小只与图中边数有关，而与顶点个数无关

8.下面的叙述中，不正确的是____。

A．任何关键活动不按期完成就会影响整个工程完成的时间
B．任何一个关键活动提前完成，将使整个工程提前完成
C．所有关键活动都提前完成，将使整个工程提前完成、
D．所有关键活动不按期完成就会影响整个工程的完成时间

9.下面的排序算法中，具有稳定性的是____排序。

A．直接选择　　B. 希尔　　C. 冒泡　　D. 堆

10.一组记录的关键字为（25,50,15,35,80,85,20,40,36,70），其中含有 5 个长度为 2 的有序表，用归并排序方法对该序列进行一趟归并后的结果为____。

A. 15,25,35,50,20,40,80,85,36,70　　B. 15,25,35,50,80,20,85,40,70,36

C. 15,25,50,35,80,85,20,36,40,70　　D. 15,25,35,50,80,20,36,40,70,85

三、简答题（35 分）

1.将表达式((a+b)-c*(d+e)-f)*(g+h)改写成后缀表达式（2 分）。

2. 给出如下稀疏矩阵的三元组表示。（4 分）

$$\begin{bmatrix} 0 & 0 & 6 & 0 \\ 4 & 0 & 0 & 0 \\ 0 & 9 & 0 & 7 \\ 0 & 0 & 0 & 8 \end{bmatrix}$$



3. 已知一棵二叉树中序和前序序列如下，画出该二叉树，并求该二叉树后序序列。（4 分）

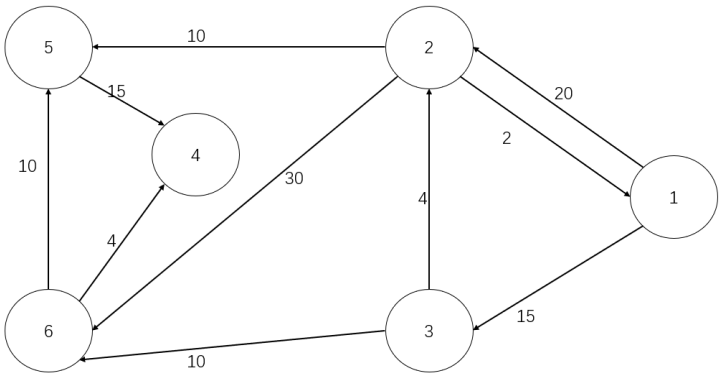
中序序列：c b d e a g i h j f

前序序列：a b c e d f g h i j

4. 已知序列 17,31,13,11,20,35,25,8,4,11,24,40,27，请画出该序列的二叉查找树。（7 分）

5. 一组记录的关键字为(52,56,26,12,69,85,33,48,70)，给出堆排序的过程，即画出每次堆调整后堆的状态。（8 分）

6. 对于如下的有向图，请利用 Dijkstra 算法求出从源点 1 到其他各顶点的最短路径，并写出执行该算法过程中扩充点集的每次状态。（10 分）



三、 算法题（共 35 分）

1. 编写一个算法，在带表头结点 head 的单链表中单链表中寻找第 i（i>=1）个结点；若找到，则函数返回指向第 i 个结点的指针；若找不到，则函数返回 NULL。（7 分）

2. 编写算法计算并输出以 root 为根的二叉树中叶结点个数。（8 分）

3. 编写算法求任意二叉树中第一条最长的路径，并输出此路径上各结点的值。（10 分）



4. 设计一个算法，判断无向图 G 是否是一棵树。若是，则返回 1；否则返回 0。（10 分）

# 2009 级《数据结构》期末试题 A 卷

一、判断题（5*2 分）

1. 算法的优劣与所用的计算机无关。但是与算法描述的语言有关。
2. 根据算法的时间复杂性，人们常常把算法分成两类：多项式阶算法、指数阶算法。当 n 很大时，可以证明有如下关系：
   O(1)<O(loglog n)<O(log n)<O(n*log n)<O(n)<O(n^2)<O(2^n)
3. 分析排序算法的最好和最坏时间复杂性时，当排序文件已经是排好序时，则算法对此文件执行排序具有最好时间复杂性；当排序文件是逆排列时，算法对此文件执行排序具有最坏时间复杂性。
4. N 个结点的有向图，若它有 N(N-1)条边，则它一定是强连通的。在 n 个结点的无向图中，若边数大于 N-1，则该图必是连通图。
5. 中序遍历一棵二叉查找树可以得到一个排好序的结点序列。

二、单选题（7*2 分）

1.一棵左右子树均不空的二叉树在后序线索化后，其空指针域的个数为____。
A.0 B.1 C.2 D.3

2.下列排序算法中，某一趟排序后未必能选出一个元素放在最终位置上的是____。
A.堆排序 B.直接插入排序 C.分划交换排序 D.冒泡排序

3.一棵具有 N 个顶点，K 条边的无向图是一个森林(N>K)，则该森林必有____棵树。
A.K B.N C.N-K D.1

4.若一组记录关键字为(34,60,55,30,32,85)，则利用堆排序的方法建立的初始堆为____。
A.{60,34,55,30,32,85} B.{85,60,55,30,32,34}
C.{85,60,55,34,32,30} D.{85,55,60,32,34,30}

5.在下述结论中正确的有____个。
1)只有一个结点的二叉树的度为 0
2)二叉树的度为 2
3)二叉树的左右字数可任意交换
4)深度为 K 的完全二叉树的结点个数小于或等于深度相同的二叉树
5)有 4 个不同关键字的结点可以构成 14 种不同的二叉树
A. 1 B. 2 C. 3 D.4

6.在一个有向图中，所有顶点的入度之和等于所有顶点出度之和的____倍。
A.1 B.2 C.1.5 D.不能确定

7.设有数组 A[i,j]，数组的每一个元素长度为三个字节，i 的值为 1 到 8，j 的值为 1 到 10，数组从内存首地址 S 开始顺序存放，当按列优先存放时，元素 A[5,8]的存储首地址为____。
A.S+141 B.S+180 C.S+222 D.S+225

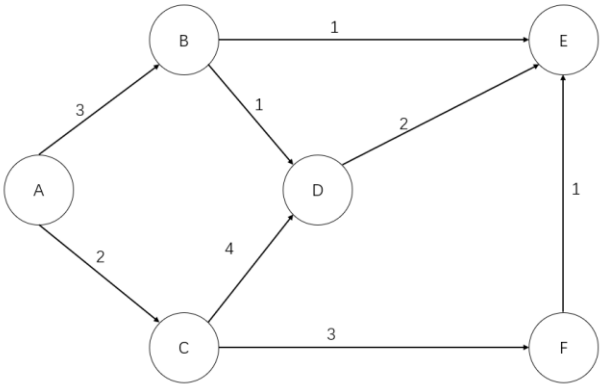
三、简答题（46 分）

1. 说明顺序存储和链式存储的区别及各自的优缺点。（4 分）

2. 将表达式(a+b)*c+d/(e*f-g)-h 改写成前缀和后缀表达式。（6 分）

3. 已知一棵二叉树 T 的诸结点在先根次序下的排列为：ABCEDFGHI，在中根次序下的排序为:ECBDFAHIG，画出此树形，并给出其后根序列。（6 分）

4. 一组记录的关键字为(48,75,8,50,33,42,86,26,70)，给出快速排序的过程（只需给出每次分划后的结果）。（6 分）
5. 一个 AOV 网如下图所示：（8 分）




1) 计算出所有活动的最早开始时间 e(a)和最迟开始时间 l(a);
2) 基于以上计算结果，给出该网络中的所有关键路径。
6. 设有序顺序表为{10,20,30,40,50,60,70,80}。画出折半查找算法对应的二叉判定树，并计算采用折半查找方法查找以上顺序表时，查找成功和查找失败情况下的平均查找次数。（6 分）
7. 设散列表长度为 11，散列函数 h(x)=x %11，给定的关键字序列是 1，13，12，34，38，33，27，22.试画出用线性探查法解决冲突时所构造的散列表。（6 分）

8. 填充如下排序算法中的方框，并讨论该算法的稳定性。（4 分）

算法 C(R，n)

/*比较计数，本算法按关键词 K1，K2，…，Kn 排序记录 R1，R2，…，Rn．一维数组 COUNT[1 : n]用来记录各个记录的排序位置*/

［C1］ FOR i=1 TO n DO ____①____.

［C2］ FOR i=n TO 2 ____②____ DO

FOR j=i–1 TO 1 STEP –1 DO

IF____③____THEN

COUNT[j] ←COUNT[j]+1

ELSE ____④____.

四、算法题（30 分）

1. 设计一个算法，将链表的链接完全颠倒。（10 分）

2. 对以左儿子—右兄弟链接表示的树，编写计算树的深度的算法。（10 分）

3. 图采用邻接表存储结构，编写一个算法，判别无向图中任意给定的两个顶点之间是否存在一条长度为 k 的简单路径。（10 分）

# 卓越班数据结构复习参考题

一、判断题

1. 能够通过在链表的表头不断地插入新结点来创建一个单向链表。
2. 双向循环链表的头或尾加入结点不需要任何特殊的处理。
3. 与单链表相比，双向链表添加和删除数据元素的能力更高。
4. 堆栈和队列不属于线性结构。
5. 环状队列是一种物理结构。
6. 循环链表可以实现环状队列。
7. 多维数组元素之间的关系是线性的。
8. 树中结点的深度等于它的祖先的个数。
9. 一个结点是叶结点，当且仅当它的度为 0 。
10. 如果 x 是 y 的后代，则 x 的深度大于 y 的深度。
11. 如果 x 的深度大于 y 的深度，则 x 是 y 的后代。
12. 满二叉树的每棵子树是完全二叉树。
13. 在任意一棵二叉树中，分支结点的数目一定少于叶结点的数目。
14. 树是非线性结构，没有明确的“下一个”概念，因此不能进行顺序遍历。
15. 任何 AOV 网的拓扑序列都是唯一的。
16. 无向图的邻接矩阵是对称的，有向图的邻接矩阵一定不对称。
17. 用邻接矩阵法存储一个图时，在不考虑压缩存储的情况下，所占用的存储空间大小只与图中结点个数有关，而与图的边数无关。
18. 没有任何一种通过比较元素重排序列的排序算法在最坏情况下，时间复杂性好于 O（nlog2n）。
19. 快速排序的速度在所有方法中为最快，而且所需的附加空间也最少
20. （101，88，46，70，34，39，45，58，66，10）是堆
21. 在最大堆中，值最大的元素在根，值最小的元素在某个叶结点处
22. 对一个堆，按二叉树层次进行遍历，可以得到一个有序序列

二、选择题

1.已知一个顺序存储的线性表，设每个结点需占 m 个存储单元，若第一个结点的地址为 da,则第 i 个结点的地址为(　)。

A. da+(i-1)*m　　B. da+i*m　　C. da-i*m　　D. da+(i+1)*m

**2.**设长度为 n 的链队列用单循环链表表示，若只设头指针，则入队操作的时间复杂度为(　)。

A. O(1)　　B. O(log2n)　　C. O(n)　　D. O(n2)

3.设长度为 n 的链队列用单循环链表表示，若只设尾指针，则入队操作的时间复杂度为(　)。

A. O(1)　　B. O(log2n)　　C. O(n)　　D. O(n2)

4.若字符串“1234567”采用链式存储，假设每个字符占用 1 个字节，每个指针占用 2 个字节，则该字符串的存储密度为（　　）。

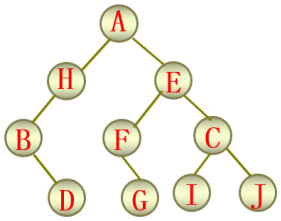
A. 20％　　B. 40％　　C. 50％　　D. 33.3％

5.设栈的输入序列是（1，2，3，4），则(　)不可能是其出栈序列。

A. 1243　　B. 2134　　C. 1432　　D. 4312

6.下图所示的 T2 是由森林 F 转换而来的二叉树，那么森林 F 共有(　)个叶结点。

A. 4　　B. 5　　C. 6　　D. 7




7.深度为 5 的二叉树最多有(　)个结点。
A. 64　　B. 63　　C. 32　　D. 31
8.将一棵有 100 个结点的完全二叉树从上到下，从左到右依次对结点进行编号，根结点编号为 1，则编号为 49 的结点的左孩子的编号为(　)。
A. 98　　B. 99　　C. 50　　D. 48
9.任何一棵二叉树的叶结点在其先根、中根、后根遍历序列中的相对位置(　)。
A. 肯定发生变化　　B. 有时发生变化　　C. 肯定不发生变化　　D. 无法确定
10.具有 35 个结点的完全二叉树的深度为(　)。
A. 5　　B. 6　　C. 7　　D. 8
11.设深度为 k 的二叉树上只有度为 0 和度为 2 的结点，则这类二叉树上所含结点总数最少(　)个。
A. k+1　　B. 2k　　C. 2k-1　　D. 2k+1
12.对于二叉树来说，第 i 层上至多有(　)个结点。
A. 2^i　　B. 2^i-1　　C. 2^i-1　　D. 2^(i-1)-1
13.某二叉树的前序和后序序列正好相同，则该二叉树一定是(　)的二叉树。
A. 空或只有一个结点
B. 高度等于其结点数
C. 任一结点无左孩子
D. 任一结点无右孩子
14.若某线性表中最常用的操作是取第 i 个元素和删除最后一个元素，则采用(　)存储方式最节省时间。
A. 顺序表　　B. 单链表　　C. 双链表　　D. 单循环链表
15.用邻接表表示图进行广度优先遍历时，通常采用(　)来实现算法的。
A. 栈　　B. 队列　　C. 树　　D. 图
16.用邻接表表示图进行深度优先遍历时，通常采用(　)来实现算法的。
A. 栈　　B. 队列　　C. 树　　D. 图
17.栈和队列都是（　）
A.顺序存储的线性表
B.链式存储的线性表
C.限制存取点的线性结构
D.限制存取点的非线性结构
18.任何一个无向连通图的最小生成树(　)。
A. 只有一棵
B. 有一棵或多棵
C. 一定有多棵
D. 可能不存在

19.下列排序算法中,某一趟结束后未必能选出一个元素放其最终位置上的是(　)。

A. 堆排序

B. 冒泡排序

C. 快速排序

D. 直接插入排序

20.下列排序算法中,第一趟排序完毕后,其最大或最小元素一定在其最终位置上的算法是(　)排序。

A. 归并　　B. 直接插入　　C. 快速　　D. 冒泡

21.当初始序列已经按键值有序时，用直接插入算法进行排序，需要比较的次数为(　)。

A. n2　　B. nlog2n　　C. log2n　　D. n-1

22.若用冒泡排序法对序列{18,16,14,12,10,8}从小到大进行排序，需要进行(　)次比较。

A.10　　B.15　　C.21　　D.34

23.下列四个关键字序列中,(　)不是堆。

A. {05,23,16,68,94,72,71,73}

B. {05,16,23,68,94,72,71,73}

C. {05,23,16,73,94,72,71,68}

D. {05,23,16,68,73,71,72,94}



24.层数为 2 的四阶 B 树（默认根结点在第 0 层）最多可以存(　)个关键字。

A.60　　B.30　　C.63　　D.66

25.层数为 3 的五阶 B 树（默认根结点在第 0 层）至少能存(　)个关键字。

A.50　　B.53　　C.55　　D.56

三、简答题

1.设计求带表头结点的单链表 head 中所有 data 域值为 x 的元素个数的算法。

2.假设线性表用带表头结点的单向链表表示，试写出删除表中所有 data 域值为零的元素的算法。

3.对于一个栈，如果输入项序列由 1，2，3 所组成，试给出全部可能的输出序列。

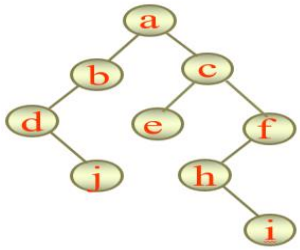
4.对于一个队列，如果输入项序列由 1,2,3,4 所组成，试给出全部可能的输出序列。

5.内存中一片连续空间，提供给两个栈 S1 和 S2 使用，怎样分配这部分存储空间，使得对任一个栈，仅当这部分空间全满时才发生上溢。

6.有二叉树先序序列为：ABCDEF,中序序列为：CBAEDF,试画出该二叉树并给出其后序序列。

7.试画出具有 3 个结点的二叉树的所有不同形态。

8.由下图给出的二叉树，求出先序、中序、后序遍历的结点序列。






9.有一份电文中共使用 5 个字符：a、b、c、d、e,它们出现的频率依次为 4、7、5、2、9,试化出对应的哈夫曼树，并求其加权路径长度 WPL，求出每个字符的哈夫曼编码。

10. 对于下图所示的 AOE 网。求出：
1).每个事件的最早发生时间和最迟发生时间；
2).每项活动的最早开始时间和最迟开始时间；
3).完成整个工程至少需要多长时间；
4).画出由所有关键活动所构成的图；
5).哪些活动加速可使整个工程提前。

2017~2018 学年 第二学期

# 2017 级**班《数据结构》期末试题（A 卷）

考试时间:2018 年 6 月

班级_____ 学号_____ 姓名_____

**请将答案写在答题纸上，写明题号，不必抄题，字迹工整、清晰；

**请在答题纸和试题纸上都写上你的班级，交卷时请将试题纸、答题纸和草纸一并交上来。

一、填空题（5*2 分）

1. 循环队列的容量为 100，经过一系列的入队和出队操作后，指向队首元素的 front=20，指向队尾元素的 rear=10，则队列中元素个数是_____。
2. 设 n 行 n 列的上三角矩阵 A 压缩存储到一维数组 T[1…n(n+1)/2]中，若按列优先顺序存储，则 A[i，j](i>=1,j<=n)对应的在 T 中的存储位置是_____。
3. 设森林 F 有四棵树，第 1、第 2、第 3、第 4 棵树的结点个数分别为 N1、N2、N3 和 N4,T 为森林 F 自然对应的二叉树,则 T 的根节点的右子树有_____结点。
4. 有向图 G 的拓扑序列个数为_____，其中 V(G)={1 ， 2 ， 3 ， 4 ， 5 ， 6 }；E(G)={<1 ， 2>,<1 ， 3>,<1 ， 4>,<2 ， 3>,<4 ， 5>,<4 ， 6>}。
5. 一组记录的关键字{46，79，56，38，40，84}，利用快速排序，以第一个记录的关键字为基准得到的一次分划结果为_____。

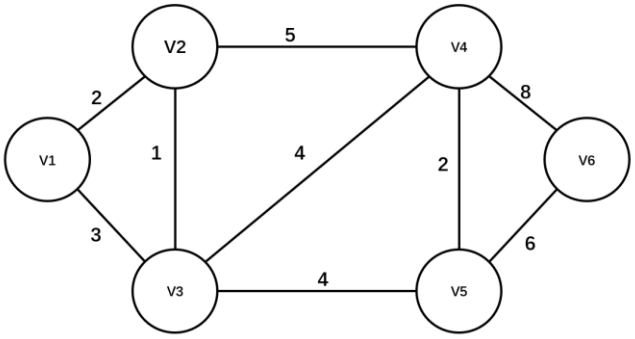
二、简答题（55 分）

1. （6 分）用栈将算术表达式 a + (b - c) * (d + e) / f 转换为后缀表达式。画出转换过程，每步包含当前处理运算符、操作符栈和当前输出序列。

2. （11 分）一棵二叉树的扩展先根序列为：A B C # # # D E # # #，#代表空指针。
   a)画出这棵二叉树；
   b)给出其中根序列和后根序列；
   c)画出其自然对应的树（或森林）；
   d)画出由其对应的中序线索二叉树。

3. （7 分）由 6 个分别带有权值 10、6、12、3、8、2 的叶子结点构造一棵哈夫曼树，画出其构造过程，并计算其加权路径长度。

4. （13 分）已知图 G=(V，E，C)，如下图所示；
   a) 画出该图的邻接矩阵表示；
   b)画出该图的邻接链表表示；
   c)给出图 G 的一个广度优先序列；
   d)求图 G 的最小生成树（过程）。




5. （5 分）排序文件含 8 个整数{8，3，6，5，4，11，4*，3*}，*标识关键字重复出现。画出归并排序过程，并讨论其稳定性。

6. （6 分）用输入序列（50，40，10，30，20，35）建立一棵高度平衡树，画出该树构造过程。

7. （7 分）设散列表长度 m = 11，散列函数 H(K) = K % m，给定关键码序列：33，21，8，22，43，17，12，13，11。试画出用拉链法消除冲突所构造的散列表，并计算查找成功和不成功的平均查找长度。



三、算法题（35 分）

**答题要求：给出算法的基本思想；描述算法可选择 ADL、C 或 C++等语言；对算法中的关键步骤给出注释；**

1. （10 分）MOVE-TO-FRONT 单链表。
   （1） 设计算法，实现使用 MOVE-TO-FRONT 策略的自组织表，即对于给定的关键字 K，如果检索成功，便把该记录移动表的开头。设查找表有 n 个记录，使用单链表存储，结点结构为 NODE(data，next)；
   （2） MOVE-TO-FRONT 不总有效，请给出一个失效的例子。

2. （10 分）树高 Parent 链接。树有 n 个结点，使用 Parent 链接存储结构。使用高效算法计算树的高度，分析算法时间复杂度。

3. （15 分）最小边数最短路。最短路可能有多条，边数最少的最短路称为最小边数最短路。设计高效算法，在非负权图上求顶点 v 到顶点 w 的最小边数最短路。分析算法时间复杂度。

# 考前补漏卷

一、单选题

1. 设有一个二维数组 A[m][n]，假设 A[0][0]存放位置在 644(10)，A[2][2]存放位置在 676(10)。每个元素占一个空间，且数组按列优先存储。问 A[3][3](10)存放在什么位置？脚注(10)表示用 10 进制表示。____

A．688　　B．678　　C．692　　D．696

2. 树最适合用来表示(　　)。

A.有序数据元素

B.无序数据元素

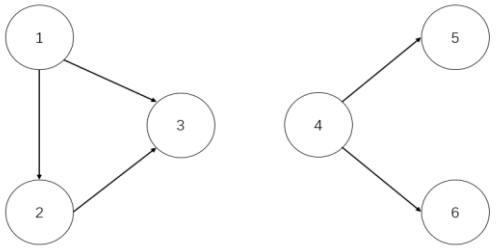
C.元素之间具有分支层次关系的数据

D.元素之间无联系的数据

3. 二叉树的第 k 层的结点数最多为(　)。

A．2k-1　　B.2K+1　　C.2K-1　　D. 2k-1

4.由下图给定的图的拓扑排序有(　)种。




A.40　　B.20　　C.30　　D.24

5. 若有 18 个元素的有序表存放在一维数组 A[19]中，第一个元素放 A[1]中，现进行二分查找，则查找 A［3］的比较序列的下标依次为(　)。

A. 1，2，3　　B. 9，5，2，3　　C. 9，5，3　　D. 9，4，2，3

6. 对 n 个记录的文件进行快速排序，所需要的辅助存储空间大致为(　)。

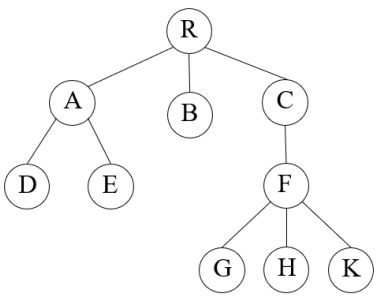
A. O（1）　　B. O（n）　　C. O（1og2n）　　D. O（n2）

二、简答题

1.把后缀算式 9 2 3 +- 10 2 / - 2 *转换为中缀表达式并计算其值。

2.已知目标串 s=“abcabcabcabcabcbabc”，模式串 pat= “abcabcb”，写出模式串失败函数 f 的值，并由此画出 KMP 算法匹配的全过程。

3. 由下图给出的一棵树，写出其先根序列和后根序列，并画出将其转化为二叉树的图形。

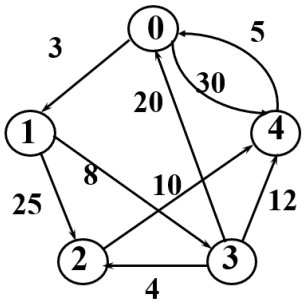


4.已知一个图的邻接矩阵表示为：

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 0 | 1 | ∞ | 4 |
| 1 | ∞ | 0 | 9 | 2 |
| 2 | 3 | 5 | 0 | 8 |
| 3 | ∞ | ∞ | 6 | 0 |

请给出 Floyd 算法求解过程。

5.已知图 G=(V,E)如下图所示：




试给出 Dijkstra 算法求顶点 0 到其他各点的最短路的过程。



6. 画出向大根堆中加入数据 4, 2, 5, 8, 3,7 时，每加入一个数据后堆的变化。

7.设文件包含的关键字为(5,4,9,6,3,7,2)，试画出一致对半查找树和斐波那契查找树。

8.设散列表的长度 M=11，散列函数为 H(x)=x % M。给定的关键码序列为：(2,5,8,10,1,23,16,27,7)。试分别画出用拉链法和线性探查法解决冲突时所构建的散列表。在等概率的情况下分别求出查找成功和查找失败的平均查找长度。

三、算法题（3*10 分）

1.试写出判断两棵二叉树是否相似的算法。

2.给定含有 N 关键字的文件，试给出文件第 M 小的关键字（M<N）的算法。

4. poj2377 题意就是要求出图的最大生成树，试给出最大生成树的算法；并给出其一些性质。（2017 卓越班考题）

# 2015 数据结构 A 参考答案

一、DBCBD

二、

1.数据的逻辑结构,数据的存储结构,运算(操作)

顺序存储结构：把一组结点存放在地址相邻的存储单元里,结点间的逻辑关系用存储单元的自然顺序关系表达

链接存储结构： 在结点的存储结构中附加指针字段，两个结点的逻辑后继关系用指针的指向来表达。

索引存储结构：索引表把整数索引值映射到结点的存储地址。索引表存储一串指针，每个指针指向存储区域的一个数据结点。

散列存储结构：索引存储的一种延伸和扩展，利用散列函数进行索引值的计算，并通过索引表求出结点的指针地址。

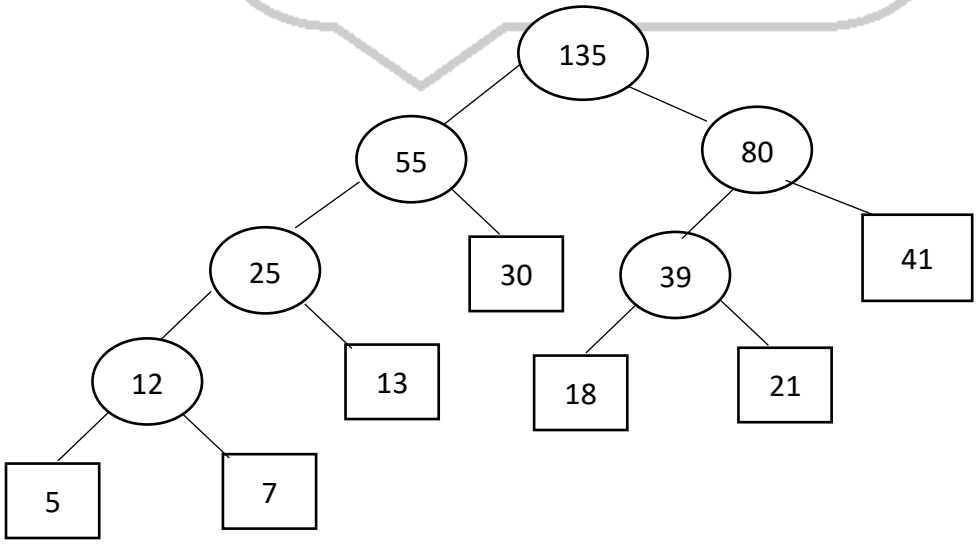
2.

| | i | j | V |
|---|---|---|---|
| 0 | 0 | 0 | 50 |
| 1 | 1 | 0 | 10 |
| 2 | 1 | 2 | 20 |
| 3 | 3 | 0 | -30 |
| 4 | 3 | 2 | -60 |
| 5 | 3 | 3 | 5 |



3. 一种可能的方法（编码不唯一，但是无论哪种方案相同的权值编码长度一定相同）

5:0000　7:0001　13:001　30:01　41:11　21:101　18:100




4. （1） 数组中一共有 5*10=50 个元素, 50*2=100

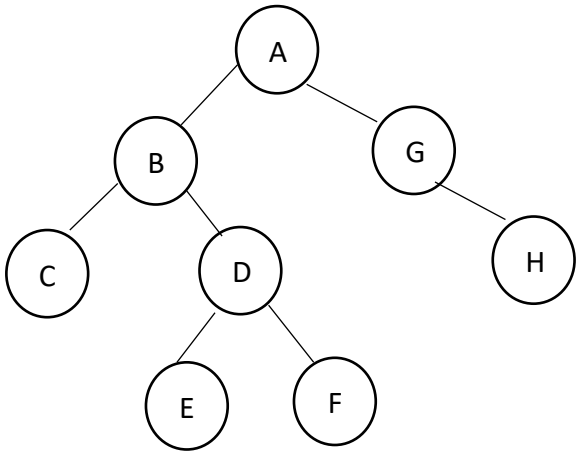
（2） 5*2=10

（3） S+((2-1)*10+(4-1))*2=S+26

（4） S+((7-1)*5+(4-1))*2=S+66

Jose & Austin

5. (1)



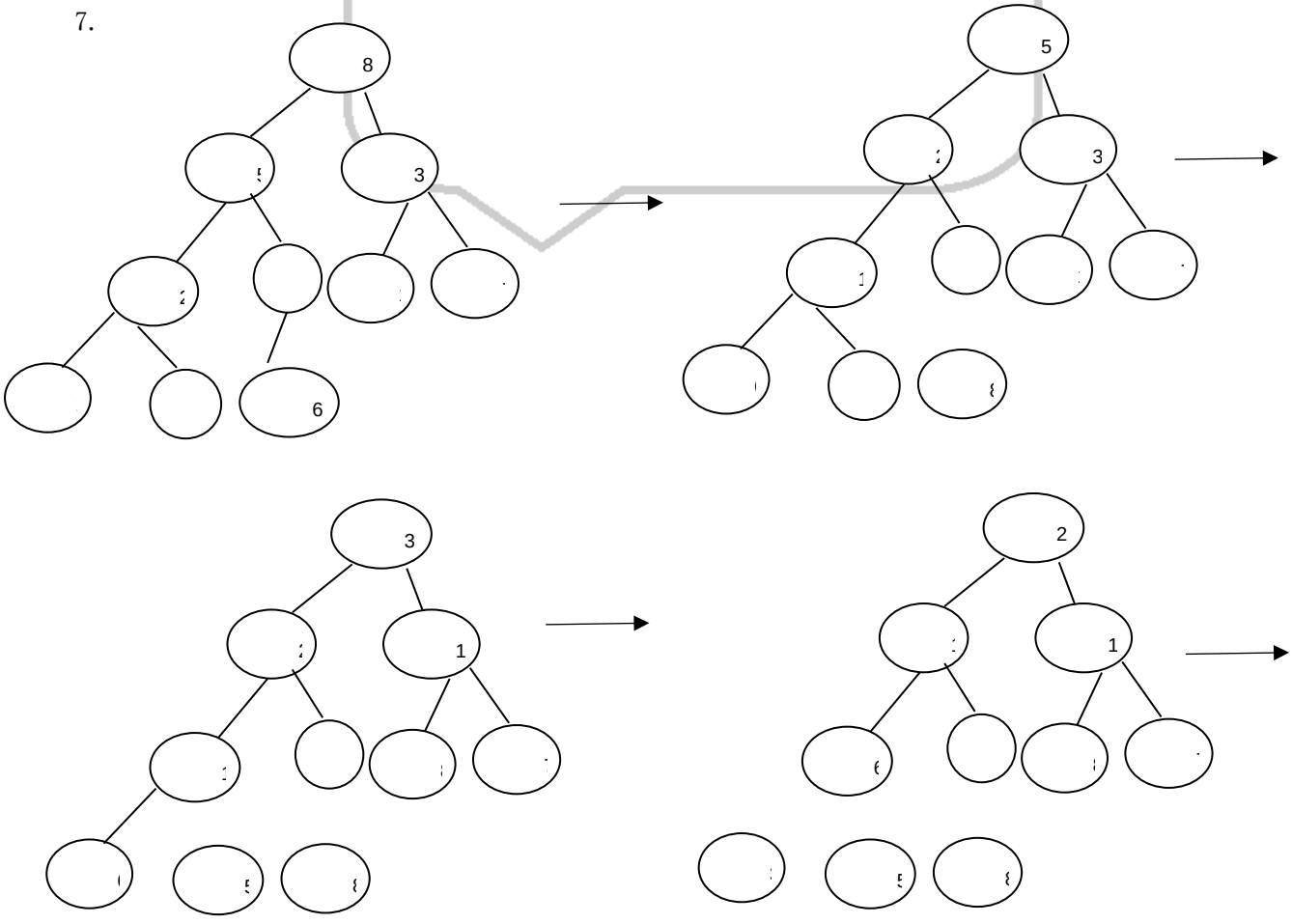

(2)ABCDEFGH

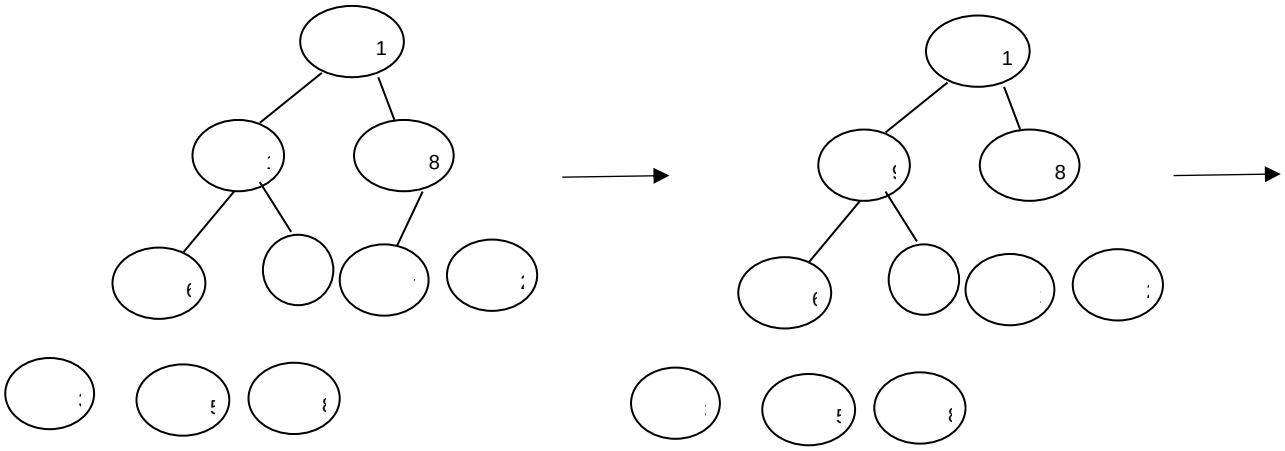
6.

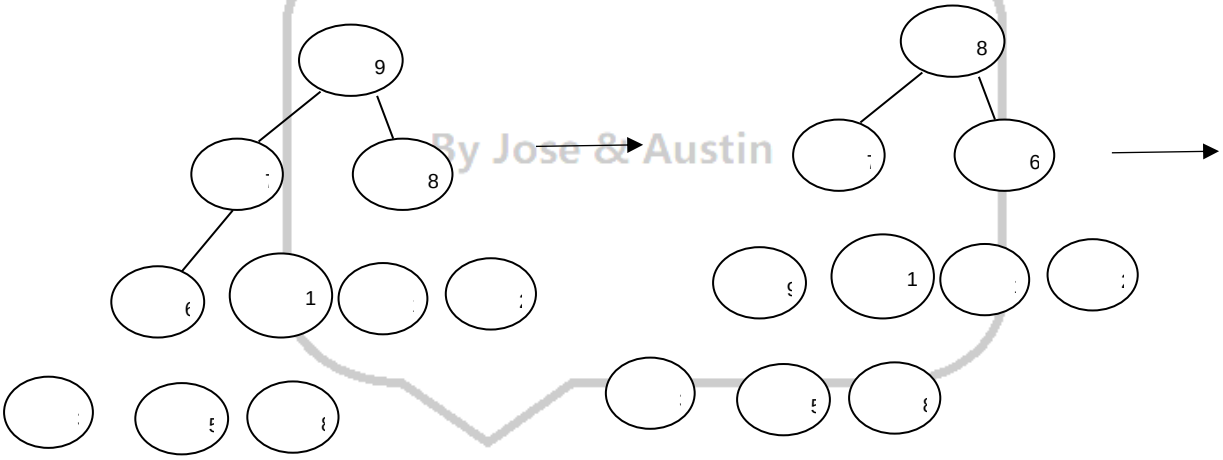
| 下标 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 关键字 | 33 | 1 | 13 | 12 | 34 | 38 | 27 | 22 | | | |



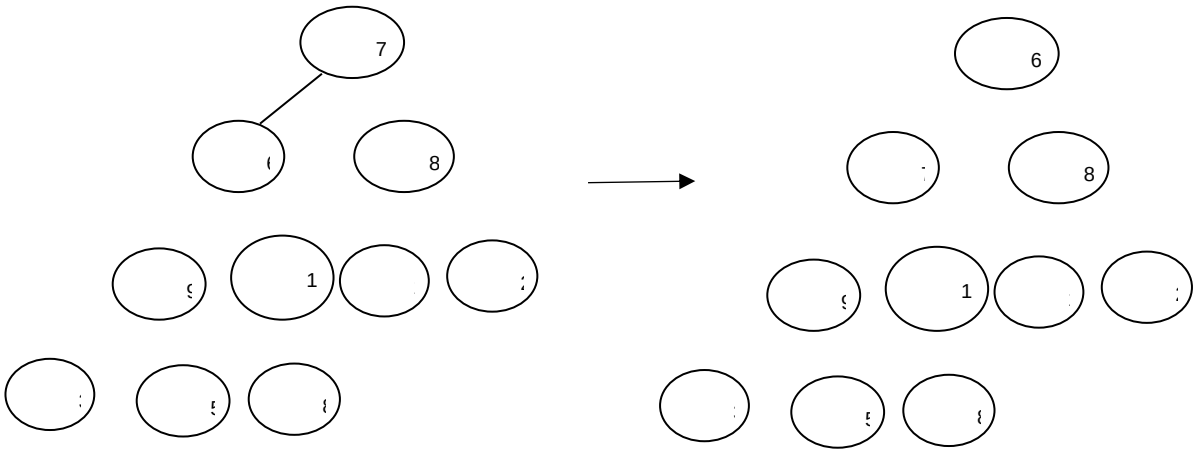
7.





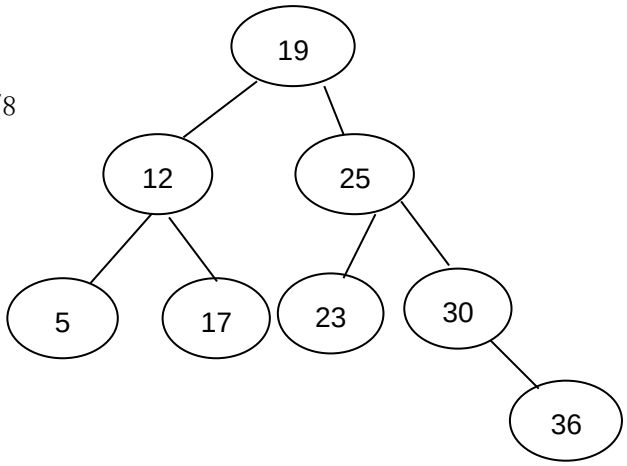






By Jose & Austin

8. 查 30:比较 3 次
成功 SUC:
(1+2*2+3*4+1*4)/8=21/8
失败 USUC:
(3*7+4*2)/9=29/9




三、算法题
1.
（1）基本思想：以 K 为基准记录，使用快速排序的一次分划过程即可，时间复杂度为 O(n)
（2）算法 Part(A,n.A)
/*以 K 为基准元素一次分划*/
P1[ 初始化]
i←1 . j←n.
P2[以 0 分划]
WHILE i<j DO
(
WHILE $K_i$<K AND i<j DO i←i+1 .
WHILE $K_j$>=K AND i<j DO j←j－1.
IF i＜j THEN $R_i \leftrightarrow R_j$ . ) ▉



2.
(1)遍历链表记录学生人数和考试成绩即可计算所有学生平均分
(2）设链表储存方式是

```
struct StuInfo{
    int number;     //学生学号
    int score;      //考试分数
    StuInfo *next;
};
```

则求学生平均分算法 AverageScore 是：

```
float AverageScore(StuInfo *head){
    if(head == NULL)    return 0;
    int sum = 0;        //记录分数和
    int StuNumb = 0;    //记录学生人数
    StuInfo *p = head;
    while(p != NULL){
        StuNUmb ++;
```

```
        sum += p.score;
        p = p->next;
    }
    float x = StuNumb;
    return sum/x;
}
```

3.分析省略。在这里谈一下算法题的大体大致步骤：先给出算法思路；再用 C/C++语言给出算法代码（可以用 ADL 语言写），给出关键步骤的注释；最后必须得分析算法复杂度。尽量给出复杂度低的算法。不必要写头文件和调试函数。

一种参考代码（代码经过调试，答题不应该写一些多余代码）：

```
#include <iostream>
#include <cstdlib>
#include <cstring>
#include <queue>
using namespace std;
typedef int DataType;
struct Tree
{
    Tree* Left;
    DataType data;
    Tree* Right;
    Tree() :Left(NULL),Right(NULL),data(0) {}
    Tree(DataType data) :data(data),Left(NULL),Right(NULL) {}
};

int maxLen=0;
Tree* maxPath[1024];

int len=0;
Tree* path[1024];

void dfs(Tree *cur)
{
    if(cur->Left)
    {
        path[len++]=cur->Left;
        dfs(cur->Left);
        len--;
    }
    if(cur->Right)
    {
        path[len++]=cur->Right;
        dfs(cur->Right);
        len--;
```

```
    }
    if(len>maxLen && cur->Left==NULL && cur->Right==NULL)     //更深的叶子节点
    {
        maxLen=len;
        memcpy(maxPath,path,sizeof(path));
    }
}

void solve(Tree *root)
{
    path[len++]=root;
    dfs(root);
    cout<<"Height: "<<maxLen<<endl;
    cout<<"Path: ";
    for(int i=0;i<maxLen;i++)
    {
        cout<< maxPath[i]->data<<' ';
    }
}

bool insert(Tree *root , const char *path,DataType data)
{
    Tree* cur=root;
    int len=strlen(path);
    for(int i=0 ; i<len-1 ; i++)
    {
        if(path[i]=='L' && cur->Left != NULL)
            cur=cur->Left;
        else if(path[i]=='R' && cur->Right != NULL)
            cur=cur->Right;
        else
            return false;
    }
    if(path[len-1]=='L' && cur->Left == NULL)
    {

        cur->Left=new Tree(data);
    }
    else if(path[len-1]=='R' && cur->Right == NULL)
    {
        cur->Right=new Tree(data);
    }
    else
        return false;
```

```
        return true;

}
void input(Tree* root)
{
        char str[1024];
        int data;
        while(cin>>str && *str!='#' )
        {
                cin>>data;
                if(insert(root,str,data)) cout<<"Good"<<endl;
                else cout<<"Bad"<<endl;
        }
}

int main()
{
        Tree* root=new Tree;//0;
        input(root);
        solve(root);
        return 0;
}
```



# 2014 数据结构 A 参考答案

一、

1-5　错对错错对　　6-10 错对对对对

二、

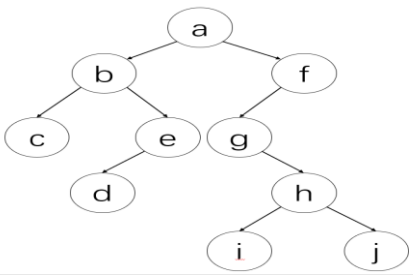
DADBCCABCA

三、1. ab+cde+*-f-gh+*

2.

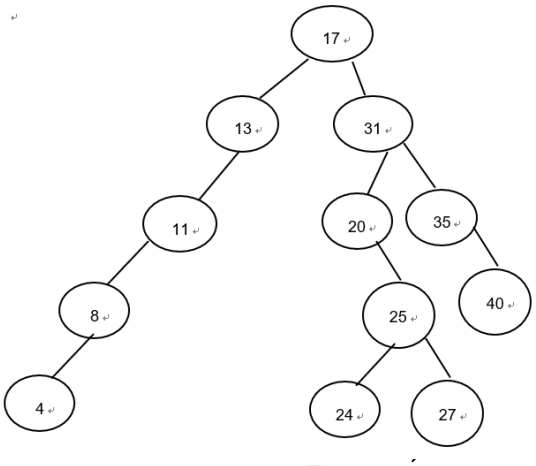
| | i | J | v |
|---|---|---|---|
| 0 | 0 | 3 | 6 |
| 1 | 1 | 0 | 5 |
| 2 | 2 | 1 | 9 |
| 3 | 2 | 3 | 7 |
| 4 | 3 | 3 | 8 |

3.

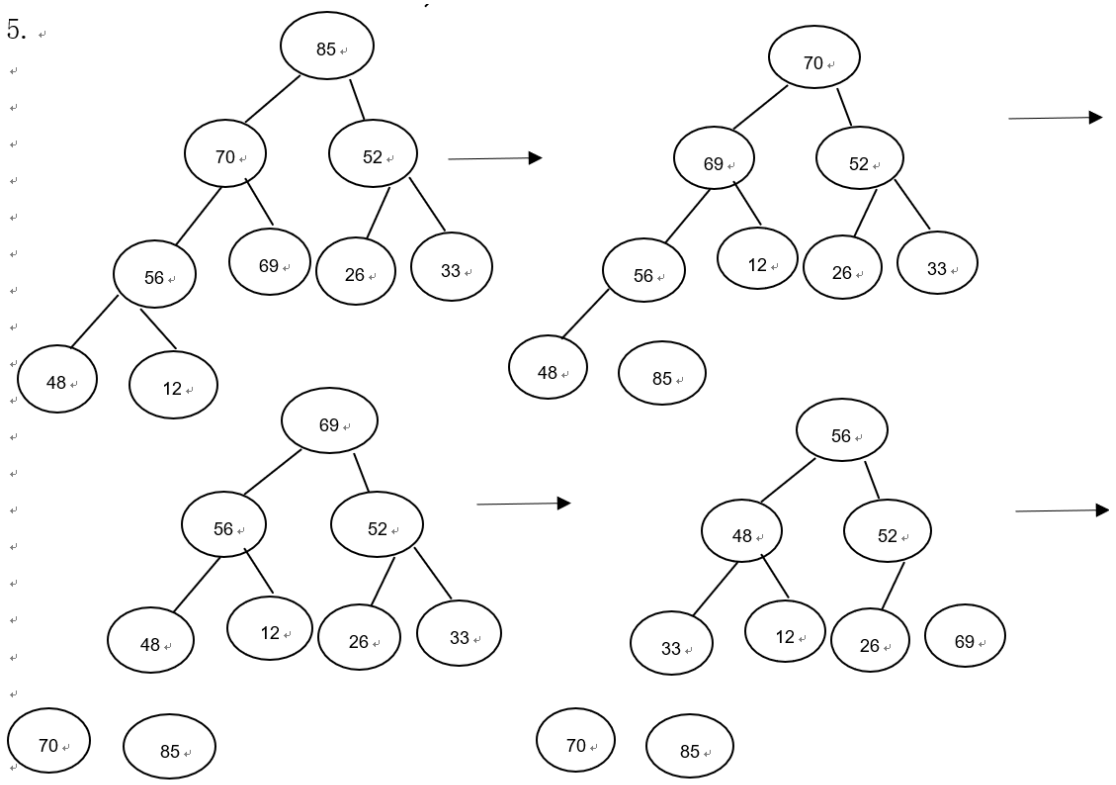


后序序列是：cdebijhgfa

4.

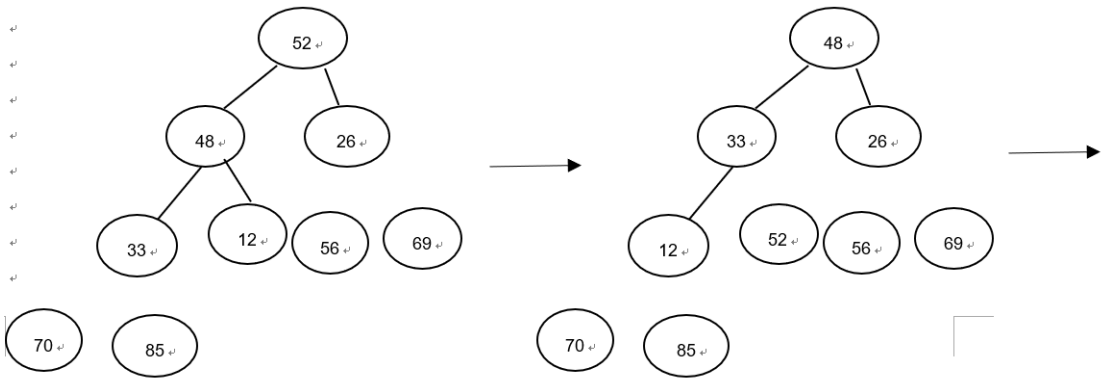


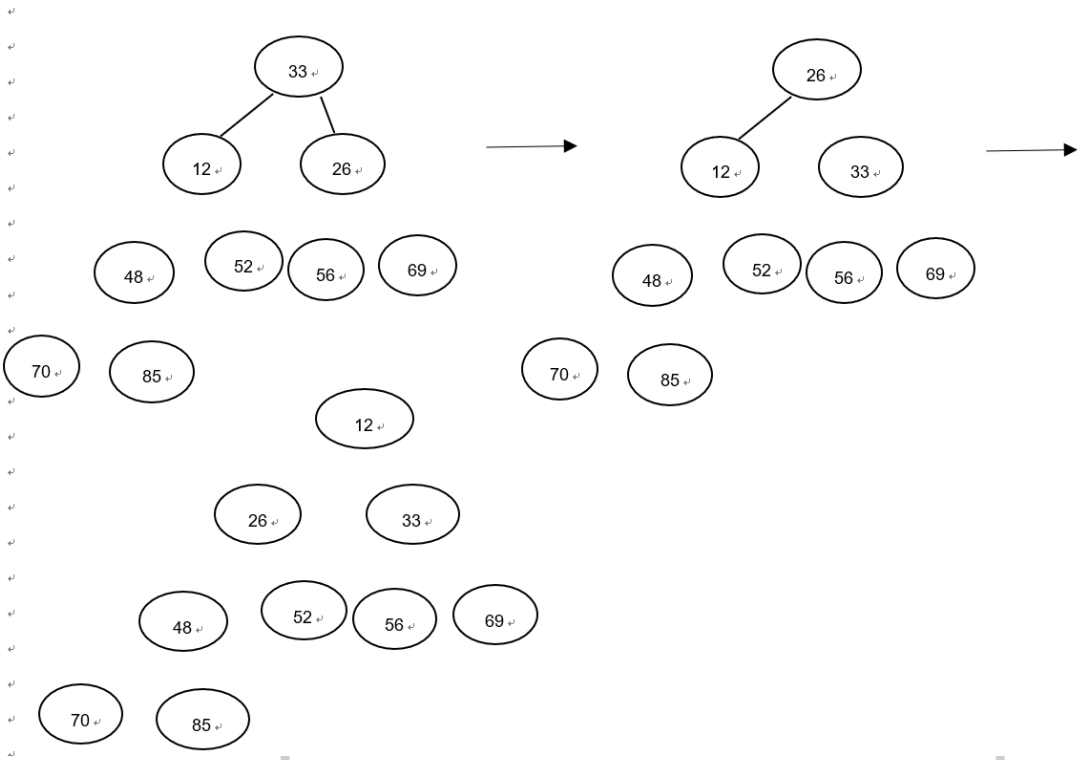

5.














6.参考画法（1）：

| | 点集 | D[1] | D[2] | D[3] | D[4] | D[5] | D[6] | P[1] | P[2] | P[3] | P[4] | P[5] | P[6] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 初始 | {1} | 0 | 20 | 15 | ∞ | ∞ | ∞ | -1 | 1 | 1 | -1 | -1 | -1 |
| 1 | {1,3} | 0 | 19 | 15 | ∞ | ∞ | 25 | -1 | 3 | 1 | -1 | -1 | 3 |
| 2 | {1,3,2} | 0 | 19 | 15 | ∞ | 29 | 25 | -1 | 3 | 1 | -1 | 2 | 3 |
| 3 | {1,3,2,6} | 0 | 19 | 15 | 29 | 29 | 25 | -1 | 3 | 1 | 6 | 2 | 3 |
| 4 | {1,3,2,6,4} | 0 | 19 | 15 | 29 | 29 | 25 | -1 | 3 | 1 | 6 | 2 | 3 |
| 5 | {1,3,2,6,4,5} | 0 | 19 | 15 | 29 | 29 | 25 | -1 | 3 | 1 | 6 | 2 | 3 |
| 6 | {1,3,2,6,4,5} | 0 | 19 | 15 | 29 | 29 | 25 | -1 | 3 | 1 | 6 | 2 | 3 |

| | 最短路径 | 距离 |
|---|---|---|
| 1->2 | 132 | 19 |
| 1->3 | 13 | 15 |
| 1->4 | 1364 | 29 |
| 1->5 | 1325 | 29 |
| 1->6 | 136 | 25 |

注：尽量不使用此画法。应该参考书本 P189 的画法，用 s（用作标记）、dist 和 path 三个数组来表示最短路生成过程。由于时间原因，不在这里显示具体过程，参考书本即可。

三、算法题

1.假设该链表存储方式是

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

则找域值为 i 的 Find_i 算法是

```
struct link * Find_i(struct link *p , int i)
{
    while(p != NULL){
        if(p->data == i)    return p;
        else    p = p->next;
    }
    return NULL;
}
```

2.算法基本思想：采用层次遍历，当一个结点左右子树都为空时即是叶结点

分析算法复杂度：由于采用的类似层次遍历的思想，所以复杂度为 O(n)

```
queue<BinTreeNode *> Q;
int CountTreeNode(BinTreeNode *root){
    if(root == NULL)    return 0;
    Q.push(root);
    if(root ->left ==NULL && root->right == NULL){
        Q.pop();
        return 1;
    }
    int count = CountTreeNode(root->left) + CountTreeNode(root->right);
    Q.pop();
    return count;
}
```



3.

4.算法思想：通过不断删除度为 1 的点来判断

设图的储存结构为邻接矩阵方式，一种参考代码如下：

```
typedef int DataType;
const int MAXN=1024;
int G[MAXN][MAXN],n;
```

算法 solve：

```
int solve()
{
    bool flag=true;
    int cnt=0;
    while(flag)
    {
```

```
        flag=false;
        for(int i=0;i<n;i++)
        {
            int sum=0;
            for(int j=0;j<n;j++)
            {
                if(G[i][j] && i!=j ) sum++;
            }
            if(sum==1)//只有一条边,则删除这个点
            {
                for(int j=0;j<n;j++)
                {
                    G[i][j]=0;
                    G[j][i]=0;
                }
                cnt++;flag=true;//重复
            }
        }
    }
    if(cnt==n-1) return 1;//删除了 n-1 条边,剩下一个孤立点
    else      return 0;
}
```



## 2009《数据结构》期末试题（A 卷）参考答案

一、正确的有:4 5

错误的有:1 2 3

二、BDCBCAB

[5.正确的是(1)(4)(5)]

二、1. 顺序存储结构: 把一组结点存放在地址相邻的存储单元里,结点间的逻辑关系用存储单元的自然顺序关系表达

链接存储结构: 在结点的存储结构中附加指针字段，两个结点的逻辑后继关系用指针的指向来表达。

顺序存储时，相邻数据元素的存放地址也相邻（逻辑与物理统一）；要求内存中可用存储单元的地址必须是连续的。

优点：存储密度大，存储空间利用率高。

缺点：插入或删除元素时不方便。

链式存储时，相邻数据元素可随意存放，但所占存储空间分两部分，一部分存放结点值，另一部分存放表示结点间关系的指针

优点：插入或删除元素时很方便，使用灵活。

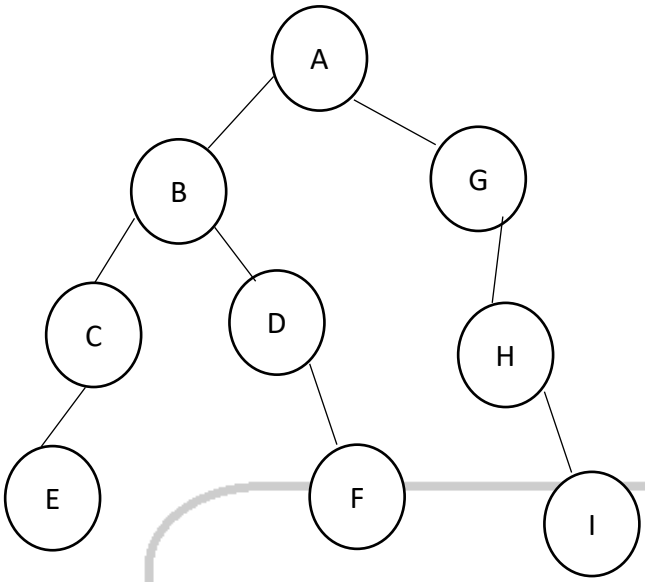
缺点：存储密度小，存储空间利用率低。

2.

前缀: -+*+abc/d-*efgh

后缀：ab+c*def*g-/h-

3.




后序:ECFDBIHGA

4.

48 75 8 50 33 42 86 26 70

33 26 8 42 48 50 86 75 70

8 26 33 42 48 50 86 75 70

8 26 33 42 48 50 70 75 86

8 26 33 42 48 50 70 75 86



5.

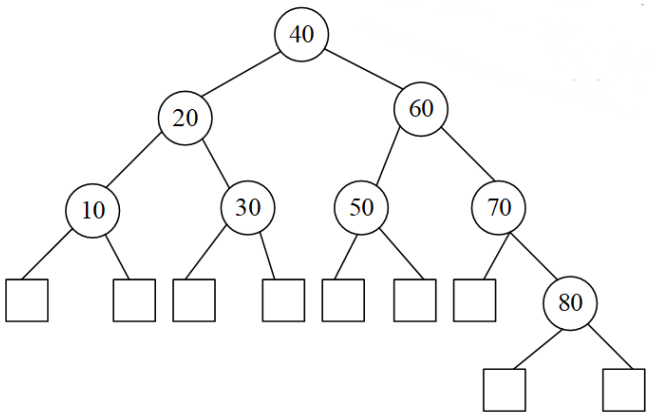
(1)

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| Ve | 0 | 3 | 2 | 6 | 8 | 5 |
| Vl | 0 | 5 | 2 | 6 | 8 | 7 |

| | <A,B> | <A,C> | <B,D> | <B,E> | <C,D> | <C,F> | <D,E> | <F,E> |
|---|---|---|---|---|---|---|---|---|
| e(a) | 0 | 0 | 3 | 3 | 2 | 2 | 6 | 5 |
| l(a) | 2 | 0 | 5 | 7 | 2 | 4 | 6 | 6 |

(2)关键路径为 ACDE

6.




成功：（1+2*2+3*4+4）/8=21/8
失败：（7*3+2*4）/9=29/9

7.
**不推荐下列画法，该做法只是方便给出答案，具体画法以课件画法为准**

| 下标 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 关键字 | 33 | 1 | 13 | 12 | 34 | 38 | 27 | 22 | | | |



8.① count[i]<-1
② STEP-1
③ Kj>Ki
④ count[i]<-count[i]+1

## 四、算法题

**注：每个算法题完成后理论上都要分析算法复杂度，在这我们只给出解题思路，算法复杂度留给读者自己分析**

1．算法 Reverse ( head.head)
/*将指针 head 所指向的链表倒置*/
RV1[空链表]
IF( next(head)=NULL) RETURN.
RV2[指针初始化]
//P1,P2 分别指向两个连续的节点
P1 ← NULL.
P2 ← next(head).
RV3[反转链表]
WHILE( P2 ≠ NULL ) DO
( P3 ← next(p2)
next(P2) ← P1. //反转节点指针
P1 ← P2. P2 ← P3. //移动 3 个指针 ）
RV4[head 指向反转链表]
next(head) ←P1 . ▌

2.算法 Depth(t. d)

//[解题思路 1]　对树做层次遍历，每遍历一层树的深度+1.
D1 [判断 t 是否为 NULL]
IF　t=NULL　THEN　( d← -1 . RETURN )
D2 [创建辅助队列，根结点入队]
CREATE(Q). Q ⇐( t,0) .
D3 [利用队列 Q 遍历第 d 层结点]
WHILE NOT (IsEmpty(Q)) DO
　(　(p,d) ⇐ Q .
　　WHILE p≠NULL DO
　　(　　IF FirstChild(p)≠NULL THEN Q⇐(FirstChild(p),d+1)
　　p←NextBrother(p) .) )

# 唐敖庆/卓越班《数据结构》复习参考题答案



一、判断题
1-5　对对对错错　　6-10　对对对对对
11-15 错对错对错　　16-20 错对对错对　　21-22 对错
二、选择题
1-5 ACADD　　6-10 CBACA　　11-15 DAAAB
16-20　ACBDD　　21-25 DBCCB
**点评：25 和 25 题可能属于知识盲点，需要特别关注一下**
三、简答题
1.参考代码：

```
struct link{
    int data;
    struct link *next;
}; //链表存储结构
struct link *head;                    //链表头指针
int count_x(struct link *p , int x) //p 是传的 head 指针
{
    int counter = 0;                  //记录链表 x 的个数
    while(p != NULL)
    {
        if(p->data == x)              //找到计数器+1
            counter++;
        p = p->next;                  //移动指针
    }
    return counter;
}
```

2.参考代码：

```
void del_zero(){
```

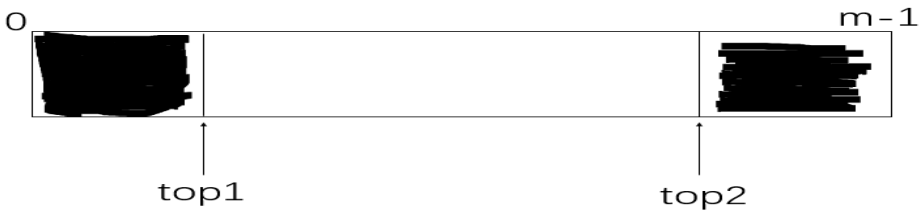
```
    struct link *p , *q;                    //q 是 p 的前驱结点
    if(head == NULL)       return ;
    p = head;
    while(head->data ==0) {                 //找到新 head 位置
        head = head->next;
        free(p);
        p = head;
    }
    q = p;
    while(p != NULL) {                      //在链表中删除 0 元素
        if(p->data == 0){
            q->next = p->next;
            free(p);
            p = q->next;
         }
         else{
            q = p;
            p = p->next;
        }
    }
}
```



3. 123，132，213，231，321

4. 1234

5.如下图：

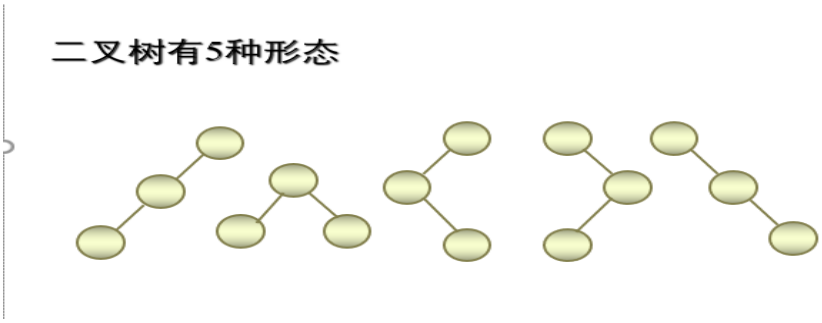



栈满的条件： top1＋1＝top2

栈空的条件： top1＝－1 &&top2＝m

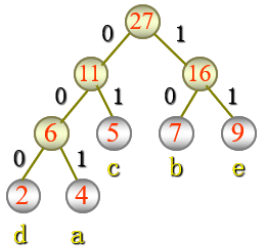
6.画图略，后序序列：CBEFDA

7.




8. 先序序列：abdjcefhi　中序序列：djbaechif　后序序列：jdbeihfca

9.




哈夫曼编码:

a (001)

b (10)

c (01)

d (000)

e (11)

WPL：60

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Ve | 0 | 19 | 15 | 29 | 38 | 43 |
| Vl | 0 | 19 | 15 | 37 | 38 | 43 |

| | <1,2> | <1,3> | <3,2> | <2,4> | <2,5> | <3,5> | <4,6> | <5,6> |
|---|---|---|---|---|---|---|---|---|
| e | 0 | 0 | 15 | 19 | 19 | 15 | 29 | 38 |
| l | 17 | 0 | 15 | 27 | 19 | 27 | 37 | 38 |
| l-e | 17 | 0 | 0 | 8 | 0 | 12 | 8 | 0 |



10.

此工程最早完成时间为 43；关键路径为：<1,3> 、<3,2> 、<2,5> 、<5,6>。

## 考前补漏卷参考答案

一、CCDCDC

二、

1. (9-(2+3)-10/2)*2=-2

2.此题考查 KMP 算法。能否会 KMP 是学没学好数据结构的标志。所以建议大家都会此题。首先你得会写失败函数，然后知道如何滑动字符串的位置。具体答案本作者省略，因为作者学的失败函数与书本的失败函数求法不同。如果有不会 KMP 的请看下面的博客：

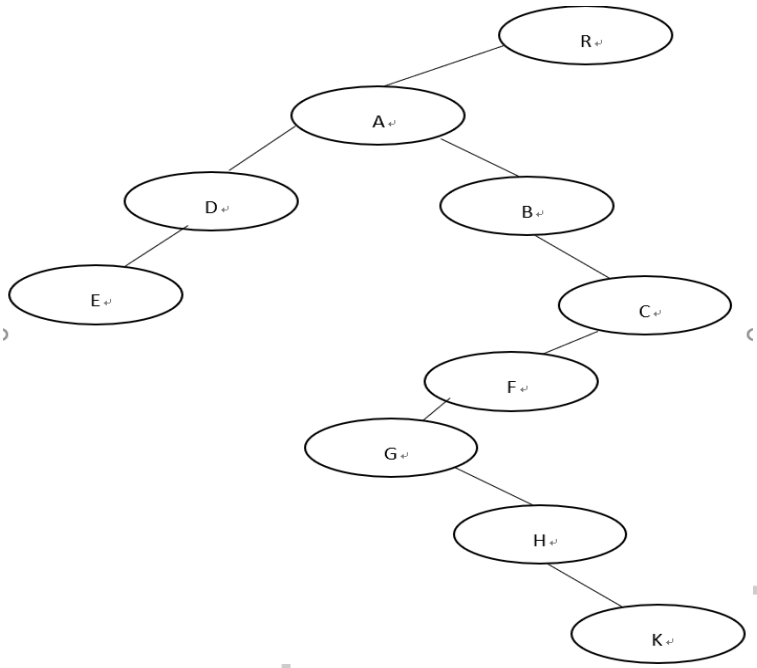
https://blog.csdn.net/starstar1992/article/details/54913261

也可以看这篇

https://www.cnblogs.com/yjiyjige/p/3263858.html

3. 先根序列：RADEBCFGHK　　后根序列：DEABGHKFCR

二叉树:




4.

| | $A^{(-1)}$ | | | | $A^{(0)}$ | | | | $A^{(1)}$ | | | | $A^{(2)}$ | | | | $A^{(3)}$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 3 |
| 0 | 0 | 1 | ∞ | 4 | 0 | 1 | ∞ | 4 | 0 | 1 | 10 | 3 | 0 | 1 | 10 | 3 | 0 | 1 | 9 | 3 |
| 1 | ∞ | 0 | 9 | 2 | ∞ | 0 | 9 | 2 | ∞ | 0 | 9 | 2 | 12 | 0 | 9 | 2 | 12 | 0 | 8 | 2 |
| 2 | 3 | 5 | 0 | 8 | 3 | 4 | 0 | 7 | 3 | 4 | 0 | 7 | 3 | 4 | 0 | 7 | 3 | 4 | 0 | 7 |
| 3 | ∞ | ∞ | 6 | 0 | ∞ | ∞ | 6 | 0 | ∞ | ∞ | 6 | 0 | 9 | 11 | 6 | 0 | 9 | 11 | 6 | 0 |

| | $Path^{(-1)}$ | | | | $Path^{(0)}$ | | | | $Path^{(1)}$ | | | | $Path^{(2)}$ | | | | $Path^{(3)}$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 3 |
| 0 | -1 | 0 | -1 | 0 | -1 | 0 | -1 | 0 | -1 | 0 | 1 | 1 | -1 | 0 | 1 | 1 | -1 | 0 | 3 | 1 |
| 1 | -1 | -1 | 1 | 1 | -1 | -1 | 1 | 1 | -1 | -1 | 1 | 1 | 2 | -1 | 1 | 1 | 2 | -1 | 3 | 1 |
| 2 | 2 | 2 | -1 | 2 | 2 | 0 | -1 | 0 | 2 | 0 | -1 | 0 | 2 | 0 | -1 | 0 | 2 | 0 | -1 | 0 |
| 3 | -1 | -1 | 3 | -1 | -1 | -1 | 3 | -1 | -1 | -1 | 3 | -1 | 2 | 2 | 3 | -1 | 2 | 2 | 3 | -1 |

| | 最短距离 | 最短路径 |
|---|---|---|
| 0->1 | 1 | 01 |
| 0->2 | 9 | 0132 |
| 0->3 | 3 | 013 |
| 1->0 | 12 | 120 |
| 1->2 | 8 | 132 |
| 1->3 | 2 | 13 |
| 2->0 | 3 | 20 |
| 2->1 | 4 | 201 |
| 2->3 | 7 | 203 |
| 3->0 | 9 | 320 |
| 3->1 | 11 | 321 |
| 3->2 | 6 | 32 |

5.

| | 点集 | D[0] | D[1] | D[2] | D[3] | D[4] | P[0] | P[1] | P[2] | P[3] | P[4] |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 初始 | {0} | 0 | 3 | ∞ | ∞ | 30 | -1 | 0 | -1 | -1 | 0 |
| 1 | {0,1} | 0 | 3 | 28 | 11 | 30 | -1 | 0 | 1 | 1 | 0 |
| 3 | {0,1,3} | 0 | 3 | 15 | 11 | 23 | -1 | 0 | 3 | 1 | 3 |
| 2 | {0,1,3,2} | 0 | 3 | 15 | 11 | 23 | -1 | 0 | 3 | 1 | 3 |
| 4 | {0,1,3,2,4} | 0 | 3 | 15 | 11 | 23 | -1 | 0 | 3 | 1 | 3 |

| | 最短距离 | 最短路径 |
|---|---|---|
| 0->1 | 3 | 01 |
| 0->2 | 15 | 0132 |
| 0->3 | 11 | 013 |
| 0->4 | 23 | 0134 |

另一种解法（推荐该解法，上面解法不推荐，此解法是书本上的解法）：

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| s | 0 | 0 | 0 | 0 | 0 |
| dist | 0 | ∞ | ∞ | ∞ | ∞ |
| path | | | | | |

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| s | 1 | 0 | 0 | 0 | 0 |
| dist | 0 | 3 | ∞ | ∞ | 30 |
| path | | $v_0$ | | | $v_0$ |

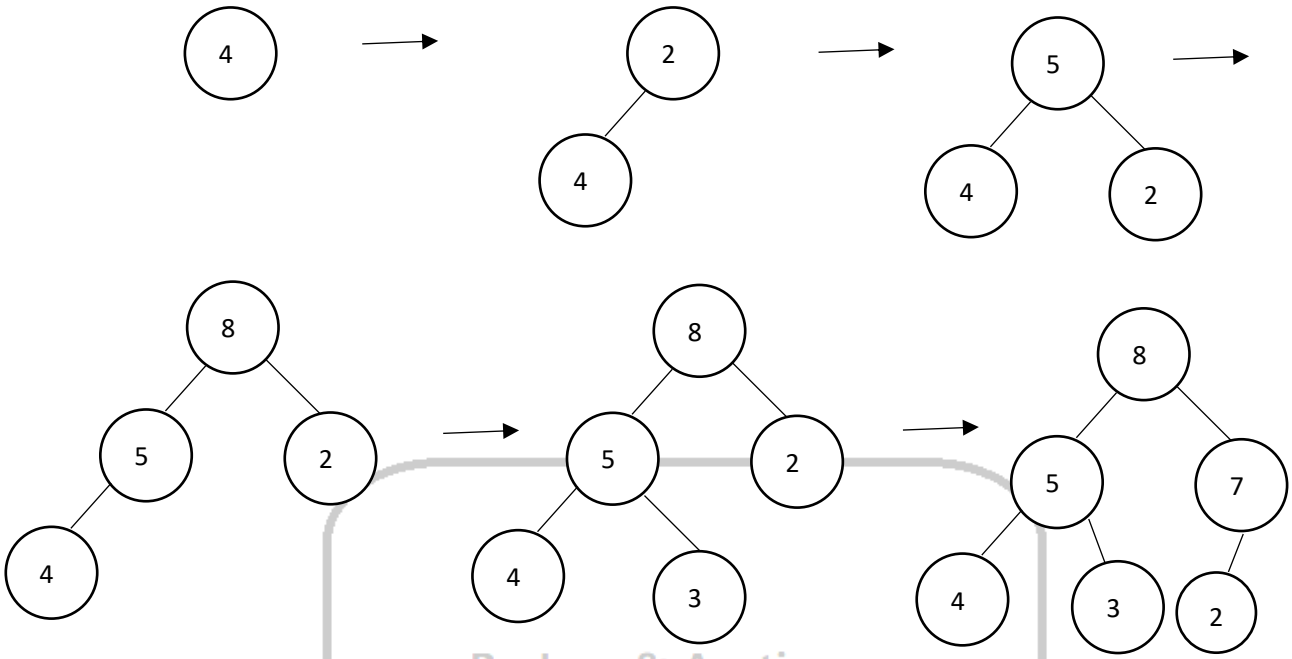
| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| s | 1 | 1 | 0 | 0 | 0 |
| dist | 0 | 3 | 28 | 11 | 30 |
| path | | $v_0$ | $v_1$ | $v_1$ | $v_0$ |

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| s | 1 | 1 | 0 | 1 | 0 |
| dist | 0 | 3 | 15 | 11 | 23 |
| path | | $v_0$ | $v_3$ | $v_1$ | $v_3$ |

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| s | 1 | 1 | 1 | 1 | 0 |
| dist | 0 | 3 | 15 | 11 | 23 |
| path | | $v_0$ | $v_3$ | $v_1$ | $v_3$ |

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| s | 1 | 1 | 1 | 1 | 1 |
| dist | 0 | 3 | 15 | 11 | 23 |
| path | | $v_0$ | $v_3$ | $v_1$ | $v_3$ |

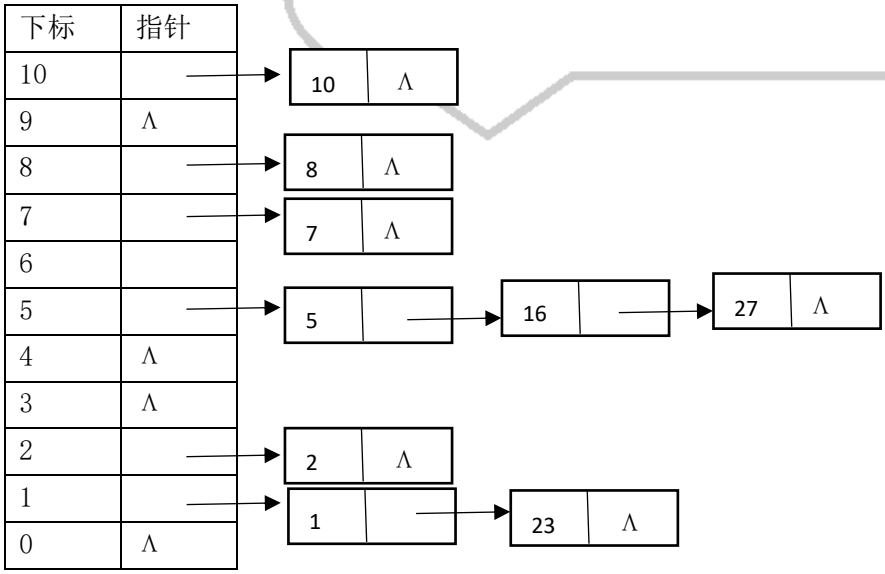
6.






7.

8. 拉链法:




成功: (1*6+2*2+3*1)/9=13/9

失败: 9/11

线性探查法:

| 比较次数 | | 1 | 1 | 3 | | 1 | 2 | 3 | 1 | 3 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 下标 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 关键字 | | 1 | 2 | 23 | | 5 | 16 | 27 | 8 | 7 | 10 |

成功: (1*5+2*1+3*3)/9=16/9

失败：(1+2+3+1+2+3+4+5+6)/10=31/10

三、算法题

1. **算法 Like(t1,t2)**

/*判断两棵二叉树是否相似,t1,t2 表示两棵树的根节点。若相似，返回值为 true，否则为 false*/

L1[递归出口]

IF t1=NULL AND t2=NULL THEN RETURN true.

IF t1=NULL OR t2=NULL THEN RETURN false.

L2[递归调用]

RETURN Like(left(t1),left(t2)) AND Like(right(t1),right(t2)). ▉

时间复杂度为 O(n1+n2)

2. **算法思路：取数组第 m 个数也就是 a[m-1]作为每一次循环的一个基准元，将大于 a[m-1]的全部放在数组的右边，小于 a[m-1]的全部放在数组的左边，如果一直这样循环下去，那么最后 a[m-1]这个单元中存放的就一定是第 m 小的数据。（快排的应用）**

**算法的复杂度为 O(n)。**

```
int findm_min(int a[], int n, int m) //n 代表数组长度,m 代表找出第 m 小的数据
{
    int left, right, privot, temp;
    int i, j;
    left = 0;
    right = n- 1;
    while(left < right)     //快速排序思想
    {
        privot = a[m-1];
        i = left;
        j = right;
        do{
            while(privot < a[j]) j--;
            while(privot > a[i]) i++;
            if(i < j){
                temp = a[i];
                a[i] = a[j];
                a[j] = temp;
                i ++;
                j--;
            }
            if(i == j && a[i] == privot)
                return privot;
        }while(i < j);
        if(j < m-1)       left = i;
        if(i > m-1)       right = j;
    }
    return a[m-1];
}
```

另外可以采用递归写法，具体写法留给读者。写出来可以看到的是递归求 K 小元素代码量比较小，比较容易理解。

**3. 使用并查集方法，思路参考克鲁斯卡尔算法（不推荐使用 Prim 算法）**

**算法复杂度为：**O(eloge)

```
struct edge    //边
{
    int u, v, w; //边的顶点、权值
}edges[MAXM]; //边的数组

int parent[MAXN];    //parent[i]为顶点 i 所在集合对应的树中的根结点
int n, m;    //顶点个数、边的个数
int i, j;    //循环变量
void UFset( )    //初始化
{
    for( i=1; i<=n; i++ )
        parent[i] = -1;
}
int Find( int x ) //查找并返回节点 x 所属集合的根结点
{
    int s; //查找位置
    for( s=x; parent[s]>=0; s=parent[s] );
    while( s!=x ) //优化方案－压缩路径，使后续的查找操作加速。
    {
        int tmp = parent[x];
        parent[x] = s;
        x = tmp;
    }
    return s;
}

//将两个不同集合的元素进行合并，使两个集合中任两个元素都连通
void Union( int R1, int R2 )
{
    int r1 = Find(R1), r2 = Find(R2); //r1 为 R1 的根结点，r2 为 R2 的根结点
    int tmp = parent[r1] + parent[r2]; //两个集合结点个数之和(负数)
    //如果 R2 所在树结点个数 > R1 所在树结点个数(注意 parent[r1]是负数)
    if( parent[r1] > parent[r2] ) //优化方案－－加权法则
    {
        parent[r1] = r2;
        parent[r2] = tmp;
    }
    else
```

```
    {
        parent[r2] = r1;
        parent[r1] = tmp;
    }
}
bool cmp( edge a, edge b ) //实现从大到小排序的比较函数
{
    return a.w  >=  b.w;
}
void Kruskal( )
{
    int sumweight = 0;   //生成树的权值
    int num = 0;   //已选用的边的数目
    int u, v;   //选用边的两个顶点
    UFset( ); //初始化  parent[]数组
    for( i=0; i<m; i++ )
    {
        u = edges[i].u; v = edges[i].v;
        if( Find(u) != Find(v) )
        {
            printf( "%d %d %d\n", u, v, edges[i].w );
            sumweight += edges[i].w; num++;
            Union( u, v );
        }
        if( num>=n-1 ) break;
    }
    printf( "weight of MST is %d\n", sumweight );       //输出权值
}
```

使用这个算法得给边权值排序（调用系统的排序函数即可）：

```
    sort(edges,edges+m,cmp);     //cmp 函数为想要排序方式
```

**数据结构的算法题一般是线性表、树和图这三章各出一道题目，偶尔会涉及到排序和查找问题。树的算法题大都围绕遍历出题，图的算法题大都与遍历和最短路相关。所以要掌握深度遍历和广度优先遍历算法。**

# 2017 级卓班/唐班数据结构试题答案

Please wait!
We are working on it!