天勤计算机考研高分笔记系列

天勤论坛

2014NIAN
JISUANJI ZHUANYE JICHU
ZONGHE LINIAN TONGKAO
ZHENTI JI SILU FENXI

2014年
计算机专业基础
综合历年统考真题及思路分析

周伟　王征勇　刘泱　主编
清航考研培训教学组　审核

第2版

重要更新

▲ 为本书建立互动更新计划
请选择适合你的渠道反馈问题或接受最新更新信息：

天勤论坛微信二维码

更多计算机考研和学习交流尽在www.csbiji.com

天勤论坛

天勤论坛，取名自古训“天道酬勤”，意为考研路上，困苦实多，然而天自有道，勤恳付出者，必有应得之酬劳。

由天勤论坛组编的高分笔记系列计算机考研辅导书，融入了论坛答疑的精华内容，论坛组织了高分考生进行勘误，不断完善此套书籍。考生在书中遇到疑问，也可在线与作者进行交流。

机械工业出版社
CHINA MACHINE PRESS

天勤计算机考研高分笔记系列

2014年计算机专业基础综合历年统考真题及思路分析

第2版

周 伟 王征勇 刘 泱 主编

机械工业出版社

本书汇集了2009～2013年的全国硕士研究生入学计算机专业课统考试题（编号408）。首先，编者不但对所有试题均给出了详细解答，而且对部分试题做到了一题多解，部分试题的解法甚至比标准答案的解法更简捷、更省时省力。其次，编者仍然沿用高分笔记系列书籍的特色，从心理学角度出发，为考生指出了一些可能的错误解法，并点评错因，提醒考生引以为鉴。最后，针对每道真题中涉及的大纲知识点都进行了详细的归纳总结，强化了考生对考题中经常出现的知识点的理解。

本书可作为参加计算机专业研究生入学考试的复习指导用书。

（编辑邮箱：jinacmp@163.com）

图书在版编目（CIP）数据

2014年计算机专业基础综合历年统考真题及思路分析 / 周伟，王征勇，刘泱主编. —2版. —北京：机械工业出版社，2013.9

（天勤计算机考研高分笔记系列）

ISBN 978-7-111-43712-3

Ⅰ. ①2… Ⅱ. ①周… ②王… ③刘… Ⅲ. ①电子计算机—研究生—入学考试—题解 Ⅳ. ①TP3-44

中国版本图书馆CIP数据核字（2013）第195434号

机械工业出版社（北京市百万庄大街22号 邮政编码 100037）

策划编辑：吉 玲 责任编辑：吉 玲 吴超莉 卢若薇

封面设计：鞠 杨 责任印制：张 楠

北京诚信伟业印刷有限公司印刷

2013年9月第2版第1次印刷

184mm×260mm · 17.25印张 · 438千字

标准书号：ISBN 978-7-111-43712-3

定价：39.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务 网络服务

社服务中心：（010）88361066 教材网：http://www.cmpedu.com

销售一部：（010）68326294 机工官网：http://www.cmpbook.com

销售二部：（010）88379649 机工官博：http://weibo.com/cmp1952

读者购书热线：（010）88379203 **封面无防伪标均为盗版**

前　言

计算机统考从2009～2014年即将跨过6个年头。纵观2009～2013年这5年的题型，2009～2012年每年试题的综合性、灵活性都呈平稳上升趋势，但是在2013年有个急速地下降，也许和部分学校自主命题有关。尽管2013年题目难度有所下降，但是真题仍然是复习的重点，只有把握好真题的出题思路，才能以不变应万变。基于此，天勤计算机考研辅导书编写组萌生了写一本关于历年统考真题详解的想法，历经两个月，终于于2012年9月成书，并于2013年7月进行了修订，并将2013年的真题收录其中。该书不但将历年真题的每个考题、考点进行透彻分析，而且还将近5年的所有综合题的评分点进行了详细阐述，准确地帮助考生抓住得分点。

为什么要研究真题命题思路？

考研学子应该清楚，不管是考研数学、考研英语还是考研政治，命题人并不是年年都换，而每位命题人一定会有思维定式，即某年考题的命题思路会与前几年考题的命题思路出现雷同，甚至一模一样。这就是为什么会在考研届流传一句话：研究真题才是王道！计算机专业课统考也不例外，编者对5年真题的分析发现，每年都会有一些题与前几年的命题思路雷同，甚至一样。所以考生非常有必要花时间去研究常考的命题思路。

本书有以下特色：

一、全真模拟环境

本书附录为编者精心编制的5年统考真题试卷版，考生可以用固定的3小时对自己进行全真模拟，并且每套试卷都配备了阅卷老师原版的评分细则，考生可以对自己的答卷自行评分，进而准确地测试复习水平。

二、真题的精心讲解

针对大部分真题，不仅仅是教会考生怎么去做，重点是教会考生为什么要这么去做，深入地分析命题老师的命题思路。

三、一题多解

编者尽量从多个角度去分析试题，尽量做到一题多解，而且部分试题的解法甚至比标准答案的解法更简捷、更省时省力。

四、归纳总结

针对每道真题中涉及的大纲知识点都给出了详细的归纳总结，使考生对于考题中经常出现的知识点得到了强化。并且对于考生容易疏忽的地方，都给予了重点提醒。

五、融入心理学

编者仍然沿用高分笔记系列书籍的特色，从心理学角度出发，指出了考生一些可能的错误解法，并点评错因，提醒考生引以为戒。

本书符号说明：

（1）符号一：1.1　数据结构的基本概念^{未考}

解释：“未考”代表此知识点在历年真题中未考查过。

（2）符号二：1.2　算法及分析$^{[2,0]}$

解释：[2,0]代表此知识点考查过 2 道选择题和 0 道综合题。以此类推，[N，M]代表已考查 N 道选择题与 M 道综合题。

（3）符号三：每道真题之后都会有类似于【11-1】这样的符号

解释：【11-1】代表此题为 2011 年真题中的第 1 题。以此类推，【N-M】代表此题为第 N 年真题中的第 M 题。

本书编写分工为：数据结构部分由王征勇编写，计算机组成原理部分由周伟编写，操作系统部分由刘泱编写，计算机网络部分由周伟编写。参加编写的人员还有王征兴，霍宇驰，董明昊，王辉，郑华斌，王长仁，刘相，章露捷，刘建萍，刘炳瑞，刘菁，孙琪，施伟，金苍宏，蔡明婉，吴雪霞，周政强，孙建兴，周政斌，叶萍，孔蓓，张继建，胡素素，邱纪虎，率方杰，李玉兰，率秀颂。

针对本书，3 位编者为所有读者提供在线答疑的服务，对于书中任何的疑问都可以通过天勤论坛（www.csbiji.com）与编者进行交流，欢迎广大考生批评指正！

最后，预祝所有考生金榜题名！

编　者

目　录

第 3 篇 操作系统

第 4 篇 计算机网络

第 1 篇　数据结构

第 1 章　绪　　论

1.1　数据结构的基本概念[未考]

1.2　算法及其分析[2,0]

1．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是（　　）。

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

A．$O(\log_2 n)$　　B．$O(n)$　　C．$O(n\log_2 n)$　　D．$O(n^2)$

【11-1】

A。容易看出，循环体内的基本操作为 x=2*x;，基本操作执行的次数即为程序的时间复杂度，因此可设基本操作执行 k 次结束，则有：

执行第 1 次：$x=2\times 2=2^{1+1}=4$；

执行第 2 次：$x=4\times 2=2^{2+1}=8$；

执行第 3 次：$x=8\times 2=2^{3+1}=16$；

⋮

执行第 k 次：$x=2^{k+1}$。

由循环结束条件知：x<n/2，

即 $2^{k+1}<n/2$ 时结束，

即 $k<\log_2 n-2$，

即 $k=\log_2 n+C$（为方便说明，其中 C 为起修正作用的常数）。

综上得，时间复杂度为 $O(\log_2 n)$。

【总结】

做这类题目，一定要推出循环结束时，执行次数 k 与问题规模 n 的关系后进行求解。切忌直接作答。若把代码改为

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

```
    x=x+2;
```

k 与 n 的关系为：k<n/4-1。考生自己按照上面的解法写出过程。

答案应该为 O(n)。

2．求整数 n（n≥0）阶乘的算法如下，其时间复杂度是（　　）。

```
int fact(int n)
{   if(n<=1) return 1;
    return n*fact(n-1);
}
```

A．$O(\log_2 n)$　　B．$O(n)$　　C．$O(n\log_2 n)$　　D．$O(n^2)$

【12-1】

B。给出的是一个递归函数。设执行时间为 T(n)，则有下面的等式。

T(n)=1　　当 n=1 时

T(n)=T(n-1)+1　　当 n>1 时

所以 T(n)=T(n-1)+1=T(n-2)+2=T(n-3)+3=…=T(1)+n-1=n=O(n)。

【总结】

做题时，如果一时想不出正规解法，可以举几个特例用排除法进行解题。例如，本题可分别考虑 n=2、n=4 时所需要的执行次数。这里的执行次数分别为 2 和 4，得出执行次数与 n 线性相关，即可选出正确答案 B。

1.3 递归算法设计[未考]

第2章 线性表

2.1 线性表的定义[未考]

2.2 顺序表[0,2]

1. 设将 n（n>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都尽可能高效的算法，将 R 中保存的序列循环左移 p（0<p<n）个位置，即将 R 中的数据序列由（x_0，x_1，…，x_{n-1}）变换为（x_p，x_{p+1}，…，x_{n-1}，x_0，x_1，…，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。 【10-42】

【答案要点】

（1）算法的基本设计思想。先将这 n 个元素的数据序列（x_0，x_1，…，x_p，x_{p+1}，…，x_{n-1}）逆置，得到的数据序列为（x_{n-1}，…，x_p，x_{p-1}，…，x_0），然后将前 n–p 个元素（x_{n-1}，…，x_p）和后 p 个元素（x_{p-1}，…，x_0）分别逆置，得到最终结果（x_p，x_{p+1}，…，x_{n-1}，x_0，x_1，…，x_{p-1}）。

例如，R=（1，2，3，4，5，6，7），p=3，过程如下：

整个逆置： 7 6 5 4 3 2 1

逆置前 n–p（4）个： 4 5 6 7 3 2 1

逆置后 p（3）个： 4 5 6 7 1 2 3

（2）算法实现。算法可以用两个函数，即 Reverse()和 LeftShift()实现相应的功能，后者调用 Reverse()函数 3 次。

算法如下：

```
void Reverse(int R[],int left,int right)//将R[left,…,right]段逆置
{
    int i=left,j=right,temp;      //i 等于左边界 left，j 等于右边界 right
    while(i<j)                    //当 i>=j 时，说明逆置完成。若不写，又会恢复为原序列
    {
        //交换R[i]与R[j]
        temp=R[i];                //将R[i]值赋给中间变量 temp
        R[i]=R[j];                //将R[j]值赋给R[i]
        R[j]=temp;                //最后将原来保存在 temp 中的R[i]值赋给R[j]
        i++;                      //i 右移一个位置
```

```
        j--;                                    //j 左移一个位置
    }
}
void LeftShift(int R[],int n,int p) //将 R[0,…,n-1]整数序列循环左移 p 个元素
{
    if(p>0&&p<n)                                //p 的合法值范围:0<p<n
    {
        Reverse(R,0,n-1);                       //将全部元素逆置
        Reverse(R,0,n-p-1);                     //将前 n-p 个元素逆置
        Reverse(R,n-p,n-1);                     //将后 p 个元素逆置
    }
}
```

（3）算法复杂性。Reverse(R,left,right)算法的时间复杂度为 O(n–m)，所以 LeftShift(R,n,p)算法的时间复杂度=O(n)+O(p)+O(n–p)=O(n)。整个算法只在 Reverse(R,left,right)函数中定义了 3 个变量，因此空间复杂度为 O(1)。

【总结】

（1）代码实现只要与算法步骤描述一致，都为正确的代码。

（2）本题也可以将前 p 个元素逆置得到（x_{p-1}，…，x_0，x_{p+1}，…，x_{n-1}），再将后 n–p 个元素逆置得到（x_{p-1}，…，x_0，x_{n-1}，…，x_p），然后将这 n 个元素逆置变为（x_p，x_{p+1}，…，x_{n-1}，x_0，x_1，…，x_{p-1}）。算法实现就是把原来的代码中的 Reverse(R，0，n–1)一句，改到 Reverse(R，n–p，n–1)之后即可。算法的时间复杂度和空间复杂度都不变。

（3）本题还很容易想到一个次优解法，其中用了一个中间数组，算法如下：

```
void CReverse(int R[],int n,int p) //将 R[0,…,n-1]整数序列循环左移 p 个元素
{
    int *R1=(int *)malloc(p*sizeof(int));//申请了一个大小为 p 的 int 数组 R1
    int i;
    for(i=0;i<p;i++)                    //将 R[0,…,p-1]复制到 R1 中
        R1[i]=R[i];
    for(i=0;i<n-p;i++)                  //将 R[n-p,…,n-1]前移 p 个元素位置
        R[i]=R[i+p];
    for(i=n-p;i<n;i++)                  //将 R1[0,…,p-1]中元素复制到 R[n-p,…,n-1]中
        R[i]=R1[i-(n-p)];
    free(R1);                           //释放 R1 数组
}
```

给出算法的时间复杂度为 O(n)，空间复杂度为 O(p)。

说明：主要就是 3 个 for 循环，算法的时间复杂度=O(p)+O(n–p)+O(p)=O(n)。申请了大小为 p 的数组 R1，定义了一个临时变量 i，所以空间复杂度为 O(p)+O(1)=O(p)。

这种方法很容易想出，但要记住数据结构的算法题的最优算法都不是那么容易想出来的。考生在想到次优算法后，可以再花点时间考虑一下有没有更优的，实在一时想不出，就只能先用次优算法答题。次优算法虽不能得满分，但可以得到大部分的分数。

2. 已知一个整数序列 A=（a_0，a_1，…，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 $m>n/2$（$0 \leqslant p_k < n$，$1 \leqslant k \leqslant m$），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。

【13-41】

【答案要点】

（1）算法的基本设计思想。算法的策略是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(说明已扫描过的元素中，非 Num 元素的个数等于 Num 的个数，可从后面元素段中进行候选主元素的选取)，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）第一步选出的候选元素不一定是主元素，因为在计数器清零后，那一段的信息即忽略了，无法判断最后的候选元素的个数在那一段是否也是多于（或等于）其他元素的个数，因此还需要进行一次判断，即判断 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) 算法复杂性。参考答案中实现的程序的时间复杂度为 O (n)，空间复杂度为 O (1)。

【总结】

这里给出次优解法。

```
int Majority1(int A[ ],int n)       //采用计数排序思想，时间：O(n)；空间：O(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;         //计数数组初始化为 0
    max=0;
    for(k=0;k<n;k++)
    {
        p[A[k]]++;                  //计数器+1
        if(p[A[k]]>p[max])max=A[k]; //记录出现次数最多的元素
    }
    if(p[max]>n/2)
        return max;
    else
        return -1;
}
```

对多数考生来说，本题的最优算法 Majority 较难想到，但次优算法 Majority1 很容易想到，并且从评分标准可知，次优算法只要答题正确，仅比最优算法少 1 分，因此从答题时间上考虑，该题采用次优算法非常可取。

2.3 单链表[1,2]

1. 已知一个带有表头结点的单链表，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，算法输出该结点的 data 域的值，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。 【09-42】

【答案要点】

（1）算法的基本设计思想。定义两个指针变量p和q，初始时均指向头结点的下一个结点，p指针沿链表向后移动。增加一个计数变量count，当p指针移动到第k个结点时（count==k），q指针开始与p指针同步移动。当p指针移动到链表尾结点时，q指针所指元素即为倒数第k个结点。

上述过程对链表仅需一遍扫描。

（2）算法的详细实现步骤。

1）置count=0，p和q都初始指向链表表头结点的下一个结点。

2）若p为空，则转向5）。

3）若count=k，则q指向下一个结点；否则，置count=count+1。

4）置p指向下一个结点，转向2）。

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

6）算法结束。

（3）算法实现。

```
typedef struct LNode                    //单链表中结点类型定义
{
    int data;                           //存放数据元素
    struct LNode *link;                 //指向后继结点
}*LinkList;
int Searchk(LinkList list,int k)
{
    LinkList *p,*q;                     //定义两个指针变量p和q
    int count=0;                        //计数器赋初值
    p=q=list->link;                     //p和q指针都指向链表表头结点的下一个结点
    while(p!=NULL)                      //p指针指向一个单链表的结点
    {
        if(count<k)                     //p指针移动到第k个结点之前
            count++;                    //计数器+1
        else                            //p指针移动到第k个结点之后
            q=q->link;                  //q移到下一个结点
        p=p->link;                      //p移到下一个结点
    }
    if(count<k)                         //如果链表的长度小于k
        return(0);                      //查找失败
    else
```

```
    {
        printf("%d",q->data);          //输出倒数第 k 个结点的 data 域的值
        return(1);                     //查找成功
    }
}
```

【总结】

1）代码实现只要与算法步骤描述一致，都为正确的代码。

2）本题还有一个次优解法，即先求出结点个数 n，再从头找到第 n–k+1 个结点，代码如下：

```
int Searchk(LinkList list,int k)
{   //定义两个指针变量 p 和 q，且 p 指向链表表头结点的下一个结点
    LinkList *p=list->link,*q;
    int n=0,j=0;                       //定义链表长度计数器 n，临时计数器 j
    while(p!=NULL)                     //先求出结点个数 n
    {
        n++;                           //计数器 n+1
        p=p->link;                     //p 指针后移
    }
    q=list;                            //q 指向链表表头结点
    if(n<k)                            //k 合法的取值范围为 k≤n
        return(0);                     //查找失败
    while(q!=NULL&&j<n-k+1)            //再从头找到第 n-k+1 个结点
    {
        j++;                           //计数器 j+1
        q=q->link;                     //q 指针后移
    }
    printf("%d",q->data);              //输出倒数第 k 个结点的 data 域的值
    return(1);                         //查找成功
}
```

这种解法需要两次扫描单链表，该解法应该是很容易想出的。次优算法虽然不能拿满分，但也能拿到大部分的分数。

2．假定采用带头结点的单链表保存单词，当两个单词有相同的后缀时，则可共享相同的后缀空间，例如，“loading”和“being”的存储映像如图 2-1 所示。

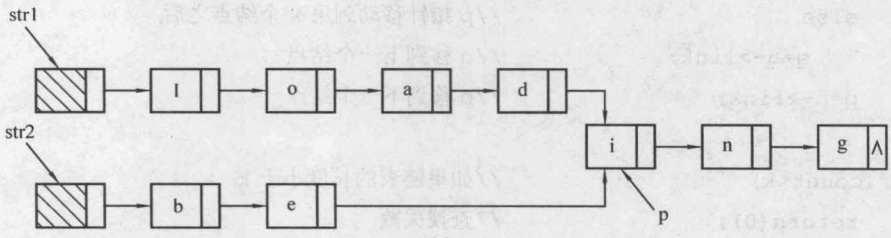

图 2-1