学府考研
中国优秀高端教育品牌

理工社®

最新版
计算机考研
历年真题权威解析
及名校考题精练

◎主编　杨航空

◎策划　学府考研计算机命题研究组

- 或巧或拙，各存其解
- 所得所失，只在本心
- 书籍＋在线——研途好伙伴

赠

北京理工大学出版社
BEIJING INSTITUTE OF TECHNOLOGY PRESS

学府考研
中国优秀高端教育品牌

理工社®

最新版
计算机考研
历年真题权威解析
及名校考题精练

◎主编　杨航空

◎策划　学府考研计算机命题研究组

北京理工大学出版社
BEIJING INSTITUTE OF TECHNOLOGY PRESS

图书在版编目(CIP)数据

计算机考研历年真题权威解析及名校考题精练/杨航空主编. —北京:北京理工大学出版社,2015.11
ISBN 978-7-5682-1392-9

Ⅰ. ①计…　Ⅱ. ①杨…　Ⅲ. ①电子计算机—研究生—入学考试—习题集　Ⅳ. ①TP3-44

中国版本图书馆 CIP 数据核字(2015)第 244579 号

出版发行 /北京理工大学出版社有限责任公司
社　　址 /北京市海淀区中关村南大街 5 号
邮　　编 /100081
电　　话 /(010)68914775(总编室)
　　(010)82562903(教材售后服务热线)
　　(010)68948351(其他图书服务热线)
网　　址 /http://www.bitpress.com.cn
经　　销 /全国各地新华书店
印　　刷 /三河市佳星印装有限公司
开　　本 /787 毫米×1092 毫米　1/16
印　　张 /19
字　　数 /500 千字
版　　次 /2015 年 11 月第 1 版　2015 年 11 月第 1 次印刷
定　　价 /37.80 元

责任编辑 / 张慧峰
文案编辑 / 张慧峰
责任校对 / 周瑞红
责任印制 / 边心超

前言 Preface

学府考研在考研辅导领域有着极好的口碑。学府考研·计算机系列丛书(《2016 计算机考研通关必做 1200 题》等)自问世之后,受到了广大计算机考研同学的诸多赞许及好评,正在帮助着越来越多的同学考上心仪的计算机名校。

在《2016 计算机考研通关必做 1200 题》系列丛书中,经常会使用近几年的名校计算机考研真题,尽管这些题目都是各个学校出题老师多年教学经验的结晶,其中被同类考研辅导书采用的经典题目众多,但尚缺一本习题书来对这些经典题目进行整理归类,供考研学子进行实战提升。

从 2012 年开始,越来越多的学校脱离统考,开始进行自主命题,而市面上还没有出现一本习题书是专门针对自主命题编写的。于是,我们有了这个想法,希望可以将各大名校近几年的真题进行精选,编写出一本适合非统考学生的习题书籍。

编者从 2012 年年底开始筹划本书,几乎收集了市面上全部的非统考真题。在收集题目的过程中,我们发现这些题目重题率相当高,这点应该让考研学子感到欣喜,因为这说明各个学校对知识点的考查方式无非就几种,只要将其中的典型题目精选出来,考生就可以达到举一反三的学习效果。

在对选题进行解析的过程中,我们还是按照系列丛书所遵循的通俗易懂的原则,进行详尽讲解和扩展。建议读者使用本书的方法是:第一遍,自测做题,只需要看错题的解析;第二遍,阅读所有题目解析,即使对曾经做对的题目,也要看一下解析,对比一下你的解法和书中解法哪个更优,不少题目还给出了扩展知识点和扩展题目。

本书特点如下:

(1)全真试题:本书中所有习题均选自计算机名校近几年的考研真题,并且对题目没有进行任何修改,即使题目有不当之处也予以保留,只在解析中进行讨论。

(2)统考为主线:对各自主命题院校的多年考研真题分析发现,自主命题院校出题依然还是以统考大纲为主线,所以本书中对 2009—2015 年 408 统考真题中的经典真题也做了整理归纳,方便广大计算机考研同学学习整理。

(3)举一反三:虽然近几年各自主命题院校的真题量非常大,但存在不少重复类型的题目,如果要考生一一做的话,必然会拖后考生整个考研复习进度,因此我们对每种类型的题目都选出其中的经典代表,然后进行详细解析,如有必要还会在解析中附上扩展题目。用最短的

时间和最少题量让考生达到举一反三的做题效果。

(3)视频讲解:本书中对同学们较难理解或历年出题频率较高的典型试题进行了视频讲解,对考生常常遇到的疑问、易混淆的知识点进行了总结,并将其融入相应的习题讲解中。

(4)实时答疑:考生对书中遇到的任何疑问,都可通过考试点论坛(bbs. kaoshidian. com)与本书作者和学长们进行交流,目前我们的答疑速度在同类网站中可以称得上是"实时",我们承诺考生的疑问均能在两天内得到解决。

由于本书是第一次出版,其中肯定还存在很多不足之处,希望广大读者多给予批评及建议。

杨航空

目录

Content

第一部分　数据结构

第二部分　计算机组成原理

第三部分 计算机操作系统

第四部分　计算机网络

第一部分
数据结构

第一章　线性表

考点概述

(1)线性表的定义和基本操作

(2)线性表的实现

1)顺序存储

2)链式存储

3)线性表的应用

考试题型

自2009年计算机全国统考至今,线性表的内容除了作为后续章节的基础知识外,还通常在试卷最后以综合题的形式对线性表的应用进行考查。

就近几年国内各自主命题院校计算机考研试卷可以看出,虽各院校试卷题型略有不同,但线性表的内容在选择题、填空题、判断题、算法题及综合题中均有出现。

复习建议

线性表这一章的知识点不多,但要做到深刻理解,能够应用相关知识点解决实际问题。链表上插入、删除结点时的指针操作是选择题的一个常考点,诸如双向链表等一些相对复杂的链表的操作也是可以出现在综合应用题当中的。

2016预测

经对历年考研真题中本章知识点的考查形式及考查内容分析可知,本章考点主要集中在线性表的应用。其表现在两个方面:一是一维数组元素的存取移位操作;二是单链表中数据元素的查找操作。要求均为设计算法思想并用C或C++进行描述以及分析该算法求解的时间复杂度和空间复杂度。

408统考历年真题权威解析

例1　(2009年第42题)已知一个带有头结点的单链表,结点结构为

data	link

假设该链表只给出了头指针list。在不改变链表的前提下,请设计一个尽可能高效的算法,查找链表中倒数第k个位置上的结点(k为正整数)。若查找成功,输出该结点的data值,并返回1;否则,只返回0。要求:

(1)描述算法的基本设计思想;

(2)描述算法的详细实现步骤;

(3)根据设计思想和实现步骤,采用程序设计语言描述算法(使用 C 或 C ++,或 Java 语言实现),关键之处请给出简要注释。

考点 本题主要考查单链表结点遍历算法。

答案与解析

(1)算法基本思想如下:

从头至尾遍历单链表,并用指针 p 指向当前结点的前 k 个结点。当遍历到链表的最后一个结点时,指针 p 所指向的结点即为所查找的结点。

(2)详细实现步骤:

增加两个指针变量和一个整型变量:

指针 p1 指向当前遍历的结点;指针 p 指向 p1 所指向结点的前 k 个结点,如果 p1 之前没有 k 个结点,那么 p 指向表头结点。整型变量 i 表示当前遍历了多少结点,当 i > k 时,指针 p 随着每次遍历,向后移动一个位置。当遍历完成时,p 或者指向表头结点,或者指向链表中倒数第 k 个位置上的结点。

(3)算法描述:

```
int Locate Element(linklist list,int k)){
p1 =list ->link;
i =1;
while(p1){
p1 =p1 ->link;i ++;
if(i >k)
p =p ->next;  //如果 i >k,则 p 也往后移
}
if(p ==list)  return 0;  //说明链表没有 k 个结点
else{
     printf("% d\n",p ->data);
     return 1;}
}
```

名师点拨 本题属于比较基础的算法题。下面,给出详细的解答过程。

(1)算法的基本思想:定义两个指针 p 和 q,开始时均指向链表表头结点的下一个结点位置;令 p 指针沿着链表移动,并且启动计数器 count 开始计数(初始时,count = 0);当 p 指针移动到链表结尾时,q 指针所指向的结点即倒数第 k 个结点。

(2)算法的详细实现步骤如下:

(A)令 p 和 q 均指向链表表头结点的下一个结点,令 count = 0;

(B)若 p 为空,即该单链表只有一个表头结点,则转(E);

(C)若 count = k,则 q 指向下一个结点;否则,count = count + 1;

(D)p 指向下一个结点,转(B);

(E)若 count = k,则查找成功,输出该结点的 data 域的值,并返回 1;否则,查找失败,返回 0。

例 2 (2010 年第 42 题)设将 $n(n \geq 1)$ 个整数存放到一维数组 R 中,试设计一个在时间和空间两方面尽可能有效的算法,将 R 中保存的序列循环左移 $p(0 < p < n)$ 个位置,即将 R 中的数据由 $(X_1, X_2, \cdots, X_n)$ 变换为 $(X_p, X_{p+1}, X_n, X_1, \cdots, X_{p-1})$。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 语言表述算法,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。

考点 本题主要考查线性存储中位数遍历算法。

答案与解析

(1)基本设计思想:

将数组$\{a_1,a_2,a_3,\cdots,a_p,a_{p+1},\cdots,a_n\}$先进行全部的逆转,然后分别对$\{a_{p+1},\cdots,a_{n-1},a_n\}$和$\{a_1,a_2,a_3,\cdots,a_p\}$进行再次逆转。

(2)算法描述:

```
void sift_left(int a[],int n,int p){
Reverse(a,0,n-1);  //移动了3n/2次数据
Reverse(a,0,n-p-1);  //移动了3(n-p)/2次数据
Reverse(a,n-p,n-1);}  //移动了3p/2次数据
void Reverse(int a[],int left,int right){
int n=right-left+1;  //设置一个辅助空间
if(n<=1)return 0;  //数组为空
for(int i=0;i<n/2;i++){  //进行逆转
        int temp=a[i];
        a[i-1]=a[n-i+1];
        a[n-i+1]=temp;  }
}
```

(3)算法的时间复杂度和空间复杂度:

算法的时间复杂度为$O(n)$;

算法的空间复杂度为$O(1)$。

名师点拨 建立一个可以放下p个整数的辅助队列,使数组R中的前p个整数依次进入辅助队列,将R中后面的$n-p$个整数依次前移p个位置,使辅助队列中的数据依次出队,依次放入R中第$n-p$个整数开始的位置。

例3 (2011年第42题)一个长度为$L(L\geqslant 1)$的升序序列S,处在第$L/2$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包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。

考点 本题主要考查顺序表结点遍历算法。

答案与解析

此题考察的知识点是基本算法的灵活运用。

(1)算法的基本设计思想:

1)比较笨的方法:

将两个升序序列归并排序,然后求其中位数,时间复杂度是$O(n)$,空间复杂度$O(n)$。

2)高效的方法:

分别求两个升序序列A和B的中位数,设为a和b。

如果$a=b$,则a或者b即为所求的中位数。

原因:如果将两序列归并排序,则最终序列中,排在序列 a、b 前边的元素为先前两序列中排在 a 和 b 前边的元素;排在序列 a、b 后边的元素为先前两序列排在 a 和 b 后边的元素。所以序列 a、b 一定位于最终序列的中间,又因为 $a=b$,显然 a 就是中位数。

如果 $a\neq b$(假设 $a<b$),中位数只能出现在(a,b)范围内。

原因:同样可以用归并排序后的序列来验证,归并排序后必然有形如…a…b…的序列出现,中位数必然出现在(a,b)范围内。因此可以做如下处理:舍弃 a 所在序列 A 之中比较小的一半,同时舍弃 b 所在序列 B 之中比较大的一半。在保留的两个升序序列中求出新的中位数 a 和 b,重复上述过程,直到两个序列只含一个元素为止,则较小者即为所求的中位数。

(2)算法实现(高效方法):

```
int Search(int A[],int B[],int n){
int s1,e1,mid1,s2,e2,mid2;
s1=0;
e1=n-1;
s2=1;
e2=n-1;
while(s1!=e1||s2!=e2){
    mid1=(s1+e1)/2;
mid2=(s2+e2)/2;
    if(A[mid1]==B[mid2])
      return A[mid1];
if(A[mid1]<B[mid2]){  //分别考虑奇数和偶数,保持两个子数组元素个数相等
if((s1+e1)%2==0){//若元素个数为奇数
s1=mid1;    //舍弃A中间点以前部分且保留中间点
e2=mid2;    //舍弃B中间点以后部分且保留中间点
}
else{  //若元素个数为偶数
s1=mid1+1;    //舍弃A中间点以前部分且不保留中间点
e2=mid2;      //舍弃B中间点以后部分且保留中间点
}
}
else{
        if((s1+e1)%2==0){  //若元素个数为奇数
            e1=mid1;      //舍弃A中间点以后部分且保留中间点
s2=mid2;              //舍弃B中间点以前部分且保留中间点
}
        else{  //若元素个数为偶数
            e1=mid1+1;//舍弃A中间点以后部分且不保留中间点
    s2=mid2;//舍弃B中间点以前部分且保留中间点
}
}
return(A[s1]<B[s2]? A[s1]:B[s2]);
```

(3)上述所给算法的时间和空间复杂度分别是 $O(\log_2 n)$ 和 $O(1)$。

因为每次总的元素个数变为原来的一半,所以有:

第一次:元素个数为 $n/2 = n/(2^1)$;

第二次:元素个数为 $n/4 = n/(2^2)$;

……

第 k 次:元素个数为 $n/(2^k)$

最后元素个数为 2

则有 $n/(2^k) = 2$,解得 $k = \log_2 n - 1$。

因此:时间复杂度为 $O(\log_2 n)$,而空间复杂度从上述程序中可看出为 $O(1)$。

名师点拨 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:

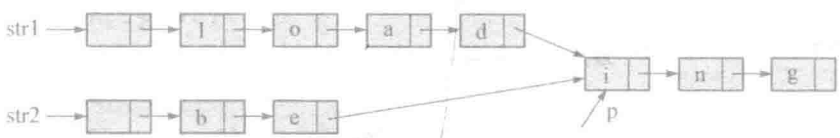
1)若 $a = b$,则 a 或 b 即为所求中位数,算法结束;

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;

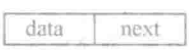
3)若 $a > b$,则舍弃序列 A 中较大的一半,同时舍弃序列 B 中较小的一半,要求舍弃的长度相等;

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为止,较小者即为所求的中位数。

例 4 (2012 年第 42 题)假定采用带头结点的单链表,如果有共同的后缀时,则可以共享相同的后缀存储空间。例如,"loading"和"being"如下图所示。



设 str1 和 str2 分别指向两个单词所指单链表的头结点,链表结点结构为:



设计一个时间上尽可能高效的算法,找出由 str1 和 str2 所指向两个链表共同后缀的起始位置(如图中字符 i 所在结点的位置 p)。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。

考点 本题主要考查链表存储结构。

答案与解析

(1)算法的基本设计思想:

1)分别求出 str1 和 str2 所指的两个链表长度 m 和 n;

2)将两个链表以表尾对齐:令指针 p,q 分别指向 str1 和 str2 的头结点,若 $m > n$,则使 p 指向 str1 链表中的第 $m - n + 1$ 个结点;若 $m < n$,则使 q 指向 str2 链表中的第 $n - m + 1$ 个结点。若 p 和 q 指向同一结点,则该点即为所求的共同后缀的起始位置。

(2)算法实现:

```
typedef struct Node{
        char data;
        struct Node * next;
}SNODE;
```

```
SNODE * findlist(SNODE * str1,SNODE * str2){
int m,n;
SNODE * p, * q;
m = listlen(str1);
n = listlen(str2);
for(p = str1;m > n;m --)
      p = p -> next;
for(p = str2;m < n;n --)
      q = q -> next;
while(p -> next ! = NULL && p -> next ! = q -> next){
       p = p -> next;
       q = q -> next;
}
return  p -> next;  }
int listlen(SNODE * head){
  int len = 0;
  while(head -> next ! = NULL){
      len ++;
      head = head -> next;
}
return  len;
}
```

(3)算法的时间复杂度为 $O(m+n)$。

名师点拨　顺序遍历两个链表到尾结点时,并不能保证两个链表同时到达尾结点。这是因为两个链表的长度不同。假设一个链表比另一个链表长 k 个结点,我们先在长链表上遍历 k 个结点,之后同步遍历两个链表,这样就能保证它们同时到达最后一个结点。由于两个链表从第一个公共结点到链表的尾结点都是重合的,所以它们肯定同时到达第一个公共结点。算法的基本设计思想:

1)分别求出 str1 和 str2 所指的两个链表的长度 m 和 n;

2)将两个链表以表尾对齐:令指针 p、q 分别指向 str1 和 str2 的头结点,若 $m \geqslant n$,则使 p 指向 str1 链表中的第 $m-n+1$ 个结点;若 $m < n$,则使 q 指向 str2 链表中的第 $n-m+1$ 个结点,即使指针 p 和 q 所指的结点到表尾的长度相等。

3)反复将指针 p 和 q 同步向后移动,并判断它们是否指向同一结点。若 p 和 q 指向同一结点,则该点即为所求的共同后缀的起始位置。

例 5　(2013 年第 41 题)已知一个整数序列 $A=(a_0,a_1,\cdots,a_{n-1})$,其中 $0 \leqslant a_i < n(0 \leqslant i < n)$。若存在 $a_{p1}=a_{p2}=\cdots=a_{pm}=x$ 且 $m > n/2(0 \leqslant pk < n, 1 \leqslant k \leqslant m)$,则称 x 为 A 的主元素。例如 $A=(0,5,5,3,5,7,5,5)$,则 5 为主元素;又如 $A=(0,5,5,3,5,1,5,7)$,则 A 中没有主元素。假设 A 中的 n 个元素保存在一个一维数组中,请设计一个尽可能高效的算法,找出 A 的主元素。若存在主元素,则输出该元素;否则输出 -1。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。

考点 本题主要考查一维数组中指定数据元素序列的排序。

答案与解析

(1)给出算法的基本设计思想:

先从前向后扫描数组元素,标记出一个可能成为主元素的元素 Num。然后重新计数,确认 Num 是否是主元素。

算法可分为以下两步:

1)选取候选的主元素:依次扫描所给数组中的每个整数,将第一个遇到的整数 Num 保存到 c 中,记录 Num 的出现次数为 1;若遇到的下一个整数仍等于 Num,则计数加 1,否则计数减 1;当计数减到 0 时,将遇到的下一个整数保存到 c 中,计数重新记为 1,开始新一轮计数,即从当前位置开始重复上述过程,直到扫描完全部数组元素。

2)判断 c 中元素是否是真正的主元素:再次扫描该数组,统计 c 中元素出现的次数,若大于 $n/2$,则为主元素;否则,序列中不存在主元素。

(2)算法实现:

```
int  Majority(int A[],int n){
int  i,c,count =1;          //c 用来保存候选主元素,count 用来计数
c =A[0];                 //设置 A[0]为候选主元素
for(i =1;i <n;i ++)     //查找候选主元素
        if(A[i] ==c)
          count ++;         //对 A 中的候选主元素计数
        else{
          if(count >0)       //处理不是候选主元素的情况
             count --;
          else               //更换候选主元素,重新计数
            {c =A[i];
             count =1;
            }
        }
if(count >0)
for(i =count =0;i <n;i ++)  //统计候选主元素的实际出现次数
        if(A[i] ==c)
          count ++;
        if(count >n/2)  return  c;     //确认候选主元素
        else  return  -1;              //不存在主元素
  }
```

(3)算法的时间复杂度为 $O(n)$,空间复杂度为 $O(1)$。

名师点拨 除此方法之外,本题还可以通过计数排序的思想来实现,具体算法如下:

```
int  Majority1(int A[],int  n){
  int  k,*p,max;
    p =(int *)malloc(sizeof(int) *n);          //申请辅助计数数组
    for(k =0;k <n;k ++){  p[k] =0;             //计数数组清 0
        max =0;
```