掌心博阅电子书重点名校考研专业课精品资料系列

【冲刺】2024年广西大学085410人工智能《828程序设计与数据结构之数据结构》考研仿真模拟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

【冲刺】2024 年广西大学 085410 人工智能《828 程序设计与数据结构之数据结构》考研仿真模拟 5 套卷（一）.....4

【冲刺】2024 年广西大学 085410 人工智能《828 程序设计与数据结构之数据结构》考研仿真模拟 5 套卷（二）.....15

【冲刺】2024 年广西大学 085410 人工智能《828 程序设计与数据结构之数据结构》考研仿真模拟 5 套卷（三）.....26

【冲刺】2024 年广西大学 085410 人工智能《828 程序设计与数据结构之数据结构》考研仿真模拟 5 套卷（四）.....37

【冲刺】2024 年广西大学 085410 人工智能《828 程序设计与数据结构之数据结构》考研仿真模拟 5 套卷（五）.....48

【冲刺】2024 年广西大学 085410 人工智能《828 程序设计与数据结构之数据结构》考研仿真模拟 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

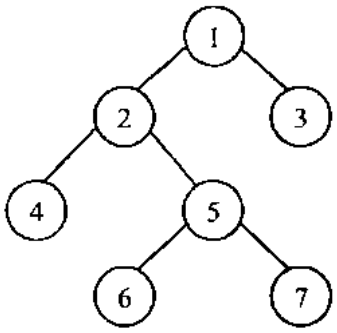
1．**一个含有 n 个顶点和 e 条边的简单无向图，在其邻接矩阵存储结构中共有_____个零元素。**

A.e
B.2e
C. n^2-e
D.n^2-$2e$

【答案】D

【解析】由邻接矩阵的定义可知，无向图的邻接矩阵是对称的，即图中的一条边对应邻接矩阵中的两个非零元素。因此，在一个含有 n 个顶点和 e 条边的简单无向图的邻接矩阵中共有n^2-$2e$个零元素。

2．**给定二叉树如下图所示。设 N 代表二叉树的根，L 代表根节点的左子树，R 代表根节点的右子树。若遍历后的节点序列为 3,1,7,5,6,2,4，则其遍历方式是。_____。**



图

A.LRN
B.NRL
C.RLN
D.RNL

【答案】D

【解析】遍历后的结果是先右子树，再根节点，最后为左子树，即为 RNL。

3．**已知关键序列** 5,8,12,19,28,20,15,22**是小根堆(最小堆)，插入关键字 3,调整后得到的小根堆是_____。**

A. 3,5,12,8,28,20,15,22,19
B.3,5,12,19,20,15,22,8,28
C.3,8,12,5,20,15,22,28,19
D. 3,12,5,8,28,20,15,22,19

【答案】A

【解析】将 3 插入后，从序列的最后一个非叶子结点 19 开始进行调整，直至根结点。

4. **若一个有向图的邻接矩阵中，主对角线以下的元素均为零，则该图的拓扑有序序列_____。**

A.存在

B.不存在

【答案】A

【解析】把 V 进行拓扑排序，直观上说，就是把 V 的所有顶点排成一个序列$V_{i_1},V_{i_2},...,V_{i_n}$(其中$i_1,i_2,...,i_n$，是 1、2、3...n 的一种排列。)且使得任何$<V_{i_k},V_{i_j}>\in A$，则在序列中V_{i_k}排在V_{i_j}之前。

这个有向图的邻接矩阵 A 主对角线以下的元素均为零，则说明对于$<V_i,V_j>\in A$，且 i > j，$a_{ij}=0$，即V_i到V_j不存在有向边。所以按顶点号从小到大排列顶点，而得到的序列一定能满足拓扑有序序列的要求。

5. **数组通常具有的两种基本操作是_____。**

A.查找和修改

B.查找和索引

C.索引和修改

D.建立和删除

【答案】A

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

A.1 和 1

B.3 和 1

C.1 和 2

D.2 和 3

【答案】B

7. **二维数组 M 的元素是 4 个字符(每个字符占一个存储单元)组成的串，行下标 i 的范围为 0～4,列下标 j 的范围为 0～5，M 按行存储时元素 M[3][5]的起始地址与 M 按列存储时元素_____的起始地址相同。**

A.M[2][4]

B. M[3][4]

C.M[3][5]

D.M[4][4]

【答案】B

【解析】M 为 5 行 6 列，在按行存储时，元素 M[3][5]的起始地址为(3×6+5)×4=92。在按列存储时，元素 M[2][4]的起始地址为(4×5+2)×4=88，元素 M[3][4]的起始地址为(4×5+3)×4=92，元素 M[3][5]的起始地址为(5×5+3)×4=112，元素 M[4][4]的起始地址为(4×5+4)×4=96。本题答案为 B。

8. **在有向图的邻接表存储结构中，顶点 v 在链表中出现的次数是_____。**

A.顶点 v 的度

B.顶点 v 的出度

C.顶点 v 的入度

D.依附于顶点 v 的边数

【答案】C

【解析】有向图邻接表第 i 个单链表中结点的个数表示顶点v_i的出度；而顶点 v 在整个邻接表中出现的次数则表示顶点v_i的入度。

9. **具有$n(n>0)$个顶点的无向图最多含有_____条边。**

A. $n(n-1)$

B. $\frac{n(n+1)}{2}$

C. $\frac{n(n-1)}{2}$

D. $n(n+1)$

【答案】C

【解析】具有 n 个结点的无向图边最多的图是无向完全图，在无向完全图中，每个顶点与其他的 n-1 个顶点都有边。因此，含有 n 个顶点的无向完全图共有 $n(n-1)/2$条边。

二、填空题

10. **链接存储的特点是利用_____来表示数据元素之间的逻辑关系。**

【答案】指针

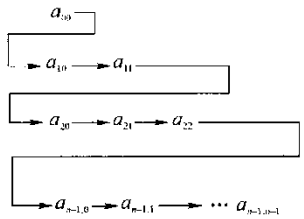
11. **已知矩阵 A[0..9,0..9]的每个元素占 5 个存储单元，将其按行优先次序存储在起始地址为 1000 的连续内存单元中，则元素 A[6,5]的地址是_____。**

【答案】1325

12. **有一个 10 阶对称矩阵 A，采用压缩存储方式(以行序为主序存储，且$A[0][0]=1$)，则$A[8][5]$的地址是_____。**

【答案】42

【解析】对称矩阵的压缩一般是把矩阵的下三角部分(包括对角线)进行线性存储，行序为主序的方法如图所示。



图

显然$A[8][5]$前有 8 行元素，总和为$1+2+3+\cdots+8=36$；在$A[8]$这一行，$A[8][5]$前面有 5 个列数，所以$A[8][5]$前面一共有 41 个数。因为第一个数$A[0][0]$地址为 1，所以$A[8][5]$的地址为 42。

13. **用 S 表示入栈操作，X 表示出栈操作，若元素入栈顺序为 1234,为了得到 1342 出栈顺序，相应的 S 和 X 操作串为_____。**

【答案】SXSSXSXX

14. **在单链表 L 的表头插入指针 S 所指结点的语句序列是_____。**

【答案】S^.next := L# L := S#

【解析】注意添加结点时的指针操作，首先应建立新结点和其后继链接关系。然后再建立前面的连接关系，否则 L 就被覆盖了

15．**在具在 n 个节点的二叉树中如果有 m 个叶子节点，则一定有_____个度为 1 的节点，_____个度为 2 的节点。**

【答案】n-2m+1、m-1

【解析】由二叉树性质 1 可知，$n_0=n_2+1$，这样$n_2=n_0-1=m-1$，又有$n=n_0+n_1+n_2$，所以得到$n_1=n-n_0-n_2=n-m-(m-1)=n-2m+1$。

三、判断题

16．**存放在磁盘、磁带上的文件，既可以是顺序文件，也可以是索引结构或其他结构类型的文件。_____**

【答案】×

17．**构成数据的最小单位是数据元素。_____**

【答案】×

18．**中序遍历一棵二叉排序树的结点就可得到排好序的结点序列。_____**

【答案】√

19．**在某种遍历的线索二叉树中，进行这种遍历时可以直接沿右指针找下去，从而访问所有节点。_____**

【答案】×

20．**对顺序找进行进栈、出栈操作，不涉及元素的前、后移动问题。_____**

【答案】√

21．**最佳二叉排序树的任何子树都是最佳的。_____**

【答案】√

四、应用题

22．**对于一个有向图，除了进行拓扑排序，还可以采用什么办法判断图中是否存在回路？请简述判断原则。**

【答案】图的深度优先遍历可用于判断图中是否存在回路。若从有向图某顶点 V 出发进行遍历，在 dfs(v)结束之前出现从顶点 W 到顶点 V 的回边，由于 W 在生成树上是 V 的子孙，则向图中必定存在包含顶点 V 和 W 的环。

23．**有一种简单的排序算法，叫做计数排序**（Count Sorting）**。这种排序算法对一个待排序的表(用数组表示)进行排序，并将排序结果存放到另一个新的表中。必须注意的是，表中所有待排序的关键码互不相同。计数排序算法针对表中的每个记录，扫描待排序的表一趟，统计表中有多少个记录的关键码比该记录的关键码小。假设针对某一个记录，统计出的计数值为 c，那么，这个记录在新的有序表中的合适的存放位置即为 c。**

(1)给出适用于计数排序的数据表定义。

(2)使用 Pascal 或 C 语言编写实现计数排序的算法。

(3)对于有 n 个记录的表，关键码比较次数是多少？

(4)与简单选择排序相比较，这种方法是否更好？为什么？

【答案】(1)数据表定义为：

TYPE sortlist=ARRAY [0..n－1] OF key。
或 typedef int key[MAXSIZE];

(2)用 C 语言实现的计数排序算法为：

```
void countsort( key *oldlist，key *newlist，int n)
{int i，j，count;
      for (i=0; i<n; i++){
            count=0;
          for (j=0; j<n; j++)
                if (*(oldlist+j)<*(oldlist+i)) count++;
            *(newlist+count)=*(oldlist+i);
      }
  }
```

(3)对于有 n 个记录的表，关键码比较次数是 n^2。

(4)简单选择排序比这种排序好，因为对具有 n 个记录的数据表进行简单选择排序只需进行 $1+2+\cdots+(n-1)=\frac{n(n-1)}{2}$ 次比较，且可在原地进行排序。

24．**已知三个带头结点的线性链表 A、B 和 C 中的结点均依元素值自小至大非递减排列(可能存在两个以上值相同的结点)，编写算法对 A 表进行如下操作：使操作后的链表 A 中仅留下三个表中均包含的数据元素的结点，且没有值相同的结点，并释放所有无用结点。限定算法的时间复杂度为 $O(m+n+p)$，其中 m、n 和 p 分别为三个表的长度。**

【答案】要求留下三个链表中的公共数据，首先查找两表 A 和 B 中的公共数据，再去 C 中查找有无该数据。要消除重复元素，应记住前驱，要求时间复杂度为 $O(m+n+p)$，在查找每个链表时，指针不能回溯。算法主要由 while（pa && pb && pc)控制。若三表有一个到尾，则结束循环。A 表头结点作监视哨，否则第一个结点要特殊处理。算法最后要给新 A 表置结尾标记，同时若原 A 表没到尾，还应释放剩余结点所占的存储空间。

25．**举例说明“拓扑排序”的算法思路。**

要求：例中顶点个数不得少于七个。

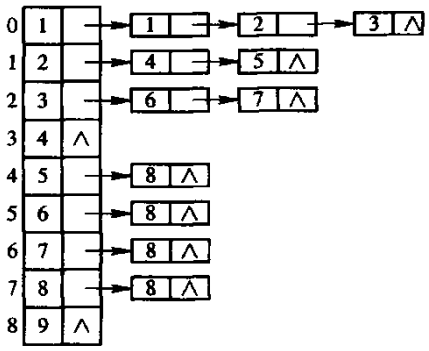
【答案】拓扑排序的思想如下：

(1)在 AOV 网中选择一个入度为零(没有前驱)的顶点且输出它；

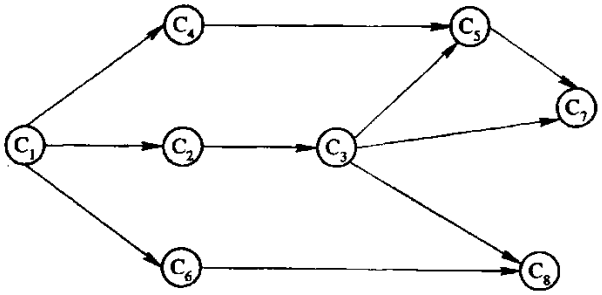
(2)从网中删除该顶点及其与该顶点有关的所有边；

(3)重复上述两步，直至网中不存在入度为零的顶点为止。

如下图所示的一个有向图，其拓扑排序为：C_1，C_2，C_3，C_4，C_5，C_6，C_7，C_8。



邻接表



图有向图

26. **一个算法所需时间由下述递归方程表示，试求出该算法的时间复杂性的级别(或阶)(以大“O”形式表示)。**

$$T(n)=\begin{cases}1, & n=1\\ 2T(\frac{n}{2})+n, & n>1\end{cases}$$

其中:n 是问题的规模，为简单起见，设 n 是 2 的整数幂。

【答案】设$n=2^k$，亦即$k=\log_2 n$。

$$\begin{aligned}T(2^k)&=2T(2^{k-1})+2^k\\&=2[2T(2^{k-2})+2^{k-1}]+2^k\\&=4T(2^{k-2})+2*2^k\\&=4[2T(2^{k-3})+2^{k-2}]+2*2^k\\&=8T(2^{k-3})+3*2^k\\&\cdots\\&=2^kT(2^0)+k*2^k\\&=(k+1)2^k\end{aligned}$$

所以有$T(n)=n(\log_2 n+1)$，所以该算法的时间复杂度为$O(n\log_2 n)$。

27. **一个算法所需时间由下述递归方程表示，试求出该算法的时间复杂性的级别(或阶)。(以大 O 形式表示。)**

$$T(n)=\begin{cases}1 & \text{若}n=1\\ 2T\left(\frac{n}{2}\right)+n & \text{若}n>1\end{cases}$$

其中：n 是问题的规模，为简单起见，设 n 是 2 的整数幂。

【答案】$O(n\log_2 n)$

设$n=2^k(k\geqslant 0)$，根据题目所给定义，有：

$$T(2^k)=2T(2^{k-1})+2^k=2^2T(2^{k-2})+2*2^k$$

由此，可得一般递推关系式：

$$T(2^k)=2^iT(2^{k-i})+i*2^k$$

进而，可得：

$$T(2^k)=2^kT(2^0)+k*2^k=(k+1)2^k$$

即$T(n)=n(\log_2 n+1)=n(\log_2(2n))=O(n\log_2 n)$

五、算法设计题

28．**请利用两个栈 S1 和 S2 来模拟一个队列。已知栈的三个运算定义如下：PUSH(ST,x):元素 x 入 ST 栈；POP(ST,x):ST栈顶元素出栈，赋给变量 x; Sempty(ST):判 ST 栈空否。那么如何用栈的运算来实现该队列的三个运算：enquence:插入一个元素入队列；dequence:删除一个元素出队列； quence_empty判队列为空。(请写明算法的思想及必要的注释。)**

【答案】利用两个栈模拟一个队列运算的思想是用一个栈作为输入，而另一个栈作为输出。当进队列时，总是将数据进入到作为输入的栈中。在输出时，如果作为输出的栈已空，则从输入栈将已输入到输入栈的所有数据输入到输出栈中，然后由输出栈输出数据；如果作为输出的栈不空，则就从输出栈输出数据。显然，只有在输入、输出栈均为空时队列才为空。

一个栈 S1 用来插入元素，另一栈 S2 用来删除元素，删除元素时应将前一栈的所有元素读出，然后进入到第二个栈中。

算法描述如下：

```
PROCEDURE enquence(VAR S1:stack; x:integer)
BEGIN
  IF (S1.top=m0) THEN writeln('队列上溢出')
  ELSE   push(S1,x);
END;
PROCEDURE dequence(VAR S1,S2:stack; VAR x:integer)
BEGIN
   S2.top=0;  {将 S2 清空，将 S1 的元素退栈后压入 S2，此时 S1 为空}
   WHILE (not sempty(S1)) DO push(S2,pop(S1));
   pop(S2,x);  {将 S1 的栈顶元素退栈并赋给 x}

   WHILE (not sempty(S2)) DO push(S1,pop(S2))   {将 S2 的所有元素推栈并压入 S1 中}
 END;

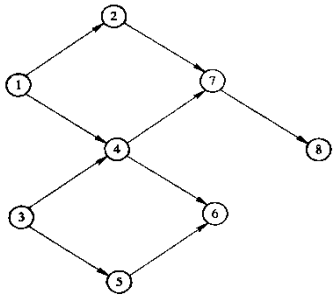
 FUNCTION quence_empty (VAR s1: stack)
 BEGIN
   IF sempty (s1) THEN quence_empty: =true;
   ELSE quence_smpty: =false;
 END;
```

29．**试设计算法在O(n)时间内将数组A[1..n]划分为左右两个部分，使得左边的所有元素值均为奇数，右边的所有元素值均为偶数，要求所使用的辅助存储空间大小为O(1)。**

【答案】算法如下：

```
void partition(int a[],int n)
{
  int i=1, j=n,k;
  while (i<j) do
  {
    while (i<j&&odd(a[i])) do i=i+1;
    while (i<j&&! odd(a[i])) do j=j-1;
    if(i<j)
    {
      k=a [i];
      a[i]=a[j];
      a [j]=k;
      i++;
      j++;
    }
  }
}
```

30．**写出从下图的邻接表表示转换成邻接矩阵表示的算法，用类 Pascal 语言(或 C 语言)写成过程形式。**



图

【答案】假设 n 个顶点的有向图G的邻接表头结点数组为 adlist[1..n]，邻接矩阵为 A[1..n,1..n]。

```
PROCEDURE ListToArray(VAR A:ARRAY[1..n,1..n] OF 0..1);
  BEGIN
    FOR i: =1 TO n DO
      FOR j: =1 TO n DO
        A[i,j]:=0;                {初始化}
    FOR i: =1 TO n DO
      BEGIN
        p:=adlist[i].firstarc;        {将邻接表中的链表转换成邻接矩阵中的一行}
        WHILE p<>NIL DO
          BEGIN
            A[i,p^.val]: =1;
            p: =p^.nextarc
          END
      END
  END;
```

31．**编写算法，求指定结点在给定的二叉排序树中的层次。**

【答案】算法如下:

```
int level(TreeNode *R, int D)
 {
   if (R = NULL)
       return 0;
   else if (R->data = D)
       return 1;
  else if (R->data < D)
     return (1+level(R->leftchild,D);
  else
     return(1+level(R->rightchild,D);
 }
```

32．**编写算法，判断一个用邻接表存储的有向图是否存在回路，并写出算法的基本思路。**

【答案】可以在深度优先遍历时引入标记数组 mark 来标记访问过的顶点，当mark[u]为 1 时，表示顶点 u 已经被访问。如果深度优先遍历时发现遍历到一个已经被访问过的顶点，则说明图中存在回路。其算法描述如下:

```
Bool circle(Graph &G, int u){
    if(G.Mark[u]==VISITED) return TRUE;
    G.Mark[u] = VISITED;
    for(Edge e=G.FirstEdge(u); G.IsEdge(e); e=G.NextEdge(e)){
        Bool flag = circle(G, G.ToVertex(e));
        if(flag) return true;
```

```
        }
        return FALSE;
    }
```

33．**写出和下列递归过程等价的非递归过程：**

```
PROCEDURE test(VAR sum:integer);
    VAR a:integer;
BEGIN
    read(a);
    IF a=0 THEN sum:=1
    ELSE BEGIN
            test(sum);
            sum:=sum*a;
         END
    write(sum);
END;
```

【答案】

```
PROCEDURE test;
VAR stack: array[1..maxsize] of integer;
    top,sum,a: integer;
BEGIN
    top:=0;
    sum:=1;
    read(a);
    WHILE a<>0 DO
    BEGIN
        top:=top+1;
        stack[top]:=a;
        read(a);
    END
    write(sum);
    WHILE top<>0 DO
    BEGIN
        sum:=sum*stack[top];
        top:=top-1;
        write(sum)
    END
END
```

34．**假定二叉树用链式存储结构表示，设计一个判断一棵二叉树是否为平衡二叉树的算法。**

【答案】计算二叉树深度的算法：

```
int depth(Node *b)
{
  if ( b == NULL)  return 0;
  else if ( b->left != NULL  ||  b->right != NULL )
  {
      Ldep = 1 + depth(b->left);
       Rdep = 1 + depth(b->right);
       return  Ldep? Rdep : (Ldep > Rdep);
  }
  else  return 1;
}
```

判断是否平衡的算法：

```
void judge(Node *root)
{ //Ldep 和 Rdep 初值为零
    if (root->left != NULL)  Ldep = depth(root->left);
```

```
    if (root->right != NULL)  Rdep = depth(root->right);
   if(abs(Ldep-Rdep) > 1)  printf("%s\n", "It's not a AVLtree");
    else
      {
       if (root->left != NULL)  judge(root->left);
       if (root->right != NULL)  judge(root->right);
      }
}
```

35．**已知 Ackermann 函数定义如下：**

$$Ack(m,n)=\begin{cases} n+1 & m=0 \\ Ack(m-1,1) & m\neq 0, n=0 \\ Ack(m-1,Ack(m,n-1)) & m\neq 0, n\neq 0 \end{cases}$$

(1)写出递归算法；

(2)写出非递归算法。

【答案】(1)递归算法:

```
FUNCTION Ackm(m,n:integer):integer;
{计算 Ackm(m,n)，如果 m<0 或 n<0，则函数返回值为-1}
BEGIN
   a:= -1;
   IF (m>=0) and (n>=0) THEN
     CASE
        m=0: a:=n+1;
        n=0: a:=Ackm(m-1,1);
        ELSE   a:=Ackm(m-1,Ackm(m,n-1))
     END;
  return(a);
END;
```

(2)非递归算法使用一个栈来实现。

```
FUNCTION Ackm(m,n:integer):integer;
{s[1..max]为附加栈，max 足够大，栈中每个元素有两个数据域，分别存放 m 和 n 的值}
BEGIN
   top:=1;
   s[top].mval:=m;
   s[top].nval:=n;        {栈的初始化}
   REPEAT
      WHILE s[top].mval<>0 DO
         BEGIN
            WHILE s[top].nval<>0 DO        {m≠0，n≠0 时，先求 Ack(m,n-1)}
               BEGIN
                  top:=top+1;
                  s[top].mval:=s[top-1].mval;
                  s[top].nval:=s[top-1].nval
               END;
            {n=0，m≠0 时，求 Ack(m-1,1)}
            s[top].mval:=s[top].mval-1;
            s[top].nval:=1
         END;
   {m=0 时，回溯}
   IF top>1 THEN
     BEGIN
        top:=top+1;          {计算 Ack(m-1,Ack(m,n-1))}
        s[top].mval:=s[top].mval-1;
```

```
            s[top].nval:=s[top].nval+1          {m=0 时，Ack(m,n-1)=n+1}
        UNTIL (top=1 and s[top].mval=0);
        Ackm:=s[top].nval+1;          {m=0 时，Ack(m,n)=n+1}
        top:=top+1
END;
```

【冲刺】2024年广西大学085410人工智能《828程序设计与数据结构之数据结构》考研仿真模拟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．**在有向图G的拓扑序列中，若顶点V_i在顶点V_j之前，则下列情形不可能出现的是_____。**

A.G中有弧<V_i，V_j>

B.G中有一条从V_i到V_j的路径

C.G中没有弧<V_i，V_j>

D.G中有一条从V_j到V_i的路径

【答案】D

2．**一棵度为k的树有n个结点，在该树的标准形式的存储结构中，空指针的个数为_____个。**

A.n+k

B.n×k

C.$n\times(k-1)$

D.$n\times(k-1)+1$

【答案】D

【解析】度为k的树的标准形式是指，每个树结点有k个指针域。一棵度为k的树有n个结点，则有n×k个指针域，其中有个n-1指针有指向(除根结点外，其他结点都是由一个指针域所指向的)，则有$n\times k-(n-1)=n\times(k-1)+1$个空指针域。

3．**稀疏矩阵的三元组存储方法_____。**

A.实现转置运算很简单，只需将每个三元组中的行标和列标交换

B.是一种链式存储方法

C.矩阵的非零元个数和位置在操作过程中变化不大时较有效

D.比十字链表法更高效

【答案】C

【解析】稀疏矩阵还要实现三元组之间的重新排列；不是链式存储，利用数组储存；与十字链表相比使用情况不同。

4．**若用邻接矩阵存储有向图，矩阵中主对角线以下的元素均为零，则关于该图拓扑序列的结论是_____。**

A.存在，且唯一

B.存在，且不唯一

C.存在，可能不唯一

D.无法确定是否存在

【答案】C

【解析】这样的有向图中只有顶点i到顶点j（i<j)可能有边，而顶点j到顶点i一定没有边，也就是说这样的有向图中一定没有回路，所以可以产生拓扑序列，但拓扑序列不一定唯一。本题答案为C。

5. **将递归程序转换成非递归程序，一般都要使用栈，但是消除_____可以不使用栈。**

A.直接递归

B.间接递归

C.尾递归

【答案】C

6. **设 n 是描述问题规模的非负整数，下面程序片段的时间复杂度为_____。**

```
x=2;
while (x<n/2)
    x=2*x;
```

A. $O(\log_2 n)$

B. $O(n)$

C. $O(n\log_2 n)$

D. $O(n^2)$

【答案】A

【解析】基本算法是语句x=2*x，设其执行时间为$T(n)$，则有：$2^{T(n)} \leqslant n/2$，即$T(n)<\log_2 n/2=O(\log_2 n)$。本题答案为 A。

7. **表达式3＊2^(4＋2＊2－6＊3)－5求值过程中当扫描到 6 时，操作数栈和运算符栈为_____，其中^为乘幂，#表示表达式的开始符。**

A.3,2,4,1,1,和#＊^(＋＊－

B.3,2,8和#＊^(

C.3,2,4,2,2,和#＊^(－

D.3,2,8和#＊^－

【答案】C

【解析】扫描到 6 时数值栈应该是3,2,4,2,2。

8. **用 ISAM 和 VSAM 组织文件属于_____。**

A.顺序文件

B.索引文件

C.哈希文件

D.倒排文件

【答案】B

9. **在文件“局部有序”或文件长度较小的情况下，最佳内部排序方法是_____。**

A.直接插入排序

B.冒泡排序

C.简单选择排序

D.归并排序

【答案】A

二、填空题

10．**主串**S="abbacbabbcabbcabbcabcaabbc"，**子串**="abbcabcaa",**若用简单模式匹配算法，查找成功需要比较多少次_____？若用 KMP 算法，查找成功需要比较多少次_____？并计算出相应的**NEXT[ ]**数组_____和**NEXTVAL[ ]**数组值_____。**

【答案】39 次、27 次、011112312、011101302

【解析】简单模式匹配算法，查找成功需要比较 15 趟 39 次。若用 KMP 算法，查找成功需要比较 7 趟 27 次。NEXT[ ]和NEXTVAL[ ]数组值分别是 011112312 和 011101302。

11．**设长度为 n 的链队列用单循环链表表示，若只设头指针，则出队列操作的时间复杂度为_____。**

【答案】O(1)

12．**带头结点的双循环链表 L 中只有一个元素结点的条件是：_____。**

【答案】L->next->next==L && L->prior->prior==L && L->next!=L

13．**设** s="abcd"，**则执行语句**s2=SubStr(s,4,2)**后，s2=_____。**

【答案】空串或""

14．**有 n 个结点，高度(二叉树的高度定义为叶结点的最大层数，根结点为第 1 层)为 n 的二叉树有_____棵。**

【答案】2^{n-1}

15．**评价哈希函数好坏的标准是_____。**

【答案】哈希函数取值是否均匀

三、判断题

16．**二叉树是树的一种特殊情况。**_____

【答案】×

17．**对一棵树进行先根遍历和后根遍历时，其中叶子节点出现的相对次序是相同的。**_____

【答案】√

【解析】因为树的先根遍历和后根遍历过程中，都是按从左到右的次序遍历子树的。

18．**给出不同的输入序列构造二叉排序树时，一定得到不同的二叉排序树。**_____

【答案】×

19．**在平衡二叉树中，向某个平衡因子不为零的结点的树中插入一新结点，必引起平衡旋转。**_____

【答案】×

20．**在哈夫曼编码中，当两个字符出现的频率相同时，其编码也相同，对于这种情况应作特殊处理。**_____

【答案】×

【解析】哈夫曼编码中不允许出现两字符编码相同的情况。

21．**算法最终必须由计算机程序实现。**_____

【答案】×

【解析】算法不必由计算机程序实现。

四、应用题

22．**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)

第六趟匹配：aabaabaabaac
a(*i*=6, *j*=1)

第七趟匹配：aabaabaabaac
（成功） aabaac(*i*=13, *j*=7)

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

23．**对于待排序序列**{12, 11, 13, 49, 26, 14, 8, 7}

(1)以快速排序方法对该序列进行排序，写出各趟排序后的结果。

(2)以该序列为输入序列建立平衡二叉搜索树(即 AVL 树)，并求出其搜索成功的平均搜索长度 ASLsucc。

【答案】待排序序列: [12] , 11, 13, 49, 26, 14, 8, 7

(1)一趟排序: 7, 11, 8, [12] , 26, 14, 49, 13

二趟排序: [7] , 11, 8　13, 14, [26], 49

三趟排序　8, [11] [13] ,14

排序结果: 7, 8, 11, 12, 13, 14, 26, 49

(2)初始为空，插入 12 为根结点，11 插入为根结点的左子女，13 插入为根结点的右子女，49 插入为结点 13 的右子女，当 26 插入为结点 49 的左子女时失衡，最小子树的根结点是 13,做 RL 旋转。继续输入 14 为结点 13 的右子女后再次失衡，最小子树的根结点是 12,做 RL 旋转。继续 8 插入为结点 11 的左子女后第三次失衡，最小子树的根结点是 12,做 LL 旋转。最后将 7 插入结点 8 的左子女。除结点 11 和 8 的平衡因子为 1 外，其余结点平衡因子都是 0。13、11、26 是度为 2 的结点，12、14、49 和 7 是叶子。

24．**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**。

25．**将关键字序列**{1,9,25,11,12,35,17,29}**存放到一个哈希表中。**

(1)采用除留余数法构造哈希函数，线性探测法处理冲突。要求新插入记录的平均查找次数不多于 2.5 次。试确定哈希表的表长 m 及相应的哈希函数H(key)。

(2)由(1)构造出哈希表，并分别计算查找成功和不成功时的平均查找次数。

(3)采用(1)的哈希函数 H(key),但用链地址法处理冲突。构造哈希表，并分别计算此时查找成功和不成功时的平均查找次数。

【答案】(1)n=8,α=n/m。依题意，要求新插入记录的平均查找次数不多于 2.5 次，即不成功的查找长度小于等于 2.5,对于线性探测法，不成功的$ASL_{unsucc}=\left(1+\frac{1}{(1-\alpha)^2}\right)/2$，即有$ASL_{unsucc}\leqslant 2.5$，求得 $\alpha=n/m\geqslant 1/2$，$m\leqslant 2n=16$，可取 m=13,p=13。这样的哈希函数为： H(key)=key MOD 13。

(2) H(1)=1 MOD 13=1
H(9)=9 MOD 13=9
H(25)=25 MOD 13=12
H(11)=11 MOD 13=11
H(12)=12 MOD 13=12　　冲突
H_1(12)=(12+1) MOD 13=0
H(35)=35 MOD 13=9　　冲突
H_1(35)=(9+1) MOD 13=10
H(17)=17 MOD 13=4
H(29)=29 MOD 13=3

对应的哈希表如表所示。

下标	0	1	2	3	4	5	6	7	8	9	10	11	12
关键字	12	1		29	17					9	35	11	25
探测次数	2	1		1	1					1	2	1	1

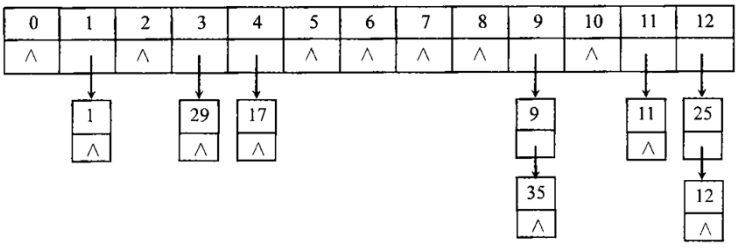
在等概率情况下： $ASL_{succ}=(6\times1+2\times2)/8=1.25$。

$ASL_{unsucc}=(3+2+1+3+2+1+1+1+1+7+6+5+4)/13=2.85$

(3)采用拉链法处理冲突建立的哈希表如图所示。在等概率情况下查找成功时各关键字的比较次数分别为：

关键字	12	1	29	17	9	35	11	25
比较次数	2	1	1	1	1	2	1	1

所以有： $ASL_{succ}=(2+1+1+1+1+2+1+1)/8=1.25$。



在等概率情况下查找不成功时各位置的比较次数分别为：

下标位置	0	1	2	3	4	5	6	7	8	9	10	11	12
比较次数	0	1	0	1	1	0	0	0	0	2	0	1	2

所以有：$ASL_{unsucc}=(0+1+0+1+1+0+0+0+0+2+0+1+2)/13=0.62$。

26．**试推导出总盘数为 n 的 Hanoi 塔的移动次数。**

【答案】设H_n为 n 个盘子的 Hanoi 塔的移动次数(假定 n 个盘子从钢针 X 移到钢针 Z，可借助钢针 Y)，则$H_n=2H_{n-1}+1$//先将 n-1 个盘子从 X 移到 Y，第 n 个盘子移到 Z,再将那 n-1 个移到 Z

$=2(2H_{n-2}+1)+1=2^2H_{n-2}+2+1=2^2(2H_{n-3}+1)+2+1=2^3H_{n-3}+2^2+2+1$

…

$=2^kH_{n-k}+2^{k-1}+2^{k-2}+\cdots+2^1+2^0$

$=2^{n-1}H_1+2^{n-2}+2^{n-3}+\cdots+2^1+2^0$

因为$H_1=1$，所以$H_n=2^{n-1}+2^{n-2}+\cdots+2^1+2^0=2^n-1$。

故总盘数为 n 的 Hanoi 塔的移动次数是2^n-1。

27．**设输入序列为 a，b，c，d，试写出借助一个栈可得到的两个输出序列和两个不能得到的输出序列。**

【答案】n 个元素的排列有 $n!$种，但借助栈结构，n 个入栈元素只可得到 $1/(n+1)((2n)!/(n!*n!))$种出栈序列，这个数小于 $n!$。本题 4 个元素，可有 14 种出栈序列，abcd 和 dcba 就是其中两种；有10（4!-14=10）种排列得不到，其中 dabc 和 adbc 是不可能得到的两种。

五、算法设计题

28．**设以整数序列 a1，a2，a3，a4 作为栈 S 的输入，利用 push 和 pop 操作，写出所有可能的输出并编程实现算法。**

【答案】设a1，a2，…，ai是已出栈的编号，b1，b2，…，bj是已进找的编号，c1，c2，…，ck则是尚未入栈的编号。现有两种可能，一是将c1进栈，另一个则是将b1出栈，因此可采用状态递归的方法，从初始状态出发，逐步递归，当所有编号出栈时，所得的就是一种可能的输出序列，由此得到所有的可能输出序列。

```
TYPE Tstack=RECORD
       number:integer;
       next:^Tstack
   END;
ptostack=^Tstack;
PROCEDURE Main(VAR p:ptostack; VAR total:integer);
BEGIN
    p:=NIL;
    IF total>=1 THEN
      BEGIN
          push(1,p);
          process(1,'')
      END;
    Clear_stack(p);
END;
PROCEDURE Push(n:integer; VAR p:ptostack);
VAR tmp:ptostack;
BEGIN
    new(temp);
    temp^.number:=n;
    temp^.next:=p;
    p:=temp
END;
PROCEDURE Pop(VAR p:ptostack):integer;
VAR temp:ptostack;
```

```
BEGIN
    temp:=p^.next;
    pop:=p^.nember;
    dispose(p);
    p:=temp
END;
PROCEDURE process(position:integer; path:STRING);
VAR m:integer;
BEGIN
    IF position<total THEN      {编号进栈时的递归}
        BEGIN
            push(position+1,p);
            process(position+1,path);
            pop(p)
        END;
    IF p<>NIL THEN      {编号出栈时的递归}
        BEGIN
            m:=pop(p);
            process(position,path+"+IntToStr(m));
            push(m,p)
       END;
    IF (position=total) and (p=NIL) THEN      {输出一种可能的编号方案}
        writeln(path);
END;
PROCEDURE Clear_stack(p:ptostack);
VAR temp:ptostack;
BEGIN
    WHILE p<>NIL DO
        BEGIN
            temp:=p;
            p:=p^.next;
            dispose(temp)
        END
END;
```

29. **一个长度为$L(L\geqslant 1)$的升序序列 S,处在第$\lceil L/2\rceil$个位置的数称为 S 的中位数。例如，若序列**S1=(11,13,15,17,19)**，则 S1 的中位数是 15。两个序列的中位数是含它们所有元素的升序序列的中位数。例如，若**S2=(2,4,6,8,20)**，则**S1**和**S2**的中位数是 11。现在有两个等长升序序列 A 和 B，试设计一个在时间和空间两方面都尽可能高效的算法，找出两个序列 A 和 B 的中位数。要求：**

(1)给出算法的基本设计思想。

(2)根据设计思想，采用 C 或C++或 jAVA 语言描述算法，关键之处给出注释。

(3)说明你所设计算法的时间复杂度和空间复杂度。

【答案】(1)算法的基本设计思想如下。

分别求出序列 A 和 B 的中位数，设为 a 和 b 求序列 A 和 B 的中位数过程如下。

①若 a=b 则 a 或 b 即为所求中位数，算法结束。

②若 a<b，则舍弃序列 A 中较小的一半，同时舍弃序列 B 中较大的一半，要求舍弃的长度相等。

③若 a>b,则舍弃序列 A 中较大的一半，同时舍弃序列 B 中较小的一半，要求舍弃的长度相等。

在保留的两个升序序列中，重复过程①、②、③，直到两个序列中只含一个元素时为止，较小者即为所求的中位数。

(2)算法的实现如下：

```
int M_Search (int A[],int B[],int n){
  int   s1 = 0, d1 = n - 1, m1, s2 = 1, d2 = n - 1,m2;
    //分别表示序列 A 和 B 的首位数、末位数和中位数

    while( s1!= d1||s2! 1 = d2){
    m1 = (s1 + d1)/2;
    m2 = (s2 + d2)/2;
    if (A[m1] == B[m2])
    return A[m1];                  //满足条件 1)
    if(A[m1]<B[m2]){               //满足条件 2)
    if((s1 + d1) % 2 == 0){        //若元素个数为奇数
    s1 = m1;                       //舍弃 A 中间点以前的部分且保留中间点
    d2 = m2;                       //舍弃 B 中间点以后的部分且保留中间点
       }
    else{                          //元素个数为偶数
    s1 = m1 + 1;                   //舍弃 A 中间点及中间点以前部分
    d2 = m2;                       //舍弃 B 中间点以后部分且保留中间点
       }
    }
    else{                          //满足条件 3)
    if( (s1 + d1) % 2 == 0){       //若元素个数为奇数
    d1 = m1;                       //舍弃 A 中间点以后的部分且保留中间点
    s2 = m2;                       //舍弃 B 中间点以前的部分且保留中间点
    }
    else{                          //元素个数为偶数
    d1 = m1 + 1;                   //舍弃 A 中间点以后部分且保留中间点
    s2 = m2;                       //舍弃 B 中间点及中间点以前部分
          }
       }
}
return A[s1]<B[s2]? A[s1]:B[s2];
}
```

(3)算法的时间复杂度为$O(\log_2 n)$，空间复杂度为$O(1)$。

30．已知一棵二叉树的先序遍历序列和中序遍历序列分别存于两个一维数组中，试编写算法来建立该二叉树的二叉链表。

【答案】算法描述如下：

```
PROCDURE BuildTree(front1,rear1,fornt2,rear2:integer;VAR root:bitreptr);
{front1,rear1,fornt2,rear2 分别指向前序、中序序列在数组 a1 和 a2 中始末位置}
BEGIN
   IF front1>rear1
     THEN t:=NIL
     ELSE BEGIN
              new(t);
              root^.data:=a1[front1];      {建根}
              point:=front2;          {在中序中找根}
              WHILE a2[point]<>a1[front1] DO point:=point+1;
              {以下建立左子树}
              BuildTree(fornt1+1,point - front2+front1,fornt2,point - 1,root^.lchild);
              {以下建立右子树}
              BuildTree(point - fornt2+front1+1,rear1,point+1,rear2,root^.rchild)
END;
```

31．**已知一个线性链表的数据元素为整型，试编写一个法将该链表分割成两个单循环链表，其中一个仅包含偶数元素，另一个仅包含奇数元素。**

【答案】

```
void decompose(LinkList &L, LinkList &La, LinkList &Lb)
{  //  含头结点
   p=L->next;
   La=L;  La->next=La;
   Lb=(LinkList)malloc(sizeof(LNode));
   if (Lb==NULL)  exit(OVERFLOW);
   Lb->next=Lb;
   while(p!=NULL)  {
      q=p->next;
      if(p->data%2==0)  {   // 偶数
         p->next=La->next;
         La->next=p;
         }
      else {
         p->next=Lb->next;
         Lb->next=p;
         }
      p=q;
      }
   }
```

32．**已知 Q 是一个非空队列，S 是一个空栈。仅用队列和栈的 ADT 函数和少量工作变量，使用 Pascal 或 C 语言编写一个算法，将队列 Q 中的所有元素逆置。**

栈的 ADT 函数有：

makeEmpty(s:stack);置空栈

push(s:stack;value:datatype);新元素 value 进栈

pop(s:stack):datatype;出栈，返回栈顶值

isEmpty(s:stack):boolean;判栈空否

队列的 ADT 函数有：

enQueue(q:queue;value:datatype);元素 value 进队

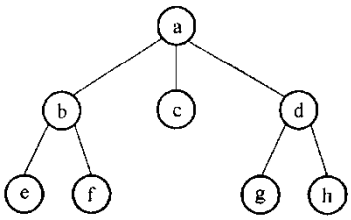
deQueue(q:queue):datatype;出队列，返回队头值

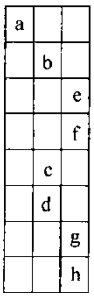
isEmpty(q:queue):boolean;判队列空否

【答案】算法描述如下：

```
PROCEDURE Reserval(VAR Q:queue);
VAR s:stack;
    i:integer;
BEGIN
   makeEmpty(s);
   FOR i:=1 TO maxsize DO
     IF isEmpty(Q) THEN push(s,deQueue(Q));
   FOR i:=1 TO maxsize DO
     IF isEmpty(s) THEN enQueue(Q,pop(s))
END;
```

33．**设树上每个结点所包含的数据元素为一个字母，并且以孩子兄弟表示法为树的存储结构，试写一个按图所示的凹入表示式显示一棵树的算法。**





【答案】具体算法如下。

```
//按凹入表示式显示一棵树,level 为层次数,根结点层数为 1
void print_tree(Tree T, int level = 1){
    if(T!=NULL){                    //只显示非空树
        cout<<endl;                 //新树换行
        for(int i=0; i<level-1; i++)
            cout<<" ";              //在 level 列输出结点
        cout<<T->data;              //输出结点
        //显示子树森林
        print_tree(T->lchild,level+1);
        //显示除第一棵树而外其他树形成的森林
        print_tree(T->rchild,level);
    }
}
```

34．**采用顺序结构存储串，设计一个算法求串 s 中出现的第一个最长重复子串的下标和长度。**

【答案】采用简单匹配算法的思路，先给最长重复子串的下标 maxi 和长度 maxlen 均赋值为 0。设 s="$a_1...a_n$"，扫描通过串 s，对于当前字符a_i，判定其后是否有相同的字符，若有记为a_j，再判定a_{i+1}是否等于a_{j+1}，a_{i+2}是否等于a_{j+2}，…，直至找到一个不同的字符为止，即找到一个重复出现的子串，把其下标 i 与长度 len 记下来，将 len 与 maxlen 相比较，保留较长的子串 maxi 和 maxlen。再从a_{j+len}之后查找重复子串。然后对于a_{i+1}之后的字符采用上述函数，最后的 maxi 与 maxlen 即记录下最长重复子串的下标与长度，将其复制到串 t 中并返回。对应算法如下:

```
SqString Maxsubstr(SqString s)
{   int maxi=0,maxlen=0,len,i=0,j,k;
    SqString t;
    while (i<s.length)
    {   j=i+1;
        while (j<s.length)
        {   if (s.data[i]==s.data[j])      //找一个子串,其序号为 i,长度为 len
            {   len=1;
                for(k=1;s.data[i+k]==s.data[j+k];k++)
                    len++;
                if (len>maxlen)            //将较大长度者赋给 maxi 与 maxlen
                {   maxi=i;
                    maxlen=len;
                }
                j+=len;
                }
                else j++;
            }
            i++;                           //继续扫描第 i 字符之后的字符
        }
```

```
        t.length=maxlen;                    //将最长重复子串赋给 t
        for (i=0;i<maxlen;i++)
            t.data[i]=s.data[maxi+i];
        return t;
    }
```

35．**已知哈希函数** H(key)=2key MOD 11，**用线性探测法处理冲突。试在 0～10 的哈希地址空间中对关键字序列**{6,8,10,17,20,23,53,41,54,57}**构造哈希表，并求等概率情况下查找成功时的平均查找长度。**

【答案】由给定的哈希函数计算如下：

H(6)=2×6 MOD 11=1
H(8)=2×8 MOD 11=5
H(10)=2×10 MOD 11=9
H(17)=2×17 MOD 11=1　　冲突
$H_1(17)=(1+d_1)$ MOD 11=(1+1) MOD 11=2
H(20)=2×20 MOD 11=7
H(23)=2×23 MOD 11=2　　冲突
$H_1(23)=(2+d_1)$ MOD 11=(2+1) MOD 11=3
H(53)=2×53 MOD 11=7　　冲突
$H_1(53)=(7+d_1)$ MOD 11=(7+1) MOD 11=8
H(41)=2×41 MOD 11=5　　冲突
$H_1(41)=(5+d_1)$ MOD 11=(5+1) MOD 11=6
H(54)=2×54 MOD 11=9　　冲突
$H_1(54)=(9+d_1)$ MOD 11=(9+1) MOD 11=10
H(57)=2×57 MOD 11=4

对应的哈希表如表所示。

下标	0	1	2	3	4	5	6	7	8	9	10
关键字	0	6	17	23	57	8	41	20	53	10	54
探测次数	0	1	2	2	1	1	2	1	2	1	2

则在等概率情况下：$ASL_{succ}=\frac{5\times1+5\times2}{10}=1.5$。

与从位置 1 开始向后直至位置 3 的关键字相比较，由于位置 3 的关键字为空，所以不再向后比较，共比较 3 次，其他的类推即可得到，所以有：

$$ASL_{不成功}=(1+3+2+1+9+8+7+6+5+4+3+2+1)/13=4。$$

【冲刺】2024 年广西大学 085410 人工智能《828 程序设计与数据结构之数据结构》考研仿真模拟 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．**对于有 n 个顶点 e 条边的有向图，求最短路径的 Dijkstra 算法的时间复杂度为_____。**

A.O(n)
B.O(n+e)
C.$O(n^2)$
D.O(ne)
【答案】C

2．**一棵完全二叉树上有 768 个节点，则该二叉树中叶子节点的个数是_____。**

A.257
B.258
C.384
D.385
【答案】C
【解析】对于完全二叉树，n=768 为偶数，所以 m=1，而$n_0=n_2+1$，$n=n_0+n_1+n_2$，即有：$n_0=n/2=384$。本题答案为 C

3．**对于一个线性表既要求能够进行较快速的插入和删除，又要求存储结构能反映数据之间的逻辑关系，则应该用_____。**

A.顺序存储方式
B.链式存储方式
C.散列存储方式
D.以上均可以
【答案】B
【解析】散列存储方式是集合的一种表示方法，元素之间没有“逻辑关系”。B 是正确选择。

4．**用不带头结点的单链表存储队列时，其头指针指向队头结点，其尾指针指向队尾结点，则在进行删除操作时_____。**

A.仅修改头指针
B.仅修改尾指针
C.头、尾指针都要修改
D.头、尾指针可能都要修改
【答案】D
【解析】当链表中不止一个元素时，只需要修改头指针；当链表中只有一个元素时，就需要将头指针和尾指针都赋值为空。

5．**需要预分较大空间，插入和删除不需要移动元素的线性表，其存储结构是_____**

A.单链表

B.静态链表

C.线性链表

D.顺序存储结构

【答案】B

6. **非空的循环单链表 L 的尾节点(由 p 所指向)满足_____。**

A. p->next==NULL

B.p==NULL

C. p->next==L

D.p==L

【答案】C

【解析】在非空的循环单链表 L 中，尾节点*p的 next 域指向头节点，即有p->next==L。

7. **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 个元素出列的顺序是 b,d,c,f,e,a,g,则栈 S 的容量至少是_____。**

A.1

B.2

C.3

D.4

【答案】C

【解析】由于队列不改变进出序列，通过一个栈将 a,b,c，d,e,f，g 序列变为 b,d，c，f，e，a,g 序列时栈空间至少多大。其过程如下表所示，从中可以看到，栈中最多有 3 个元素，即栈大小至少为 3。

操作 1	栈 S	出栈序列
a 进栈	a	
b 进栈	ab	
b 出栈	a	b
c 进栈	ac	b
d 进栈	acd	b
d 出栈	ac	bd
c 出栈	a	bdc
e 进栈	ae	bdc
f 进栈	aef	bdc
f 出栈	ae	bdcf
e 出栈	a	bdcfe
a 出栈		bdcfea
g 进栈	g	bdcfea
g 出栈		bdcfeag

表

8. **对 8 个元素的线性表进行快速排序，在最好的情况下，元素间的比较次数为_____。**

A.7 次

B.8 次

C.12 次

D.13 次

【答案】D

【解析】对 8 个元素排序的最好情况是：第一次找到的元素将原表分成长度为 3 和 4 的表，用到 7 次比较 第二层，对于长度为 3 的表，最少需要 2 次，对于长度为 4 的表，继续分成长度为 1 和长度为 2 的表，最少需要 3 次 第三层，需要对长度为 2 的表进行排序，最少需要 1 次比较，所以总共需要 7+2+3+1=13 次比较。

9. **一般情况下，将递归算法转换成等价的非递归算法应该设置_____。**

A.堆栈

B.队列

C.堆栈或队列

D.数组

【答案】A

二、填空题

10. **直接选择排序算法在最好的情况下所做的交换元素的次数为_____。**

【答案】0

【解析】直接选择排序每次得到一个最大(最小)的元素，如果序列本身就是有序的，则不需要进行交换。

11. **已知一棵完全二叉树中共有 768 个结点，则该树中共有_____个叶子结点。**

【答案】384

12. **n 个顶点的连通无向图，其边的条数至少为_____。**

【答案】n-1

13. **若一组记录的排序码为**(46, 79, 56, 38, 40, 84)**，利用堆排序建立的初始堆是_____。(注:堆顶元素取最大值。)**

【答案】 84, 79, 56, 38, 40, 46

14. **对于顺序表 s,将其初始化为空串的操作是_____。**

【答案】 s.length=0

15. **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)。

三、判断题

16. **当各边上权值均相等时，广度优先搜索算法可以用来解决单源点最短路径问题。_____**

【答案】√

【解析】当图中所有边上的权值相等时，两个顶点 i、j 的距离即为它们之间的边数，因此在这种情况下，广度优先遍历算法可以解决单源点最短路问题。

17．**数据的逻辑结构与各数据元素在计算机中如何存储有关。**_____

【答案】×

【解析】数据的逻辑结构与存储结构无关

18．**稀疏矩阵压缩存储后，必会失去随机存取功能。**_____

【答案】√

【解析】要解决这个问题，就必须深入了解随机存取的定义和稀疏矩阵压缩存储时采用的存储结构。如果存取结构中任一元素的时间是相等的，那么我们把具有这种特点的存储结构称为随机存取结构。而对稀疏矩阵进行压缩存储，所用存储结构是三元组表或十字链表。十字链表是链表结构，因此不能随机存取。对于使用三元组顺序表存储的矩阵，现假定要获取第 i 行第 j 列的元素，必须扫描该顺序表，逐个将其中的元素的行、列下标分别与 i、j 比较。显然取得排在顺序表前部的元素与取得排在后部的元素所耗费的时间是不同的。所以三元组顺序表也不是一个顺序存储的结构。

19．**数据的逻辑结构是按使用需要而建立的，与实际的存储形式无关。**_____

【答案】×

【解析】数据逻辑结构是指数据元素间的相互关系，并不涉及数据元素在计算机存储设备中的具体存储方式，是独立于计算机的。

20．**文件系统采用索引结构是为了节省存储空间。**_____

【答案】×

21．**m 阶 B-树中所有叶子节点都在同一层。**_____

【答案】√

四、应用题

22．**仔细阅读下面的 Pascal 过程，并回答有关问题。**

```
PROCEDURE UnknownName(VAR A:ARRAY[1..500] OF integer; n:integer);
VAR i,j,x:integer;
    b:boolean;
BEGIN
    b:true;
    i:=1;
    WHILE (i<n) AND b DO
      BEGIN
        b:=false;
        FOR j:=1 TO  ①  DO
          IF  ②
            THEN BEGIN
                   x:=A[j];
                   A[j]:=A[j+1];
                   A[j+1]:=x;
                    ③
                 END;
        i:=i+1;
      END
END;
```

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/835303032001011230