文都教育™

备考计算机专业研究生考试通用教材

2014

全国硕士研究生入学统考

计算机学科专业基础综合

辅导讲义

文都考研命题研究中心 编

主编◎宋雨姗

2014QUANGUOSHUOSHIYANJIUSHENGRUXUETONGKAOJISUANJIXUEKEZHUANYEJICHUZONGHEFUDAOJIANGYI

- 最近考纲与命题规律 完美结合
- 全方位解读考查内容 突出重点
- 多角度剖析典型例题 提取关键

中国时代经济出版社

文都教育™ 备考计算机专

2014

全国硕士研究生入学统考

计算机学科专业基础综合

辅导讲义

文都考研命题研究中心 编
主编◎宋雨姗

中国时代经济出版社

图书在版编目(CIP)数据

计算机学科专业基础综合辅导讲义/宋雨姗主编.
—北京：中国时代经济出版社，2012.5(2013.4 重印)
全国硕士研究生入学考试辅导用书
ISBN 978-7-5119-1108-7

Ⅰ.①计… Ⅱ.①宋… Ⅲ.①电子计算机－研究生－入学考试－自学参考资料 Ⅳ.①TP3

中国版本图书馆 CIP 数据核字(2012)第 077803 号

书　　名：计算机学科专业基础综合辅导讲义
主　　编：宋雨姗

出版发行：中国时代经济出版社
社　　址：北京市丰台区右安门外玉林里 25 号
邮政编码：100069
发行热线：(010)83910203
传　　真：(010)83910203
网　　址：www.cmepub.com.cn
电子邮箱：zgsdjj@hotmail.com
经　　销：各地新华书店
印　　刷：北京市银祥福利印刷厂
开　　本：787×1092　1/16
字　　数：600 千字
印　　张：29
版　　次：2012 年 5 月第 1 版
印　　次：2013 年 4 月第 2 次印刷
书　　号：ISBN 978-7-5119-1108-7
定　　价：55.00 元

前言

Preface

本书以2013年教育部考试中心发布的《计算机专业基础综合考试大纲》为蓝本，对大纲中要求的知识点进行框架梳理，辅以图表形象展示，并配以例题解析及真题演练。全书共分四个部分：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。

计算机学科专业基础综合考试重点考查考生对计算机专业基础知识、基本理论、基础方法的掌握。考生在准备这门考试过程中，应该注意以下几个问题：

1. 大纲中要求掌握哪些内容？

2. 考试中的重点内容有哪些？

3. 重要知识点怎样考查？

4. 历年考试中，经常出现的考点有哪些？

为回答上述问题，本书每一小节都分为：考纲要求、知识点讲解、例题解析与参考答案，真题演练与答案。考纲要求能够帮助考生了解大纲中要求掌握的知识点；知识点讲解帮助考生迅速理清理论脉络，掌握重要考点；例题解析中精选了与考试难度相近的例题，帮助考生了解知识点在考试中的考查形式和重点题型；真题演练中给出了2009—2013年的部分计算机专业基础综合真题，考生可以通过对真题的练习掌握考试的出题形式及答题技巧。

本书具有以下特点：

1. 依据最新考试大纲编写，对考纲要求的知识点进行全面归纳，并对重点和难点做了标注。

2. 使用表格对知识点进行归纳，方便考生记忆重要考点，梳理知识脉络，形成对计算机基础综合科目宏观上的把握。

3. 以插图的形式讲解知识点，方便考生形象地理解计算机基础综合中涉及的原理。

4. 选取了计算机专业基础综合考试中的部分真题，通过真题与知识点的紧密配合，考生能够更明确地了解考点，把握考试规律。

本书适合基础复习阶段和强化复习阶段使用，在基础复习结束之后，建议考生好好研究书中真题，将真题弄通，能够举一反三，并结合重要考点，多加练习。

通过有的放矢地辅导和训练，加上考生的刻苦努力，梦想定会变成现实。

预祝各位考生在2014年考研中笑傲考场，书写自己的传奇！

编　者

2013年4月

目 录

Contents

第一部分　数据结构

第一章　算法与算法分析 …… 3
第一节　常用的算法介绍 …… 3
第二节　算法时间复杂度和空间复杂度的概念以及计算 …… 5
第二章　线性表 …… 8
第一节　线性表的逻辑结构 …… 8
第二节　线性表的顺序存储结构 …… 9
第三节　线性表的链式存储结构 …… 13
第三章　栈、队列和数组 …… 32
第一节　栈与队列 …… 32
第二节　数　　组 …… 39
第四章　树与二叉树 …… 46
第一节　树的概念 …… 46
第二节　二叉树 …… 47
第三节　树和森林 …… 59
第四节　树的应用 …… 62
第五章　图 …… 74
第一节　图的概念 …… 74
第二节　图的存储及基本操作 …… 76
第三节　图的遍历 …… 82
第四节　图的基本应用 …… 84
第六章　查找 …… 93
第一节　查找的基本概念 …… 93
第二节　顺序查找 …… 94
第三节　折半查找 …… 95
第四节　分块查找 …… 97
第五节　B－树和 B＋树 …… 99
第六节　散列表查找 …… 103
第七章　排序 …… 109
第一节　排序的基本概念 …… 109
第二节　插入排序 …… 110

第三节　冒泡排序 …… 113
第四节　简单选择排序 …… 114
第五节　希尔排序 …… 115
第六节　快速排序 …… 116
第七节　堆排序 …… 119
第八节　二路归并排序 …… 121
第九节　基数排序 …… 121
第十节　外部排序 …… 123
第十一节　各种内部排序算法的比较 …… 125

第二部分　计算机组成原理

第一章　计算机系统概述 …… 131
第一节　计算机发展历程 …… 131
第二节　计算机系统层次结构 …… 134
第三节　计算机性能指标 …… 136
第二章　数据的表示和运算 …… 139
第一节　数制与编码 …… 139
第二节　定点数的表示和运算 …… 147
第三节　浮点数的表示和运算 …… 153
第四节　算术逻辑单元 ALU …… 156
第三章　存储系统 …… 160
第一节　存储器的分类 …… 160
第二节　存储器的层次化结构 …… 162
第三节　半导体随机存取存储器 …… 166
第四节　只读存储器 …… 169
第五节　主存储器与 CPU 的连接 …… 170
第六节　双口 RAM 和多模块存储器 …… 174
第七节　高速缓冲存储器(Cache) …… 177
第八节　虚拟存储器 …… 183
第四章　指令系统 …… 193
第一节　指令格式 …… 193
第二节　指令的寻址方式 …… 197
第三节　CISC 和 RISC 的基本概念 …… 200
第五章　中央处理器 …… 205
第一节　CPU 的功能和基本结构 …… 205
第二节　指令执行过程 …… 207
第三节　数据通路的功能和基本结构 …… 212
第四节　控制器的功能和工作原理 …… 214

第五节　指令流水线 …… 222
第六章　总线 …… 230
第一节　总线概述 …… 230
第二节　总线仲裁 …… 232
第三节　总线操作和定时 …… 235
第四节　总线标准 …… 237
第七章　输入输出系统 …… 240
第一节　I/O 系统基本概念 …… 240
第二节　外部设备 …… 240
第三节　I/O 接口(I/O 控制器) …… 244
第四节　I/O 方式 …… 247

第三部分　计算机操作系统

第一章　计算机操作系统概述 …… 259
第一节　操作系统的概念、特征、功能和提供的服务 …… 259
第二节　操作系统的发展与分类 …… 261
第二章　进程管理 …… 265
第一节　进程与线程 …… 265
第二节　进程同步 …… 277
第三节　处理机调度 …… 290
第四节　死　锁 …… 296
第三章　存储管理 …… 306
第一节　内存管理基础 …… 306
第二节　虚拟内存管理 …… 318
第四章　文件管理 …… 329
第一节　文件系统基础 …… 329
第二节　文件系统实现 …… 337
第三节　磁盘组织与管理 …… 338
第五章　输入输出管理 …… 345
第一节　I/O 管理概述 …… 345
第二节　I/O 核心子系统 …… 352

第四部分　计算机网络

第一章　计算机网络概述 …… 363
第一节　计算机网络概述 …… 363
第二节　计算机网络体系结构与参考模型 …… 365
第二章　物理层 …… 369

第一节　通信基础 …… 369
第二节　传输介质 …… 374
第三节　物理层设备 …… 377
第三章　数据链路层 …… 379
第一节　数据链路层的功能 …… 379
第二节　组　帧 …… 380
第三节　差错控制 …… 381
第四节　流量控制与可靠传输机制 …… 383
第五节　介质访问控制 …… 386
第六节　局域网 …… 391
第七节　广域网 …… 395
第八节　数据链路层设备 …… 397
第四章　网络层 …… 402
第一节　网络层的功能 …… 402
第二节　路由算法 …… 403
第三节　IPv4 …… 406
第四节　IPv6 …… 415
第五节　路由协议 …… 416
第六节　IP 组播 …… 421
第七节　移动 IP …… 422
第八节　网络层设备 …… 423
第五章　传输层 …… 431
第一节　传输层提供的服务 …… 431
第二节　UDP 协议 …… 433
第三节　TCP 协议 …… 434
第六章　应用层 …… 442
第一节　网络应用模型 …… 442
第二节　DNS 系统 …… 443
第三节　FTP …… 445
第四节　电子邮件 …… 446
第五节　WWW …… 448

第一部分 数据结构

第一章 算法与算法分析

第一节 常用的算法介绍

知识点讲解

这一部分在考试中主要是与线性表一章结合在一起考查，通常出现在第42题算法设计这道题目中。在复习中要掌握基本的思想，利用这些思想来设计算法。

算法的定义如下：

算法是对特定问题求解步骤的一种描述，是指令的有限序列，其中每条指令表示一个或多个操作。

算法的特性：

- 有穷性：算法必须在有限步内结束。
- 确定性：组成算法的操作必须清晰无二义性。
- 可行性：组成算法的操作必须能够在计算机上实现。
- 输入：0个或多个输入。
- 输出：1个或多个输出。

在考试中，常用的算法有以下几种：

1. 分治算法

<table>
<tr><td rowspan="4">分治算法</td><td colspan="2">将一个规模为N的问题分解为K个规模较小的子问题，这些子问题相互独立且与原问题性质相同。求出子问题的解，就可得到原问题的解。</td></tr>
<tr><td>基本思想</td><td>➢ 当我们求解某些问题时，由于这些问题要处理的数据相当多，或求解过程相当复杂，使得直接求解法在时间上相当长，或者根本无法直接求出。
➢ 对于这类问题，我们往往先把它分解成几个子问题，找到求出这几个子问题的解法后，再找到合适的方法，把它们组合成求整个问题的解法。
➢ 如果这些子问题还较大，难以解决，可以再把它们分成几个更小的子问题，依此类推，直至可以直接求出解为止。</td></tr>
<tr><td rowspan="2">应用场景</td><td>(1)分解，将要解决的问题划分成若干规模较小的同类问题；
(2)求解，当子问题划分得足够小时，用较简单的方法解决；
(3)合并，按原问题的要求，将子问题的解逐层合并构成原问题的解。</td></tr>
<tr><td>(1)原问题可以分解为多个子问题
这些子问题与原问题相比，只是问题的规模有所降低，其结构和求解方法与原问题相同或相似。
(2)原问题在分解过程中，递归地求解子问题
由于递归都必须有一个终止条件，因此，当分解后的子问题规模足够小时，应能够直接求解。</td></tr>
</table>

续表

分治算法	应用场景	(3)在求解并得到各个子问题的解后 应能够采用某种方式、方法合并或构造出原问题的解。 不难发现,在分治策略中,由于子问题与原问题在结构和解法上的相似性,用分治方法解决的问题,大都采用了递归的形式。在各种排序方法中,如归并排序、堆排序、快速排序等,都存在有分治的思想。

2. 递归算法

递归算法	递归算法的特点	(1) 递归就是在过程或函数里调用自身。 (2) 在使用递归策略时,必须有一个明确的递归结束条件,称为递归出口。 (3) 递归算法解题通常显得很简洁,但递归算法解题的运行效率较低。 (4) 在递归调用的过程当中系统为每一层的返回点、局部量等开辟了栈来存储。递归次数过多容易造成栈溢出等。所以一般不提倡用递归算法设计程序。
	递归算法的要求	递归算法所体现的“重复”一般有三个要求: 一是每次调用在规模上都有所缩小(通常是减半); 二是相邻两次重复之间有紧密的联系,前一次要为后一次做准备(通常前一次的输出就作为后一次的输入); 三是在问题的规模极小时必须用直接给出解答而不再进行递归调用,因而每次递归调用都是有条件的(以规模未达到直接解答的大小为条件),无条件递归调用将会成为死循环而不能正常结束。

3. 动态规划

动态规划	基本思想	➢ 动态规划算法与分治法类似,其基本思想也是将待求解问题分解成若干个子问题,先求解子问题,然后从这些子问题的解得到原问题的解。 ➢ 与分治法不同的是,适合于用动态规划求解的问题,经分解得到子问题往往不是互相独立的。若用分治法来解这类问题,则分解得到的子问题数目太多,有些子问题被重复计算了很多次。如果我们能够保存已解决的子问题的答案,而在需要时再找出已求得的答案,这样就可以避免大量的重复计算,节省时间。 ➢ 我们可以用一个表来记录所有已解的子问题的答案。不管该子问题以后是否被用到,只要它被计算过,就将其结果填入表中。
	解题步骤	(1)确定问题的决策对象。 (2)对决策过程划分阶段。 (3)对各阶段确定状态变量。 (4)根据状态变量确定费用函数和目标函数。 (5)建立各阶段状态变量的转移过程,确定状态转移方程。
	适用条件	(1)最优化原理(最优子结构性质) 最优化原理可这样阐述:一个最优化策略具有这样的性质,不论过去状态和决策如何,对前面的决策所形成的状态而言,余下的诸决策必须构成最优策略。简而言之,一个最优化策略的子策略总是最优的。一个问题满足最优化原理又称其具有最优子结构性质。 (2)无后效性 将各阶段按照一定的次序排列好之后,对于某个给定的阶段状态,它以前各阶段的状态无法直接影响它未来的决策,而只能通过当前的这个状态。换句话说,每个状态都是过去历史的一个完整总结。这就是无后向性,又称为无后效性。 (3)子问题的重叠性 动态规划将原来具有指数级复杂度的搜索算法改进成了具有多项式时间的算法。其中的关键在于解决冗余,这是动态规划算法的根本目的。动态规划实质上是一种以空间换时间的技术,它在实现的过程中,不得不存储产生过程中的各种状态,所以它的空间复杂度要大于其他的算法。

4. 回溯法

回溯法	基本思想	回溯法(探索与回溯法)是一种选优搜索法,按选优条件向前搜索,以达到目标。但当探索到某一步时,发现原先选择并不优或达不到目标,就退回一步重新选择,这种走不通就退回再走的技术称为回溯法,而满足回溯条件的某个状态的点称为“回溯点”。
	解题步骤	(1)针对所给问题,定义问题的解空间; (2)确定易于搜索的解空间结构; (3)以深度优先方式搜索解空间,并在搜索过程中用剪枝函数避免无效搜索。

例题解析

评价一个好的算法,您是从哪几方面来考虑的?

答案与解析

答案:评价一个好的算法有四个方面。

一是算法的正确性;

二是算法的易读性;

三是算法的健壮性;

四是算法的时空效率(运行)。

第二节 算法时间复杂度和空间复杂度的概念以及计算

知识点讲解

1. 算法复杂度

算法复杂度分为时间复杂度和空间复杂度。

作用:时间复杂度是度量算法执行的时间长短的指标;

空间复杂度是度量算法所需存储空间的大小的指标。

2. 算法的时间复杂度

所谓算法的时间复杂度,是指执行算法所需要的计算工作量。

一般情况下,算法的基本操作重复执行的次数是模块 n 的某一个函数 f(n),因此,算法的时间复杂度记做:T(n) = O(f(n)) 。

随着模块 n 的增大,算法执行的时间的增长率和 f(n)的增长率成正比,所以 f(n)越小,算法的时间复杂度越低,算法的效率越高。

在计算时间复杂度的时候,先找出算法的基本操作,然后根据相应的各语句确定它的执行次数,再找出 T(n)的同数量级,找出后,f(n) = 该数量级,若 T(n)/f(n)求极限可得到一常数 c,则时间复杂度 T(n) = O(f(n)) 。

按数量级递增排列,常见的时间复杂度有:

常数阶 $O(1)$；

对数阶 $O(\log_2 n)$；

线性阶 $O(n)$；

线性对数阶 $O(n\log_2 n)$；

平方阶 $O(n^2)$；

立方阶 $O(n^3)$；

k 次方阶 $O(n^k)$；

指数阶 $O(2^n)$。

随着问题规模 n 的不断增大，上述时间复杂度不断增大，算法的执行效率越低。

3. 算法的空间复杂度

一个算法的空间复杂度，一般是指执行这个算法所需要的内存空间。

程序执行时所需存储空间包括以下两部分：

(1)固定部分。这部分空间的大小与输入/输出的数据的个数多少、数值无关。主要包括指令空间(即代码空间)、数据空间(常量、简单变量)等所占的空间。这部分属于静态空间。

(2)可变空间。这部分空间主要包括动态分配的空间，以及递归栈所需的空间等。这部分的空间大小与算法有关。

例题解析

1. 在下面的程序段中，对 x 的赋值语句的频度为(　　)。

```
FOR  i: =1  TO  n  DO
  FOR  j: =1  TO  n  DO
    x: =x+1;
```

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

2. 斐波那契数列 F_n 定义如下：

$F_0=0$，$F_1=1$，$F_n=F_{n-1}+F_{n-2}$，$n=2,3\ldots$ 请就此斐波那契数列回答下列问题。

(1)(7 分) 在递归计算 F_n 的时候，需要对较小的 F_{n-1}，F_{n-2}，…，F_1，F_0 精确计算多少次？

(2)(5 分) 如果用大 O 表示法，试给出递归计算 F_n 时递归函数的时间复杂度为多少？

答案与解析

1. 答案：C

解析：执行 2 次 FOR 循环，每次都需要执行 n 次加 1 的操作，执行 n 遍，时间复杂度即为 $O(n^2)$。

2. 答案：

(1)由斐波那契数列的定义可得：

$$
\begin{aligned}
F_n &= F_{n-1}+F_{n-2}\\
&= 2F_{n-2}+F_{n-3}\\
&= 3F_{n-3}+2F_{n-4}\\
&= 5F_{n-4}+3F_{n-5}\\
&= 8F_{n-5}+5F_{n-6}
\end{aligned}
$$

……

$= pF_1 + qF_0$

设 F_m 的执行次数为 $B_m(m=0,1,2,\cdots,n-1)$，由以上等式可知，F_{n-1} 被执行一次，即 $B_{n-1}=1$；F_{n-2} 被执行两次，即 $B_{n-2}=2$；直至 F_1 被执行 p 次、F_0 被执行 q 次，即 $B_1=p$，$B_0=q$。B_m 的执行次数为前两等式第一因式系数之和，即 $B_m=B_{m-1}+B_{m-2}$，再有 $B_{n-1}=1$ 和 $B_{n-2}=2$，这也是一个斐波那契数列。

(2)根据上式可以解得：时间复杂度为 O(n)。

真题演练

1.(2011-1)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. $O(n^2)$

2.(2012-1)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. $O(n^2)$

3.(2013-1)已知两个长度分别为 m 和 n 的升序链表，若将它们合并为一个长度为 m+n 的降序链表，则最坏情况下的时间复杂度是(　　)。

A. O(n)　　B. O(m×n)

C. O(min(m, n))　　D. O(max(m, n))

参考答案

1. A.　　2. B　　3. D

第二章 线性表

第一节 线性表的逻辑结构

考纲要求

线性表的定义和基本操作。线性表的定义包括线性表的逻辑结构定义，线性结构的特点，前驱和后继的主要含义。

知识点讲解

1. 线性表的定义

线性表的定义	线性表(linear list)是具有相同数据类型的 $n(n\geq 0)$ 个数据元素 $a_1, a_2, \cdots a_n$ 组成的有限序列。其中 n 称为数据元素的个数或线性表的长度。
	➢ 当 $n=0$ 时称为空表，$n>0$ 时称为非空表。 ➢ 数据元素 $a_i(1\leq i\leq n)$ 是线性表中第 i 个数据元素，它是一个抽象的符号，其数据类型可以根据具体情况而定，i 称为数据元素 a_i 在线性表中的位序。

2. 线性结构的特点

线性结构的特点是数据元素之间是一种线性关系：

(1)存在唯一的一个被称为“第一个”的数据元素；

(2)存在唯一的一个被称为“最后一个”的数据元素；

(3)除第一个之外，集合中的每个数据元素均只有一个前驱；

(4)除最后一个之外，集合中的每个数据元素均只有一个后继。

例题解析

线性表是具有 n 个(　　)的有限序列($n>0$)。

A. 表元素　　B. 字符　　C. 数据元素

D. 数据项　　E. 信息项

答案与解析

答案：C

解析：本题考查的是线性表的基本概念，要熟记。

第二节 线性表的顺序存储结构

考纲要求

(1)线性表的顺序存储结构,靠元素存储的先后位置反映数据元素的逻辑关系。

(2)在具体语言环境下有两种不同实现:表空间的静态分配和动态分配。

(3)用向量(一维数组)表示,即给定下标可以存取相应元素,属于随机存取的存储结构。

(4)线性表的顺序存储结构实现插入、删除、定位等运算的算法。

知识点讲解

1. 顺序表的定义

顺序表的基本概念	定义	在内存中开辟一片连续的存储空间,将线性表中数据元素按照数据元素之间的逻辑顺序依次存放其中,用这种存储形式存储的线性表称其为顺序表。需注意的是该连续存储空间所能容纳的数据元素个数不得小于顺序表的长度。
	特点	在顺序表中,逻辑关系相邻的两个元素在物理位置上也相邻。 顺序表用物理上的相邻实现数据元素之间的逻辑相邻关系是既简单又自然的。
	数据位置确定	线性表的顺序存储结构很容易确定每个数据元素在存储单元中起始地址的相对位置: 假设线性表中元素为$(a_1,a_2,\cdots,a_{i-1},a_i,a_{i+1},\cdots,a_n)$,设第一个元素 a_1 的内存地址为 $LOC(a_1)$,而每个元素在计算机内占 t 个存储单元,则第 i 个元素 a_i 的首地址 $LOC(a_i)$ 为: $$LOC(a_i)=LOC(a_1)+(i-1)\times t \quad (\text{其中 } 1\leqslant i\leqslant n)$$ 这就是说只要知道顺序表首地址和每个数据元素所占地址单元的个数就可求出第 i 个数据元素的地址,这也是顺序表具有按数据元素的序号随机存取的特点。

2. 顺序表的基本操作

(1)顺序表的初始化

```
int    Init_SqList(SqList  &L){
        L.elem = (ElemType *)malloc(LIST_INIT_SIZE * sizeof(ElemType));
        if(!L.elem) exit(OVERFLOW);//存储分配失败
       L.length = 0;
       L.listsize = LIST INIT_SIZE ; //初始存储容量
       return  OK;
       }
```

顺序表的初始化即构造一个空表,这对表是一个加工型的运算,因此,将 L 设为引用参数。首先动态分配存储空间,然后,将 length 置为 0,表示表中没有数据元素。

(2)在顺序表中插入数据元素

线性表的插入是指在表的第 i(取值范围:$1\leqslant i\leqslant n+1$)个位置上插入一个值为 item 的新元素,插入后使原表长为 n 的表:$(a_1,a_2,\cdots,a_{i-1},a_i,a_{i+1},\cdots,a_n)$ 成为表长为 n+1 的表:$(a_1,a_2,\cdots,a_{i-1},item,a_i,a_{i+1},\cdots,a_n)$。

在顺序表中插入一个元素需要经过以下几个步骤: