掌心博阅电子书重点名校考研专业课精品资料系列

【基础】2025年福州大学083500软件工程《408计算机学科专业基础之数据结构》考研基础检测5套卷

主编：掌心博阅电子书

青岛掌心博阅电子书 出品

特别说明

本书严格按照该科目今年考研专业课真题题型、试题数量和考试难度出题，结合本专业考研大纲整理编写，由考研学长严格审核校对。其内容涵盖了本科目考研常考试题及重点试题，针对性强，是报考本校该科目考研专业课复习的重要资料。

版权声明

青岛华研教育旗下掌心博阅电子书依法对本书享有专有著作权，同时我们尊重知识产权，对本电子书部分内容参考和引用的市面上已出版或发行图书及来自互联网等资料的文字、图片、表格数据等资料，均要求注明作者和来源。但由于各种原因，如资料引用时未能联系上作者或者无法确认内容来源等，因而有部分未注明作者或来源，在此对原作者或权利人表示感谢。若使用过程中对本书有任何异议请直接联系我们，我们会在第一时间与您沟通处理。

因编撰此电子书属于首次，加之作者水平和时间所限，书中错漏之处在所难免，恳切希望广大考生读者批评指正。

目录

【基础】2025 年福州大学 083500 软件工程《408 计算机学科专业基础之数据结构》考研基础检测 5 套卷（一）.....4

【基础】2025 年福州大学 083500 软件工程《408 计算机学科专业基础之数据结构》考研基础检测 5 套卷（二）.....15

【基础】2025 年福州大学 083500 软件工程《408 计算机学科专业基础之数据结构》考研基础检测 5 套卷（三）.....27

【基础】2025 年福州大学 083500 软件工程《408 计算机学科专业基础之数据结构》考研基础检测 5 套卷（四）.....41

【基础】2025 年福州大学 083500 软件工程《408 计算机学科专业基础之数据结构》考研基础检测 5 套卷（五）.....51

【基础】2025 年福州大学 083500 软件工程《408 计算机学科专业基础之数据结构》考研基础检测 5 套卷（一）

说明：本书按照考试大纲、历年真题、指定参考书等公开信息潜心整理编写，由学长严格审核校对，仅供考研备考使用，与目标学校及研究生院官方无关，如有侵权请联系我们立即处理。

一、单项选择题

1．**若用冒泡排序法对序列（10,14,26,29,41,52）从大到小进行排序，需要进行_____比较。**

A.3
B.10
C.15
D.25

【答案】C

【解析】由排序规则直接求解，假设参与排序的数据元素共有 n 个，则冒泡排序需要进行的比较次数为 n(n+1)/2。

2．**一个图中包含 k 个连通分量，若按深度优先搜索方法访问所有顶点，则必须调用_____次深度优先遍历算法。**

A.k
B.1
C.k-1
D.k+1

【答案】A

【解析】从图中一个顶点出发进行深度优先遍历，会将与这个顶点连通的所有顶点都访问到，对于含有 k 个连通分量的图来说，要访问到图中所有的顶点，需要调用 k 次深度优先遍历算法。

3．**已知序列 25，13，10，12，9 是大根堆，在序列尾部插入新元素 18，将其再调整为大根堆，调整过程中元素之间进行的比较次数是_____。**

A.1
B.2
C.4
D.5

【答案】B

【解析】插入新元素 18 时，只须和它的所有祖先节点进行比较，它有两个祖先节点。本题答案为 B。

4．**_____不是算法所必须具备的特性。**

A.有穷性
B,确定性
C.高效性
D.可行性

【答案】C

【解析】算法的五个重要特征是：有穷性、确定性、可行性、输入和输出。

5．**数组SZ[－3...5，0...10]含有元素数目为_____。**

A.88
B.99

C.80
D.90
【答案】B
【解析】考查二维数组的概念。该数组共有 9 行、11 列，故共有元素 99 个。

6. **下面不正确的说法是_____。**

(1)求从指定源点到其余各顶点的迪杰斯特拉(Dijikstra)最短路径算法中弧上权不能为负的原因是在实际应用中无意义；

(2)利用Dijikstra求每一对不同顶点间的最短路径的算法时间是$O(n^3)$(图用邻接矩阵表示)；

(3)利用Floyd求每对不同顶点对的算法中允许弧上的权为负，但不能有权和为负的回路。

A.(1)，(2)，(3)
B.(1)
C.(1)，(3)
D.(2),(3)
【答案】A

7. **若 X 是后序线索二叉树中的叶结点，且 X 存在左兄弟结点 Y，则 X 的右线索指向的是_____。**

A.X 的父结点
B.以 Y 为根的子树的最左下结点
C.X 的左兄弟结点 Y
D.以 Y 为根的子树的最右下结点
【答案】A
【解析】根据后序线索二叉树的定义，X 结点为叶子结点且有左兄弟，那么这个结点为右孩子结点，利用后序遍历的方式可知 X 结点的后继是其父结点，即其右线索指向的是父结点。

8. **设广义表L=((a,b,c))，则 L 的长度和深度分别为_____。**

A.1和1
B.1和3
C.1和2
D.2和3
【答案】C

9. **具有 n 个权值的哈夫曼树，其结点数为_____。**

A.n
B.2n+1
C.$2(n+1)$
D.$2n-1$
【答案】D

二、填空题

10. **若广义表中的每个元素都是_____，则广义表变成为线性表。**

【答案】原子

11．**数据结构中评价算法的两个重要指标是:_____。**

【答案】算法的时间复杂度、算法的附加空间复杂度

12．**已知广义表**LS=(a,(b,c,d),e)**，运用 head 和 tail 函数取出 LS 中的原子 b 的运算是_____。**

【答案】head(head(tail(LS)))

【解析】广义表的基本运算。

tail(LS)=((b,c,d),e)
head(tail(LS))=(b,c,d)
head(head(tail(LS)))=b

13．**归并排序有两个基本阶段，第一阶段是_____，第二阶段是_____。**

【答案】生成初始归并段、对这些初始归并段采用某种归并方法，进行多遍归并

14．**阅读下述程序，指出其输出结果_____。**

```
void g(int**);
main( )
{
   int line[100],i;
   int *p=line;
   for(i=0;i<100;i++)
{
      *p=i;
      g(&p);
   }
   for(i=0;i<100;i++) printf('%d\n',line[i]);
}
void g(int**p)
{
  (**p)++;
  (*p)++;
}
```

【答案】程序输出 1，2，3，...，100。

15．**用邻接矩阵存储一个不带权有向图 G，其第 i 列的所有元素之和等于顶点 i 的_____。**

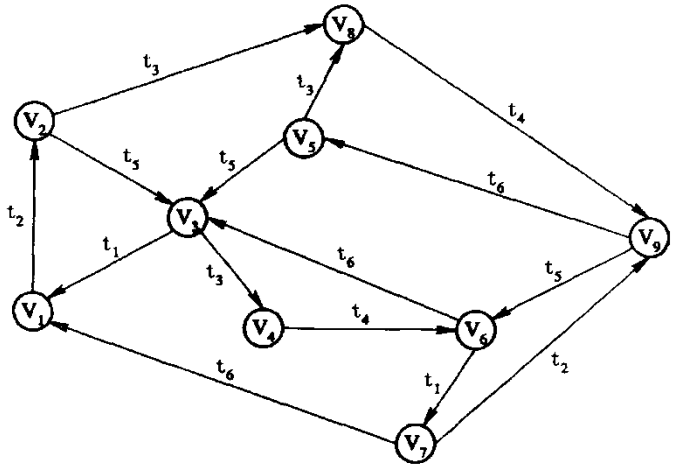
【答案】入度

三、应用题

16．**已知在某并发处理系统的 Petri 网基础上建立的可达图如下图所示。图中每个顶点表示系统运行中的一种状态，有向边(弧)表示事件(用 t 表示)，例如有向边(V_i，V_j)表示系统在状态 V_i时出现该事件将引发系统由状态 V_i到状态 V_j。**

(1)请分别给出该可达图的邻接表、邻接矩阵以及邻接矩阵的三元组形式，要求每种存储结构能够表达出该可达图的全部信息，并分别对这三种形式中的每个部分的含义(物理意义)予以简要说明。

(2)若假设每个域(包括指针域)的长度为一个字节，请分别计算出这三种结构所占用的空间大小。



图

【答案】(1)①邻接表表示方法。

表结点

adjvex	info	nextarc

头结点

vexdata	firstarc

Adjvex：指示与该结点邻接的结点在图中的位置；

nextarc:指示下一条弧的结点；

info:存储与弧有关的信息，如权值；

vexdata:存储顶点的名；

firstarc:指向表中第一个结点。

该图的邻接表表示如下图 1 所示。

②邻接矩阵的表示方法如下图 2 所示。

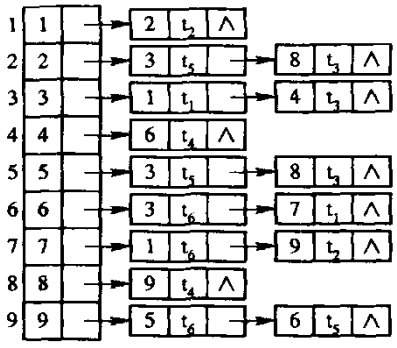


图 1

$$\begin{bmatrix} \infty & t_2 & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & t_5 & \infty & \infty & \infty & \infty & t_3 & \infty \\ t_1 & \infty & \infty & t_3 & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & t_4 & \infty & \infty & \infty \\ \infty & \infty & t_5 & \infty & \infty & \infty & \infty & t_3 & \infty \\ \infty & \infty & t_6 & \infty & \infty & \infty & t_1 & \infty & \infty \\ t_6 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & t_2 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & t_4 \\ \infty & \infty & \infty & \infty & t_6 & t_5 & \infty & \infty & \infty \end{bmatrix}$$

图 2

③邻接矩阵的三元组表示方法如下图 3 所示。

(2)邻接表表示方法需要的空间： 15×3+9×2=63 B 。

邻接矩阵表示方法需要的空间： 9×9=81 B。

三元组表示方法需要的空间： 3×(15+1)=48 B。

(9，9，15)
(1，2，t_2)
(2，3，t_5)
(2，8，t_3)
(3，1，t_1)
(3，4，t_3)
(4，6，t_4)
(5，3，t_5)
(5，8，t_3)
(6，3，t_6)
(6，7，t_1)
(7，1，t_6)
(7，8，t_2)
(8，9，t_4)
(9，5，t_6)
(9，6，t_5)

图 3

17．**设某文件中待排序记录的排序码为**72, 73, 71, 23, 94, 16, 05, 68，

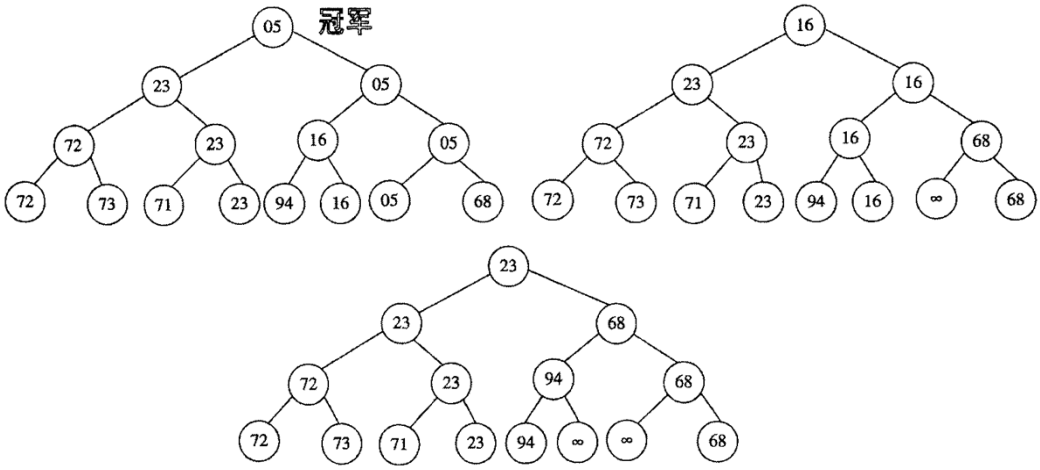
(1)试画图表示出树形选择排序(增序)过程的前三步。

(2)试说明树形选择排序的基本思想。

(3)树形选择排序与直接选择排序相比较，优缺点是什么?

(4)堆排序是如何改进树形排序方法的? 优点是什么?

【答案】(1)如下图所示：



图

(2)树形选择排序的基本思想:首先对 n 个待排序记录的关键字进行两两比较，从中选出$\lceil n/2 \rceil$个较小者再两两比较，直到选出关键字最小的记录为止，此为一趟排序。我们将一趟选出的关键字最小的记录称为“冠军”，而“亚军”是从与“冠军”比较失败的记录中找出，具体做法为:输出“冠军”后，将其叶子结点关键字改为“最大值”，然后从该叶子结点开始，和其左(或右)兄弟比较，修改从叶子结点到根结点路

径上各结点的关键字，则根结点的关键字即为次小关键字。如此下去，可依次选出从小到大的全部关键字。其时间复杂度是$O(n\log n)$，空间复杂度是$O(n)$，由于使用空间较多，以及一些多余的比较，更主要由于堆排序的出现，使得很少再用树形排序。

(3)树形选择排序与直接选择排序相比较,其优点是从求第 2 个元素开始,从$n-i+1$个元素中不必进行$n-i$次比较，比较次数远小于直接选择排序；其缺点是辅助储存空间大。

(4)堆排序基本思想是:堆是 n 个元素的序列，先建堆，即先选得一个关键字最大或最小的记录，然后与序列中最后一个记录交换，之后将序列中前$n-1$记录重新调整为堆(调堆的过程称为“筛选”)，再将堆顶记录和当前堆序列的最后一个记录交换，如此反复直至排序结束。优点是在时间性能与树形选择排序属同一量级的同时，堆排序只需要一个记录大小供交换用的辅助空间，调堆时子女只和双亲比较。避免了过多的辅助存储空间及和最大值的比较。

18．**设有 12 个数据{25,40,33,47,12,66,72,87,94,22,5,58}，它们存储在散列表中，利用双散列解决冲突，要求为插入新数据的平均查找次数不超过 3 次。**

(1)该散列表的大小 m 应设计多大?

(2)试为该散列表设计相应的散列函数(用除留余数法)和计算寻找下一个“空位”时向前跨步步长的再散列函数;

(3)顺次将各个数据散列到表中;

(4)计算查找成功的平均查找次数。

【答案】(1)线性探测再散列的哈希表查找成功的平均查找长度为$(1+1/(1-\alpha))/2$，按照题目要求它的值要小于等于 3,有$(1+1/(1-\alpha))/2\leqslant 3$，可计算出$\alpha\leqslant 0.8$，设表的长度为 m,即$12/m\leqslant 0.8$，所以$m\geqslant 15$。

(2)取 m = 16,散列函数$H(k)=k \bmod 16$，再散列函数为$H(k)+3$。

(3)顺次将各个数据散列到表中如下所示:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	33	66			5	22	87	40	25	58	72	12		94	47

(4)查找成功的平均查找次数为

$$(1+1+1+1+1+1+1+1+1+1+2+1)/12=1.08$$

19．**设字符串** S='aabaabaabaac'，P='aabaac'。

(1)给出 S 和 P 的 next 值和 nextval 值;

(2)若 S 作主串，P 作模式串，试给出利用 BF 算法和 KMP 算法的匹配过程。

【答案】(1)S 的 next 与 nextval 值分别为 012123456789 和 002002002009。

P 的 next 与 nextval 值分别为 012123 和 002003。

(2) 利用 BF 算法的匹配过程：

第一趟匹配：aabaabaabaac
aabaac(i=6, j=6)

第二趟匹配：aabaabaabaac
aa(i=3, j=2)

第三趟匹配：aabaabaabaac
a(i=3, j=1)

第四趟匹配：aabaabaabaac
aabaac(i=9, j=6)

第五趟匹配：aabaabaabaac
aa(i=6, j=2)

利用 KMP 算法的匹配过程：

第一趟匹配：aabaabaabaac
aabaac(i=6, j=6)

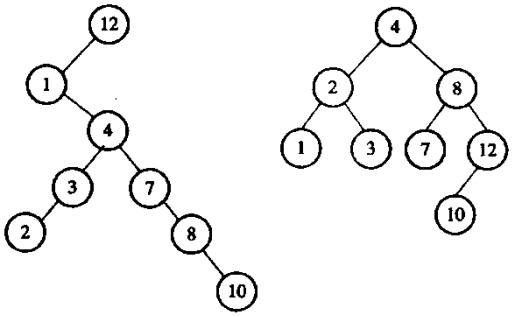
第二趟匹配：aabaabaabaac
(aa)baac

第三趟匹配：aabaabaabaac
（成功） (aa)baac

第六趟匹配：aabaabaabaac
a(i=6, j=1)
第七趟匹配：aabaabaabaac
（成功） aabaac(i=13, j=7)

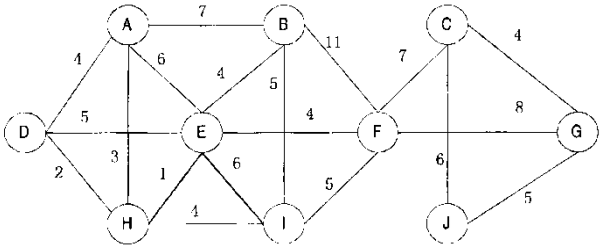
20．**试求按关键字序列(12，1，4，3，7，8，10，2)插入生成的二叉排序树和平衡二叉树。**

【答案】二叉排序树和平衡二叉树如下图所示。



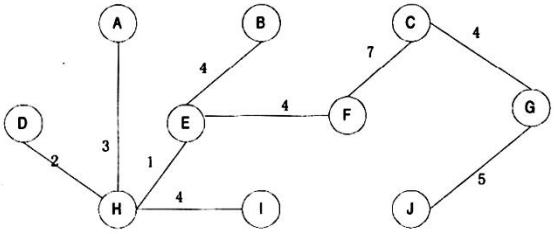
图

21．**用深度优先搜索遍历如图中所示的无向图，试给出以 A 为起点的顶点访问序列(同一个顶点的多个邻点，按字母顺序访问)，并给出一棵最小生成树。**



【答案】顶点访问序列：ABEDHIFCGJ

一棵最小生成树如下图所示。



最小生成树

四、算法设计题

22．**将十进制整数 n 转换成二进制数。以集合类型的变量表示一个 16 位二进制数。**

【答案】算法描述如下：

```
PROCEDURE Convers(VAR n:integer);
VAR A:ARRAY[1..16] OF 0..1;
      i,j:integer;
BEGIN
   FOR i:=1 TO 16 DO
      A[i]:=0;
   j:=j+1;
   REPEAT
      n:=n div 2;
```

```
        A[j]:=n;
        j:=j+1
    UNTIL n=0;
    FOR j:=19 DOWNTO 1 DO
        write(A[j]);
END;
```

23．**如果字符串的一个子串(其长度大于 1)的各个字符均相同，则称之为等值子串。试设计一个算法:输入字符串 S，以“!”为结束标志，如果串 S 中不存在等值子串，则输出信息:“无等值子串”，否则求出(输出)一个长度最大的等值子串。**

例如，若S＝"abc123abc123!"，则输出:“无等值子串”；

又如，若S＝"abcaabcccddddaaadd 123abc123!"，则输出等值子串:“dddd”。

【答案】

```
void EquivalentSubstring(char S[n]) {
for (k = 0;;k++)  {
    scanf("%c",&S[k]);
  if(S[k] == '!')      break;
  }//读入字符串 S
for( i = 0,j = 1,head = 0,max = 1;S[i] != '!' && S[j] != '!';
  i = j, j++)  {
    count = 1;
  while(S[i] == S[j])    {
      j++;
        count++;
      } //统计当前等值子串的长度
  if(count > max)       {
          head = i;
      max = count;   //发现更长等值子串，更新 head 和 max
        }
    }
if(max > 1)  {
    for(k = head; k < (head + max); k++)
       printf("%c",S[k]);
    }
else
   printf("There is no equivalent substring in S");
  }
```

24．**设计一个算法，将不带权图的邻接表表示转换为邻接矩阵表示。**

【答案】先将邻接矩阵所有元素置初值零。在邻接表上查找邻接节点，找到后修改相应邻接矩阵元素的值。对应算法如下。

```
void ListToMat(AGraph *G,MGraph &g)      //将邻接表 G 转换成邻接矩阵 g
{   int i,j;
    g.n=G->n;g.e=G->e;
    ArcNode *p;
    for (i=0;i<g.n;i++)                   //g.edges[i][j]赋初值 0
        for (j=0;j<g.n;j++)
            g.edges[i][j]=0;
    for (i=0;i<g.n;i++)                   //遍历每一个顶点
    {   p=G->adjlist[i].firstarc;
        while (p!=NULL)                   //遍历每个邻接点
        {   g.edges[i][p->adjvex]=1;
            p=p->nextarc;                 //找下一个邻接点
        }
```

```
    }
}
```

25．**假设有如下的关于职工信息的结构体：**

```
struct Employee{
  long eID;
  char eName[10];
  struct Employee * pNext;
};
```

其中，eID 表示职工编号，请编写一个函数：

```
struct Employee * eIDSequence(struct Employee * pHead)
```

其中，pHead 为关于 Employee 的单链表的头指针，该链表中已存放有若干名职工的信息。该函数的功能是：对链表中的职工信息基于职工编号进行升序排序而得到新的链表的头指针。

【答案】利用插入排序解决排序问题。

```
Employee * eIDSequence(Employee * pHead)
{
    Employee * p, * q, * r;
    int i,j;
    q = pHead->pNext;
    j = 1;
    while(q->pNext){
        p = pHead;
        for(i = 0;i<j;i++)
        {
            if(p->pNext->eID>q->pNext->eID)
            {
                r = q->pNext;
                q->pNext = r->pNext;
                r->pNext = p->pNext;
                p->pNext = r;
                i = j;//跳出 for 循环
            }
            p = p->pNext;
        }
        q = q->pNext;
        j++;
    }
    return pHead;
}
```

26．**设计一个算法，计算一个顺序串 s 中每个字符出现的次数。**

【答案】

```
typedef struct
{   char ch;                          //存放字符
    int count;                        //存放字符出现的次数
} Cch;
```

用 i 扫描串 s，判断s.data[i]是否在C[]数组(其元素类型为 Cch)中，若在其中，只需将相应的 count 域增 1 即可，若不在其中，则将该字符添加到 C[]中，并将相应的 count 域置 1。对应算法如下。

```
int Count(SqString s,Cch C[])              //返回不同字符个数
    {   int i,j,n=0;
        for (i=0;i<s.length;i++)
        {   for (j=0;j<=n;j++)                  //在 C[]中查找等于 s.data[i]的字符
                if (C[j].ch==s.data[i])         //若找到了，累计值增 1
                {   C[j].count++;
                    break;
```

```
            }
        if (j>n)                                    //若未找到，在C中增加一个元素
        {   C[n].ch=s.data[i];
            C[n].count=1;
            n++;
        }
    }
    return n;
}
```

27．**假设一个算术表达式中包含圆括号、方括号和花括号三种类型的括号，编写一个算法，判别表达式中括号是否正确配对。**

【答案】算法如下：

```
Status AllBrackets_Test(char *str)//判别表达式中三种括号是否匹配
{
  InitStack(s);
  for(p=str; *p; p++)
  {
    if(*p=='('||*p=='['||*p=='{') push(s,*p);
    else if(*p==')'||*p==']'||*p=='}')
    {
      if(StackEmpty(s)) return ERROR;
      pop(s,c);
      if(*p==')'&&c!='(') return ERROR;
      if(*p==']'&&c!='[') return ERROR;
      if(*p=='}'&&c!='{') return ERROR; //必须与当前栈顶括号匹配
    }
  }//for
  if(!StackEmpty(s)) return ERROR;
  return OK;
}//AllBrackets_Test
```

28．**在已知链表中按内容查找某元素，若找到，则删除之；否则输出失败信息。**

【答案】算法描述如下：

```
PROCEDURE Delx(VAR h:linklist; x:elementtype);
BEGIN
  IF h<>NIL THEN
    BEGIN
      p:=h;
      WHILE p^.next<>h DO
        IF p^.next^.data=x THEN p^.next:=p^.next^.next   {删除}
                           ELSE p:=p^.next               {后移}
    END
  IF p^.next=NIL THEN write('无删除的元素，失败')
END;
```

29．**假设散列函数为H(key)，编写用链地址法解决冲突的散列表的插入和删除算法。**

【答案】链地址法把每一个相同散列值的元素串成一个链表，插入和删除算法即为对链表的操作。算法描述如下：

```
typedef struct ListNode{
    int data;
    struct ListNode *next;
}ListNode;
```

```
List Node *HL[k];
void insert(int key){
    List Node *p,*q;
    int i;
    p = (ListNode *)malloc(sizeof (ListNode));
    p->data = key;
    i = H(key);
    if (!HL[i]) {                    //没有冲突
        HL[i] = p;
        p->next = null;
    }
    else{                            //有冲突，按数值大小顺序插入
        q = HL[i];
        while (q->next&&q->next->data<key) q = q->next;
        p->next = q->next;
        q->next = p;
    }
}

void delete (int key){
    int i;
    List Node *q,*p;
    i = H[key];
    if(HL[i]->data==key){            //找到了关键字等于 key 的元素，删除之
        HL[i] = HL[i]->next;
        free(HL[i]);
     }
     else{                           //有冲突，继续找
       q = HL[i];
       while(q->next&&q->next->data<key) q = q->next;
       if(q->next->data==key){
          p = q;
          q = q->next;
          p->next = q->next;
          free(q);
 }
 }
```

【基础】2025 年福州大学 083500 软件工程《408 计算机学科专业基础之数据结构》考研基础检测 5 套卷（二）

说明：本书按照考试大纲、历年真题、指定参考书等公开信息潜心整理编写，由学长严格审核校对，仅供考研备考使用，与目标学校及研究生院官方无关，如有侵权请联系我们立即处理。

一、单项选择题

1．**对有 3600 个记录的索引顺序表进行查找，最理想的块长是_____。**

A.1800
B.60
C.1200
D.1000

【答案】B

【解析】索引顺序表的查找方式为先按顺序来查找块，然后按顺序在块中查找，设块长为 k，则平均查找长度为$(n/k+1)/2+(k+1)/2$，对其求导，得到最佳的 k 值为$\sqrt{n}$，题目中 n=3600,所以，最佳的 k 为 60。

2．**一个稀疏矩阵采用压缩后，和直接采用二维数组存储相比会失去_____特性。**

A.顺序存储
B.随机存取
C.输入输出
D.以上都不对

【答案】B

【解析】当稀疏矩阵采用二维数组存储时，它具有随机存取特性，而采用三元组或稀疏矩阵存储时，尽管占用的空间会减少，但不再具有随机存取特性。

3．**串是任意有限个_____。**

A.符号构成的集合
B.符号构成的序列
C.字符构成的集合
D.字符构成的序列

【答案】D

【解析】串是任意字符的有限序列。

4．**具有 n 个顶点的有向图最多有_____条边。**

A.n
B.$n(n-1)$
C.$n(n+1)$
D.n^2

【答案】B

5．**以下不属于存储结构是_____。**

A.栈
B.线索树
C.哈希表

D.双链表

【答案】A

【解析】栈属于一种逻辑结构，通常有顺序栈和链栈两种存储结构。

6. **哈希表长度为 m，哈希函数$H(K)=K\%P$，一般来说 P 应该取小于 m 的最大_____。**

A.奇数

B.偶数

C.素数

D.合数

【答案】C

7. **在带头结点的链队列中，队头指针指向链表的_____**

A.最后一个元素结点

B.第一个元素结点

C.头结点

D.都不是

【答案】C

8. **线性表的链式存储结构和顺序存储结构相比，优点是_____。**

A.所有的操作算法实现简单

B.便于随机存取

C.便于插入和删除元素

D.节省存储空间

【答案】C

【解析】顺序存储结构在插入和删除元素时需大量移动元素，而链式存储结构则不需要这样。

9. **(1)静态链表既有顺序存储的优点，又有动态链表的优点。所以，它存取表中第 i 个元素的时间与 i 无关**

(2)静态链表中能容纳的元素个数的最大数在定义时就确定了，以后不能增加；

(3)静态链表与动态链表在元素的插入、删除上类似，不需作元素的移动；

以上错误的是_____。

A.(1)、(2)

B.(1)

C.(1)、(2)、(3)

D.(2)

【答案】B

二、填空题

10. **一个算法具有 5 个特性：_____、_____、_____、输入和输出。**

【答案】可行性、有穷性、确定性

11．**折半查找要求数据元素_____，存储方式采用_____。**

【答案】有序、顺序存储

12．**一个无序序列可以通过构造一棵_____树而变成一个有序序列，构造树的过程即为对无序序列进行排序的过程。**

【答案】二叉排序树

13．**在一个单链表中 p 所指节点之后插入*s节点时，应执行** s->next = _____**和** p->next = _____ **的操作。**

【答案】p->next、s

14．**树在计算机内的表示方式有_____、_____、_____。**

【答案】双亲表示法、孩子表示法、孩子兄弟表示法

15．**在一个无向图的的邻接表中，若表结点的个数是 m，则图中边的条数是_____条。**

【答案】m/2

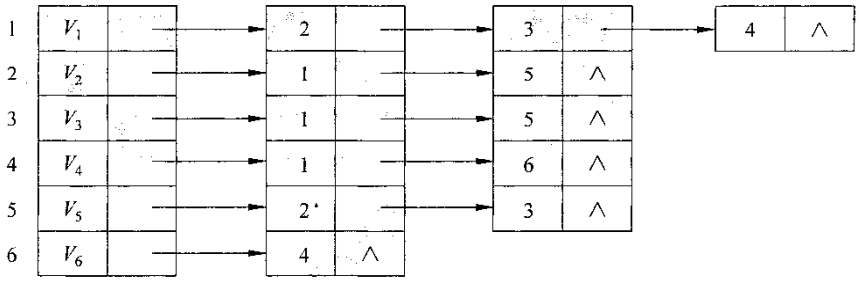
三、应用题

16．**已知某图的邻接表如图所示。**

(1)写出此邻接表对应的邻接矩阵。

(2)写出由V_1开始的深度优先遍历序列。

(3)写出由V_1开始的广度优先遍历序列。



图

【答案】(1)从图的邻接表中，可以得出该图中有 6 个顶点，对应的邻接矩阵为

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

(2)由V_1开始的深度优先遍历序列为$V_1V_2V_5V_3V_4V_6$。

(3)由V_1开始的广度优先遍历序列为$V_1V_2V_3V_4\ V_5V_6$。

17．**推导并求解 n 阶 Hanoi 塔问题至少执行 move 操作次数。**

【答案】设 Hanoi(n)表示 n 阶 Hanoi 需要移动的次数，当 n=l 时，问题较简单，只要移动一次即可。当**n>l**时，需要利用中间的辅助塔，将压在编号 n 上的圆盘从第一个塔移动到第二个塔上，移动的次数为

Hanoi(n-1)；然后将编号为 n 的圆盘移动到第三个塔上，需要移动一次即可；最后将第二个塔上的 n-1 个圆盘移动到第三个塔上，需要移动的次数也为 Hanoi (n-1)。可得

$$\text{Hanoi}(n)=\begin{cases}1 & n=1\\ 2\text{Hanoi}(n-1)+1 & n>1\end{cases}$$

解得

$$\text{Hanoi}(n)=2^n-1$$

18．**判别下面的每个结点序列是否表示一个堆，如果是堆，请把它调整为堆。**

(1)100，90，80，60，85，75，20，25，10，70，65，50

(2)100，70，50，20，90，75，60，25，10，85，65，80

【答案】(1)画出此序列对应的完全二叉树，如下图 1 所示。

由图可以知道每个顶点都大于它的儿子结点。所以，这是一个大顶堆。

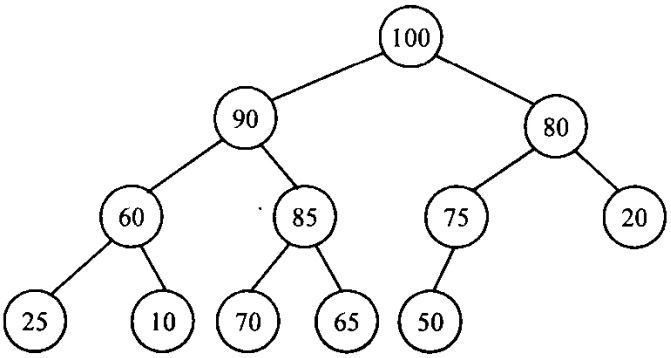


图 1

(2)画出此序列对应的完全二叉树，如图 2 所示。

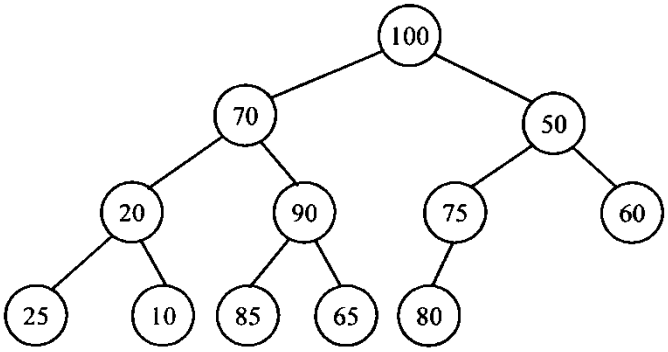


图 2

由图 2 知道，此序列既不是大顶堆也不是小顶堆。

调整后的堆的二叉树表示如图 3 所示。

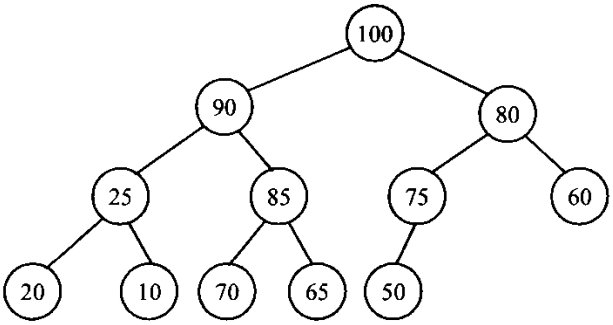


图 3

对应的序列是：(100，90，80，25，85，75，60，20，10，70，65，50)

19．**在单链表和双向链表中，能否从当前结点出发访问到任何一个结点?**

【答案】在单链表中不能从当前结点(若当前结点不是第一结点)出发访问到任何一个结点。因为链表运算只能从头指针开始，访问到链表中每个结点。在双链表中从当前结点反向可以到第一结点，正向可以到最后结点，因而从当前结点出发可以访问到链表中任何一个结点。

20．**二叉树的带权路径长度(WPL)是二叉树中所有叶结点的带权路径长度之和。给定一棵二叉树 T,采用二叉链表存储,结点结构为**

left	weight	right

其中叶结点的 weight 域保存该结点的非负权值。设 root 为指向 T 的根结点的指针,请设计求 T 的 WPL 的算法,要求:

(1)给出算法的基本设计思想。

(2)使用 C 或C ++语言,给出二叉树结点的数据类型定义。

(3)根据设计思想，采用 C 或C ++语言描述算法,关键之处给出注释。

【答案】二叉树的带权路径长度为每个叶子结点的深度与权值之积的总和，可以使用先序遍历或层次遍历解决问题。

(1)算法的基本设计思想。

①基于先序递归遍历的算法思想是用一个 static 变量记录 wpl,把每个结点的深度作为递归函数的一个参数传递,算法步骤如下:

若该结点是叶子结点,那么变量 wpl 加上该结点的深度与权值之积。

若该结点是非叶子结点,那么若左子树不为空,对左子树调用递归算法,若右子树不为空,对右子树调用递归算法,深度参数均为本结点的深度参数加 1。

最后返回计算出的 wpl 即可。

②基于层次遍历的算法思想是使用队列进行层次遍历,并记录当前的层数。

当遍历到叶子结点时,累计 wpl。

当遍历到非叶子结点时把该结点的子树加入队列。

当某结点为该层的最后-一个结点时,层数自增 1。

队列空时遍历结束,返回 wpl。

(2)二叉树结点的数据类型定义如下:

```
typedef struct BiTNode{
  int weight;
  struct BiTNode * lchild,* rchild;
}BiTNode,* BiTree;
```

(3)算法代码如下:

①基于先序遍历的算法。

```
int WPL(BiTree root){
  return wpl_PreOrder(root, 0);
}
```

```
int wpl_PreOrder(BiTree root, int deep){
  static int wpl=0;                          //定义一个 static 变量存储 wpl
  if(root->lchild ==NULL && root->lchild ==NULL)
                                             //若为叶子结点,累计 wpl
wpl +=deep * root->weight;
if(root->lchild !=NULL)                      //若左子树不空,则对左子树递归遍历
wpl_PreOrder(root->lchild, deep+1);
if(root->rchild !=NULL)                      //若右子树不空,则对右子树递归遍历
wpl_PreOrder(root->rchild, deep+1);
return wpl;
}
```

②基于层次遍历的算法

```
#define MaxSize 100                          //设置队列的最大容量
int wpl_LevelOrder(BiTree root){
  BiTree q[MaxSize];                         //声明队列,end1 为头指针,end2 为尾指针
  int end1, end2;                            //队列最多容纳 MaxSize-1 个元素
  end1=end2=0;                             //头指针指向队头元素,尾指针指向队尾的后一个元素
  int wpl=0, deep=0;                         //初始化 wpl 和深度
  BiTree lastNode;                           //lastNode 用来记录当前层的最后一个结点
  BiTree newlastNode;                        //newlastNode 用来记录下一层的最后一个结点
   lastNode=root;                                  //lastNode 初始化为根结点
   newlastNode=NULL;                               //newlastNode 初始化为空
   q[end2++]=root;                                 //根结点入队
   while(end1 !=end2){                             //层次遍历,若队列不空则循环
     BiTree t=q[end1++];                           //拿出队列中的头一个元素
     if(t->lchild ==NULL && t->rchild ==NULL){
       wpl +=deep * t->weight; }                   //若为叶子结点,统计 wpl
     if(t->lchild !=NULL){                         //若非叶子结点,则把左结点入队
       q[end2++]=t->lchild;
     newlastNode=t->lchild; }                  //并设下一层的最后一个结点为该结点的左结点
   if(t->rchild !=NULL){                         //处理叶结点
     q[end2++]=t->rchild;
     newlastNode=t->rchild; }
   if(t ==lastNode){                           //若该结点为本层最后一个结点,则更新 lastNode
     lastNode=newlastNode;
     deep +=1;                                 //层数加 1
       }
     }
     return wpl;                                     //返回 wpl
   }
```

21. **假设二叉树采用二叉链存储结构存储，设计一个算法求其中节点值为 x 的节点个数。**

【答案】设f(b,x)返回二叉树 b 中所有节点值为 x 的节点个数，其递归模型f(b,x)如下。

f(b,x) = 0　　b=NULL
f(b,x) = 1+ f(b->lchild,x)+f(b->rchild,x);　　当 b->data=x
f(b,x) = f(b->lchild,x)+f(b->rchild,x);　　其他情况

对应的递归算法如下。

```
int FindCount(BTNode *b,ElemType x)
{   if (b==NULL)
```

```
        return 0;
    else if (b->data==x)
        return(1+FindCount(b->lchild,x)+FindCount(b->rchild,x));
    else
        return(FindCount(b->lchild,x)+FindCount(b->rchild,x));
}
```

四、算法设计题

22．二叉排序树采用二叉链表存储。写一个算法，删除结点值是 X 的结点。要求删除该结点后，此树仍然是一棵二叉排序树，并且高度没有增长。(注:可不考虑被删除的结点是根的情况)

【答案】算法如下:

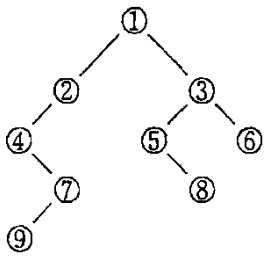
```
int DeleteBtreeNode(btree * root, keytype x)
{
  btree * t = root;
  while (t! = NULL)
  {
    if (t - > key > x) t = t - > lchild;
    else if (t - > key < x) t = t - > rchild;
    else {Delete(t); return 1;}
  }
}
void Delete(btree * p)
{
  if (!p - > rchild)
  {
  //右子树为空则只要重接它的左子树
  q = p;p = p - > lchild;free(q);
}
else if (!p - > lchild)
{
  //左子树为空则只要重接它的右子树
  q = p;p = p - > rchild;free(q);
}
else
{
  //左右子树均不空,找 p 的中序前驱 pre,用 pre 代替 p
  q = p;pre = p - > lchild;
  while (pre - > rchild)
  {
    //p 的中序前驱是它的左孩子结点的最右端结点
    q = pre;pre = pre - > rchild;
  }
 p - > data = pre - > data;
 if (q! = p) q - > rchild = pre - > lchild;
 else q - > lchild = pre - > lchild;
}
}
```

23．**下面给出的是一个在二叉树中查找值为 x 的结点，并打印该结点所有祖先结点的算法。在此算法中，假设值为 x 的结点不多于一个。此算法排序的非递归遍历形式。因退栈时需要区分其左、右子树是否已经遍历，故在结点进栈时附带有一个标志，进入左子树标志=0，进入右子树标志=1，进入右子树时栈 ST 保存结点指针 ptr 以及标志 tag，top 是指向栈顶的指针。**

```
void print (BinTreeNode * t; Type &x)
{
  stack ST; int i, top;
  top = 0; //置空栈
  while (t! = NULL&&t - > data! = x || top! = 0)
  {
    //寻找值为 x 的结点
    while(t! = NULL&&t - > data! = x)
    {
      ①_____;
      ST[top].ptr = t; //进栈
      ST[top].tag = 0;
      ②_____;
    }
    if (t! = NULL&&t - > data = = x) //找到值为 x 的结点
    for (i = 1;③_____;i ++ )
      cout < < ST[i].ptr - > data < < endl;
    else
    {
      while (top > 0&&ST[top].tag = = 1) //未找到值为 x 的结点
      top - - ;
      if (top > 0)
      {
        ST[top].tag = 1; //转向右子树
        ④_____;
      }
    }
  }
}
```

(1)请将空缺的语句补上:

(2)请给出对于下图所示的二叉树,使用上述算法搜索值为9的结点和值为10的结点的结果,以及在栈ST中的变化(top是栈顶指针)。



图

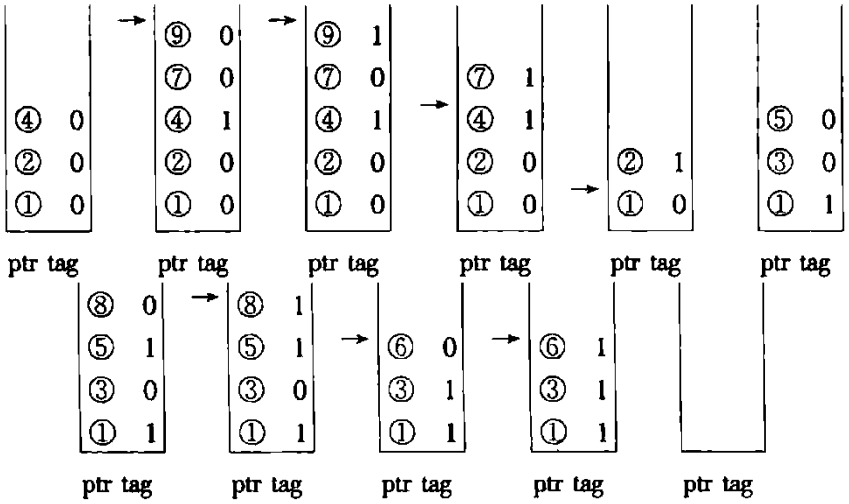
【答案】(1)① top + +

② t = t - > leftChild

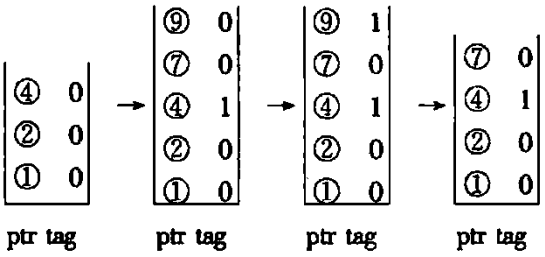
③ i < = top

④ t = ST[top].ptr - > rightChild

(2)搜索值为10的结点:



搜索值为9的结点：



24．**有两个递增有序表采用顺序表存储，设计一个算法将它们归并成一个有序表(假设每个表中没有重复关键字的元素，归并时重复关键字的元素只归并一次)。分析算法的时间复杂度和空间复杂度。**

【答案】将含 n 个元素的有序表 R1、含 m 个元素的有序表 R2 归并成含 k 个元素的有序表 R3 的算法如下。

```
void MergeSort1(SqList R1[],int n,SqList R2[],int m,SqList R3[],int &k)
{   int i=0,j=0;                    //i 扫描 R1,j 扫描 R2
    k=0;                            //k 保存归并到 R3 中的元素个数
    while (i<n && j<m)
    {   if (R1[i].key<R2[j].key)
            R3[k++]=R1[i++];
        else if (R1[i].key>R2[j].key)
            R3[k++]=R2[j++];
        else
        {   R3[k++]=R1[i++];
            j++;
        }
    }
    while (i<n)                     //R1 未归并完时循环
        R3[k++]=R1[i++];
    while (j<m)                     //R2 未归并完时循环
        R3[k++]=R2[j++];
}
```

本算法的时间复杂度为$O(m+n)$，空间复杂度为$O(1)$。

25．**自由树(即无环连通图)T=(V，E)的直径是树中所有点对间最短路径长度的最大值，即 T 的直径定义为 MAX d(u，v)，这里 d(u，v)表示顶点 u 到顶点 v 的最短路径长度(路径长度为路径中所含的边数)。试写一算法来求 T 的直径，并分析算法的时间复杂度(时间复杂度越小，得分越高)**

【答案】 求树的直径的时间复杂度可为$O(n)$，$O(n^2)$和$O(n^3)$。

解法一：先调用求所有点对间最短路径算法(每边权值为 1)，如 Floyd 算法，然后对指代矩阵求最大的 COST[i,j]作为直径。

解法二：修改 BFS 算法，使之遍历时记录当前访问结点的深度(离根的边数)，用存在的度为 1 的结点作起点调用 BFS,求出其它非根结点的深度，在各次调用 BFS 算法中求最大深度，即为树的直径。时间 $O(n(n+e))=O(n^2+ne)$，这里 $O(ne)$是一次外部调用 BFS 的运行时间，调用 BFS 的最多次数(指外部调用)不超过 $O(n)$(存储结构为邻接表时)。

解法三：用邻接表作为存储结构依次删去树叶(度为一的结点)，将与树叶相连的结点度数减 1。设在第一轮删去原树 T 的所有树叶后，所得树为 T1;再依次做第二轮删除，即删除所有 T1 的叶子；如此反复，若最后剩下一个结点，则树直径应为删除的轮数乘以 2。具体算法描述如下：

```
m:=0;
FOR i:=1 TO n DO
  IF du(v[i])=1 THEN
    BEGIN
      enqueue(Q,v[i])    {叶子 vi 入队 m:=m+1;m 记录当前一轮叶子数}
    END;
r:=0;     {表示删除叶子轮数}
WHILE m>=2 DO    {当前叶子轮数}
  BEGIN
      j:=0;    {j 计算新一轮叶子数目}
       FOR i:=1 TO m DO
         BEGIN
           dequeue(Q,v)://出队,表示删去叶子 v，将与 v 相邻的结点 w 的度数减 1
           IF du(w)=1 THEN    {w 是新一轮的叶子}
             BEGIN
               j:=j+1;
               enqueue(Q,w)    {w 入队}
             END
  END;
  r:=r+1;    {轮数加 1}
  m:=j    {新一轮叶子总数}
END;
IF m=0 THEN return 2×r-1          {m=0,直径为轮数×2-1}
       ELSE return 2×r;           {m=1,直径为轮数×2}
```

26．**设计一个算法，从一给定的顺序表 L 中删除元素值在 x 到 y（x≤y）之间的所有元素，要求以较高的效率来实现，空间复杂度为 O(1)。**

【答案】可以采用上例的解法,只是将L.data[i]==x的条件改为L.data[i]>=x && L.data[i]<=y。

对应算法如下。

```
void delallxy1(SqList &L,ElemType x, ElemType y)
{   int i,k=0;
    for (i=0;j<L.length;i++)
        if (!(L.data[i]>=x && L.data[i]<=y))
        {   L.data[k]=L.data[i];
            k++;
        }
    L.length=k;                          //表长置为 k
}
```

27．**二项式$(a+b)^n$展开式的系数为**

$$\begin{cases} C(n,0)=1,\ C(n,n)=1, & \text{对于 } n\geqslant 0 \\ C(n,k)=C(n-1,k)+C(n-1,k-1), & \text{对于 } 0<k<n \end{cases}$$

形成著名的杨辉三角形，如下所示：

```
n=0                                  1
n=1                               1     1
n=2                            1     2     1
n=3                         1     3     3     1
n=4                      1     4     6     4     1
n=5                   1     5    10    10     5     1
n=6                1     6    15    20    15     6     1
n=7             1     7    21    35    35    21     7     1
n=8          1     8    28    56    70    36    28     8     1
```

(1)试写一个递归算法，根据以上公式生成$C(n,k)$；

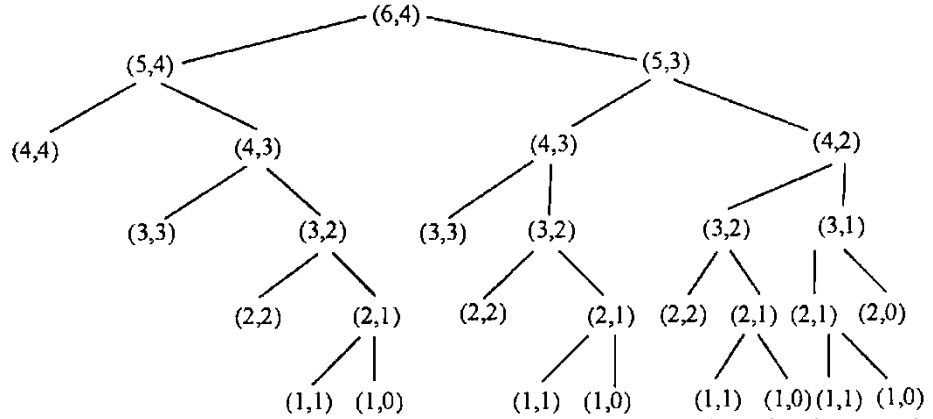
(2)试画出计算$C(6,4)$的递归树；

(3)试写一个递归算法，既不用数组也不用栈，对于任意的$0\leqslant k\leqslant n$计算 $C(n,k)$。

【答案】(1)求$C(n,k)$的递归算法如下：

```
int c(int n, int k)
{
  if (k==0 || k==n) return(1);
  return(c(n-1,k)+c(n-1,k-1));
}
```

(2)C(6,4)的递归树如下图所示：



(3)求 $C(n,k)$的非递归算法如下(根据二项式定理 $C(n,k)=\frac{n(n-1)\cdots(n-k+1)}{k!}$)：

```
int c(int n, int k)
{
   long x=1,y=1;
   int i;
   for (i=1;i<=k;i++) x*=i;
   for (i=n;i>=n-k+1;i--) y*=i;
   return(y/x);
}
```

28．**设有一对刚出生的小兔子，假定两个月以后每个月便可繁殖雌雄各一的 k 对小兔子，请输出计算第 n 个月后兔子数目的递推公式，并写出计算 F(n)的程序。**

【答案】递推公式为：

$$F(n)=\begin{cases}0, n<0\\ 1, 0\leqslant n\leqslant 2\\ F(n-1)+kF(n-1)\end{cases}$$

C 程序如下:

```
int fb(int k,int n)
{
  if(n<0) return(0);
  else if(n<3) return(1);
       else return(fb(k,n-1)+k*fb(k,n-1));
}
main( )
{
   int k,n;
   printf("input n:");
   scanf("%d",&n);
   prentf("input k:");
   scanf("%d",&k);
  printf("f(%d,&d)=&d\n",k,n,fb(k,n));
}
```

29. **输入一个字符串，内有数字和非数字字符，如:** ak123x456 17960?302gef4563**，将其中连续的数字作为一个整体，依次存放到一数组 a 中，例如 123 放入** $a[0]$**，** 456**放入** $a[1]$**，……编程统计其共有多少个整数，并输出这些数。**

【答案】核心是如何将数从字符串中分离出来。从左到右扫描字符串，初次碰到数字字符时，作为一个整数的开始。然后进行拼数，即将连续出现的数字字符拼成一个整数，直到碰到非数字字符为止，一个整数拼完，存入数组，再准备下一整数，如此下去，直至整个字符串扫描到结束。核心语句段如下:

```
while(ch!='#')                          //'#'是字符串结束标记,字符 ch 已输入
  if(isdigit(ch))                       //是数字字符
    {num=0;                             //数初始化
      while(isdigit(ch)&& ch!='#')      //拼数
       {num=num*10+'ch'-'0';   cin>>ch;  }
        a[i++]=num;                     //i 是整数个数, 初值是 0
        if(ch!='#')cin>>ch; ;           //若拼数中输入了'#', 则不再输入
       }
   else cin>>ch;                        //输入非数字且非#时, 继续输入字符
```

【基础】2025 年福州大学 083500 软件工程《408 计算机学科专业基础之数据结构》考研基础检测 5 套卷（三）

说明：本书按照考试大纲、历年真题、指定参考书等公开信息潜心整理编写，由学长严格审核校对，仅供考研备考使用，与目标学校及研究生院官方无关，如有侵权请联系我们立即处理。

一、单项选择题

1．**若串** $s='bioinformatics'$，**其字串的个数是_____。**

A.15
B.95
C.35
D.105

【答案】D

【解析】对于长度为 n 的字符串来说，长度为 1 的子串有 n 个，长度为 2 的子串有 n-1 个，长度为 3 的子串有 n-2 个，…，长度为 n 的子串有 1 个。所以总共的子串数为 $n(n+1)/2$。题目中 n 为 14,故答案为 D

2．**假设有 60 行 70 列的二维数组** $a[1..60,1..70]$ **以列序为主序顺序存储，其基地址为 10000,每个元素占 2 个存储单元，那么第 32 行第 58 列的元素** $a[23,58]$ **的存储地址为_____。(无第 0 行第 0 列元素)**

A.16902
B.16904
C.14454
D.ABC 均不对

【答案】A

3．**一个栈的入栈序列为 A，B，C，D，E，则栈的不可能出栈序列是_____。**

A.ABCDE
B.EDCBA
C.DECBA
D.DCEAB

【答案】D

4．**设 7 行 6 列的数组 a 以列序为主序顺序存储，其基地址为 1024,每个元素占 2 个存储单元，第 4 行第 5 列的元素(假定无第 0 行第 0 列)的存储地址是_____。**

A.1068
B.1086
C.1084
D.1066

【答案】B

【解析】该数组以列序为主序顺序存储，其第 4 行第 5 列的元素为第 $(5-1)\times 7+4$ 个元素，其地址为 $1024+(4\times 7-4)\times 2=1086$

5．**具有 6 个顶点的无向图，当有_____条边时能确保是一个连通图。**

A.8
B.9
C.10
D.11

【答案】D

【解析】确保是一个连通图等价于最大边数的非连通图的基础上再加一条边。5 个结点为一个连通分量且具有最多边时即是最大边数的非连通图，所以图非连通时最多有 $5\times(5-1)/2=10$条边，故当有 10+1=11 条边时确保是一个连通图。

6. **B+树不同于 B 树的特点之一是_____。**

A.能支持顺序查找

B.结点中含有关键字

C.根结点至少有两个分支

D.所有叶结点都在同一层上

【答案】A

7. **对包含 n 个关键字的散列表进行查找，平均查找长度是_____。**

A. $O(\log_2 n)$

B. $O(n)$

C. $O(n\log_2 n)$

D.不直接依赖于 n

【答案】D

【解析】哈希表的平均查找长度是装填因子的函数，而不是 n 的函数。

8. **循环队列是 _____。**

A.不会产生下溢出

B.不会产生上溢出

C.不会产生假溢出

D.以上都不对

【答案】C

9. **串的连接运算不满足_____。**

A.分配率

B.交换率

C.结合率

D.都不满足

【答案】D

二、填空题

10. **以下程序的功能是实现带附加头结点的单链表数据结点的逆序连接，请填空完善之。**

```
void reverse(pointer h)
  /* h 为附加头结点指针 */
 { pointer p, q;
   p=h->next;  h->next=NULL;
   while(_(1)_____)
      {q=p; p=p->next; q->next=h->next; h->next=_(2)_____; }
 }
```

【答案】(1) p!=null　　//或 p，链表未到尾就一直作

(2) q　　//将当前结点作为头结点后的第一元素结点插入

11．**设有广义表** LS=((a,b,c),(d,e,f))，**取出原子 e 的运算是_____。**

【答案】访问 LS 的子表 tail(LS)的头指针指向的第二个原子，即为 e。

【解析】考查广义表的定义以及运算。由于广义表是由原子或者子表构成的聚集，因此访问深层的结构时一般采用类似于顺序表或者树形结构访问的方法。

要访问原子 e，首先访问 LS 表的尾结点(d,e,f)，这是个子表，然后利用子表的头指针进行遍历，遍历到第二个结点的时候返回此时的原子，即为 e。

12．**在字符串运算中的"模式匹配"是常见的，KMP 匹配算法是有用的办法。**

①其基本思想为_____。

②对模式串 $p(=p_1p_2\cdots p_n)$，求数组时，$next[i]$是满足下列性质的 k 的最大值或为 0:_____。

【答案】①不回溯主串指针，而用 next 数组标示出的下一个比较位置来避免不必要的时间浪费。② $p_{(1\ldots k-1)}=p_{(j-k+1\ldots j-1)}$(其中 j 表示模式串中与主串字符失配位置)。

13．**在结点个数大于 1 的单循环链表中，指针 p 指向表中的某个结点，当执行以下程序后，指针 s 指向结点* p的直接前驱结点，请在下画线处填入合适语句使得程序完整。**

```
s = p;
while(_______)s = s->link;
```

【答案】 s->link->link != p

14．**对 n 个记录的表进行简单选择排序，所需的关键字间的比较次数为_____，最坏情况下所需的记录移动次数为_____。**

【答案】$n(n-1)/2$ 、$3(n-1)$

【解析】简单选择排序的基本思想是，每一趟在 $n-i+1(i=1,2,\cdots,n-1)$个记录中选取关键字最小的记录作为第 i 个记录，所以第 i 趟排序需要进行 n-i 次比较。所以总的比较次数为 $\sum_{i=1}^{n-1}(n-i)=n(n-1)/2$。对于移动次数，最好的情况是不用进行交换，所以移动次数是"0"。在最坏情况下，每次都进行对换，要进行次对换，一次对换需要三次移动，所以最坏情况下，所需的记录移动次数为 3(n-1)。

15．**n 个顶点的连通无向图，其边的条数至少为_____。**

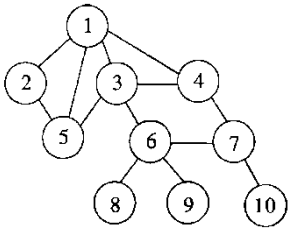
【答案】n-1

三、应用题

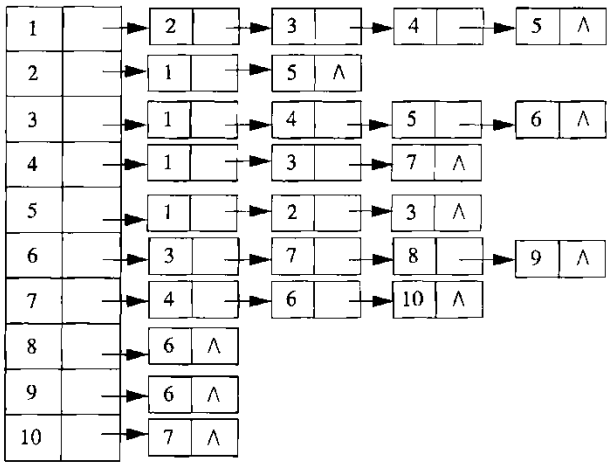
16．**给出图 G:**

(1)画出 G 的邻接表表示图;

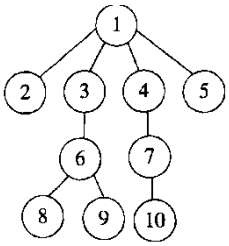
(2)根据你画出的邻接表，以顶点①为根，画出 G 的深度优先生成树和广度优先生成树。



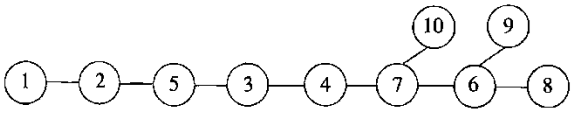
【答案】(1)如下图所示



(2)广度优先生成树，深度优先生成树，为节省篇幅，生成树横画，下同。



广度优先生成树



深度优先生成树

17．**在外部分类时，为了减少读、写的次数，可以采用K路平衡归并的最佳归并树模式。当初始归并段的总数不足时，可以增加长度为零的“虚段”。请问增加的“虚段”的数目为多少？请推导之。设初始归并段的总数为m。**

【答案】对于k-路归并，就是要构成一个归并树。这个归并树，如果是k叉树，则由树的性质可以得出其叶结点个数与中间结点个数的关系。因为初始归并段都是叶结点，设为m，由k叉树性质可知有(m-1)/(k-1)个中间结点，也就是说(m-1)/(k-1)为整数时，构成k叉树。所以满足条件(m-1)MOD(k-1) = 0，不需要增加虚段。而此条件不满足时，就要增加虚段，设 x = (m-1)MOD(k-1)，则可知需要增加的虚段的个数是 **k-x-1**，即 **k-(m-1)MOD(k-1)-1**。

18．**下面给出一个算法，其中 n 是数组 A[]中元素总数。**

```
template<class Type>void unknown(Type a[], int n){
      int d = 1, j;
      while (d < n/3) d = 3 * d + 1;
      while (d > 0) {
        for (int i = d; i < n; i++) {
          Type temp = a[i];
          j = i;
          while (j >= d && a[j - d] > temp) {
              a[j] = a[j - d];
              j -= d;
          }
          a[j] = temp;
```

以上内容仅为本文档的试下载部分，为可阅读页数的一半内容。如要下载或阅读全文，请访问：https://d.book118.com/808116011001006101