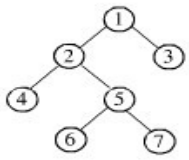
- (2 分)为解决计算机和打印机速度不匹配问题，通常设置一个打印数据缓冲区，主机将要输出的数据依次写入缓冲区，而打印机依次从该缓冲区中取出数据．该缓冲区的逻辑结构应该是?

  A. 栈　　B. 队列　　C. 树　　D. 图
- (2 分)设栈 S 和队列 Q 的初始状态均为空，元素 abcdefg 依次进入栈 S．若每个元素出栈后立即进入队列 Q．且 7 个元素出对的顺序是 bdcfeag，则栈 S 的容量至少是?

  A．1　　B: 2　　C. 3　　D, 4

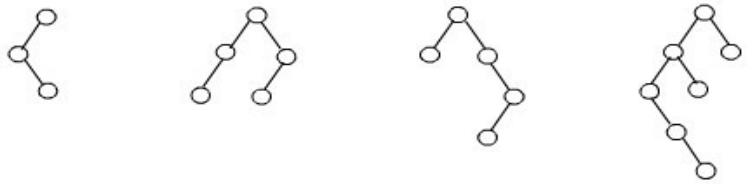
3．给定二叉树油图所示。设N代表二叉树的根，L代表根结点的左子树，R代表根结点的右子树。若遍历后的结点序列为3, 1, 7, 5, 6, 2, 4，则其遍历方式是____。



A．LRN　　B．NRL　　C．RLN　　D．RNL

4．下列二叉排序树中，满足平衡二叉树定义的是____。

A.　　B.　　C.　　D.



- (2 分)已知完全二叉树的第六层(根节点视为第一次)有 8 个节点．则此完全二叉树节点个数最多为

  A.　39　　B. 52　　C. 111　　D. 119
- 将森林转换为对应的二叉树．若在二叉树中节点 u 是节点 v 的父节点的父节点．则在原来的森林中, u 与 v 的可能关系为

  甲) 父子关系.　　乙)兄弟关系　　丙) u 的父节点与 v 的父节点是兄弟关系

  A. 只有 甲　　B. 甲 和 乙　　C. 甲 和 丙　　D 甲乙丙
- 下面关于无向连通图特性的叙述中，正确的是:

  甲) 所有定点度数之和为偶数.　　乙) 边数大于顶点个数减 1

  丙) 至少有一个顶点的度为 1.

  A.　只有 甲　　B.只有 乙　　C.甲 和 乙　　D.甲 和 丙

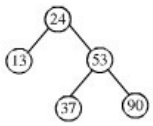
参考答案:

1)B　2)C　3)D　4)B　5)C　6)B　7) A

- 下面叙述中，不符合 m 阶 B 树定义要求的是:
  A. 根节点最多有 m 棵子树　　B.所有叶节点都在同一层上.
  C.各节点内关键字均升序或降序排列　　D. 叶节点之间通过指针链接.
- 已知关键字序列 5, 8, 12, 19, 28, 20, 15, 22 为极小堆(小根堆，最小堆). 添加关键字 3 调整后得到的极小堆是:
  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
- 若数据元素序列 11,12,13,7,8,9,23,4,5 是采用下列排序算法之一得到的第二趟排序后的结果，则该排序算法只能是:
  A. 冒泡排序　　B. 插入排序　　C.选择排序　　D.二路归并排序
- 如元素 abcdef 依次进栈，允许进栈出栈操作交替进行，但不允许连续三次退栈. 则不可能得到的出栈序列为:
  A. dcebfa　　B. cbdaef　　C.bcaefd　　D. afedcb
- 某队列允许在其两端进行入队操作，但仅允许在一端进行出队操作. 若元素 abcde 依次入队后再进行出队操作，则不可能的出队序列为
  A. bacde　　B. dbace　　C.dbcae　　D. ecbad
-

参考答案:
1)D 2)A 3)B 4) D　5)C

4. 在右图所示的平衡二叉树中，插入关键字48后得到一棵新平衡二叉树。在新平衡二叉树中，关键字37所在结点的左、右子结点中保存的关键字分别是____。



A．13、48　　B．24、48　　C．24、53　　D．24、90

5. 在一棵度为4的树T中，若有20个度为4的结点，10个度为3的结点，1个度为2的结点，10个度为1的结点，则树T的叶节点个数是____。

A．41　　B．82　　C．113　　D．122

6. 对n(n≥2)个权值均不相同的字符构造哈夫曼树。下列关于该哈夫曼树的叙述中，错误的是____。

A．该树一定是一棵完全二叉树

B．树中一定没有度为1的结点

C．树中两个权值最小的结点一定是兄弟结点

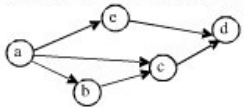
D．树中任一非叶结点的权值一定不小于下一层任一结点的权值

CBA

7. 若无向图G=(V,E)中含7个顶点，要保证图G在任何情况下都是连通的，则需要的边数最少是____。

A．6　　B．15　　C．16　　D．21

8. 对下图进拓扑排序，可以得到不同的拓扑序列的个数是____。



A．4　　B．3　　C．2　　D．1

9. 已知一个长度为16的顺序表L，其元素按关键字有序排列。若采用折半查找法查找一个L中不存在的元素，则关键字的比较次数最多是____。

A．4　　B．5　　C．6　　D．7

CBB

- 采用递归方式对顺序表进行快速排序. 下列关于递归次数的叙述中, 正确的是:
  A. 递归的次数与初始数据的排列次序无关.
  B. 每次划分后先处理较长的区间可以减少递归次数;
  C. 每次划分后先处理较短的区间可以减少递归次数;
  D. 递归次数与处理划分后得到的区间的次序无关.
- 对一组数据(2,12,16,88,5,10)进行排序. 如果前三趟排序结果如下
  第一趟(2,12,16,5,10,88)
  第二趟(2,12,5,10,16,88)
  第三趟(2,5,10,12,16,88)
  则采用的排序算法可能是:
  A. 冒泡排序　　B. 希尔排序　　C.归并排序　　D. 基数排序

DA

1. （　）数据的逻辑结构是指数据的各数据项之间的逻辑关系。
2. （　）KMP 算法的特点是在模式匹配时指示主串的指针不会变小。
3. （　）强连通分量是无向图的极大强连通子图。
4. （　）查找相同结点的效率折半查找总比顺序查找高。
5. （　）求 n 个数中最大的 k(k<<n)个数，起泡排序比直接选择排序要好。
6. （　）平衡二叉树(AVL 树)的中序遍历值是递增的。
7. （　）外排中使用置换选择排序的目的，是为了增加初始归并段的长度。
8. （　）链表的每个结点都恰好有一个指针。
9. （　）用六叉链表表示 30 个结点的六叉树，则树中共有 151 个空指针。
10. （　）若完全二叉树的某个结点没有左子，则此结点必是叶子结点。

批注 [曹魏腾1]: 应为数据元素间的关系

批注 [曹魏腾2]: 有向图

批注 [曹魏腾3]: 单链表

FTFFF
TFFTT

11. （　）栈和队列都是线性表，只是在插入和删除时受到了一些限制。
12. （　）数据的逻辑结构与数据元素本身的形式和内容无关。
13. （　）若把堆看成是一棵完全二叉树，则该树一定是一棵二叉排序树。
14. （　）若装填因子 α 为 1，则向散列表中散列元素时一定会产生冲突。
15. （　）霍夫曼树的所有子树也均是霍夫曼树。
16. （　）平衡二叉树(AVL 树)的中序遍历值是递增的。
17. （　）若有向图不存在回路，即使不用访问标志位同一结点也不会被访问两次。
18. （　）归并排序在任何情况下都比所有简单排序速度快。
19. （　）外排中使用置换选择排序的目的，是为了增加初始归并段的长度。
20. （　）任何一个关键活动提前完成，则整个工程也会提前完成。

批注 [曹魏腾4]: 都是线性表，都是逻辑结构

TTFFT
TFFTF

1.（　）在对线性表的插入、删除操作较多，随机访问较少的情况下，采用链式存储结构优于顺序存储结构。
2.（　）线性表的逻辑顺序与存储顺序总是一致的。
3.（　）顺序存储方式只能用于存储线性结构。
4.（　）顺序存储的线性表可以按序号随机存取。
5.（　）非空循环链表中每一个元素都有后继。
6.（ F ）在对队列做出队操作时，不会改变 front 指针的值。
7.（ F ）数据结构包含数据的逻辑结构、数据的存储结构以及数据集合上定义的运算。
8.（　）若一个树叶是某二叉树的先序遍历序列最后一个结点，则它必是该二叉树的中序遍历序列最后一个结点。
9.（　）已知一棵树的先序序列和后序序列，一定能构造出该树。
10.（　）字符串是数据对象特定的线性表。

批注 [曹魏腾5]: 明显错了 树也可以用线性结构

批注 [曹魏腾6]: front++，不会改变 rear

批注 [曹魏腾7]: 不包括存储结构

批注 [曹魏腾8]:

TFFTT
FFFTT

选择题:

1. 在数据结构中，从逻辑上可以把数据结构分成（　　）
A．动态结构和静态结构　　B．紧凑结构和非紧凑结构
C．线性结构和非线性结构　　D．内部结构和外部结构

2．线性表若采用链式存储结构时，要求内存中可用存储单元的地址（　　）。
A.必须是连续的　　B.部分地址必须是连续的
C.一定是不连续的　　D.连续或不连续都可以

3. 下列数据中，（　　）不是线性数据结构。
A．队列　　B. 栈　　C. 完全二叉树　　D. 循环队列

4. 除了考虑存储数据结构本身所占用的空间外，实现算法所用辅助空间的多少称为（　　）
A．时间效率　　B. 空间效率　　C. 硬件效率　　D. 软件效率

5．带头结点的单链表 h 为空的判断条件是（　　）
A.h==NULL　　B. h->next==h
C. h->next==NULL　　D. h!=NULL

6、以下算法的时间复杂度为_____。

```
for(i=1;i<=100;i++)
  for(j=i;j<=1000;j++)
      x=x+1;
```

（A）O(1)　（B）O(n)　（C）O(n^2)（D）O(n^3)

7、数据元素是数据的基本单位，其内_____数据项。
（A）只能包括一个　（B）不包含
（C）可以包含多个　（D）必须包含多个

8、线性表的链式存储和顺序存储相比，最有利于进行_____。
（A）查找（B）表尾插入或删除（C）按值插入（D）表头插入或删除

9、在一个单链表中，如果要在 p 所指向的节点之后插入一个新的节点，则需要相继修改_____个节点的指针域的值。
（A）1　（B）2　（C） 3　（D）4

10、栈的插入和删除操作在 _____进行。

（A）栈顶 （B）栈底（C）任意位置（D）指定位置

11、假定一个不设队列长度变量的顺序队列的队首和队尾指针分别为 f 和 r，则判断队空的条件是_____。

（A）f+1 = = r （B）r+1 = = f （C）r = = f （D）f = = 0

12、在一棵树中，每个节点最多有_____个前驱节点。

（A）0 （B）1 （C）2 （D）任意多个

13、二叉树的中序遍历的顺序是_____。

（A）根结点，左子树，右子树 （B）左子树，根结点，右子树

（C）右子树，根结点，左子树 （D）左子树，右子树，根结点

14、在一棵有 35 个结点的完全二叉树中，该树深度是_____。

（A）5 （B）6 （C）7 （D）8

15、顺序查找适用于存储结构为_____的线形表

（A）顺序存储 （B）链式存储

（C）顺序存储或链式存储 （D）索引存储

16、设有两个串 p 和 q，其中 q 是 p 的子串，求 q 在 p 中首次出现的位置的算法称为(　　)。

（A）求子串 （B）联接 （C）模式匹配 （D）求串长

17、具有 10 个记录的序列，采用冒泡排序最少的比较次数为_____ 。

（A）1 （B）100 （C）9 （D）55

18、快速排序在_____ 情况下最不利于发挥其长处。

（A）要排序的数据量太大 （B）要排序的数据中含有多个相同值

（C）要排序的数据已经基本有序 （D）要排序的数据是逆序

19、一组记录的排序关键字为（46，79，56，38，40，84），利用快速排序方法，以第一个记录为基准得到的第一次划分的结果为_____ 。

（A）38，40，46，56，79，84 （B）40，38，46，79，56，84

（C）40，38，46，56，79，84 （D）40，38，46，84，56，79

20、对 n 个元素进行直接选择排序，需要进行_____ 趟选择和交换。

（A）n （B）n+1 （C）n-1 （D）n/2

21、折半查找一个具有 n 个数据元素的线性表，其时间复杂度为_____。

（A）O(n) （B）O(n^2) （C）O($\log_2 n$) （D）O(n^3)

22、在一棵非空二叉树的中序遍历序列中，根结点的右边_____ 。

（A）只有右子树上的所有结点 （B）只有右子树上的部分结点

（C）只有左子树上的所有结点 （D）只有左子树上的部分结点

23、采用顺序查找法检索长度为 n 的线性表，则检索每个元素的平均比较次数为_____。

（A）n （B）n/2 （C）(n+1)/2 （D）(n-1)/2

24、采用顺序查找时，设置“监测哨”的好处是_____。

（A）降低平均查找长度 （B）查找失败时，不必每次判断表是否检测完，统一算法

（C）对于查找成功的情况，可以提高效率 （D）可以使算法对存储没有要求

25、在一棵度为 3 的树中,度为 3 的结点个数为 2,度为 2 的结点个数为 1,度为 1 的结点个数为 0,则度为 0 的结点个数为_____。

A．4 B．5 C．6 D．7

26、快速排序在平均情况下的时间复杂度为 _____ 。

（A）O(n) （B）O(n^2) （C）O($n\log_2 n$) （D）O(n^3)

27、每次从无序表中挑选出一个最大或最小元素，把它交换到有序表的一端，这种排序方法

叫做_____。

（A）插入排序 （B）交换排序 （C）选择排序 （D）冒泡排序

28、快速排序是一种（ ）。

A.插入排序 B.交换排序 C. 选择排序 D.归并排序

29、将一棵有 100 个结点的完全二叉树，从根这一层开始，每一层从左到右依次对结点编号，根结点的编号为 1，则编号为 49 的结点的双亲的编号为_____ 。

（A）24 （B）25 （C）23 （D）无法确定

30、深度为 6（根节点的层次为 1 ）的二叉树至多有_____个结点。

（A）64 （B）63 （C）31 （D）32

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
C	D	C	B	C	A	C	D	B	A	C	B	B	B	C	C	C	C	C	C

21	22	23	24	25	26	27	28	29	30
C	A	C	B	C	C	C	B	A	B

1. 六个**不同**元素依次进栈，能得到_____种不同的出栈序列。

   A. 42　　**B.** 82　　**C.** 132　　**D.** 192

2. 数据对象是具有相同特性的_____的集合，它是数据的子集。

   A. 数据项　　**B.** 数据元素　　**C.** 数据对象　　**D.** 数据结构

3. 含有 4 个元素值**均不相同**的结点的二叉排序树有_____种。

   A. 4　　**B.** 6　　**C.** 10　　**D.** 14

4. 有 345 个元素的有序表，等概率顺序查找成功的平均查找长度为_____。

   A. 173　　**B.** 172　　**C.** 86　　**D.** 345

5. 一棵 Huffman 树共有 215 个结点，对其进行 Huffman 编码，共能得到_____个**不同**的码字；

   A. 107　　**B.** 108　　**C.** 214　　**D.** 215

6. 一个具有 n 个顶点的无向图，最少有_____条边就**一定**连通。

   A. n-1　　**B.** n　　**C.** n(n-1)/2　　**D.** (n-1)(n-2)/2+1

7. 若一个有向图的邻接矩阵是_____，则该有向图**一定**存在拓扑排序。

   A. 稀疏矩阵　　**B.** 对称矩阵　　**C.** 对角矩阵　　**D.** 三角矩阵

8. 选用时间最快的稳定排序算法是_____。

   A. 归并排序　　**B.** 堆排序　　**C.** 快速排序　　**D.** 简单插入排序

9. 18 个初始归并段进行 5 路平衡归并，需要增加_____个虚拟归并段。

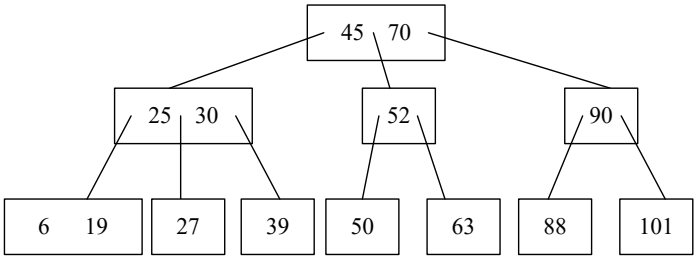
   A. 1　　**B.** 2　　**C.** 3　　**D.** 4

10. 一棵 m 阶的 B+树中，每个结点**最多**有_____棵子树。

    A. m/2　　**B.** m　　**C.** m-1　　**D.** m+1

CBDAB
DDACB

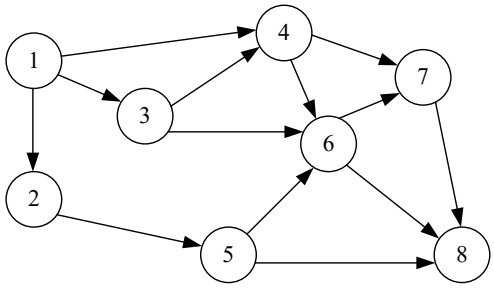
一. 已知一棵 3 阶的 B－树初态如下所示，若此后依次向此树中插入关键字 55、61、7，然后再删除关键字 50、101，请画出**每一步**执行后 B－树的状态。



二. 请回答以下关于有向图的一些问题：

（1）某有向无环图如下所示，请写出该图全部的拓扑序列；

（2）对于一个有向图，不用拓扑排序，如何判断图中是否存在环？



三． 已知一组待排关键字序列{**83120，52366，7037，43126，50921，5731，83265，73192，8235，49198**}，试根据基数排序原理写出每一趟排序的结果。

四． 求模式串 P='decddecgdecdegf'的 next 和 nextval 数组各元素的值，填入下表中。

J	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
P	d	e	c	d	d	e	c	g	d	e	c	d	e	g	f
next[j]															
nextval[j]															

五． 设已知散列表的地址空间为 0-10，散列函数为 H(K)=K MOD 11，解决冲突的方法为线性探测再散列法，试将下列关键字集合{22,41,53,33,46,30,13,01,67}依次插入到如下所示的散列表中。并分别求出等概率下查找成功时和查找失败时的平均查找长度.

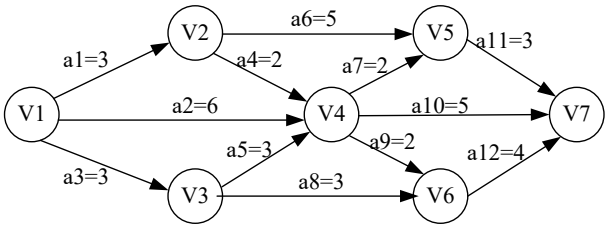
散列地址	**0**	**1**	**2**	**3**	**4**	5	**6**	**7**	**8**	**9**	**10**
关键字											

六． 已知一棵树的先根遍历序列和后根遍历序列如下所示，请画出这颗树对应的二叉树。

先根遍历序列：A B C D E F G H I J K L

后根遍历序列：B D E F C G J K I L H A

七． 已知一图如下图所示，以 v1 为源点, 以 v7 为汇点，求出所有事件允许发生的最早时间和最晚时间填入下表中，并求出所有的关键路径；



事件	V1	V2	V3	V4	V5	V6	V7
Ve(j)							
Vl(j)							

八． 已知采用某种内部排序方法对关键字序列{3，87，12，61，70，97，26，45}进行排序，可得到如下几组中间结果，请问该方法为何种排序法，并利用该排序方法填写空白处的内容。

初态：3，87，12，61，70，97，26，45

（1）97，87，26，61，70，12，3，45

（2）45，87，26，61，70，12，3，97

（3）3，70，26，61，45，12，87，97

（4）12，61，26，3，45，70，87，97

（5）_____

（6）_____

（7）_____

（9）_____

（9）3，12，26，45，61，70，87，97

九) 现有关键字序列(m，g，b，e，d，c，a，p，n，h，q，k)，按此顺序建立平衡二叉树，画出此树，并求在等概率情况下查找成功的平均查找长度。

十) 已知有向网络的邻接矩阵为：

顶点	A	B	C	D	E
A	0	15	30	∝	∝
B	∝	0	10	8	∝
C	∝	∝	0	15	10
D	20	∝	∝	0	8
E	∝	∝	∝	∝	0

1. 画出此图；
2. 求该图的强连通分量；
3. 从顶点 A 出发用 DFS 和 BFS 法遍历图，分别写出遍历后的顶点序列并画出 DFS 生成树和 BFS 生成树；

用 Dijkstra 算法求出从源点 A 到其它各点的最短路径及它们的长度。

十一) 已知一组记录的关键字为{18,2,10,6,78,56,45,50,110,8}. 按输入顺序画出此组记录的平衡二叉树，求等概率情况下查找成功的平均查找长度。若查找每个元素的概率不等，此时的平衡二叉树是否是最佳查找树？为什么？

十二) 有字符集合{A，B，C，D，E，F，G，H}，各字符的使用频度依次为{12，8，2，35，5，15，20，3}，设计它们的赫夫曼编码，并求相对于这些字符采用等长编码，传输的压缩率是多少？

十三) 求字符串 " aabcdaaa " 的 next 和 nextval 数组值。

- (10 分)求 s 到 t 的加权最短路径(权值非否)算法如下

  u = s;
  while(u != t)
  {
  选择从 u 出发的权值最小的弧边 e = (u,v);
  u = v
  }
  请问上述算法是否正确. 若正确请给出证明, 若不正确, 请给出反例.
- (10 分) 设计一个函数, 返回带头结点的单链表的倒数第 k(>=0) 个结点的值. (假设该链表至少有 k 个结点, 函数不能修改原链表)
  甲)描述算法思想
  乙)描述实现步骤
- 将 n(>0) 个整数存放在数值 R 中. 试设计一个算法在时间和空间两个方面尽可能高的算法, 将 R 中元素循环左移 p (0<p<n) 个位置. 即将 R 中的数据有(x_0, x_1,… ,x_{n-1}) 变换为(x_p,x_{p+1}, …x_{n-1},x_0x_1…x_{p-1}) . 要求给出算法的基本思想; 根据设计思想, 采用 C/C++/Java 语言描述算法; 关键处给出注释; 说明算法的时间和空间复杂度.

  参考答案:

  (1) 先将 R 就地倒置. 在将前后两段分别倒置.

  （2）算法实现

```
void Reverse(int r[ ],int left,int right)
{
    int k=left,j=right,temp;            /* k 等于左边界 left，j 等于右边界 right */
    while(k<j){
        temp=r[k];
        r[k]=r[j];
        r[j]=temp;
        k++;                            /* k 右移一个位置 */
        j--;                            /* j 左移一个位置 */
    }
}
void LeftShift(int r[ ],int n,int p)
{
    if(p>0 && p<n){
        Reverse(r,0,n-1);               /* 将全部数据逆置 */
        Reverse(r,0,n-p-1);             /* 将前 n-p 个元素逆置 */
        Reverse(r,n-p,n-1);             /* 将后 p 个元素逆置 */
    }
}
```

  （3）说明算法复杂性：上述算法的时间复杂度为 O(n)，空间复杂度为 O(1)。

- 试编写算法，对一棵以孩子—兄弟链表表示的树统计叶子节点的个数。

```
typedef    struct    node {
    etype    data;
    node    *child,   *sibling;
```

```
} node,    *bitptr;

int   countleaf（ bitptr   root ）//root 为根节点，函数返回值为叶子结点个数
{
      if(root==NULL)
            return(0);
      else if(t->child==NULL)   //左子空，结点必为叶子
            return(1+Count(t->nextsibling))；
            //1（当前结点为叶子）+兄弟子树上的叶子结点个数
      else
            return(Count(t->firstchild)+Count(t->nextsibling));
            //左子树的叶子个数+兄弟子树的叶子个数
}
```

算法思想：统计以孩子—兄弟链表表示的树的叶子数目，及求对应二叉树中没有左孩子结点的数目，可采用递归来解决，递归模型如下：若结点空，叶子数目=0；若左子空，则叶子数目=1+兄弟子树上的叶子结点个数；若左子不空，则叶子数目=左子树的叶子个数+兄弟子树的叶子个数。

- 已知由整数组成的、单调递增的有序表以带头结点的单链表存储，试编写算法将链表中元素值大于 n 且小于 m 的部分**就地**置逆（n>m），其它部分保持不变。

- 假设一棵平衡二叉树的每个结点都标明了平衡因子 bf,试设计一个算法，利用 bf 的值求平衡二叉树的高度。

```
 typedef   struct   node
 {
     int    bf, data；
     struct   node *left，*right;
  } BiTNode，*BSTree；

 int    height(BSTree t)  //函数返回值为平衡二叉树 t 的高度
```

- 若待排序列用带头结点的单链表存储，试给出简单选择排序算法。**（15 分）**

```
typedef    struct    node
{
    int       key;
    node     *next;
} node,    *pointer;

void selectsort(pointer &h)//h 为头指针
```

- 已知一棵二叉树的中序遍历序列和按层次遍历的序列，试编写算法生成此二叉树的二叉链表。
- 设整型数组[a_0 .. a_{n-1}]中的数据呈随机分布，利用快速排序的思想设计算法求其中第 k 小元素，要求平均时间复杂度为 *O(n)*。

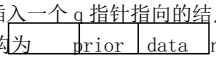
```
int k_th(int a[], int n, int k)
//返回第 k 小元素
```

模拟试题

一．选择题：选择正确的答案填入下面的_____中。（10 分）

1．长度为 n 的顺序存储线性表，当在任何位置上插入或删除一个元素的概率都相等时，插入一个元素所需移动元素的平均个数为 A:_____，删除一个元素所需移动元素的平均个数为 B:_____。

A、B：①(n-1)/2　②n　③n+1　④n-1　⑤n/2　⑥(n+1)/2

2．在双向循环链表 p 指针指向的结点之前插入一个 q 指针指向的结点，其修改指针的操作为_____。（注：双向链表结点的结构为 

其中 prior 与 next 分别为指向前驱结点和后继结点的指针）

①p->prior=q; q->next=p; p->prior->next=q; q->prior=p->prior;

②p->prior=q; p->prior->next=q; q->next=p; q->prior=p->prior;

③q->next=p; q->prior=p->prior; p->prior->next=q; p->prior=q;

3．下列排序方法中时间复杂度为 0(nlog$_2$n)，且使用辅助空间最少的是 A:_____；时间复杂度不受待排序序列的初始状态影响，恒为 0(n^2)的是 B:_____。

A、B：①堆排序　②冒泡排序　③快速排序　④希尔排序

⑤直接插入排序　⑥简单选择排序

4．顺序文件在数据量很大的情况下适合于____。

①按关键字存取　②直接存取　③成批处理　④随机存取

5．二维数组 a[0..8][1..10]中，每个数组元素占用 6 个存储单元，a 的第 8 列和第 5 行共占用 A:_____个存储单元。若 a 按行优先存放，元素 a[8][5]的起始地址与 a 按列优先存放时元素 B:_____的起始地址相同。

A：①108　②114　③54　④60　⑤160

B：①a[8][5]　②a[3][10]　③a[5][8]　④a[0][9]

6．假设某文件经过内部排序得到 27 个初始归并段，若要使多路归并 3 趟完成，则应取归并的路数为_____。

①2　②3　③4　④5

7．倒排文件含有若干个倒排表，倒排表的每一个元素是_____。

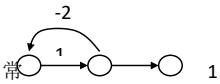
①一个主关键字值及其记录地址

②一个次关键字值及具有此次关键字值的记录个数

③一个次关键字值及所有具有此次关键字值的记录地址

二．回答下列问题（10 分）

1. 设 T 是一棵二叉树，共有 11 个结点，除叶子结点外其它结点的度数皆为 2，试问 T 的最大可能高度 h_{max} 和最小可能高度 h_{min} 是多少？T 中共有多少非叶结点？
2. 设目标串 s='abaabcdabfgp'，模式串 t='abcd'。试说明简单的模式匹配过程。设目标串长为 n，模式串长为 m，这种匹配方法在什么情况下时间复杂度最大？应是多少？KMP 匹配方法的特点是什么？时间复杂度是多少？
3. 如果有向图含有路径长度为负的回路（如图题Ⅱ-1 所示），试问用 Floyd 方法求每对顶点之间的最短路径能否正常工作？为什么？



图题Ⅱ-1

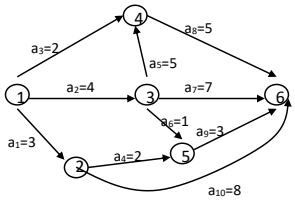
4. 高度为 4 的 3 阶 B-树至少有多少个结点？为什么？
5. 外排序的归并排序中使用的选择树和堆排序中的堆有什么区别？

三．已知一棵二叉树按层次遍历序列为 ABCDEFGHIJ，中序遍历序列为 BGDHAECIFJ。(13 分)

1．画出此二叉树的后序后继线索树；
2．画出此二叉树对应于完全二叉树的顺序存储结构；
3．将此二叉树还原成森林。

四．某工程的 AOE 网络如图题Ⅱ-2 所示。图中边上的权值为活动 a_1~a_{10} 的期限（即完成活动所需的天数）。（12 分）

1. 求该工程各事件的最早发生时间 v_e 和允许的最晚发生时间 v_l 及各活动的最早开始时间 e 和允许的最晚开始时间 l。
2. 完成此项工程至少需要多少时间？哪些活动是关键活动？



图题Ⅱ-2

五．有一组记录的关键字为{78,12,45,98,23,109,85,68,89,256,34}。(15 分)

1．写出对这组记录进行一趟快速排序的结果，并说明这趟排序中关键字比较的次数为多少？
2．将这组记录关键字建成一个小根堆；
3．将这组记录排序后作为有序查找表，画出此查找表的二叉判定树，求在等概率情况下查找成功的平均比较长度是多少？若查找给定关键为 45 和 90 的记录各需比较多少次？

六．有一组数据元素存于数组 L[1..n]中，阅读下面的算法，写出该算法的功能，并分析其时间复杂度是多少？（10 分）

```
void maxmin(int i,int j,ElemType *max,ElemType *min)
//此为递归算法，调用初值 i=1，j=n
{  ElemType max1,max2,min1,min2;
   int mid;
   if (i==j) *max=*min=L[i];
```

```
        else
        {  mid=(i+j)/2;
           maxmin(i,mid,&max1,&min1);
           maxmin(mid+1,j,&max2,&min2);
           if (max1>max2) *max=max1;
           else *max=max2;
           if (min1<min2) *min=min1;
           else *min=min2;
        }
    }//maxmin
```

七．试写一个算法，判断某二叉树（以二叉链表方式存储）是否为完全二叉树。(15 分)

八．已知有向图的邻接表，试写出求此有向图逆邻接表的算法。(15 分)

[模拟试题参考答案]

一．选择题：

1．A：⑤ B：①　　2．③　　3．A：① B：⑥　　4．③

5．A：① B：②　　6．②　　7．③

二．问答题：

1．　非叶结点的个数为 5；h_{max} =6；h_{min} =4

2．简单的模式匹配过程如下：

```
i   1   2   3   4   5   6   7   8   9   10  11  12
s   a   b   a   a   b   c   d   a   b   f   p   g
    ‖   ‖   ╱
    a   b   c   d
        ╱
        a   b   c   d
            ‖   ╱
            a   b   c   d
                ‖   ‖   ‖   ‖
                a   b   c   d   (匹配成功，index(s,t)=4)
```

简单的模式匹配方法，最坏的情况出现在每趟比较都到模式串的最后一个字符时才失配，此时的时间复杂度是 O(m*n)。

KMP 法的特点是每趟比较目标串的下标 i 不回溯，时间复杂度为 O(m+n)。

3．如果有向图含有路径长度为负的回路时，Floyd 方法不能正常工作。虽然 Floyd 方法允许路径为负数的情况，但不得包含路径长度为负的回路，如图题Ⅱ-1 的有向图，在求顶点①→③的最短路径过程中，当加入中间点②测试时，因为回路②→①→②的长度为-1,而①→③之间可无数次经过这个回路，每经过一次，其路径长度就减去 1，因此使得 A[1,3]=-∞。

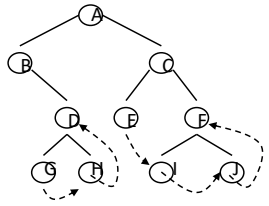
4．高度为 4 的 3 阶 B-树至少有 15 个结点。因为 m 阶 B-树除根之外的每个结点至少有⌈m/2⌉=2 棵子树，而根至少有两棵子树，所以对 3 阶 B-树而言最少的结点数应与高度相同的满二叉树结点数相同。

5．选择树是由参加比较的 n 个元素作为叶子结点而构造的完全二叉树，而堆是 n 个元素的序列表示的完全二叉树，它满足 $R_i \le R_{2i}$ 且 $R_i \le R_{2i+1}$ 或 $R_i \ge R_{2i}$ 且 $R_i \ge R_{2i+1}$。

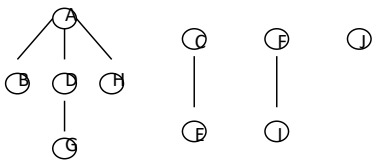
三．1．此二叉树的后序后继线索树如图题Ⅱ-3 所示；

2．此二叉树的顺序存储结构如图题Ⅱ-4 所示；

3．由此二叉树转换后的森林如图题Ⅱ-5 所示。

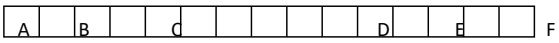


图题Ⅱ-3



图题Ⅱ-5

1　2　3　4　5　6　7　8　9　10　11　12　13　14　15



图题Ⅱ-4

四．1．某工程 AOE 网络的各事件最早发生时间和允许的最晚发生时间如表题Ⅱ-1 所示；
各个活动的最早开始时间和允许的最晚开始时间如表题Ⅱ-2 所示。

表题Ⅱ-1

事件	1	2	3	4	5	6
ve	0	3	4	9	5	14
vl	0	6	4	9	11	14

表题Ⅱ-2

活动	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}
e	0	0	0	3	4	4	4	9	5	3
l	3	0	7	9	4	10	7	9	11	6

2．完成此项工程至少需 14 天。关键活动为 a_2、a_5、a_8，事件 1-3-4-6 为关键路径。

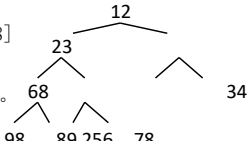
五．1．对这组记录关键字进行一趟快速排序后的结果是：

［34, 12, 45, 68, 23］78［85, 109, 89, 256, 98］

此趟排序中关键字比较 10 次。

2．由此组记录关键字建成的小根堆如图题Ⅱ-6 所示。

3．此组记录排序后组成的有序查找表为：

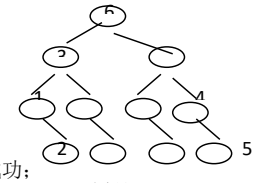


图题Ⅱ-6

表题Ⅱ-3

序号	1	2	3	4	5	6	7	8	9	10	11
key	12	23	34	45	68	78	85	89	98	109	256

二叉判定树如图题Ⅱ-7 所示。



图题Ⅱ-7

等概率情况下查找成功的平均查找长度为：

ASL=(1+2×2+4×3+4×4)/11=33/11=3

查找 K=45 的记录需要比较 3 次，查找成功；

查找 K=90 的记录需要比较 4 次，查找失败。

六．本算法的功能是求数组 L 中元素的最大值和最小值。算法的时间复杂度分析如下：

算法的语句频度 T(n)= $\begin{cases} 1 & n=1 \\ T(n/2)+T(n/2)+3 & n\geq 2 \end{cases}$

为分析方便假设 $n=2^k$（k 为正整数），且每次将序列分为两个长度相等的子序列，则：

$T(n)= T(n/2)+T(n/2)+3$

$=2T(n/2)+3$

$=2(2T(n/4)+3)+3 =4T(n/4)+2*3+3$

$=4(2T(n/8)+3)+2*3+1*3=8T(n/8)+2^2*3+2^1*3+2^0*3 =8T(n/8)+(2^2+2^1+2^0)*3$

$= \cdots\cdots$

$=2^kT(n/2^k)+(2^{k-1}+2^{k-2}+ \cdots\cdots+2^1+2^0)*3$

$=2^kT(1)+(2^k-1)*3$

$=2^k+(2^k-1)*3$

$=4*2^k-3$

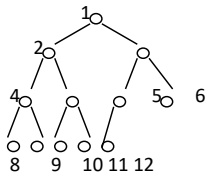
$=4n-3$

$=O(n)$

因此算法的时间复杂度为O(n)。

七．算法思想：

如图题Ⅱ-8所示，对于完全二叉树而言，按层次排列二叉树的结点时，结点是连续存在的；而对于非完全二叉树，如以结点5为根的子树（包括结点10和11）不存在时，即结点5为空结点，其后还会有结点存在。也就是说，非完全二叉树按完全二叉树序列层次排列结点时，会有空结点间隔实际存在结点的情况。



图题Ⅱ-8 完全二叉树示例

设一个辅助队列q存放二叉树结点的指针，另设标志变量tag，表示按层次逐层从左至右扫描结点过程中是否出现过空结点，其初值为0，意为尚未有空结点出现。

（1）若根结点存在，指针入队；

（2）队列不空时，反复以下操作：

若出队指针p为空，置tag=1；

否则p不为空，若此时tag=1，则判定为非完全二叉树，结束；

若此时tag=0，将它的左孩子和右孩子指针入队；

（3）队列为空时，判定为完全二叉树，结束。

算法描述：

```
        #define maxsize 100  //预定义队列容量
    #define FALSE 0
    #define TRUE  1
    typedef struct Bitnode
    {  ElemType data;
       struct Bitnode *lchild,*rchild;
    }Bitnode,*Bitree;  //定义二叉树结点类型和二叉树类型
    typedef struct
    {  Bitree Elem[maxsize];
       int front,rear;
    }Sequeue;  //定义队列类型

    int complete_Bintree(Bitree t)
    //判断二叉树是否为完全二叉树，t为二叉树根结点的指针。
    //该二叉树是完全二叉树，返回“真”(即1)；否则返回“假”(即0)。
    {  Sequeue q;  //定义辅助队列q
       q.front=-1; q.rear=0; //队列初始化
       tag=0; //标志变量初始化
       q.Elem[q.rear]=t; //根结点指针入队
       while (q.front!=q.rear) //队列不空则循环
       {  q.front++; p=q.Elem[q.front];  //队头指针出队
          if (p==NULL) tag=1; //若为空指针，置标志表示出现过空指针
          else
          {  if (tag==1)  //队头指针不空且之前出现过空指针
                return(FALSE);  //返回“假”并且退出
             else  //队头指针不空且之前未出现过空指针
             {  q.rear++; q.Elem[q.rear]=p->lchild;  //*p的左孩子指针入队
```

```
            q.rear++; q.Elem[q.rear]=p->rchild;  //*p 的右孩子指针入队
          }
        }
      }//endwhile
      return(TRUE);  //返回“真”并且退出
    }//complete_Bintree
```

八. 算法思想：

首先将有向图的邻接表的顶点表拷贝给逆邻接表的顶点表，依次扫描邻接表的边表，若顶点 i 存在邻接点 w，则 i 应为逆邻接表中顶点 w 的边表中的一个结点，将其插入到该边表的表头。当扫描完邻接表中的所有边表后，逆邻接表就建成了。

算法描述：

```
#define max_vertex_num 100  //预定义图的最多顶点数
typedef struct enode
{  int adjvex;          //邻接顶点序号
   struct enode *next;  //下一结点指针
}EdgeNode;  //定义边表结点类型
typedef struct vnode
{  VertexType vertex;  //顶点信息
   EdgeNode *firstedge;  //边表头指针
}VertexNode;  //定义顶点表结点类型
typedef struct
{  VertexNode adjlist[max_vertex_num];  //顶点表
   int n;  //图中的顶点数
}ALGraph;  //定义邻接表类型

void inverse_adjlist(ALGraph A, ALGraph *B)
//利用有向图的邻接表 A，建立它的逆邻接表*B
{  B->n=A.n;
   for (i=0;i<A.n;i++)  //建立逆邻接表的顶点表
   {  B->adjlist[i].vertex=A.adjlist[i].vertex;
      B->adjlist[i].firstedge=NULL;
   }
   for (i=0;i<A.n;i++) //扫描邻接表每个边表，建立逆邻接表的各边表
   {  p=A.adjlist[i].firstedge; //p 为指向顶点 i 的边表的头指针
      while (p!=NULL) //扫描顶点 i 的边表
      {  w=p->adjvex; //取它的一个邻接点 w
         s=(EdgeNode *)malloc(sizeof(EdgeNode)); //建立一个边表结点*s
         s->adjvex=i;
         s->next=B->adjlist[w].firstedge; //插入到逆邻接表顶点 w
         B->adjlist[w].firstedge=s;       //的边表之表头
         p=p->next;
      }//endwhile
   }//endfor
}//inverse_adjlist
```