一．是非题

（正确的打“√”，错误的打“×”。）

1. 数据结构可用三元式表示（D，S，P）。其中：D 是数据对象，S 是 D 上的关系集，P 是对 D 的基本操作集。×（DSP 是指抽象数据类型，数据结构用 DS 表示）
2. 线性表的链式存储结构具有可直接存取表中任一元素的优点。 ×
3. 队列是数据对象特定的线性表。×
4. 二叉树是一棵结点的度最大为二的树。 ×　　-----“度”
5. 邻接表可以用以表示无向图，也可用以表示有向图。 √
6. 可从任意有向图中得到关于所有顶点的拓扑次序。×
7. 一棵无向连通图的生成树是其极大的连通子图。×
8. 二叉排序树的查找长度至多为 $\log_2 n$ 。×
9. 对于一棵 m 阶的 B-树，树中每个结点至多有 m-1 个关键字。除根之外的所有非终端结点至少有⌈m/2⌉ -1 个关键字。 √
10. 对于目前所知的排序方法，快速排序具有最好的平均性能。 √
11. 顺序存储方式的优点是存储密度大，且插入、删除运算效率高。×
12. 二维数组是其数据元素为线性表的线性表。√
13. 连通图 G 的生成树是一个包含 G 的所有 n 个顶点和 n-1 条边的子图。×
14. 折半查找不适用于有序链表的查找。√
15. 完全二叉树必定是平衡二叉树。×
16. 中序线索二叉树的优点是便于在中序下查找直接前驱结点和直接后继结点。√
17. 队列是与线性表完全不同的一种数据结构。×
18. 平均查找长度与记录的查找概率有关。√
19. 广义表的表头和表尾都有可能是原子或广义表。×
20. 算法的时间复杂性越好，可读性就越差；反之，算法的可读性越好，则时间复杂性就越差。×

二．选择题

1. 若广义表 LS 满足 GetHead(LS)==GetTail(LS)，则 LS 为(b)。
   A. ( )　　B. (( ))　　C. (( ),( ))　　D. (( ),( ),( ))

2. 递归程序可借助于(b)转化为非递归程序。
   a:线性表　　b: 栈　　c:队列　　d:数组

3. 在下列数据结构中(c)具有先进先出（FIFO）特性，(b)具有先进后出(FILO)特性。
   a:线性表　　b:栈　　c:队列　　d:广义表

1. 若对编号为 1，2，3 的列车车厢依次通过扳道栈进行调度，不能得到 (e) 的序列。
   a:1,2,3　b:1,3,2　c:2,1,3　d:2,3,1　e:3,1,2　f:3,2,1

4. 对字符串 s=’data-structure’ 执行操作 replace(s,substring(s,6,8),’bas’) 的结果是（ d ）。
a: ‘database’　b: ‘data-base’　c: ‘bas’　d: ‘data-basucture’

5. 设有二维数组 $A_{5\times7}$，每一元素用相邻的 4 个字节存储，存储器按字节编址。已知 A 的起始地址为 100。则按行存储时，元素 A_{06} 的第一个字节的地址是（ d ）
按列存储时，元素 A_{06} 的第一个字节的地址是（ a ）
a: 220　b: 200　c: 140　d: 124

6. 对广义表 A=((a,(b)),(c,()),d) 执行操作 gettail(gethead(gettail(A)))
的结果是：（ b ） 。
a: ()　b: (())　c: d　d: (d)

7. 假设用于通讯的电文仅由 6 个字符组成，字母在电文中出现的频率分别为 7, 19, 22, 6, 32, 14。 若为这 6 个字母设计哈夫曼编码（设生成新的二叉树的规则是按给出的次序从左至右的结合，新生成的二叉树总是插入在最右），则频率为 7 的字符编码是（ g ），频率为 32 的字符编码是（ c ）。
a: 00　b: 01　c: 10　d: 11
e: 011　f: 110　g: 1110　h:1111

8. 对二叉排序树（ b ）可得到有序序列。
a:按层遍历　b:前序遍历　c:中序遍历　d:后序遍历

9. 已知某树的先根遍历次序为 abcdefg，后根遍历次序为 cdebgfa。
若将该树转换为二叉树,其后序遍历次序为（ d ）。
a: abcdefg　b: cdebgfa　c: cdegbfa　d: edcgfba

10. 对一棵完全二叉树进行层序编号。则编号为 n 的结点若存在右孩子,其位序是(d)。
编号为 n 的结点若存在双亲,其位置是(a)。
a: n/2　b: 2n　c:2n-1　d:2n+1　e:n　f: 2(n+1)

11. 关键路径是指在只有一个源点和一个汇点的有向无环网中源点至汇点（ c ）的路径。
a:弧的数目最多　b:弧的数目最少　c:权值之和最大　d:权值之和最小

12. 哈希表的查找效率取决于（ d ）。
a: 哈希函数　b:处理冲突的方法。　c:哈希表的装填因子。　d:以上都是

13. 从逻辑上可以把数据结构分成(c)。
A. 动态结构和静态结构　B. 顺序组织和链接组织
C. 线性结构和非线性结构　D. 基本类型和组合类型

14. 在计算递归函数时，如不用递归过程，应借助于(b)这种数据结构。
A. 线性表　B. 栈　C. 队列　D. 双向队列

15. 若已知某二叉树的中序和后序遍历序列分别 BCAEFD 和 CBFEDA，则该二叉树的先序

序列为（ a ）。

A. ABCDEF　　B. ABDCEF　　C. ABDCFE　　D. ACBDFE

16. 当待排序序列的关键字次序为倒序时，若需为之进行正序排序，下列方案中（ d ）为佳。

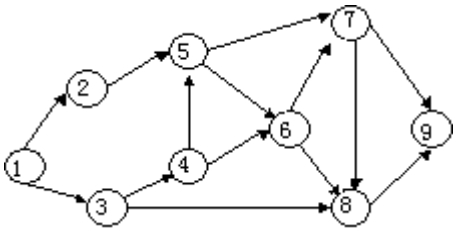
A. 起泡排序　　B. 快速排序

C. 直接插入排序　　D. 简单选择排序

17. 若从二叉树的根结点到其它任一结点的路径上所经过的结点序列按其关键字递增有序，则该二叉树是（ c ）。

A. 二叉排序树　　B. 赫夫曼树　　C. 堆　　D. 平衡二叉树

18. 下图所有可能的拓扑序列有（ b ）种。

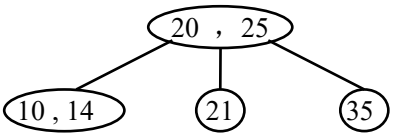


A. 2　　B. 3　　C. 4　　D. 5

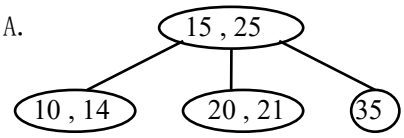
19. 下列排序算法中，（ d ）算法可能会出现：初始数据为正序时，花费的时间反而最多。

A. 堆排序　　B. 起泡排序　　C. 归并排序　　D. 快速排序

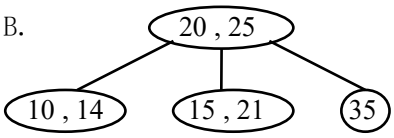
20. 右图为一棵 3 阶 B-树。在该树上插入元素 15 后的 B-树是（ c ）。



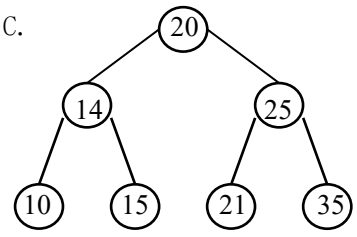
A.



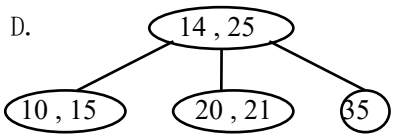
B.



C.



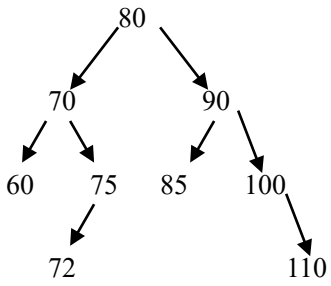
D.



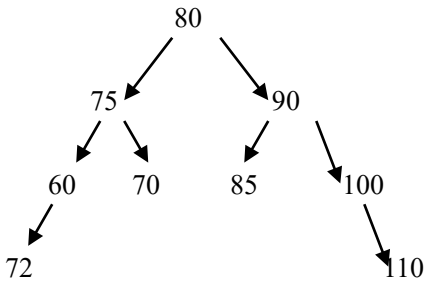
21. 设森林 F 中有三棵树，第一、第二和第三棵树的结点个数分别为 m1、m2 和 m3，则与森林 F 对应的二叉树根结点的右子树上的结点个数是（ d ）。

A. m1　　B. m1+m2　　C. m3　　D. m2+m3

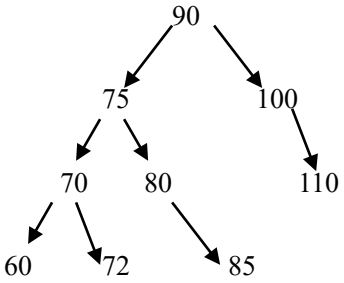
22. 根据插入次序（80，90，100，110，85，70，75，60，72）建立二叉排序树。
图（ a ）是最终变化的结果。
若仍以该插入次序建立平衡二叉树。图（ c ）是最终变化的结果。



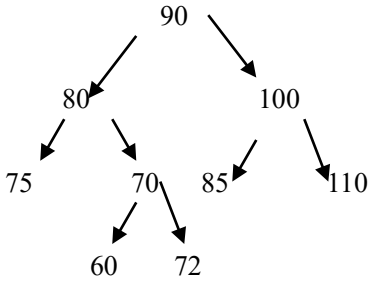
a:



b:

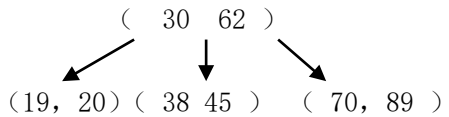


c:

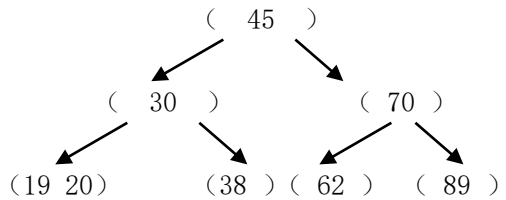


d:

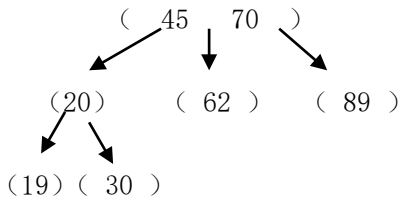
23. 设输入序列为 20, 45, 30, 89, 70, 38, 62，19 依次插入到一棵 2-3 树中(初始状态为空)，该 B-树为（ b ）。再删除 38，该 B-树为（ f ）。

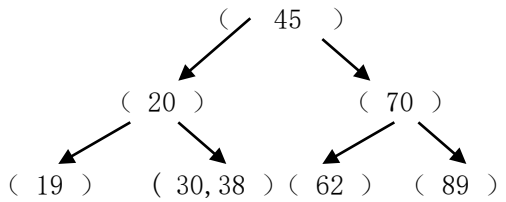


a:

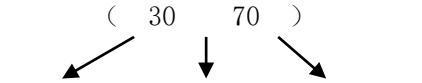


b:

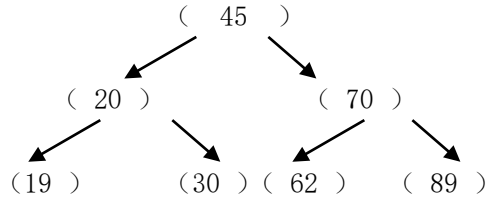




c:



d:



e:

f:

24. 已知一组待排序的记录关键字初始排列如下：45,34,87,25,67,43,11,66,27,78 。
(g)是快速排序法一趟排序的结果；
(a)是希尔排序法（初始步长为 4）一趟排序的结果；
(b)是初始堆（大堆顶）；
(d)是基数排序法一趟排序的结果。

A. 27, 34, 11, 25, 45, 43, 87, 66, 67, 78　　B. 87, 78, 45, 66, 67, 43, 11, 25, 27, 34
C. 11, 43, 34, 25, 45, 66, 27, 67, 87, 78　　D. 11, 43, 34, 45, 25, 66, 87, 67, 27, 78
E. 34, 45, 25, 67, 43, 11, 66, 27, 78, 87　　F. 87, 45, 11, 25, 34, 78, 27, 66, 67, 43
G. 27, 34, 11, 25, 43, 45, 67, 66, 87, 78　　H. 34, 11, 27, 25, 43, 78, 45, 67, 66, 87

25. 若有序表中关键字序列为：14，20，25，32，34，45，57，69，77，83，92。对其进行折半查找，则在等概率情况下，查找成功时的平均查找长度是(c)。查找 32 时需进行(c)次比较。

A. 1　　B. 2　　C. 3　　D. 4

26. 设一棵二叉树 BT 的存储结构如下:

	1	2	3	4	5	6	7	8
lchild	2	3	0	0	6	0	0	0
data	A	B	C	D	E	F	G	H
rchild	0	5	4	0	8	7	0	0

其中 lchild，rchild 分别为结点的左、右孩子指针域，data 为结点的数据域。则该二叉树的高度为(d)；
第 3 层有(a)个结点（根结点为第 1 层）。

A. 2　　B. 3　　C. 4　　D. 5

27. 一个连通图的最小生成树(b)。

A. 只有一棵　　B. 有一棵或多棵　　C. 一定有多棵　　D. 可能不存在

28. 若某二叉树有 20 个叶子结点，有 20 个结点仅有一个孩子，则该二叉树的总结点数是(c)。

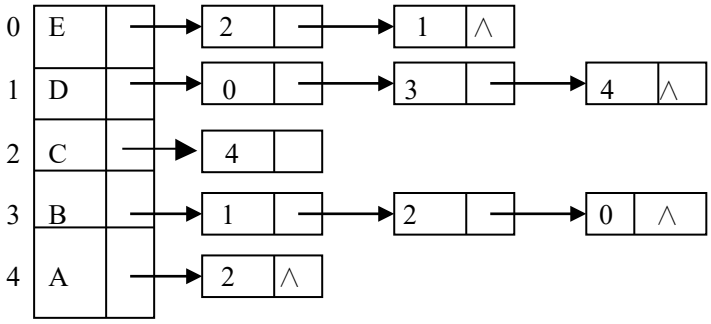
A. 40　　B. 55　　C. 59　　D. 61

29. 已知哈希表地址空间为 A[0..8]，哈希函数为 H(k)=k mod 7，采用线性探测再散列处理冲突。若依次将数据序列：76,45,88,21,94,77,17 存入该散列表中，则元素 17 存储的下标为(f)；在等概率情况下查找成功的平均查找长度为(c)。

A. 0　　B. 1　　C. 2　　D. 3

E. 4　　F. 5　　G. 6　　H. 7

30. 已知某有向图的邻接表存储结构如图所示。



根据存储结构，依教材中的算法其深度优先遍历次序为（ d ）。

广度优先遍历次序为（ c ）。各强连通分量的顶点集为（　　）。

a: abcde.　　b: edcba.　　c: ecdab.　　d: ecadb.

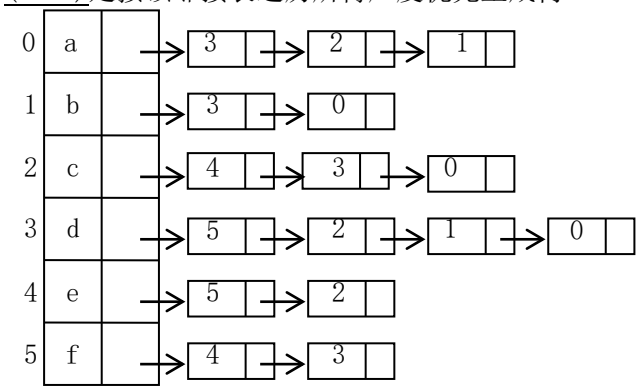
e: abc 及 ed　　f: ac 及 bed　　g: ab 及 ced　　h: bc 及 aed

31. 已知某无向图的邻接表如下所示；

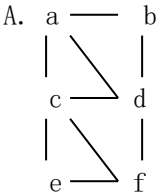
(　　)是其原图。

(　　)是按该邻接表遍历所得深度优先生成树。

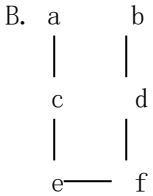
(　　)是按该邻接表遍历所得广度优先生成树。



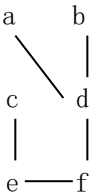
A.



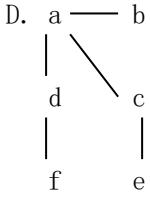
B.



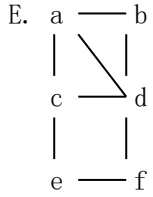
C.



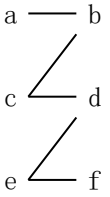
D.



E.



F.



32. 若顺序表中各结点的查找概率不等，则可用如下策略提高顺序查找的效率：若找到指定的结点，将该结点与其后继（若存在）结点交换位置，使得经常被查找的结点逐渐移至表尾。以下为据此策略编写的算法，请选择适当的内容，完成此功能。

顺序表的存储结构为：

```
typedef struct{
              ElemType *elem; //数据元素存储空间，0 号单元作监视哨
              int     length; //表长度
             }SSTable;
int search_seq(SSTable ST, KeyType key)
{ //在顺序表 ST 中顺序查找关键字等于 key 的数据元素。
  //若找到，则将该元素与其后继交换位置，并返回其在表中的位置，否则为 0。
  ST.elem[0].key=key;
  i=ST.length;
  while(ST.elem[i].key!=key)_____;
  if(_____)
    { ST.elem[i]←→ST.elem[i+1];
      _____;
    }
  return i;
}
```

A. i>0　　B. i>=0　　C. i<ST.length　　D. i<=ST.length

E. i++　　F. i--　　G. A 和 C 同时满足　　H. B 和 D 同时满足

33.下列函数为堆排序中的堆调整过程(调整 H.r[s]的关键字,使 H.r[s..m]成为一小顶堆)。请在“_____”处填上合适的内容,完成该算法。

```
Void heapadjust( heaptype & H , int s , int m ) {
    rc=H.r[s];
    for (j=2*s;j<=m;j*=2) {
      if (j<m && _____ ) ++j;
      if ( _____ ) break;
      H.r[s]=H.r[j]; s=j;
    }
    _____ ;
}//heapadjust
```

a: (H.r[j].key>H.r[j+1].key);

b: !(rc.key< H.r[j].key);

c: (H.r[j].key<H.r[j+1].key);

d: !(rc.key >H.r[j].key);

e: H.r[s]=rc;

f: rc=H.r[s];

g: h.r[j]=rc;

h: rc=H.r[j];

三．简答题

1. 设一棵二叉树的先序遍历序列为：A B D F C E G H ，中序遍历序列为： B F D A G E H C （10 分）
①、试画出这棵二叉树。 （8 分）
②、写出这棵二叉树的后序遍历序列。（2 分）
2. 对于图 1，用 Prim 算法从顶点 a 开始，求最小生成树。（10 分）
①、依次写出先后生成的各条边。（6 分）
②、画出求得的最小生成树。（2 分）
③、计算该最小生成树的代价。（2 分）
3. 设散列函数 hash(x)=x mod 11，散列表的地址空间为 0-10，现要把数据：1,13,12,34,38,33,27,22 插入到散列表中。（10 分）
①、使用线性探测再散列法构造散列表。（4 分）
②、使用链地址法构造散列表。（6 分）
4. 设待排序的关键字序列为：{83，40，63，13，84，35，96，57，39，79，61，15}。（10 分）
①写出简单选择排序对上述序列前 6 趟排序中各趟排序的结果（6 分）。
②写出应用 2-路归并排序对上述序列进行排序中各趟的结果（4 分）。

四．算法设计题

1. 单链表结点的类型定义如下：

```
typedef struct LNode {
   int data;
   struct LNode *next;
} LNode, *Linklist;
```

写一算法，Contrary(linklist &L) ，对一带头结点且仅设尾指针 L 的循环单链表就地逆置。（即首元变尾元，尾元变首元。）

```
void Contrary(linklist &L)
{ p=L->next; q=L;
  while(p!=L)
    { r=p->next; p->next=q; q=p; p=r; }
p->next=q;
}
```

2. 二叉树用二叉链表存储表示。

```
typedef struct BiTNode {
   TelemType   data;
   Struct BiTNode *lchild, *rchild;
} BiTNode, *BiTree;
```

试编写销毁二叉树 T 的算法 DestroyBiTree (BiTree &T)。

```
void DestroyBiTree ( BiTree &T)
{ if(!T) return;
  if(T-lchild) DestroyBiTree(T->lchild);
```

```
  if(T-rchild) DestroyBiTree(T->rchild);
  free(T); T=NULL;
 }
```

3．设带头结点的单链表中的元素以值非递减有序排列，试编写算法，删除表中所有值相同的多余元素。

单链表结点的类型定义如下：

```
typedef struct LNode{
    int data;
    Struct LNode    *next;
}LNode, *LinkList;
void Delete_same(LinkList L)
{ if(!L->next) return;
 p=L->next; q=p->next;
 while(q)
  if(p->data==q->data)
    { p->next=q->next; free(q); q=p->next;}
  else
    { p=q; q=q->next;}
}
```

4．设在一个带头结点的单链表中所有元素结点的数据值无序排列，设计一个算法，删除表中所有大于 min 且小于 max 的元素（若存在）。

```
typedef int DataType (0.5 分）
void rangeDelete(LinkList &L, DataType min, DataType max){ (1.5 分）
LinkNode *pr=L, *p=L->link; (1 分）
while(p!=NULL) (1 分）
    if(p->data>min && p->data<max) (1 分）
{pr->link=p->link; delete p;   p=pr->link;} (3 分）
else {pr=p; p=p->link;} (2 分）
```