PTA数据结构部分答案与解析v0.0.2

前言

由于考试时间临近，我可能实在没有精力做出剩余的部分。目前就做出来这么写。请使用比较专业的PDF阅读器（例如福昕PDF阅读器）打开，这样可以看到文档的目录结构。

由于专业之间题目不完全一样，这份文档与软件的PTA题目结构是一样的，其他专业可以通过查找功能查找特定题目。

另外本人水平有限，部分题目做出的解答可能不是很清晰。如果有任何疑问，欢迎通过QQ（1064406280）联系我一起讨论问题，指出错误什么的，我会继续完善这份文档。

同时要感谢帮我找出文档中错误的zsy, wzw,hb等同学，有了大家的帮助才能使这份文档变得更准确。

—— 软件工程2016-2 wzh

更新日志

2017-12-20 v0.0.1初版发行

2017-12-21 v0.0.2 增添了图的内容，修正了部分错误，并且为题目类别增添了序号

最新错误信息

http://www.92ac.cn/?p=529

点击此链接查看最新的一些题目错误纠正信息

一.树和二叉树作业—二叉树

0.错误纠正

1-3, 解析有误，已纠正

2-8，解析有误，已纠正

2-16，D->B，已纠正

1.判断题

1-1

某二叉树的后序和中序遍历序列正好一样，则该二叉树中的任何结点一定都无右孩子。 (2分)

解析：略，分析同1-6

答案：`T`

1-2

某二叉树的后序和中序遍历序列正好一样，则该二叉树中的任何结点一定都无左孩子。 (2分)

解析：略，分析同1-6

答案：`F`

1-3

存在一棵总共有2016个结点的二叉树，其中有16个结点只有一个孩子。 (3分)

解析：分支数$B = n_1 + 2n_2$,$B = n - 1 = 2015$, n_1为偶数，可推出矛盾

答案：`F`

1-4

若`A`和`B`都是一棵二叉树的叶子结点，则存在这样的二叉树，其前序遍历序列为`...A...B...`，而中序遍历序列为`...B...A...`。 (2分)

解析：

答案：`F`

1-5

若一个结点是某二叉树的中序遍历序列的最后一个结点，则它必是该树的前序遍历序列中的最后一个结点。 (2分)

解析：画图分析即可，考虑只有两个节点的无右子树的二叉树，中序遍历的最后一个节点是根节点，而根节点在前序遍历中一定是在开头的。

答案：`F`

1-6

某二叉树的前序和中序遍历序列正好一样，则该二叉树中的任何结点一定都无左孩子。 (2分)

解析：假如存在左孩子，那么前序遍历的第一个元素肯定不等于中序遍历的第一个元素。我们分析第一层，假设根节点的左子树不存在，那么总算前序遍历和中序遍历结果相等了，然后我们判断根节点的右子树部分，发现还是要保证左子树不存在的。递归分析一定没有左孩子的。

答案：`T`

1-7

已知一棵二叉树的先序遍历结果是ABC， 则CAB不可能是中序遍历结果。 (2分)

解析：知道中序遍历和先序遍历，是可以画出树来的。如果不是很会这种方法，反正只有三个节点，大可以画图举例。可得没有树满足先序是ABC，中序是CAB的。

我们这样去分析：由先序遍历可得A是根节点；由中序遍历可得C是左孩子，B是右孩子，而先序遍历中B是左孩子，C是右孩子，矛盾，所以不可能滴

答案：`T`

2.单选题

2-1*

如果一棵非空k（k≥2）叉树T中每个非叶子结点都有k个孩子，则称T为正则k叉树。若T的高度为h（单结点的树h=1），则T的结点数最多为：(3分)

1. (k^h-1)/(k−1)
2. $(k^{h-1}-1)$/(k−1)
3. $(k^{h+1}-1)$/(k−1)
4. 以上都不是

解析：将k=2带入，套用二叉树的结点树结论，发现A为正确形式

答案：A

2-2*

如果一棵非空k（k≥2）叉树T中每个非叶子结点都有k个孩子，则称T为正则k叉树。若T的高度为h（单结点的树h=1），则T的结点数最少为：(3分)

1. $(k^{h-1}-1)$/(k−1)+1
2. (k^h-1)/(k−1)−1
3. kh
4. $k(h-1)+1$

解析：这道题我不知道怎么正确推导答案，不过这题可以举一个正确的二叉树的例子，带答案用排除法做

答案：D

2-3

要使一棵非空二叉树的先序序列与中序序列相同，其所有非叶结点须满足的条件是：(2分)

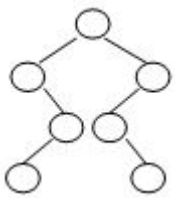
1. 只有左子树
2. 只有右子树
3. 结点的度均为1
4. 结点的度均为2

解析：略

答案：B

2-4

已知一棵二叉树的树形如下图所示，其后序序列为{ e , a , c , b , d , g , f }。树中与结点 a 同层的结点是：(3分)



1. c
2. d
3. f
4. g

解析：模拟一遍求解即可

答案：B

2-5

在下述结论中，正确的是：(2分)

① 只有2个结点的树的度为1；

② 二叉树的度为2；

③ 二叉树的左右子树可任意交换；

④ 在最大堆（大顶堆）中，从根到任意其它结点的路径上的键值一定是按非递增有序排列的。

1. ①④
2. ②④
3. ①②③
4. ②③④

解析：二叉树度数不一定为2，比如空二叉树

答案：A

2-6

若一棵二叉树的后序遍历序列是{ 1, 3, 2, 6, 5, 7, 4 }，中序遍历序列是{ 1, 2, 3, 4, 5, 6, 7 }，则下列哪句是错的？(3分)

1. 这是一棵完全二叉树
2. 2是1和3的父结点
3. 这是一棵二叉搜索树
4. 7是5的父结点

解析：根据后序遍历与中序遍历建树判断即可

答案：A

2-7

如果一棵非空k（k≥2）叉树T中每个非叶子结点都有k个孩子，则称T为正则k叉树。若T有m个非叶子结点，则T中的叶子结点个数为：(3分)

1. mk
2. m(k−1)
3. m(k−1)+1
4. m(k−1)−1

解析：$n = B + 1 = mk + 1$

$n = n_0 + n_k$

B代表分支数，n代表结点数,联立上边二式可得正确答案，故选C

答案：C

2-8

有一个四叉树，度2的结点数为2，度3的结点数为3，度4的结点数为4。问该树的叶结点个数是多少？(2分)

1. 10
2. 12
3. 20
4. 21

解析：根据题意随便画一种符合题意的图即可判断，也可以通过公式推导。

$n = n_0 + n_1 + n_2 + n_3 + n_4$

$n = B + 1 = 4n_4 + 3n_3 + 2n_2 + n_1 + 1$

B代表分支数，n代表总结点数，将上边式子联立，然后带入题目中数据，可得

$n_0 = 3n_4 + 2n_3 + 1n_2 + 1$

答案：D

2-9

若一棵二叉树的前序遍历序列是{ 4, 2, 1, 3, 6, 5, 7 }，中序遍历序列是{ 1, 2, 3, 4, 5, 6, 7 }，则下列哪句是错的？(3分)

1. 这是一棵完全二叉树
2. 所有的奇数都在叶子结点上
3. 这是一棵二叉搜索树
4. 2是5的父结点

解析：根据前序，中序遍历可以建出图来，然后根据图来判断即可

答案：D

2-10

按照二叉树的定义，具有3个结点的二叉树有几种？ (2分)

1. 3
2. 4
3. 5
4. 6

解析：画图枚举即可，这里问二叉树种类不考虑二叉树节点值的不同。

答案：C

2-11

任何一棵二叉树的叶结点在先序、中序和后序遍历序列中的相对次序 (2分)

1. 发生改变
2. 不发生改变

3. 不能确定
4. 以上都不对

解析：他们都是左-右，因为问的是相对次序，根节点插在哪里无所谓的

答案：B

2-12

二叉树中第5层（根的层号为1）上的结点个数最多为：(2分)

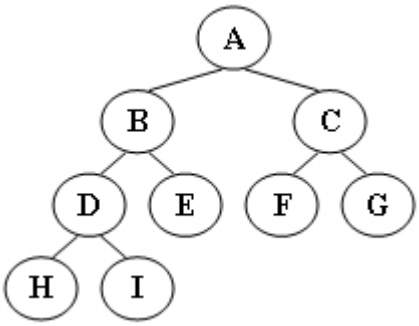
1. 8
2. 15
3. 16
4. 32

解析：$n_5 = 2^{5-1} = 16$

答案：C

2-13

先序遍历图示二叉树的结果为 (2分)



1. A , B , C , D , H , E , I , F , G
2. A , B , D , H , I , E , C , F , G
3. H , D , I , B , E , A , F , C , G
4. H , I , D , B , E , F , G , A , C

解析：先序遍历先遍历根节点，再遍历左子树，最后遍历右子树，递归进行

答案：B

2-14

三叉树中，度为1的结点有5个，度为2的结点3个，度为3的结点2个，问该树含有几个叶结点？ (3分)

1. 8
2. 10
3. 12
4. 13

解析：思路与2-8类似，在此不再阐述

答案：A

2-15

某二叉树的中序序列和后序序列正好相反，则该二叉树一定是 (2分)

1. 空或只有一个结点
2. 高度等于其结点数
3. 任一结点无左孩子
4. 任一结点无右孩子

解析：中序遍历是左-根-右，后序遍历是左-右-根，可知如果左子树不存在的话，就是恰好满足条件的

答案：`C`

2-16

某二叉树的前序和后序遍历序列正好相反，则该二叉树一定是 (2分)

1. 空或只有一个结点
2. 高度等于其结点数
3. 任一结点无左孩子
4. 任一结点无右孩子

解析：分两种情况，无左结点（根右-右根）或者无右结点（根左-左根）

答案：`B`

2-17

设n、m为一棵二叉树上的两个结点，在中序遍历时，n在m前的条件是 (3分)

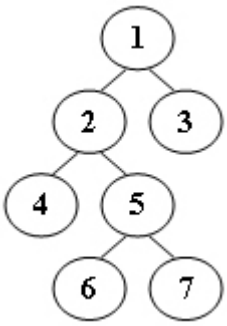
1. n在m左方
2. n在m右方
3. n是m祖先
4. n是m子孙

解析：由显然易得，选A

答案：`A`

2-18

给定二叉树如下图所示。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： (2分)



1. NRL

2. RNL
3. LRN
4. RLN

解析：由显然易得，选B

答案：B

2-19

设高为h的二叉树（规定叶子结点的高度为1）只有度为0和2的结点，则此类二叉树的最少结点数和最多结点数分别为：(3分)

1. $2h, 2^h-1$
2. $2h-1, 2^h-1$
3. $2h-1, 2^{h-1}-1$
4. $2^{h-1}+1, 2^h-1$

解析：由二叉树结点数性质，易得B答案正确, 容易误选D，当除根节点之外，每层有两个节点的时候，结点数是最少的（不方便画图就不画了，自行理解）

答案：B

2-20

在下述结论中，正确的是：(2分)

①只有一个结点的二叉树的度为0;

②二叉树的度为2；

③二叉树的左右子树可任意交换；

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满二叉树。

1. ①④
2. ②④
3. ①②③
4. ②③④

解析：考察二叉树的定义以及性质，可以参考课本P113页

答案：A

3.函数题

4.编程题

二.线性表应用

1.单选题

2-1

采用多项式的非零项链式存储表示法，如果两个多项式的非零项分别为N1和N2个，最高项指数分别为M1和M2，则实现两个多项式相乘的时间复杂度是：(2分)

1. O(N1×N2)
2. O(M1×M2)
3. O(N1+N2)
4. O(M1+M2)

解析：略

答案：A

2.编程题

三.栈

1.判断题

1-1

通过对堆栈S操作：Push(S,1), Push(S,2), Pop(S), Push(S,3), Pop(S), Pop(S)。输出的序列为：123。 (2分)

解析：水题，略

答案：F

1-2

若一个栈的输入序列为1，2，3，...，N，输出序列的第一个元素是i，则第j个输出元素是j−i−1。 (2分)

解析：第j个输出的元素应该是不确定的，可以带i=2，j=3的特指验证

答案：F

1-3

若一个栈的输入序列为{1, 2, 3, 4, 5}，则不可能得到{3, 4, 1, 2, 5}这样的出栈序列。 (2分)

解析：2一定先与1出栈的

答案：T

2.单选题

2-1

给定一个堆栈的入栈序列为{ 1, 2, ⋯, n }，出栈序列为{ p1, p2, ⋯, pn }。如果p2=n，则存在多少种不同的出栈序列？(2分)

1. 1
2. 2
3. n−1
4. n

解析：首先p1之前有n-1中选法，然后p2后边的序列一定是唯一的，因为不会再有元素进栈导致出栈序列变化了

答案：`C`

2-2

设一个堆栈的入栈顺序是1、2、3、4、5。若第一个出栈的元素是4，则最后一个出栈的元素必定是：(2分)

1. 1
2. 3
3. 5
4. 1或者5

解析：要不1-4全出栈再进5，要不进5之后全

答案：`D`

2-3

从栈顶指针为`ST`的链栈中删除一个结点且用`X`保存被删结点的值，则执行：(2分)

1. `X= ST->data;`
2. `X= ST; ST = ST->next;`
3. `X= ST->data; ST = ST->next;`
4. `ST = ST->next; X= ST->data;`

解析：考察对链栈的掌握，在链栈中，ST代表栈顶元素，其data存储的元素的值

答案：`C`

2-4

设栈S和队列Q的初始状态均为空，元素a、b、c、d、e、f、g依次进入栈S。若每个元素出栈后立即进入队列Q，且7个元素出队的顺序是b、d、c、f、e、a、g，则栈S的容量至少是：(2分)

1. 1
2. 2
3. 3
4. 4

解析： 模拟即可

答案：`C`

2-5

假设有5个整数以1、2、3、4、5的顺序被压入堆栈，且出栈顺序为3、5、4、2、1，那么为了获得这样的输出，堆栈大小至少为：(2分)

1. 2

2. 3
3. 4
4. 5

解析：模拟一遍即可

答案：C

2-6

若元素a、b、c、d、e、f依次进栈，允许进栈、退栈操作交替进行，但不允许连续三次进行退栈工作，则不可能得到的出栈序列是？ (2分)

1. b c a e f d
2. c b d a e f
3. d c e b f a
4. a f e d c b

解析：D中的d c d很明显是连续三次退栈了

答案：D

2-7

设一个栈的输入序列是1、2、3、4、5，则下列序列中，是栈的合法输出序列的是？ (2分)

1. 3 2 1 5 4
2. 5 1 2 3 4
3. 4 5 1 3 2
4. 4 3 1 2 5

解析：如果某个子序列在某个数的左边，那么在输出的时候，这个子序列的输出一定是确定的。

例如1,2,3在4的左边，那么我栈输出的时候，一定是先输出3，在输出2，最后输出1，中间可以穿插其他的元素。可以利用这个性质去判断选项是否合法

答案：A

2-8

有六个元素以6、5、4、3、2、1的顺序进栈，问哪个不是合法的出栈序列？ (2分)

1. 2 3 4 1 5 6
2. 3 4 6 5 2 1
3. 5 4 3 6 1 2
4. 4 5 3 1 2 6

解析：针对每个选项进行验证即可

答案：B

2-9

若一个栈的入栈序列为1、2、3、…、N，输出序列的第一个元素是i，则第j个输出元素是： (2分)

1. i−j−1
2. i−j

3. j−i−1
4. 不确定

解析：当i不等于N的时候，第j个输出是不确定的，因为随时有可能有新的元素插入进来并在任意位置输出改变原有的输出顺序；当i = N，输出序列才是确定的

答案：`D`

2-10

若一个栈的入栈序列为1、2、3、...、N，其输出序列为p1、p2、p3、...、pN。若p1=N，则pi为： (2分)

1. i
2. n−i
3. n−i+1
4. 不确定

解析：当p1等于N，p2-pn是唯一的一个序列，pi=n - i + 1

答案：`C`

2-11

将5个字母`ooops`按此顺序入栈，则有多少种不同的出栈顺序可以仍然得到`ooops`？ (2分)

1. 1
2. 3
3. 5
4. 6

解析：五种方案，三个o的顺序可以是3 2 1, 1 2 3, 2 1 3 , 1 3 2, 2 3 1

答案：`C`

2-12

栈的插入和删除操作在（ ）进行。 (2分)

1. 栈顶
2. 栈底
3. 任意位置
4. 指定位置

解析：略

答案：`A`

3.编程题

四.栈及其应用

1.单选题

2-1

线性表、堆栈、队列的主要区别是什么？(1分)

1. 线性表用指针，堆栈和队列用数组
2. 堆栈和队列都是插入、删除受到约束的线性表
3. 线性表和队列都可以用循环链表实现，但堆栈不能
4. 堆栈和队列都不是线性结构，而线性表是

解析：首先题意我自己感觉很迷。。。A选项，他们都可以用指针或者数组,C选项，线性表不能用循环链表实现，D选项，他们都是线性结构

答案：`B`

2.函数题

3.编程题

五.队列

1.单选题

2-1

为解决计算机主机与打印机之间速度不匹配问题，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该缓冲区，而打印机则依次从该缓冲区中取出数据。该缓冲区的逻辑结构应该是？(1分)

1. 堆栈
2. 队列
3. 树
4. 图

解析：略

答案：`B`

2-2

若已知一队列用单向链表表示，该单向链表的当前状态（含3个对象）是：`1->2->3`，其中`x->y`表示`x`的下一节点是`y`。此时，如果将对象`4`入队，然后队列头的对象出队，则单向链表的状态是：(1分)

1. `1->2->3`
2. `2->3->4`
3. `4->1->2`
4. 答案不唯一

解析：略

答案：`B`

2-3

在一个链队列中，front和rear分别为头指针和尾指针，则插入一个结点s的操作为（ ）。(2分)

1. front=front->next
2. s->next=rear;rear=s
3. rear->next=s;rear=s;
4. s->next=front;front=s;

解析：略

答案：C

2-4

依次在初始为空的队列中插入元素a,b,c,d以后，紧接着做了两次删除操作，此时的队头元素是（ ）。 (2分)

1. a
2. b
3. c
4. d

解析：略

答案：C

2-5

在一个不带头结点的非空链式队列中,假设f和r分别为队头和队尾指针,则插入s所指的结点运算是( )。 (2分)

1. f->next=s; f=s;
2. r->next=s; r=s;
3. s->next=s; r=s;
4. s->next=f; f=s;

解析：略

答案：B

2.编程题

六.循环队列及线性结构综合

1.判断题

1-1

所谓“循环队列”是指用单向循环链表或者循环数组表示的队列。 (1分)

解析：略

答案：F

1-2

在用数组表示的循环队列中，front值一定小于等于rear值。 (1分)

解析：略

答案：`F`

1-3

不论是入队列操作还是入栈操作,在顺序存储结构上都需要考虑"溢出"情况。 (2分)

解析：略

答案：`T`

2.单选题

2-1

若用大小为6的数组来实现循环队列，且当前`front`和`rear`的值分别为0和4。当从队列中删除两个元素，再加入两个元素后，`front`和`rear`的值分别为多少？ (2分)

1. 2和0
2. 2和2
3. 2和4
4. 2和6

解析：front = （ 0 + 2 ）% 6 = 2，rear = （ 4 + 2 ）% 6 = 0

答案：`A`

2-2

如果循环队列用大小为`m`的数组表示，且用队头指针`front`和队列元素个数`size`代替一般循环队列中的`front`和`rear`指针来表示队列的范围，那么这样的循环队列可以容纳的元素个数最多为： (2分)

1. `m-1`
2. `m`
3. `m+1`
4. 不能确定

解析：注意看清楚题目给定的规则，这样就不需要空一个位置来判断是否为满了

答案：`B`

2-3

如果循环队列用大小为`m`的数组表示，队头位置为`front`、队列元素个数为`size`，那么队尾元素位置`rear`为： (2分)

1. `front+size`
2. `front+size-1`
3. `(front+size)%m`
4. `(front+size-1)%m`

解析：读题，rear这次代表队尾元素啦，然后自然就选D了

答案：`D`

3.编程题

七.图--基本概念

0.错误纠正

2-28, A->C并增加了解析, 已纠正

1.判断题

1-1

无向连通图至少有一个顶点的度为1。 (1分)

解析：

答案：F

1-2

用邻接表法存储图，占用的存储空间数只与图中结点个数有关，而与边数无关。 (1分)

解析：

答案：F

1-3

用邻接矩阵法存储图，占用的存储空间数只与图中结点个数有关，而与边数无关。 (1分)

解析：

答案：T

1-4

在一个有向图中，所有顶点的入度与出度之和等于所有边之和的2倍。 (1分)

解析：

答案：T

1-5

在任一有向图中，所有顶点的入度之和等于所有顶点的出度之和。 (1分)

解析：

答案：T

1-6

如果无向图G必须进行两次广度优先搜索才能访问其所有顶点，则G中一定有回路。 (2分)

解析：

答案：F

1-7

如果无向图G必须进行两次广度优先搜索才能访问其所有顶点，则G一定有2个连通分量。 (2分)

解析：

答案：T

1-8

无向连通图所有顶点的度之和为偶数。 (1分)

解析：

答案：T

1-9

无向连通图边数一定大于顶点个数减1。 (1分)

解析：

答案：F

2.单选题

2-1

若无向图G = (V , E) 中含10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： (3分)

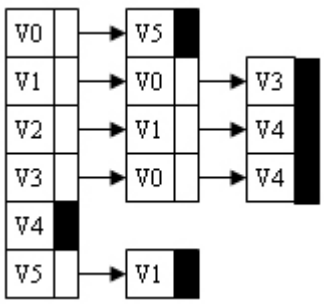
1. 45
2. 37
3. 36
4. 9

解析：题意说要保证图在任何情况下连通，我们可以考虑让n-1个点构成完全图，然后再加一条边连接第n个顶点。这种情况下再加一条边多余，再减一条边就有可能不连通了，所以$n = (n-1)*(n-2)/2+1$

答案：B

2-2

给定一个有向图的邻接表如下图，则该图有__个强连通分量。(3分)



1. 4 {{0, 1, 5}, {2}, {3}, {4}}
2. 3 {{2}, {4}, {0, 1, 3, 5}}
3. 1 {0, 1, 2, 3, 4, 5}
4. 1 {0, 5, 1, 3}

解析：建图求解,强连通分量中的点之间都可相互抵达，判断即可

答案：B

2-3

给定有向图的邻接矩阵如下：

$$\begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

顶点2（编号从0开始）的出度和入度分别是：(1分)

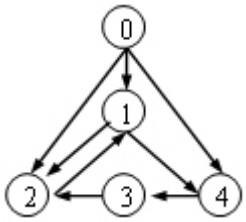
1. 3, 1
2. 1, 3
3. 0, 2
4. 2, 0

解析：略

答案：C

2-4

下面给出的有向图中，有__个强连通分量。(2分)



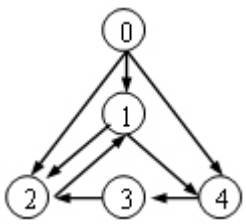
1. 1 ({0,1,2,3,4})
2. 1 ({1,2,3,4})
3. 2 ({1,2,3,4}, {0})
4. 5 ({0}, {1}, {2}, {3}, {4})

解析：略

答案：C

2-5

下面给出的有向图中，各个顶点的入度和出度分别是：(1分)



1. 入度: 0, 2, 3, 1, 2; 出度: 3, 2, 1, 1, 1

2. 入度: 3, 2, 1, 1, 1; 出度: 0, 2, 3, 1, 2
3. 入度: 3, 4, 4, 2, 3; 出度: 3, 4, 4, 2, 3
4. 入度: 0, 1, 2, 1, 1; 出度: 3, 2, 1, 1, 1

解析：略

答案：A

2-6

如果G是一个有36条边的非连通无向图，那么该图顶点个数最少为多少？(3分)

1. 7
2. 8
3. 9
4. 10

解析：思路同2-1, 36 = 9 * 8 / 2，当有9个顶点的时候保证是连通的无向图，他们之间都有边相连，再加上一个顶点后，此时保证顶点最小且非连通

答案：D

2-7

下面关于图的存储的叙述中，哪一个是正确的？ (1分)

1. 用相邻矩阵法存储图，占用的存储空间数只与图中结点个数有关，而与边数无关
2. 用相邻矩阵法存储图，占用的存储空间数只与图中边数有关，而与结点个数无关
3. 用邻接表法存储图，占用的存储空间数只与图中结点个数有关，而与边数无关
4. 用邻接表法存储图，占用的存储空间数只与图中边数有关，而与结点个数无关

解析：考察课本基础知识

答案：A

2-8

关于图的邻接矩阵，下列哪个结论是正确的？ (1分)

1. 有向图的邻接矩阵总是不对称的
2. 有向图的邻接矩阵可以是对称的，也可以是不对称的
3. 无向图的邻接矩阵总是不对称的
4. 无向图的邻接矩阵可以是不对称的，也可以是对称的

解析：无向图的邻接矩阵一定是对称的，有向图不一定对称

答案：B

2-9

设N个顶点E条边的图用邻接表存储，则求每个顶点入度的时间复杂度为： (2分)

1. O(N)
2. O(N2)
3. O(N+E)
4. O(N×E)

解析：略

答案：`C`

2-10

在一个无向图中，所有顶点的度数之和等于所有边数的多少倍？ (2分)

1. 1/2
2. 1
3. 2
4. 4

解析：对于无向图，有向图都是保证顶点的度数之和是所有边数的二倍

答案：`C`

2-11

在一个有向图中，所有顶点的入度与出度之和等于所有边之和的多少倍？ (2分)

1. 1/2
2. 1
3. 2
4. 4

解析：对于无向图，有向图都是保证顶点的度数之和是所有边数的二倍

答案：`C`

2-12

在任一有向图中，所有顶点的入度之和与所有顶点的出度之和的关系是： (1分)

1. 相等
2. 大于等于
3. 小于等于
4. 不确定

解析：略

答案：`A`

2-13

设无向图的顶点个数为N，则该图最多有多少条边？ (1分)

1. N−1
2. N(N−1)/2
3. N(N+1)/2
4. N2

解析：每个点都与其余n-1个点有连接，去除重复,边数为N*(N-1)/2

答案：`B`

2-14

下列关于无向连通图特征的叙述中，正确的是： (2分)

1. 所有顶点的度之和为偶数
2. 边数大于顶点个数减1
3. 至少有一个顶点的度为1

1. 只有1
2. 只有2
3. 1和2
4. 1和3

解析：1显然正确，2边数大于等于顶点的个数，3可能顶点的度数都大于1，反正没有度数为0的顶点就成

答案：A

2-15

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： (3分)

1. 6
2. 15
3. 16
4. 21

解析：思路同2-1

答案：C

2-16

在N个顶点的无向图中，所有顶点的度之和不会超过顶点数的多少倍？ (2分)

1. 1
2. 2
3. (N−1)/2
4. N−1

解析：顶点的度之和 = 边数的二倍 ，而边数是小于等于N*(N-1)/2，故选D

答案：D

2-17

对于一个具有N个顶点的无向图，要连通所有顶点至少需要多少条边？ (2分)

1. N−1
2. N
3. N+1
4. N/2

解析：N个点依次相连需要的边数最少

答案：A

2-18

具有N (N>0) 个顶点的无向图至多有多少个连通分量？ (2分)

1. 0
2. 1
3. N−1
4. N

解析：每个点之间都不连通，此时有N个连通分量

答案：`D`

2-19

一个有N个顶点的强连通图至少有多少条边？ (2分)

1. N−1
2. N
3. N+1
4. N(N−1)

解析：每个点首尾连接成环，边数最少

答案：`B`

2-20

如果G是一个有28条边的非连通无向图，那么该图顶点个数最少为多少？ (3分)

1. 7
2. 8
3. 9
4. 10

解析：前边已解释过，不再阐述

答案：`C`

2-21

对于有向图，其邻接矩阵表示比邻接表表示更易于： (2分)

1. 求一个顶点的入度
2. 求一个顶点的出边邻接点
3. 进行图的深度优先遍历
4. 进行图的广度优先遍历

解析：略

答案：`A`

2-22

对于一个具有N个顶点的无向图，若采用邻接矩阵表示，则该矩阵的大小是： (1分)

1. N−1
2. N
3. $(N-1)^2$
4. N^2

解析：略

答案：D

2-23

若一个有向图用邻接矩阵表示，则第i个结点的入度就是： (1分)

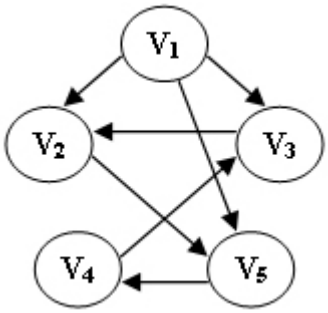
1. 第i行的元素个数
2. 第i行的非零元素个数
3. 第i列的非零元素个数
4. 第i列的零元素个数

解析：略

答案：C

2-24

下列选项中，不是下图深度优先搜索序列的是：(2分)



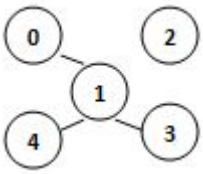
1. V1, V5, V4, V3, V2
2. V1, V3, V2, V5, V4
3. V1, V2, V5, V4, V3
4. V1, V2, V3, V4, V5

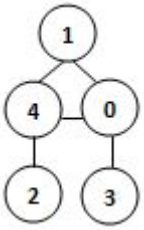
解析：略

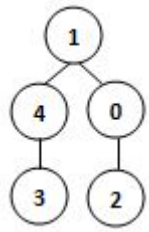
答案：D

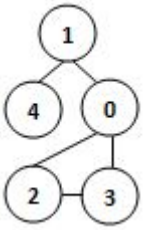
2-25

若某图的深度优先搜索序列是{V1, V4, V0, V3, V2}，则下列哪个图不可能对应该序列？ (2分)

1. 

2. 

3. 

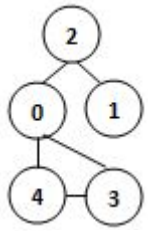
4. 

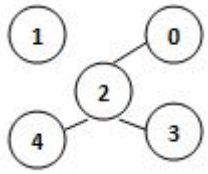
解析：略

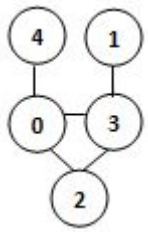
答案：C

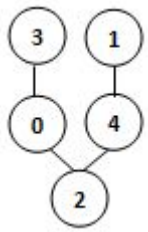
2-26

若某图的深度优先搜索序列是{V2, V0, V4, V3, V1}，则下列哪个图不可能对应该序列？ (2分)

1. 

2. 

3. 

4. 

解析：略

答案：D

2-27

已知无向图G含有16条边，其中度为4的顶点个数为3，度为3的顶点个数为4，其他顶点的度均小于3。图G所含的顶点个数至少是：(4分)

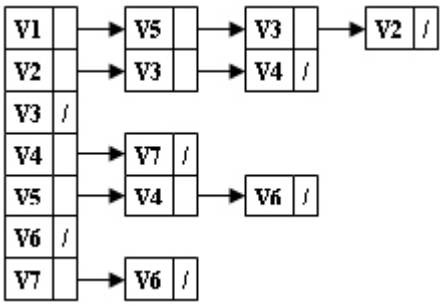
1. 10
2. 11
3. 13
4. 15

解析：要使顶点数最少且保证度数小于3, 那么令其他点度数为2即可，得出度数为2的点数量为 $(16 \times 2 - 3 \times 4 - 3 \times 4)/2 = 4$，故顶点数最少为 3 + 4 + 4 = 11

答案：B

2-28

给定一有向图的邻接表如下。从顶点V1出发按深度优先搜索法进行遍历，则得到的一种顶点序列为：(2分)



1. V1,V5,V4,V7,V6,V2,V3
2. V1,V2,V3,V4,V7,V6,V5
3. V1,V5,V4,V7,V6,V3,V2
4. V1,V5,V6,V4,V7,V2,V3

解析：注意深度优先遍历的顺序与邻接表遍历的顺序是相关的，因此不能选A，比如V1有两个邻接点V5，V3，V2，一定是先遍历V5，再遍历V3，最后遍历V2的

答案：C

2-29

图的广度优先遍历类似于二叉树的：(1分)

1. 先序遍历
2. 中序遍历
3. 后序遍历
4. 层次遍历

解析：略

答案：D

2-30

给定无向图G，从V0出发进行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)}。则下面哪条边不可能出现在G中？ (3分)

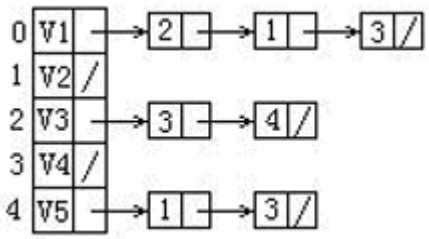
1. (V0,V2)
2. (V0,V6)
3. (V1,V5)
4. (V4,V6)

解析：首先根据边集画出树来，然后依次判断选项中的边加入后有没有可能改变边集合。假如加入(V1, V5) ,并且先遍历到的(v1, v5), 那么(v0, v1)可能就会被深度优先遍历遍历到了。

答案：C

2-31

给定一有向图的邻接表如下。从顶点V1出发按深度优先搜索法进行遍历，则得到的一种顶点序列为： (2分)



1. V1,V2,V3,V5,V4
2. V1,V3,V4,V5,V2
3. V1,V4,V3,V5,V2
4. V1,V2,V4,V5,V3

解析：略

答案：B

2-32

已知一个图的邻接矩阵如下，则从顶点V1出发按深度优先搜索法进行遍历，可能得到的一种顶点序列为： (2分)

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

1. V1,V2,V3,V4,V5,V6
2. V1,V2,V4,V5,V6,V3
3. V1,V3,V5,V2,V4,V6
4. V1,V3,V5,V6,V4,V2

解析：略

答案：B

2-33

如果从无向图的任一顶点出发进行一次深度优先搜索可访问所有顶点，则该图一定是： (2分)

1. 连通图

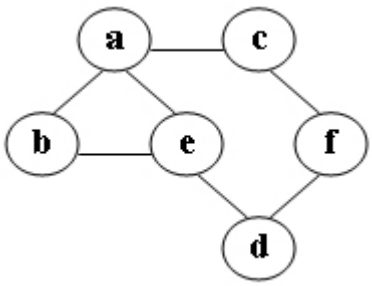
2. 完全图
3. 有回路的图
4. 一棵树

解析：略

答案：A

2-34

在图中自a点开始进行广度优先遍历算法可能得到的结果为：(2分)



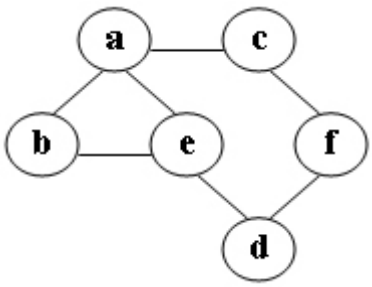
1. a, e, d, f, c, b
2. a, c, f, e, b, d
3. a, e, b, c, f, d
4. a, b, e, c, d, f

解析：略

答案：D

2-35

在图中自c点开始进行广度优先遍历算法可能得到的结果为：(2分)



1. c,a,b,e,f,d
2. c,a,f,d,e,b
3. c,f,a,d,e,b
4. c,f,a,b,d,e

解析：略

答案：C

2-36

如果无向图G必须进行两次广度优先搜索才能访问其所有顶点，则下列说法中不正确的是：(2分)

1. G肯定不是完全图

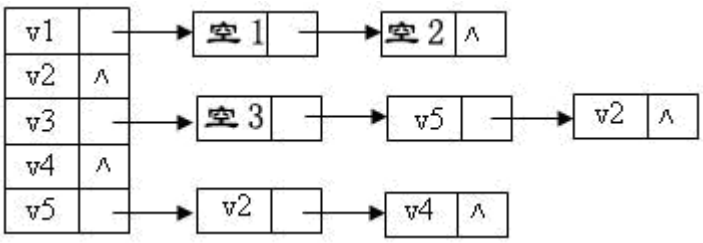
2. G中一定有回路
3. G一定不是连通图
4. G有2个连通分量

解析：略

答案：B

2-37

给定一有向图的邻接表如下。若从v1开始利用此邻接表做广度优先搜索得到的顶点序列为：{v1, v3, v2, v4, v5}，则该邻接表中顺序填空的结果应为： (3分)



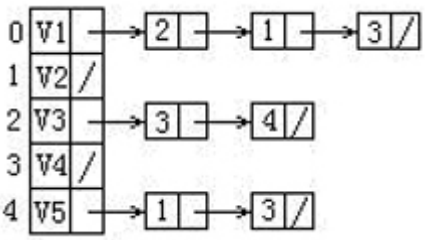
1. v2, v3, v4
2. v3, v2, v4
3. v3, v4, v2
4. v4, v3, v2

解析：略

答案：B

2-38

给定一有向图的邻接表如下。从顶点V1出发按广度优先搜索法进行遍历，则得到的一种顶点序列为： (2分)



1. V1,V2,V3,V4,V5
2. V1,V2,V3,V5,V4
3. V1,V3,V2,V4,V5
4. V1,V4,V3,V5,V2

解析：略

答案：C

2-39

已知一个图的邻接矩阵如下，则从顶点V1出发按广度优先搜索法进行遍历，可能得到的一种顶点序列为： (2分)

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

1. V1,V2,V3,V5,V4,V6
2. V1,V2,V4,V5,V6,V3
3. V1,V3,V5,V2,V4,V6
4. V1,V3,V5,V6,V4,V2

解析：略

答案：A

2-40

下列说法不正确的是：(2分)

1. 图的遍历是从给定的源点出发每一个顶点仅被访问一次
2. 遍历的基本算法有两种：深度遍历和广度遍历
3. 图的深度遍历是一个递归过程
4. 图的深度遍历不适用于有向图

解析：有向图无向图都可以

答案：D

2-41

图的深度优先遍历类似于二叉树的：(1分)

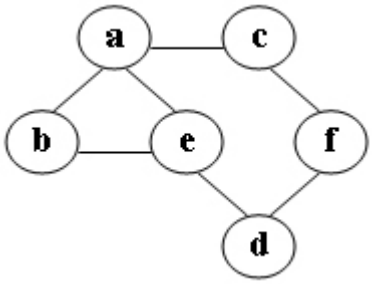
1. 先序遍历
2. 中序遍历
3. 后序遍历
4. 层次遍历

解析：略

答案：A

2-42

在图中自a点开始进行深度优先遍历算法可能得到的结果为：(2分)



1. a, b, e, c, d, f
2. a, c, f, e, b, d
3. a, e, b, c, f, d
4. a, e, d, f, c, b

解析：略

答案：`D`

3.编程题

八.最短路径

1.判断题

1-1

在一个有权无向图中，若`b`到`a`的最短路径距离是12，且`c`到`b`之间存在一条权为2的边，则`c`到`a`的最短路径距离一定不小于10。 (3分)

解析：根据题意画出图来，因为a,b 最短路径是12，所以c-b-a应大于等于12，所以c-a最短距离大于等于10

答案：`T`

2.单选题

2-1

我们用一个有向图来表示航空公司所有航班的航线。下列哪种算法最适合解决找给定两城市间最经济的飞行路线问题？ (1分)

1. Dijkstra算法
2. Kruskal算法
3. 深度优先搜索
4. 拓扑排序算法

解析：A选项解决单源最短路问题，自然可以解决两点之间最短路，B选项解决最小生成数问题，D选项解决AOE关键路径问题，C选项。。就是遍历图用的，可以生成深度优先生成树

答案：`A`

2-2

数据结构中Dijkstra算法用来解决哪个问题？ (1分)

1. 关键路径
2. 最短路径
3. 拓扑排序
4. 字符串匹配

解析：考察课本基础知识

答案：`B`

2-3

若要求在找到从`S`到其他顶点最短路的同时，还给出不同的最短路的条数，我们可以将Dijkstra算法略作修改，增加一个`count[]`数组：`count[V]`记录`S`到顶点`V`的最短路径有多少条。则`count[V]`应该被初始化为： (3分)

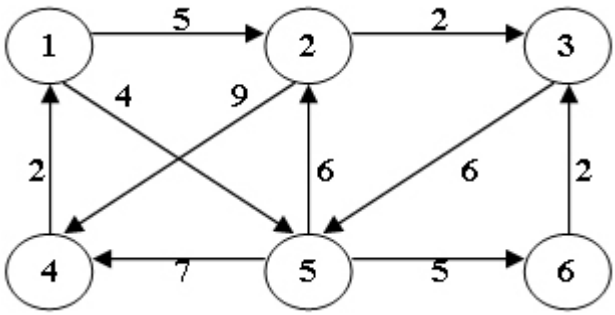
1. `count[S]=1;`对于其他顶点`V`则令`count[V]=0`
2. `count[S]=0;`对于其他顶点`V`则令`count[V]=1`
3. 对所有顶点都有`count[V]=1`
4. 对所有顶点都有`count[V]=0`

解析：计算S到V最短路径的递推式是：count[V] = count[V] + count[S]。可得A答案比较合适

答案：`A`

2-4

使用迪杰斯特拉（Dijkstra）算法求下图中从顶点1到其他各顶点的最短路径，依次得到的各最短路径的目标顶点是：(2分)



1. 5, 2, 3, 4, 6
2. 5, 2, 3, 6, 4
3. 5, 2, 4, 3, 6
4. 5, 2, 6, 3, 4

解析：模拟Dijstra算法求解即可

答案：`B`

2-5

在一个有权无向图中，如果顶点b到顶点a的最短路径长度是10，顶点c与顶点b之间存在一条长度为3的边。那么下列说法中有几句是正确的？ (3分)

1. c与a的最短路径长度就是13
2. c与a的最短路径长度就是7
3. c与a的最短路径长度不超过13

4. c与a的最短路径不小于7

1. 1句
2. 2句
3. 3句
4. 4句

解析：根据题意建出图来，显然，3,4句话是正确的

答案：`B`

3.程序填空题

4.编程题

九.拓扑排序与关键路径

0.错误修正

2-5 ,C->B, 已纠正

1.单选题

2-1

在AOE网中，什么是关键路径？ (1分)

1. 最短回路
2. 最长回路
3. 从第一个事件到最后一个事件的最短路径
4. 从第一个事件到最后一个事件的最长路径

解析：考察课本基础知识

答案：`D`

2-2

在拓扑排序算法中用堆栈和用队列产生的结果会不同吗？(1分)

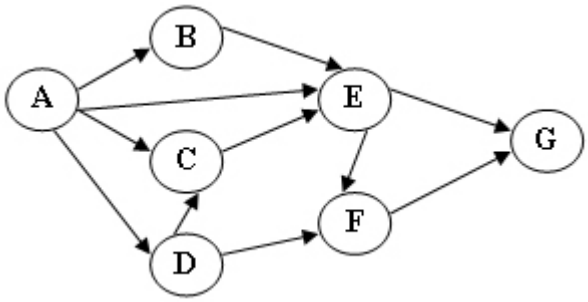
1. 是的肯定不同
2. 肯定是相同的
3. 有可能会不同
4. 以上全不对

解析：引入栈的目的仅仅是为了避免重复扫描数组，因此引入队列也可以，结果是有可能相同的，比如只有一个元素

答案：C

2-3

下图为一个AOV网，其可能的拓扑有序序列为： (2分)



1. ABCDFEG
2. ADFCEBG
3. ACDFBEG
4. ABDCEFG

解析：排除法做，首先第一个元素肯定是A, 第二个肯定不是C，假设第二个是B，那么第三个肯定是D，选D没毛病

答案：D

2-4

若将n个顶点e条弧的有向图采用邻接表存储，则拓扑排序算法的时间复杂度是：(1分)

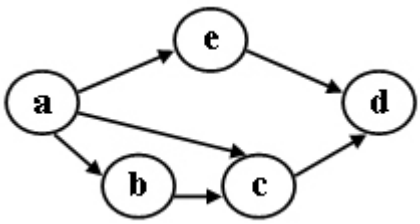
1. O(n)
2. O(n+e)
3. O(n2)
4. O(n×e)

解析：先需要O(e)求入度，然后需要O(n)求序列

答案：B

2-5

对下图进行拓扑排序，可以得到不同的拓扑序列的个数是： (2分)



1. 4
2. 3
3. 2
4. 1

解析：一个一个数，选B

答案：B

2-6

已知有向图G=(V, E)，其中`V = {v1, v2, v3, v4, v5, v6}`，`E = {<v1,v2>, <v1,v4>, <v2,v6>, <v3,v1>, <v3,v4>, <v4,v5>, <v5,v2>, <v5,v6>}`。G的拓扑序列是： (2分)

1. `v3, v1, v4, v5, v2, v6`
2. `v3, v4, v1, v5, v2, v6`
3. `v1, v3, v4, v5, v2, v6`
4. `v1, v4, v3, v5, v2, v6`

解析：建图模拟即可，画清楚图

答案：`A`

2.编程题

十.查找-顺序查找、二分查找

1.单选题

2-1

已知一个长度为16的顺序表L，其元素按关键字有序排列。若采用二分查找法查找一个L中不存在的元素，则关键字的比较次数最多是： (2分) A. 4 B. 5 C. 6 D. 7

解：二分查找的最多比较次数为 $\lfloor \log_2 n \rfloor + 1$,

故答案是B: 5

2-2

用二分查找从100个有序整数中查找某数，最坏情况下需要比较的次数是：(2分)

A. 7 B. 10 C. 50 D. 99

解：解法同上

故答案是A:7

2-3

在有n（n>1000）个元素的升序数组`A`中查找关键字x。查找算法的伪代码如下所示：

```
k = 0;
while ( k<n 且 A[k]<x )  k = k+3;
if ( k<n 且 A[k]==x )  查找成功;
else if ( k-1<n 且 A[k-1]==x ) 查找成功;
     else if ( k-2<n 且 A[k-2]==x ) 查找成功;
          else 查找失败;

```

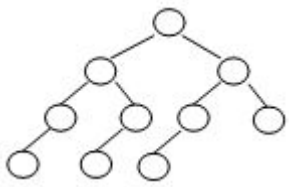
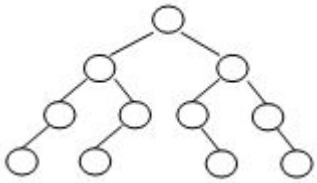
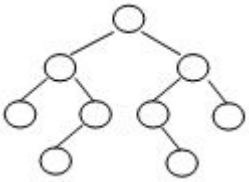
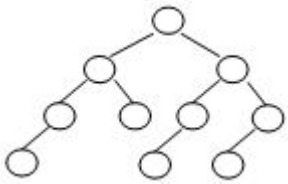
本算法与二分查找（折半查找）算法相比，有可能具有更少比较次数的情形是：(2分)

A. 当x不在数组中 B. 当x接近数组开头处 C. 当x接近数组结尾处 D. 当x位于数组中间位置

解: 显然可得，选B，这样第一个while循环可以少执行

2-4

下列二叉树中，可能成为折半查找判定树（不含外部结点）的是：(4分)

A.  B.  C.  D. 

解：通常情况下，我们是选取mid=$\lfloor(low+high)/2\rfloor$，此时叶子节点应该从最右边开始，是偏向右边的;当mid=$\lceil(low+high)/2\rceil$时，叶子节点偏向相反，应该是从最左边开始，偏向左边的。我们可以自行画图验证。综上所述，A选项正确，B（有两种偏向并存），C（有两种偏向并存，并且没有从最左，左右开始），D错误（中间有一个叶子节点不存在）

2.编程题

7-1 两个有序序列的中位数（25 分）

已知有两个等长的非降序序列S_1, S_2, 设计函数求S_1与S_2并集的中位数。有序序列A_0,A_1,⋯,A_{N-1}的中位数指$A_{(N-1)}/2$的值,即第⌊(N+1)/2⌋个数（A_0为第1个数）。

输入格式:

输入分三行。第一行给出序列的公共长度N（0<N≤100000），随后每行输入一个序列的信息，即N个非降序排列的整数。数字用空格间隔。

输出格式:

在一行中输出两个输入序列的并集序列的中位数。

输入样例1:

```
5
1 3 5 7 9
2 3 4 5 6

```

输出样例1:

```
4

```

输入样例2:

```
6
-100 -10 1 1 1 1
-50 0 2 3 4 5

```

输出样例2:

```
1
```

解题思路

利用归并排序中的两路归并思想，将两个区间归并成一个区间，然后求中位数即可

代码实现

```
#include <cstdio>
using namespace std;
struct Node {
    int v;
};
const int maxn = 2*100000 + 10;
int head[maxn];
int nexts[maxn];
int vec1[maxn], vec2[maxn], vec[maxn];
int main() {
    int n;
    scanf("%d", &n);
    for(int i = 0; i < n; i++) {
        scanf("%d", &vec1[i]);
    }
    for(int i = 0; i < n; i++) {
        scanf("%d", &vec2[i]);
    }
    int i = 0, j = 0;
    int cnt = 0;
    while(i < n && j < n) {
        if(vec1[i] < vec2[j]) {
            vec[cnt++] = vec1[i];
            i++;
        }
        else if (vec1[i] > vec2[j]) {
            vec[cnt++] = vec2[j];
            j++;
        }
        else {
            vec[cnt++] = vec2[j];
            i++;
        }
    }
    while(i < n) {
        vec[cnt++] = vec1[i++];

    }
```

```
    while(j < n) {
        vec[cnt++] = vec2[j++];
    }
    printf("%d\n", vec[(cnt - 1) / 2]);
    return 0;
}
```

十一.查找-二叉排序树new

0.错误纠正

2-3, A->D，已纠正

2-9, D->B，已纠正

1.单选题

2-1

若二叉搜索树是有N个结点的完全二叉树，则不正确的说法是：(1分)

A. 所有结点的平均查找效率是O(logN) B. 最小值一定在叶结点上 C. 最大值一定在叶结点上 `D. 中位值结点在根结点或根的左子树上`

解析：A答案显然正确；因为是完全二叉树，考虑只有两个节点的完全二叉树，可得B答案正确，C选项错误，D选项错误（两个节点的中位值好像不存在啊）

答案：`C`

2-2

若一棵二叉树的后序遍历序列是{ 1, 3, 2, 6, 5, 7, 4 }，中序遍历序列是{ 1, 2, 3, 4, 5, 6, 7 }，则下列哪句是错的？(3分)

A. 这是一棵完全二叉树

B. 2是1和3的父结点

C. 这是一棵二叉搜索树

D. 7是5的父结点

解析：根据后序遍历和中序遍历建树去判断即可

答案：`A`

2-3

将{ 32, 2, 15, 65, 28, 10 }依次插入初始为空的二叉搜索树。则该树的前序遍历结果是：(3分)

A. 2, 10, 15, 28, 32, 65

B. 32, 2, 10, 15, 28, 65

C. 10, 28, 15, 2, 65, 32

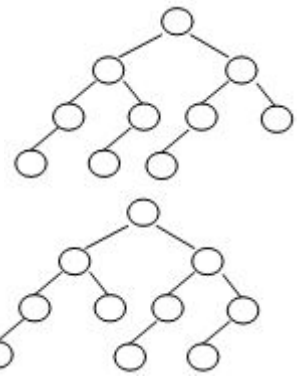
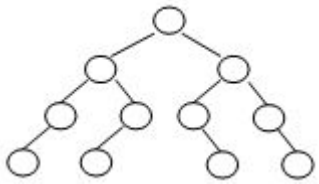
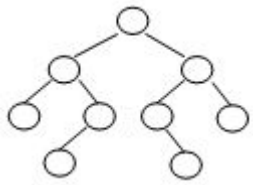
D. 32, 2, 15, 10, 28, 65

解析：建出二叉搜索树，然后去判断即可

答案：D

2-4

下列二叉树中，可能成为折半查找判定树（不含外部结点）的是： (4分)

A.  B.  C.  D.

解析：略

答案：A

2-5

若一棵二叉树的前序遍历序列是{ 4, 2, 1, 3, 6, 5, 7 }，中序遍历序列是{ 1, 2, 3, 4, 5, 6, 7 }，则下列哪句是错的？(3分)

A. 这是一棵完全二叉树 B. 所有的奇数都在叶子结点上 C. 这是一棵二叉搜索树 D. 2是5的父结点

解析：建出二叉搜索树，然后去判断即可

答案：D

2-6

将{ 5, 11, 13, 1, 3, 6 }依次插入初始为空的二叉搜索树。则该树的后序遍历结果是：(3分)

A. 3, 1, 5, 6, 13, 11 B. 3, 1, 6, 13, 11, 5 C. 1, 3, 11, 6, 13, 5 D. 1, 3, 5, 6, 13, 11

解析：建出二叉搜索树，然后去判断即可

答案：B

2-7

对二叉搜索树进行什么遍历可以得到从小到大的排序序列？ (1分)

A. 前序遍历 B. 后序遍历 C. 中序遍历 D. 层次遍历

解析：根据二叉搜索树的定理可得，左子树小于根节点，右子树大于根节点，所以应当才有中序遍历，先左，后中，再右

答案：C

2-8

若二叉搜索树是有N个结点的完全二叉树，则不正确的说法是：(1分)

A. 平均查找效率是O(logN) B. 最大值一定在最后一层 C. 最小值一定在叶结点上 D. 中位值结点在根结点或根的左子树上

解析：B选项，还是考虑只有 两个元素的完全二叉树，最大值在第一层，故选B

答案：`B`

2-9

已知8个数据元素为（34，76，45，18，26，54，92，65），按照依次插入结点的方法生成一棵二叉搜索树后，最后两层上的结点总数为： (2分)

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

解析：建树判断即可

答案：`B`

2-10

下列叙述正确的是（ ）。 (2分)

A. 在任意一棵非空二叉搜索树，删除某结点后又将其插入，则所得二叉搜索树与删除前原二叉搜索树相同。 B. 二叉树中除叶结点外, 任一结点X，其左子树根结点的值小于该结点（X）的值；其右子树根结点的值≥该结点（X）的值，则此二叉树一定是二叉搜索树。 C. 虽然给出关键字序列的顺序不一样，但依次生成的二叉搜索树却是一样的。 D. 在二叉搜索树中插入一个新结点，总是插入到最下层，作为新的叶子结点。

解析：因为对于同样的元素集合，插入的顺序不同，二叉树的形状也不同，所以A，C错误，B选项中大于等于表述不正确，根据二叉搜索树定义可知应该是大于,故选D

答案：`D`

2.编程题

7-1 是否同一棵二叉搜索树（25 分）

给定一个插入序列就可以唯一确定一棵二叉搜索树。然而，一棵给定的二叉搜索树却可以由多种不同的插入序列得到。例如分别按照序列{2, 1, 3}和{2, 3, 1}插入初始为空的二叉搜索树，都得到一样的结果。于是对于输入的各种插入序列，你需要判断它们是否能生成一样的二叉搜索树。

输入格式:

输入包含若干组测试数据。每组数据的第1行给出两个正整数N (≤10)和L，分别是每个序列插入元素的个数和需要检查的序列个数。第2行给出N个以空格分隔的正整数，作为初始插入序列。最后L行，每行给出N个插入的元素，属于L个需要检查的序列。

简单起见，我们保证每个插入序列都是1到N的一个排列。当读到N为0时，标志输入结束，这组数据不要处理。

输出格式:

对每一组需要检查的序列，如果其生成的二叉搜索树跟对应的初始序列生成的一样，输出“Yes”，否则输出“No”。

输入样例:

```
4 2
3 1 4 2
3 4 1 2
3 2 4 1
2 1
2 1
1 2
0

```

输出样例:

```
Yes
No
No

```

鸣谢青岛大学周强老师补充测试数据！

解题思路：

两棵二叉搜索树相等应该满足这样的条件：

- 左子树相等
- 右子树相等
- 根节点相等

  根据这个条件，递归去解即可，注意边界以及空子树的特殊处理，具体可见代码实现

参考代码：

```
#include <cstdio>
  using namespace std;
  struct Node {
      int v;
      Node * lchild;
      Node * rchild;
  };
  typedef Node * BSTree;
  void InsertBST(BSTree& T, int v) {
      if(T == NULL) {
          T = new Node();
          T->lchild = NULL;
          T->rchild = NULL;
          T->v = v;
      } else if (v < T->v) {
          InsertBST(T->lchild, v);
      } else if (v > T->v) {
          InsertBST(T->rchild, v);
      }
  }
  bool CompareBST(BSTree& T1, BSTree& T2) {
      if(T1 == NULL && T2 == NULL) {
```

```
        return true;
    }
    if(!T1 || !T2) {
        return false;
    }
    if(T1->v != T2->v) {
        return false;
    }
    if(!CompareBST(T1->lchild, T2->lchild) || !CompareBST(T1->rchild, T2->rchild) ) {
        return false;
    }
    return true;
}
void CreateBST(BSTree& T, int n) {
    T = NULL;
    for (int i = 0; i < n; i++) {
        int d;
        scanf("%d", &d);
        InsertBST(T, d);
    }
}
int main() {
    int N, L;
    while(~scanf("%d", &N) && N){
        scanf("%d", &L);
        BSTree T1;
        BSTree T2;
        CreateBST(T1, N);
        for(int i = 0; i < L; i++) {
            CreateBST(T2, N);
            if(CompareBST(T1, T2)) {
                printf("Yes\n");
            } else {
                printf("No\n");
            }
        }
    }
    return 0;
}
```

7-2 是否完全二叉搜索树（30 分）

将一系列给定数字顺序插入一个初始为空的二叉搜索树（定义为左子树键值大，右子树键值小），你需要判断最后的树是否一棵完全二叉树，并且给出其层序遍历的结果。

输入格式：

输入第一行给出一个不超过20的正整数 `N`；第二行给出 `N` 个互不相同的正整数，其间以空格分隔。

输出格式：

将输入的 `N` 个正整数顺序插入一个初始为空的二叉搜索树。在第一行中输出结果树的层序遍历结果，数字间以1个空格分隔，行的首尾不得有多余空格。第二行输出 `YES`，如果该树是完全二叉树；否则输出 `NO`。

输入样例1：

```
9
38 45 42 24 58 30 67 12 51

```

输出样例1：

```
38 45 24 58 42 30 12 67 51
YES

```

输入样例2：

```
8
38 24 12 45 58 67 42 51

```

输出样例2：

```
38 45 24 58 42 12 67 51
NO
```

解题思路：

根据完全二叉树的性质，最后一层可以不满，但是叶子必须从左到右顺排着，不允许中间有空节点，因此我们可以在层次遍历的过程中，通过设置标志变量的方式判断输入的树是否满足叶子节点之间有空节点

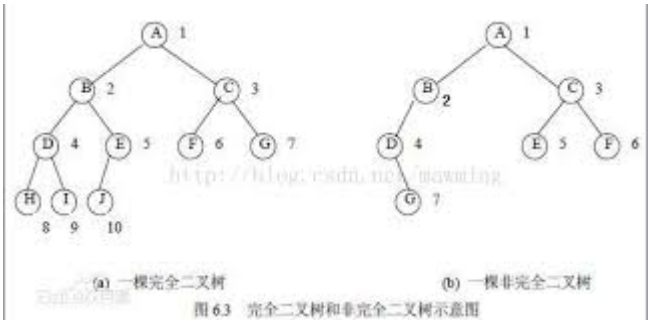


图 6.3 完全二叉树和非完全二叉树示意图

上边右图G节点与F节点中间为空，因此不是完全二叉树。要实现判断，需要虚拟一个空节点便于处理。

题实现并不简单，可以参考下边的代码

参考代码：

> 注意：代码中用了部分STL的库，这些库的使用不是必要的，容易自己去实现，这里就不再替换了

```
#include <cstdio>
#include <queue>
#include <vector>
using namespace std;
struct Node {
    int v;
    Node * lchild;
    Node * rchild;
};
```

```
typedef Node * BSTree;
BSTree NULLTree;
void InsertBST(BSTree& T, int v) {
    if(T == NULLTree) {
        T = new Node();
        T->lchild = NULLTree;
        T->rchild = NULLTree;
        T->v = v;
    } else if (v > T->v) {
        InsertBST(T->lchild, v);
    } else if (v < T->v) {
        InsertBST(T->rchild, v);
    }
}
void CreateBST(BSTree& T, int n) {
    T = NULLTree;
    for (int i = 0; i < n; i++) {
        int d;
        scanf("%d", &d);
        InsertBST(T, d);
    }
}
bool LevelBST(BSTree& T, vector<int>& vec) {
    queue<BSTree> q;
    q.push(T);
    bool is_ok = true;
    int flag = 0;
    while(!q.empty()) {
        BSTree t = q.front();
        q.pop();
        if(t != NULLTree) {
            if(flag > 0) {
                flag = -1;
            }
            vec.push_back(t->v);
            if (t->lchild) {
                q.push(t->lchild);
            }
            else {
                q.push(NULLTree);
            }
            if (t->rchild) {
                q.push(t->rchild);
            }
            else {
                q.push(NULLTree);
            }

        } else {
            if(flag >= 0) {
                flag++;
            }

        }
```

```
    }
    return flag == -1;
}
int main() {
    int N, L;
    NULLTree = new Node();
    NULLTree->lchild = NULL;
    NULLTree->rchild = NULL;
    NULLTree->v = -1;
    while(~scanf("%d", &N) && N){
        BSTree T;
        CreateBST(T, N);
        vector<int> vec;
        bool flag = LevelBST(T, vec);
        printf("%d", vec[0]);
        for(int i = 1; i < vec.size(); i++) {
            printf(" %d", vec[i]);
        }
        printf("\n");
        if(!flag){
            printf("YES\n");
        } else {
            printf("NO\n");
        }
    }
    return 0;
}
```

十二.查找-哈希查找

1.单选题

2-1

散列冲突可以被描述为：(1分)

1. 两个元素除了有不同键值，其它都相同
2. 两个有不同数据的元素具有相同的键值
3. 两个有不同键值的元素具有相同的散列地址
4. 两个有相同键值的元素具有不同的散列地址

解析：由散列冲突的定义，选C

答案：`C`

2-2

将10个元素散列到100000个单元的哈希表中，是否一定产生冲突？(1分)

1. 一定会
2. 可能会
3. 一定不会
4. 有万分之一的可能会

解析：是否冲突与散列函数有关，比如对于散列函数H(key) = $(key + 100000)$ %100000，1与-99999是冲突的

答案：B

2-3

设散列表的地址区间为[0,16]，散列函数为H(Key)=Key%17。采用线性探测法处理冲突，并将关键字序列{ 26，25，72，38，8，18，59 }依次存储到散列表中。元素59存放在散列表中的地址是： (2分)

1. 8
2. 9
3. 10
4. 11

解析：模拟线性探测法过程，遇到冲突元素，顺着往后找可以放置这个元素的位置

答案：D

2-4

采用线性探测法解决冲突时所产生的一系列后继散列地址： (1分)

1. 必须大于等于原散列地址
2. 必须小于等于原散列地址
3. 可以大于或小于但不等于原散列地址
4. 对地址在何处没有限制

解析：线性探测法是将原散列表想象成一个循环表，发生冲突时，从冲突地址的下一个位置顺序寻找空单元，可以折返到表头继续找，找不到就说明散列表已满故选C

答案：C

2-5

将元素序列{18，23，11，20，2，7，27，33，42，15}按顺序插入一个初始为空的、大小为11的散列表中。散列函数为：H(Key)=Key%11，采用线性探测法处理冲突。问：当第一次发现有冲突时，散列表的装填因子大约是多少？ (3分)

1. 0.27
2. 0.45
3. 0.64
4. 0.73

解析：根据装填因子计算公式$\alpha = \frac{表中填入的记录数}{散列表的长度}$，模拟计算即可

答案：B

2-6

给定散列表大小为11，散列函数为H(Key)=Key%11。按照线性探测冲突解决策略连续插入散列值相同的4个元素。问：此时该散列表的平均不成功查找次数是多少？ (2分)

1. 1
2. 4/11
3. 21/11
4. 不确定

解析：无论插入的是四个什么数，查找不成功的比较次数为恒值$\frac{1}{11}(11-4+1+2+3+4+5)=\frac{21}{11}$

答案：C

2-7

从一个具有N个结点的单链表中查找其值等于X的结点时，在查找成功的情况下，需平均比较多少个结点？(2分)

1. N/2
2. N
3. (N−1)/2
4. (N+1)/2

解析：这道单链表题怎么乱入？？解法略

答案：D

***2-8**

给定输入序列 {4371, 1323, 6173, 4199, 4344, 9679, 1989} 以及散列函数 h(X)=X%10。如果用大小为10的散列表，并且用开放定址法以及一个二次散列函数h2(X)=7−(X%7)解决冲突，则输入各项经散列后在表中的下标为：（-1表示相应的插入无法成功）(2分)

1. 1, 3, 3, 9, 4, 9, 9
2. 1, 3, 4, 9, 7, 5, -1
3. 1, 3, 4, 9, 5, 0, 8
4. 1, 3, 4, 9, 5, 0, 2

解析：

答案：B

2-9

设数字 {4371, 1323, 6173, 4199, 4344, 9679, 1989} 在大小为10的散列表中根据散列函数 h(X)=X%10得到的下标对应为 {1, 3, 4, 9, 5, 0, 2}。那么继续用散列函数 “h(X)=X%表长”实施再散列并用线性探测法解决冲突后，它们的下标变为：(3分)

1. 11, 3, 13, 19, 4, 0, 9
2. 1, 3, 4, 9, 5, 0, 2
3. 1, 12, 9, 13, 20, 19, 11
4. 1, 12, 17, 0, 13, 8, 14

解析：题目据说出错了

答案：

2-10

若N个关键词被散列映射到同一个单元，并且用分离链接法解决冲突，则找到这N个关键词所用的比较次数为：(2分)

1. N(N+1)/2
2. N(N−1)/2
3. N+1
4. N

解析：1+ 2 + … + N = N(N+1)/2

答案：A

2-11

给定散列表大小为17，散列函数为H(Key)=Key%17。采用平方探测法处理冲突：hi(k)=(H(k)±i^2)%17将关键字序列{ 6, 22, 7, 26, 9, 23 }依次插入到散列表中。那么元素23存放在散列表中的位置是：(3分)

1. 0
2. 2
3. 6
4. 15

解析：模拟平方探测法的过程，当插入23的时候，发现i = +1, -1, +2时，位置均被占用，此时选择i = -2,位置是23%17-4=2

答案：B

2-12

给定散列表大小为17，散列函数为H(Key)=Key%17。采用平方探测法处理冲突：hi(k)=(H(k)±i^2)%17将关键字序列{ 23, 22, 7, 26, 9, 6 }依次插入到散列表中。那么元素6存放在散列表中的位置是：(3分)

1. 154
2. 10
3. 6
4. 2

解析：同上

答案：D

2-13

将元素序列{18, 23, 4, 26, 31, 33, 17, 39}按顺序插入一个初始为空的、大小为13的散列表中。散列函数为：H(Key)=Key%13，采用线性探测法处理冲突。问：当第一次发现有冲突时，散列表的装填因子大约是多少？ (3分)

1. 0.54
2. 0.63
3. 0.31
4. 0.62

解析：模拟计算即可，第一次发现冲突时，此时表中共有4个元素，4/13=0.31，选C

答案：C

2-14

给定散列表大小为11，散列函数为H(Key)=Key%11。按照线性探测冲突解决策略连续插入散列值相同的5个元素。问：此时该散列表的平均不成功查找次数是多少？(2分)

1. 26/11
2. 5/11
3. 1
4. 不确定

解析：$ASL_{unsucc}=\frac{1}{r}\sum_{i=1}^{r}C_i$,r为散列函数取值的个数，$C_i$ 为函数取值为i时查找失败的比较次数

；比较次数就是从查找的位置开始到遇到空的位置的比较次数。带入公式解即得答案

答案：`A`

2.函数题

6-1 线性探测法的查找函数（20 分）

试实现线性探测法的查找函数。

函数接口定义：

```
Position Find( HashTable H, ElementType Key );
```

其中`HashTable`是开放地址散列表，定义如下：

```
#define MAXTABLESIZE 100000  /* 允许开辟的最大散列表长度 */
typedef int ElementType;     /* 关键词类型用整型 */
typedef int Index;           /* 散列地址类型 */
typedef Index Position;      /* 数据所在位置与散列地址是同一类型 */
/* 散列单元状态类型，分别对应：有合法元素、空单元、有已删除元素 */
typedef enum { Legitimate, Empty, Deleted } EntryType;

typedef struct HashEntry Cell; /* 散列表单元类型 */
struct HashEntry{
    ElementType Data; /* 存放元素 */
    EntryType Info;   /* 单元状态 */
};

typedef struct TblNode *HashTable; /* 散列表类型 */
struct TblNode {   /* 散列表结点定义 */
    int TableSize; /* 表的最大长度 */
    Cell *Cells;   /* 存放散列单元数据的数组 */
};

```

函数`Find`应根据裁判定义的散列函数`Hash( Key, H->TableSize )`从散列表`H`中查到`Key`的位置并返回。如果`Key`不存在，则返回线性探测法找到的第一个空单元的位置；若没有空单元，则返回`ERROR`。

裁判测试程序样例：

```
#include <stdio.h>

#define MAXTABLESIZE 100000  /* 允许开辟的最大散列表长度 */
typedef int ElementType;     /* 关键词类型用整型 */
typedef int Index;           /* 散列地址类型 */
typedef Index Position;      /* 数据所在位置与散列地址是同一类型 */
/* 散列单元状态类型，分别对应：有合法元素、空单元、有已删除元素 */
typedef enum { Legitimate, Empty, Deleted } EntryType;


```

```
typedef struct HashEntry Cell; /* 散列表单元类型 */
struct HashEntry{
    ElementType Data; /* 存放元素 */
    EntryType Info;   /* 单元状态 */
};

typedef struct TblNode *HashTable; /* 散列表类型 */
struct TblNode {   /* 散列表结点定义 */
    int TableSize; /* 表的最大长度 */
    Cell *Cells;   /* 存放散列单元数据的数组 */
};

HashTable BuildTable(); /* 裁判实现，细节不表 */
Position Hash( ElementType Key, int TableSize )
{
    return (Key % TableSize);
}

#define ERROR -1
Position Find( HashTable H, ElementType Key );

int main()
{
    HashTable H;
    ElementType Key;
    Position P;

    H = BuildTable();
    scanf("%d", &Key);
    P = Find(H, Key);
    if (P==ERROR)
        printf("ERROR: %d is not found and the table is full.\n", Key);
    else if (H->Cells[P].Info == Legitimate)
        printf("%d is at position %d.\n", Key, P);
    else
        printf("%d is not found.  Position %d is returned.\n", Key, P);

    return 0;
}

/* 你的代码将被嵌在这里 */

```

输入样例1：（注：-1表示该位置为空。下同。）

```
11
11 88 21 -1 -1 5 16 7 6 38 10
38

```

输出样例1：

```
38 is at position 9.

```

输入样例2：

```
11
11 88 21 -1 -1 5 16 7 6 38 10
41

```

输出样例2：

```
41 is not found.  Position 3 is returned.

```

输入样例3：

```
11
11 88 21 3 14 5 16 7 6 38 10
41

```

输出样例3：

```
ERROR: 41 is not found and the table is full.
```

解题思路：

线性探测法的实现，如果第一次散列得到的位置为空，那么元素不存在；如果不为空并且等于要查找的值，则找到，否则继续查找下一个位置，并且判断是否等于要查找的元素，直到遍历整个哈希表。

参考代码：

```
Position Find( HashTable H, ElementType Key ) {
  Position p;
  p = Hash(Key,H->TableSize);
  int a = p;
  int flag = 0;
  while(1) {
    if(H->Cells[p].Info != Legitimate) break;
    if(H->Cells[p].Info == Legitimate && H->Cells[p].Data == Key) break;
    if(p == a && flag == 1) return ERROR;
    ++ p;
    p = p % H->TableSize;
    flag = 1;
  }
  return p;
}
```

十三.排序-插入排序

1.单选题

2-1

对一组包含10个元素的非递减有序序列，采用直接插入排序排成非递增序列，其可能的比较次数和移动次数分别是： (2分)

1. 100, 100
2. 100, 54
3. 54, 63
4. 45, 44

解析：假设原序列是1,2,3,4,5,6,7,8,9那么比较和移动的次数经过计算都为$\frac{\sum_{i=0}^{n=9} i}{n}$ = 45。选项中没有合适的，当序列中出现重复的元素时，如1,2,2,3,4,5，这种情况，移动第2个2的时候移动次数比比较次数少了一次，故选D

答案：D

2-2

设有1000个元素的有序序列，如果用二分插入排序再插入一个元素，则最大比较次数是：(2分)

1. 1000
2. 999
3. 500
4. 10

解析：根据最大比较次数公式$\lfloor \log_2 n \rfloor + 1$，计算可得答案为10

答案：D

2-3

用直接插入排序方法对下面四个序列进行排序（由小到大），元素比较次数最少的是（ ）。 (2分)

1. 94,32,40,90,80,46,21,69
2. 21,32,46,40,80,69,90,94
3. 90,69,80,46,21,32,94,40
4. 32,40,21,46,69,94,90,80

解析：元素比较次数与逆序数相关，从第二个元素开始数每个元素有多少个数与之是逆序数，加起来便是答案，因为每个元素之前的所有元素肯定已经排好序了，其逆序数肯定都放在了后边，因为我们与大于等于当前数的元素比较，大于等于当前数的就是当前数的逆序数。因此比较次数等于逆序数的个数。

A选项逆序数和为 1 + 1 + 1 + 2 + 3 + 6 + 3 = 17

B选项逆序数和为 0 + 0 + 1 + 0 + 1 + 0 + 0 = 2

C，D选项自行计算

答案：B

2.编程题

7-1 排序（25 分）

给定N个（长整型范围内的）整数，要求输出从小到大排序后的结果。

本题旨在测试各种不同的排序算法在各种数据情况下的表现。各组测试数据特点如下：

数据1：只有1个元素；

数据2：11个不相同的整数，测试基本正确性；

数据3：103个随机整数；

数据4：104个随机整数；

数据5：105个随机整数；

数据6：105个顺序整数；

数据7：105个逆序整数；

数据8：105个基本有序的整数；

数据9：105个随机正整数，每个数字不超过1000。

输入格式:

输入第一行给出正整数N（≤105），随后一行给出N个（长整型范围内的）整数，其间以空格分隔。

输出格式:

在一行中输出从小到大排序后的结果，数字间以1个空格分隔，行末不得有多余空格。

输入样例:

```
11
4 981 10 -17 0 -20 29 50 8 43 -5
```

输出样例:

```
-20 -17 -5 0 4 8 10 29 43 50 981
```

解题思路：

参考代码中是用插入排序来实现的，但是代码会超时，应该使用更快的归并排序或者快速排序或者希尔排序或者堆排序，这里因为是想考察插入排序的内容，就用插入排序写了，同时使用了二分去减少比较次数.

参考代码:

```
#include <cstdio>
using namespace std;
int A[1000000];
int search(int *A, int l, int r) {
    while(l <= r) {
        int m = (l + r) / 2;
        if(A[0] <= A[m]) {
            r = m - 1;
        } else {
```

```
            l = m + 1;
        }
    }
    return l;
}
int main() {
    int n;
    scanf("%d", &n);
    for (int i = 1; i <= n; i++) {
        scanf("%d", &A[i]);
    }
    for (int i = 1; i <= n; i++) {
        A[0] = A[i];
        int j;
        int t = search(A, 1, i - 1);
        for (j = i; j > t; j--) {
            A[j] = A[j - 1];
        }
        A[j] = A[0];
    }
    printf("%d", A[1]);
    for (int i = 2; i <= n; i++) {
        printf(" %d", A[i]);
    }
    printf("\n");
    return 0;
}
```

十四.排序--交换类排序与选择类排序

1.判断题

1-1

对N个记录进行堆排序，需要的额外空间为O(N)。 (1分)

解析：需要O(1)的额外空间

答案：F

1-2

对N个记录进行简单选择排序，比较次数和移动次数分别为O(N^2)和O(N)。 (1分)

解析：

答案：T

1-3

希尔排序是稳定的算法。 (1分)

解析：希尔排序过程中因为不是顺需插入的，会破坏相等元素的相对位置，（意会吧，我将不出来）

答案：F

作者: DS课程组

1-4

对N个不同的数据采用冒泡排序进行从大到小的排序，当元素基本有序时交换元素次数肯定最多。(1分)

解析：基本有序时不发生交换

答案：F

2.单选题

2-1*

在快速排序的一趟划分过程中，当遇到与基准数相等的元素时，如果左右指针都会停止移动，那么当所有元素都相等时，算法的时间复杂度是多少？(2分)

1. O($logN$)
2. O(N)
3. O($NlogN$)
4. O(N^2)

解析：考察对快速排序过程的理解。首先我们要记住，当对进行完一趟排序之后，如果基准元素在接近于中间的位置，那么它的时间复杂度自然是极好的，每趟排序我们都要对元素比较n次，赋值n次，总共进行了log_2n趟排序，所以时间复杂度是log_2n；但如果我们进行完一趟排序之后，如果基准元素在接近于最左或者最右的位置，那么我们排序的趟数肯定会增加，极端条件下达到n。

回到这道题，当遇到基准元素的时候左右指针都停止了移动，那么进行完一趟排序的时候，基准元素一定位于接近于中间的位置（想一想为什么），因此时间复杂度为O($NlogN$)

答案：C

2-2*

在快速排序的一趟划分过程中，当遇到与基准数相等的元素时，如果左右指针都不停止移动，那么当所有元素都相等时，算法的时间复杂度是多少？(2分)

1. O($logN$)
2. O(N)
3. O($NlogN$)
4. O(N^2)

解析：与上题属于一个系列的，当遇到基准元素的时候左右指针都不停止移动，那么进行完一趟排序的时候，基准元素一定位于接近于右边的位置（想一想为什么），因此时间复杂度为O(N^2)

答案：D

2-3*

在快速排序的一趟划分过程中，当遇到与基准数相等的元素时，如果左指针停止移动，而右指针在同样情况下却不停止移动，那么当所有元素都相等时，算法的时间复杂度是多少？(2分)

1. O($logN$)
2. O(N)
3. O($NlogN$)
4. O(N^2)

解析：与上题属于一个系列的，当遇到基准元素的时候右指针都不停止移动，因为我们通常的写法会先从右往左找一遍元素，在这道题目的条件下就已经遍历到最左边了，此时基准元素就被安插到了最右边，，因此时间复杂度为O(N^2)

答案：D

2-4

对N个不同的数据采用冒泡算法进行从大到小的排序，下面哪种情况下肯定交换元素次数最多？ (1分)

1. 从小到大排好的
2. 从大到小排好的
3. 元素无序
4. 元素基本有序

解析：由显然易得，选A，此时每次比较都要交换，太悲催了

答案：A

2-5

对于7个数进行冒泡排序，需要进行的比较次数为： (2分)

1. 7
2. 14
3. 21
4. 49

解析：(6 + 5 + 4 + 3 +2 + 1) / 2 = 21

答案：C

2-6

有组记录的排序码为{ 46，79，56，38，40，84 }，则利用堆排序的方法建立的初始堆为： (2分)

1. 79，46，56，38，40，80
2. 84，79，56，46，40，38
3. 84，56，79，40，46，38
4. 84，79，56，38，40，46

解析：模拟堆排序建初始堆过程即可求解

答案：D

2-7

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：(2分)

1. 每次划分后，先处理较长的分区可以减少递归次数
2. 每次划分后，先处理较短的分区可以减少递归次数
3. 递归次数与每次划分后得到的分区处理顺序无关
4. 递归次数与初始数据的排列次序无关

解析：A，B选项显然很扯，两部分分区处理都是独立的，不会减少递归次数，D选项肯定是有关的，不然也不会存在最坏时间复杂度

答案：C

2-8

对N个记录进行快速排序，在最坏的情况下，其时间复杂度是：(1分)

1. O(N)
2. O(NlogN)
3. O(N^2)
4. O(N^2logN)

解析：考察课本基础知识

答案：C

2-9

有组记录的排序码为{46，79，56，38，40，84}，采用快速排序（以位于最左位置的对象为基准而）得到的第一次划分结果为：(2分)

1. {38,46,79,56,40,84}
2. {38,79,56,46,40,84}
3. {38,46,56,79,40,84}
4. {40,38,46,56,79,84}

解析：本题可以模拟快排过程去做，也可以根据快速排序第一次划分结束后，基准元素左边元素应该都小于基准元素，右边元素应该都大于基准元素，故选D

答案：D

2-10

对于序列{ 49，38，65，97，76，13，27，50 }，按由小到大进行排序，下面哪一个是初始步长为4的希尔排序法第一趟的结果？(2分)

1. 13,27,38,49,50,65,76,97
2. 49,13,27,50,76,38,65,97
3. 49,76,65,13,27,50,97,38
4. 97,76,65,50,49,38,27,13

解析：观察序列中的49,76，他们相距4并且已经有序，然后第一趟排序后应该位置不变，观察选项中，只有B选项满足

答案：B

2-11

给定初始待排序列{ 15，9，7，8，20，-1，4 }。如果希尔排序第一趟结束后得到序列为{ 15，-1，4，8，20，9，7 }，则该趟增量为： (2分)

1. 1
2. 2
3. 3
4. 4

解析：这道题的可以将选项带进去验证，比如带入D选项，发现在此增量下，元素都是有序的，故选D

答案：D

2-12

对N个元素采用简单选择排序，比较次数和移动次数分别为： (1分)

1. O(N^2), O(N)
2. O(N), O($logN$)
3. O($logN$), O(N^2)
4. O($NlogN$), O($NlogN$)

解析：考察基础课本知识

答案：A

2-13

对于10个数的简单选择排序，最坏情况下需要交换元素的次数为： (2分)

1. 9
2. 36
3. 45
4. 100

解析：当序列为逆序时，此时需要最大交换次数，为n-1

答案：A

2-14

对N个记录进行堆排序，最坏的情况下时间复杂度是： (1分)

1. O(logN)
2. O(N)
3. O(NlogN)
4. O(N2)

解析：考察课本基础知识

答案：C

2-15

对N个记录进行堆排序，需要的额外空间为： (1分)

1. O(1)
2. O(logN)

3. O(N)
4. O(NlogN)

解析：考察课本基础知识

答案：A

3.编程题

十五.排序-归并排序与基数排序

1.判断题

1-1

对N个记录进行归并排序，归并趟数的数量级是O($NlogN$)。 (2分)

解析：考察课本基础知识，并且要认真看题呀

答案：F

2.单选题

2-1

对N个记录进行归并排序，归并趟数的数量级是： (1分)

1. O(logN)
2. O(N)
3. O(NlogN)
4. O(N2)

解析： 仔细读题即可

答案：A

2-2

对N个记录进行归并排序，空间复杂度为： (1分)

1. O(logN)
2. O(N)
3. O(NlogN)
4. O(N2)

解析： 略

答案：B

2-3

给出关键字序列{ 431, 56, 57, 46, 28, 7, 331, 33, 24, 63 }，下面哪个选择是按次位优先（LSD）链式基数排序进行了一趟分配和收集的结果？ (2分)

1. →331→431→33→63→24→56→46→57→7→28
2. →56→28→431→331→33→24→46→57→63→7
3. →431→331→33→63→24→56→46→57→7→28
4. →57→46→28→7→33→24→63→56→431→331

解析： 考察对基数排序的掌握，基数排序的方式可以采用LSD（Least significant digital）或MSD（Most significant digital），LSD的排序方式由键值的最右边开始，而MSD则相反，由键值的最左边开始。

[隐藏]

答案：C

2-4

输入105个只有一位数字的整数，可以用O(N)复杂度将其排序的算法是：(1分)

1. 快速排序
2. 插入排序
3. 桶排序
4. 希尔排序

解析： 略

答案：C

2-5

数据序列{ 3, 1, 4, 11, 9, 16, 7, 28 }只能是下列哪种排序算法的两趟排序结果？(2分)

1. 冒泡排序
2. 快速排序
3. 插入排序
4. 堆排序

解析： 排除法做即可

答案：B

2-6

对10TB的数据文件进行排序，应使用的方法是：(1分)

1. 希尔排序
2. 堆排序

3. 归并排序
4. 快速排序

解析： 考察课本基础知识，课本后边讲的几种外部排序都是基于归并的

答案：C

2-7

在内部排序时，若选择了归并排序而没有选择插入排序，则可能的理由是：(2分)

1. 归并排序的程序代码更短
2. 归并排序占用的空间更少
3. 归并排序的运行效率更高

1. 仅 2
2. 仅 3
3. 仅 1、2
4. 仅 1、3

解析： 插入排序占用空间小，代码短，就是慢一点，所以这样选

答案：B

2-8

下列排序方法中，若将顺序存储更换为链式存储，则算法的时间效率会降低的是：(2分)

1.插入排序；2.选择排序；3.起泡排序；4.希尔排序；5.堆排序

1. 仅1、2
2. 仅2、3
3. 仅3、4
4. 仅4、5

解析： 因为4访问数组的元素是跳跃的，因此用链式存储会降低时间效率；因为5是用数组形式存储的完全二叉树，要想改成链式存储肯定是要变慢的，还要加一堆指针

答案：D

2-9*

{ 12，9，11，8，7，4，5，13，23 }是下列哪种方法第二趟排序后的结果？ (2分)

1. 归并排序
2. 堆排序
3. 插入排序
4. 基数排序

解析：略

答案：B

2-10

输入104个只有一位数字的整数，可以用O(N)复杂度将其排序的算法是：(1分)

1. 桶排序

2. 快速排序
3. 插入排序
4. 希尔排序

解析：略

答案：A

2-11

将序列{ 2, 12, 16, 88, 5, 10, 34 }排序。若前2趟排序的结果如下：

- 第1趟排序后：2, 12, 16, 10, 5, 34, 88
- 第2趟排序后：2, 5, 10, 12, 16, 34, 88

则可能的排序算法是：(2分)

1. 冒泡排序
2. 快速排序
3. 归并排序
4. 插入排序

解析：依次带入选项尝试即可，首先可以排除A, D，模拟一下可排除C

答案：B

2-12

将序列{ 2, 12, 16, 88, 5, 10, 34 }排序。若前2趟排序的结果如下：

- 第1趟排序后：2, 12, 16, 10, 5, 34, 88
- 第2趟排序后：2, 5, 10, 12, 16, 34, 88

则可能的排序算法是：(2分)

1. 冒泡排序
2. 归并排序
3. 插入排序
4. 快速排序

解析：依次带入选项尝试即可，首先可以排除A, C，模拟一下可排除B

答案：D

2-13

将序列{ 2, 12, 16, 88, 5, 10, 34 }排序。若前2趟排序的结果如下：

- 第1趟排序后：2, 12, 16, 10, 5, 34, 88
- 第2趟排序后：2, 5, 10, 12, 16, 34, 88

则可能的排序算法是：(2分)

1. 冒泡排序
2. 归并排序
3. 快速排序
4. 插入排序

解析：依次带入选项尝试即可，首先可以排除A, D，模拟一下可排除B

答案：C

2-14

在对N个元素进行排序时，基于比较的算法中，其“最坏时间复杂度”中最好的是： (1分)

1. O(logN)
2. O(N)
3. O(NlogN)
4. O(N^2)

解析：基于比较的算法有交换类(冒泡，快排)， 插入类（插入，希尔），归并类（归并），选择类（选择，堆排），其中归并，堆排序都有较好的最坏时间复杂度

答案：C

2-15

下列排序算法中，哪种算法可能出现：在最后一趟开始之前，所有的元素都不在其最终的位置上？（设待排元素个数N>2） (2分)

1. 冒泡排序
2. 插入排序
3. 堆排序
4. 快速排序

解析：这道题感觉自我感觉比较坑，不仔细想容易选成D，拿三个元素模拟一下，能得出B是正确答案，比如原序列 2 3 1 ，这算最后一趟之前，保证题目中的添加，然后进行后就有序了

答案：B

2-16*

若数据元素序列{ 11，12，13，7，8，9，23，4，5 }是采用下列排序方法之一得到的第二趟排序后的结果，则该排序算法只能是： (2分)

1. 冒泡排序
2. 选择排序
3. 插入排序
4. 归并排序

解析：这道题坑啊，一个是我们可以用排除法做；之所以是选C，是因为当这个前前三个元素有序时，那么进行插入排序将什么都不做（啊！）,就这样子

答案：C

2-17

数据序列{ 3，2，4，9，8，11，6，20 }只能是下列哪种排序算法的两趟排序结果？ (2分)

1. 冒泡排序
2. 选择排序
3. 插入排序
4. 快速排序

解析：首先排除A,B,C，因为他们肯定会将前两个位置排成有序的，但原序列并没有体现出这一点

答案：`D`

2-18

对一组数据{ 2，12，16，88，5，10 }进行排序，若前三趟排序结果如下： 第一趟排序结果：2，12，16，5，10，88 第二趟排序结果：2，12，5，10，16，88 第三趟排序结果：2，5，10，12，16，88 则采用的排序方法可能是： (2分)

1. 冒泡排序
2. 希尔排序
3. 归并排序
4. 基数排序

解析：逐个验证，选A

答案：`A`

2-19

就排序算法所用的辅助空间而言，堆排序、快速排序、归并排序的关系是： (1分)

1. 堆排序 < 归并排序 < 快速排序
2. 堆排序 > 归并排序 > 快速排序
3. 堆排序 < 快速排序 < 归并排序
4. 堆排序 > 快速排序 > 归并排序

解析：堆排序O(1), 快速排序O($log_2 n$),最坏O(n), 归并排序O(n)

答案：`C`

2-20

下面四种排序算法中，稳定的算法是： (1分)

1. 堆排序
2. 希尔排序
3. 归并排序
4. 快速排序

解析：归并排序，插入排序，冒泡排序是稳定的算法

答案：`C`

2-21

在基于比较的排序算法中，哪种算法的最坏情况下的时间复杂度不高于O(NlogN)？ (1分)

1. 冒泡排序
2. 归并排序
3. 希尔排序
4. 快速排序

解析：归并排序和堆排序时间复杂度比较稳定，恒为O(NlogN)，快速排序有可能达到$O(N^2)$

,冒泡排序是$O(N^2)$，希尔排序比冒泡快，但也达不到O(NlogN)的水平

答案：`B`

2-22

下列排序算法中，时间复杂度不受数据初始状态影响，恒为O(NlogN)的是：(1分)

1. 冒泡排序
2. 直接选择排序
3. 堆排序
4. 快速排序

解析：堆排序时间复杂度比较稳定，恒为O(NlogN)，快速排序有可能达到$O(N^2)$

,其他两种都是恒为$O(N^2)$

答案：`C`

2-23

输入105个只有一位数字的整数，可以用O(N)复杂度将其排序的算法是：(1分)

1. 快速排序
2. 插入排序
3. 希尔排序
4. 基数排序

解析：O(N)算法只有基数排序能做到了

答案：`D`

2-24

排序方法中，从未排序序列中依次取出元素与已排序序列中的元素进行比较，将其放入已排序序列的正确位置的方法称为：(1分)

1. 插入排序
2. 选择排序
3. 快速排序
4. 归并排序

解析：考察常用排序算法的理解

答案：`A`

3.编程题