天勤论坛

天勤计算机考研高分笔记系列

计算机考研大纲起草者

殷人昆

鼎力推荐

数据结构高分笔记之习题精析扩展

第3版

率 辉 主编

本书互动更新平台：

wechat ID：shuaihui_ds

★量身定做：涵盖历年真题必考题型；仿造出题思路编写，让考生保持一种做真题的感觉。

★梯度分类：将各章习题进行梯度分类，包括基础题与拔高题。

★论坛精品：天勤论坛高分学子的精华交流内容。基础薄弱考生常问的各种难点、易混淆点，已融入相应习题的讲解中。

机械工业出版社
CHINA MACHINE PRESS

天勤计算机考研高分笔记系列

数据结构高分笔记之习题精析扩展

第3版

率 辉 主编

机 械 工 业 出 版 社

本书所选习题，紧密围绕教育部考试中心发布的考试大纲，并以梯度的形式呈现给读者（从基础题进阶到拔高题），使考生的学习更具有针对性。

本书作者对近四年统考真题所考查的知识点进行了深入剖析，给出了复习重点和建议都给出了本章节的考点预测，使得考生可以有重点地进行复习，提高复习效率。在第 3 版中，本书增加了与高分笔记对应的“知识点讲解提纯”部分，可以使考生更加精准地定位考点并做针对性练习。此外，考生还可以关注一下信息平台来了解本书的最新更新并反馈信息：weibo.com/sijieshuai（新浪微博），微信：shuaihui_ds（微信），www.csbiji.com（天勤论坛）。

本书可作为参加计算机专业研究生入学考试的考生复习指导用书，也可作为全国各高校计算机专业或非计算机专业的学生学习相关课程的辅导用书。

（编辑邮箱：jinacmp@163.com）

图书在版编目（CIP）数据

数据结构高分笔记之习题精析扩展 / 率辉主编. —3 版. —北京：机械工业出版社，2016.4

（天勤计算机考研高分笔记系列）

ISBN 978-7-111-53403-7

Ⅰ. ①数… Ⅱ. ①率… Ⅲ. ①数据结构—研究生—入学考试—题解 Ⅳ. ①TP311.12-44

中国版本图书馆 CIP 数据核字（2016）第 064858 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 王 康 刘丽敏

封面设计：鞠 杨 责任校对：杨 林 责任印制：乔 宇

北京铭成印刷有限公司印刷

2016 年 4 月第 3 版第 1 次印刷

184mm×260mm • 13.5 印张 • 339 千字

标准书号：ISBN 978-7-111-53403-7

定价：32.00 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

服务咨询热线：010-88361066 机工官网：www.cmpbook.com

读者购书热线：010-68326294 机工官博：weibo.com/cmp1952

010-88379203 金书网：www.golden-book.com

封面无防伪标均为盗版 教育服务网：www.cmpedu.com

序

《2017 版数据结构高分笔记》《2017 版计算机组成原理高分笔记》《2017 版操作系统高分笔记》《2017 版计算机网络高分笔记》等辅导教材问世了，这对于有志考研的同学是一大幸事。“他山之石，可以攻玉”，参考一下亲身经历过考研并取得优秀成绩的师兄们的经验，必定有益于对考研知识点的复习和掌握。

能够考上研究生，这是无数考生的追求，能够以优异的成绩考上名牌大学的全国数一数二的计算机或软件工程学科的研究生，更是许多考生的梦想。如何学习或复习相关课程，如何打好扎实的理论基础、练好过硬的实践本领，如何抓住要害，掌握主要的知识点并获得考试的经验，先行者已经给考生们带路了。“高分笔记”的作者们在认真总结了考研体会，整理了考研的备战经验，参考了多种考研专业教材后，精心编写了本套系列辅导书。

“天勤计算机考研高分笔记系列”辅导教材的特点是：

✧ 贴近考生。作者们都亲身经历了考研，他们的视角与以往的辅导教材不同，是从复习考研的学生的立场理解教材的知识点——哪些地方理解有困难，哪些地方需要整理思路，叙述处处替考生着想，有很好的引导作用。

✧ 重点突出。作者们在复习过程中做了大量习题，并经历了考研的严峻考验，对重要的知识点和考试出现频率高的题型都了如指掌。因此，在复习内容的取舍上进行了精细地考虑，使得读者可以抓住重点，有效地复习。

✧ 分析透彻。作者们在复习过程中对主要辅导教材的许多习题都进行了深入分析并亲自解答过，对重要知识点做过相关验证并进行了总结，因此，解题思路明确，叙述条理清晰，问题求解的步骤详细，对结果的分析透彻，不但可以扩展考生的思路，还有助于考生举一反三。

计算机专业综合基础考试已经考过 8 年，今后考试的走向如何，可能是考生最关心的问题了。我想，这要从考试命题的规则入手来讨论。

以清华大学为例，学校把研究生入学考试定性为选拔性考试。研究生入学考试试题主要测试考生对本学科的专业基础知识、基本理论和基本技能掌握的程度。因此，出题范围不应超出本科教学大纲和硕士生培养目标，并尽可能覆盖一级学科的知识面，一般会使本学科、本专业本科毕业的优秀考生取得及格以上的成绩。

实际上，全国计算机专业研究生入学联考的命题原则也是如此，各学科的重点知识点都是命题的重点。一般知识要考，比较难的知识（较深难度的知识）也要考。通过对 2009 年以来几年的考试题进行分析可知，考试的出题范围基本符合考试大纲，都覆盖到各大知识点，但题量有所侧重。因此，考生一开始不要抱侥幸的心理去押题，应踏踏实实读好书，认认真真做好复习题，仔仔细细归纳问题解决的思路，夯实基础，增长本事，然后再考虑重点复习。这里有几条规律可供参考：

✧ 出过题的知识点还会有题，出题频率高的知识点，今后出题的可能性也大。

✧ 选择题大部分题目涉及基本概念，主要考查对各个知识点的定义和特点的理解，个别选择题会涉及相应延伸的概念。

✧ 综合应用题分为两部分：简做题和设计题。简做题的重点在设计和计算；设计题的重点在算法、实验或综合应用。

常言道："学习不怕根基浅，只要迈步总不迟"，只要大家努力了，收获总会有的。

清华大学 殷人昆

前　言

高分笔记系列书籍包括《数据结构高分笔记》《计算机组成原理高分笔记》《操作系统高分笔记》《计算机网络高分笔记》等，是一套针对计算机考研的辅导书。它于2010年夏天诞生于一群考生之手，其写作风格特色突出表现为：以学生的视角剖析知识难点；以通俗易懂的语言取代晦涩难懂的专业术语；以成功考生的亲身经历指引复习方向；以风趣幽默的笔触缓解考研压力。高分笔记系列书籍从成书的那一日起就不断接受读者的反馈意见，为了更好地与读者沟通，遂成立了天勤论坛（www.csbiji.com）。论坛名取自古训“天道酬勤”，以明示考研之路艰辛，其成功非勤而无以致。论坛中专门为高分笔记系列书籍开设了答疑专区，以弥补书中讲解的百密一疏；勘误专区，让读者成为作者的一部分，实时发现书中的不足以纠正；读者回馈专区，保留最真实的留言，用读者自己的声音向新人展示高分笔记的特色。相信高分笔记系列书籍带给考生的将是更高效、更明确、更轻松、更愉快的复习过程。

本书的特色

1．考点针对性试题

针对考纲要求的每个知识点，筛选大量相关习题供读者练习，以达到读者在考场上看到任何一个题目都能迅速定位其考查类型，进而快速解题的目的。

2．根据试题难度进行梯度分类

根据试题难度将其分成了基础题和拔高题两类，读者可以在不同的个人解题水平阶段选取适合自己水平的题目进行训练，减少因突如其来的高难度试题造成的心理冲击，提高复习体验。

参加本书编写的人员有：率辉，章露捷，刘建萍，刘炳瑞，刘菁，孙琪，施伟，金苍宏，蔡明婉，吴雪霞，孙建兴，张继建，胡素素，邱纪虎，率方杰，李玉兰，率秀颂，刘忠艳，赵建，张兆红，张来恩，张险峰，殷凤岭，于雪友，周桂芝，张玉奎，李亚静，周莉，李娅，刘梅，殷晓红，李艳红，王中静，张洪英，王艳红，周晓红，杨秋侠，秦凤利，叶萍，王辉，刘桐，王勇，周政强，王长仁，霍宇驰，董明昊，李红梅，郑华斌。

编者

目　录

第 1 章　算法复杂度相关问题专练

说明：对于算法复杂度的评估问题已经是每年必考的重点，因此本书将其独立作为一章，通过一些具体的例题进行有针对性地讲解，帮助考生在解决此类问题时形成系统的套路，以应对多变的考研题目。

如果考生已经读过《数据结构高分笔记》绪论部分，则可以直接做算法复杂度综合题目专练。

对于这部分，要牢记一句话：**将算法中基本操作的执行次数作为算法时间复杂度的度量。**这里所讨论的时间复杂度，不是执行完一段程序的总时间，而是其中基本操作的总次数。因此对于一个算法进行时间复杂度分析的要点，就是明确算法中哪些操作是基本操作，然后计算出基本操作所重复执行的次数。在考试中的算法题目里总能找到一个 n，可以称为问题的规模，如要处理的数组元素的个数为 n，而基本操作所执行的次数是 n 的一个函数 f(n)（这里的函数是数学中函数的概念，不是 C 或 C++语言中函数的概念）。对于求其基本操作执行的次数，就是求函数 f(n)。求出以后就可以取出 f(n)中随 n 增大而增长最快的项，然后将其系数变为 1，作为时间复杂度的度量，记为 **T(n)=O（f(n)中增长最快的项/此项的系数）**。如 $f(n)=2n^3+4n^2+100$，则其时间复杂度为 $T(n)=O(2n^3/2)=O(n^3)$。其实计算算法的时间复杂度，就是给出相应的数量级，当 f(n)与 n 无关时，时间复杂度 T(n)=O(1)；当 f(n)与 n 是线性关系时，T(n)=O(n)；当 f(n)与 n 是平方关系时，$T(n)=O(n^2)$；以此类推。

说明：考研的题目中常常要比较各种时间复杂度的大小，常用的比较关系如下。

$$O(1)\leqslant O(\log_2(n))\leqslant O(n)\leqslant O(n\log_2(n))\leqslant O(n^2)\leqslant O(n^3)\leqslant\cdots\leqslant O(n^k)\leqslant O(2^n)$$

通过以上分析总结出计算一个算法时间复杂度的步骤如下：

1）确定算法中的基本操作以及问题的规模。

2）根据基本操作执行情况计算出规模 n 的函数 f(n)，并确定时间复杂度为 T(n)=O(f(n)中增长最快的项/此项的系数)。

注意：有的算法中基本操作执行的次数不仅仅跟初始输入的数据规模有关，还和数据本身有关。例如，一些排序算法，同样 n 个待处理数据，数据初始有序性不同，其基本操作执行次数也会不同。如果题目不做特殊要求，一般依照使得基本操作执行次数最多的输入来计算时间复杂度，即将最坏的情况作为算法时间复杂度的度量。

例题 1：求出以下算法的时间复杂度。

```
void fun(int n)
{
    int i=1,j=100;
    while(i<n)
    {
        ++j;
        i+=2;
    }
```

```
}
```

分析：

1）找出基本操作，确定规模 n。

① 找基本操作（所谓基本操作，即其重复执行次数和算法的执行时间成正比的操作。通俗点说，这种操作组成了算法，当它们都执行完的时候算法也结束了，**多数情况下取最深层循环内的语句所描述的操作作为基本操作**），显然题目中++j;语句与 i+=2;语句都可以作为基本操作。

② 确定规模，由循环条件 i<n 可以知道，循环执行的次数即基本操作执行的次数和参数 n 有关，因此参数 n 就是规模 n。

2）计算出 n 的函数 f(n)。

显然，n 确定以后，循环的结束与否与 i 有关。i 的初值为 1，每次自增 2，假设 i 自增 m 次后循环结束，则 i 最后的值为 1+2×m，因此有 1+2×m+K=n（其中 K 为一个常数，在循环结束时 i 的值稍大于 n，为了方便表述和进一步计算，用 K 将 1+2×m 修正成 n。因为 K 为常数，所以这样做不会影响最终时间复杂度的计算），解得 m=(n−1−K)/2，即 f(n)=(n−1−K)/2。可以发现其中增长最快的项为 n/2，因此时间复杂度 T(n)=O(n)。

例题 2：分析以下算法的时间复杂度。

```
void fun(int n)
{
    int i,j,x=0;
    for(i=1;i<n;++i)
        for(j=i+1;j<=n;++j)
            ++x;
}
```

分析：

++x;语句处于最内层循环，因此取++x;语句作为基本操作。显然 n 为规模，可以算出++x;语句的执行次数为 f(n)=n(n−1)/2，变化最快的项为 n^2，因此时间复杂度 $T(n)=O(n^2)$。

例题 3：分析以下算法的时间复杂度。

```
void fun(int n)
{
    int i=0;s=0;
    while(s<n)
    {
        ++i;
        s=s+i;
    }
}
```

分析：

显然 n 为规模，基本操作为++i;语句和 s=s+i;语句。i 与 s 都从 0 开始，假设循环执行 m 次结束，则有 $s_1=1$，$s_2=1+2=3$，$s_3=1+2+3=6$，…，$s_m=m(m+1)/2$（其中 s_m 为执行到第 m 次时 s 的值），则有 m(m+1)/2+K=n（K 为起修正作用的常数），由求根公式得

$$m=\frac{-1+\sqrt{8n+1-8k}}{2}$$

即

$$f(n)=\frac{-1+\sqrt{8n+1-8k}}{2}$$

由此可知时间复杂度为

$$T(n)=O(\sqrt{n})$$

说明：在计算时间复杂度的时候有可能会出现这种情况，即对于相同的规模，因输入序列不同会出现不同的时间复杂度，这时一般取最坏情况下的输入序列（即使得基本操作执行次数最多的序列）来计算时间复杂度。

算法复杂度综合题目专练

1．设 n 为如下程序段处理的数据个数，求其时间复杂度。

```
for(i=1;i<=n;i=2*i)
    cout<<"i="<<i<<endl;
```

2．计算 n!的递归函数 fac(n)如下，试分析它的时间复杂度。

```
int fac(int n)
{
    if(n<=1)
        return 1;                    //①
    else
        return(n*f(n-1));            //②
}
```

3．设计一个算法：用不多于 3n/2 的平均比较次数，在数组 A[1，…，n]中找出最大值和最小值的元素。

4．设 A 是一个线性表（a_1，a_2，…，a_n），采用顺序存储结构，则在等概率的前提下，平均插入一个元素需要移动的元素个数是多少？若元素插入在 a_i 与 a_{i+1} 之间（$1\leqslant i\leqslant n-1$）的概率为 $\frac{n-i}{n(n+1)/2}$，则平均每插入一个元素所要移动的元素个数是多少？

5．分析以下各程序段的时间复杂度。

```
(1) void fun()
    {
        int i=1,k=0,n=10;
        while(i<=n-1)
        {
            k+=10*i;
            ++i;
        }
    }
```

```
(2) void fun(int n)
    {
        int i=1,k=0;
        do
        {
            k+=10*i; ++i;
        } while(i==n);
    }
(3) void fun(n)
    {
        int i=1,j=0;
        while(i+j<=n)
            if(i>j)
                ++j;
            else
                ++i;
    }
(4) void fun(int n)
    {
        int x=n,y=0;
        while(x>=(y+1)*(y+1))
           ++y;
    }
(5) void fun(n)
    {
        int i=1;
        while(i<=n)
            i=i*3;
    }
```

6. 分析以下程序的时间复杂度。

```
    void fun(n)
    {
        i=1;
        while(i<=n)
            i=i*2;
    }
```

7. 假设 n 为 2 的乘幂，如 n=2，4，8，16，…试求下列程序的时间复杂度及变量 count 的值（以 n 的函数形式表示）。

```
void counter()
{
```

```
    int n,x,count;
    cout<<"n: ";
    cin>>n;
    count=0;
    x=2;
    while(x<n/2)
    {
        x=2*x;
       ++count;
    }
    cout<<count<<endl;
}
```

8．某算法所需时间由下述方程表示，试求出该算法的时间复杂度（以大 O 形式表示）。注意，n 为求解问题的规模，为简单起见，设 n 是 2 的正整数幂。

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

9．分析 order 函数的时间复杂度。

```
order(int j,int n)
{
    int i,temp;
    if(j<n)
    {
        for(i=j;i<=n; ++i)
            if(a[i]<a[j])
            {
                temp=a[i];
                a[i]=a[j];
                a[j]=temp;
            }
        ++j;
        order(j,n);                    //递归调用
    }
}
```

10．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，…就此斐波那契数列，回答下列问题。

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

（2）如果用大 O 表示法，递归计算 F_n 时递归函数的时间复杂度是多少？

11．设计求解下列问题的算法，并分析其最坏情况的时间复杂度。

（1）在数组 A[1，…，n]中查找值为 K 的元素，若找到则输出其位置 i；否则输出 0 作为

标志。

（2）找出数组A[1，…，n]中元素的最大值和次最大值。

12．以下算法是在一个有n个数据元素的数组A中删除第i个位置的数组元素，要求当删除成功时数组元素个数减1，求平均删除一个数组元素需要移动的元素个数是多少？其中，数组下标从0至n-1。

```
int delete(int A[],int n,int i)
{
    int j;
    if(i<0||i>n)
        return 0;
    for(j=i+1;j<n; ++j)
        A[j-1]=A[j];
    --n;
    return 1;
}
```

算法复杂度综合题目专练答案

1.【答案要点】

其中的基本语句是第2行的cout语句，设它的执行次数为f(n)，则有$2^{f(n)}\leqslant n$，即$f(n)\leqslant \log_2 n$。

本程序段时间复杂度$T(n)=f(n)\leqslant \log_2 n=O(\log_2 n)$。

所以时间复杂度为$O(\log_2 n)$。

2.【答案要点】

设T(n)为fac(n)的时间开销函数。显然T(1)=O(1)。①的时间开销为O(1)，递归调用fac(n-1)的时间开销为T(n-1)；②的时间开销为O(1)+T(n-1)。时间复杂度为

$$\begin{aligned}T(n)&=O(1)+T(n-1)\\&=O(1)+O(1)+T(n-2)\\&=2\times O(1)+T(n-2)\\&\vdots\\&=(n-1)\times O(1)+T(1)\\&=n\times O(1)=O(n)\end{aligned}$$

3.【答案要点】

本题的算法思想是：如果在查找出最大值和最小值的元素时各遍历一遍所有元素，则至少要比较2n次，所以使用一遍遍历找出最大值和最小值的元素。实现本题功能的函数如下：

```
void maxmin(int A[],int n)
{
    int i;
    int max,min;
    max=A[1];min=A[1];
```

```
    for(i=2;i<=n; ++i)
        if(A[i]>max)
            max=A[i];
        else if(A[i]<min)
            min=A[i];
    cout << "max=" << max << ",min=" << min << endl;
}
```

在这个函数中，最坏的情况是 A 中的元素按递减次序排列，这时（A[i]>max）条件均不成立，比较的次数为 n-1。另外，每次都要比较（A[i]<min），同样比较的次数为 n-1。因此，总的比较次数为 2(n-1)。最好的情况是 A 中的元素按递增次序排列，这时（A[i]>max）条件均成立，不会再执行 else 的比较，所以总的比较次数为 n-1。

平均比较次数为

$$\frac{2(n-1)+n-1}{2}=\frac{3n}{2}-\frac{3}{2}\leqslant\frac{3n}{2}$$

由此可知，本算法的平均比较次数不多于 3n/2。

4.【答案要点】

A=（a_1，a_2，…，a_n），有 n+1 个位置可插入元素，即 a_1 之前，a_1 与 a_2 之间，…，a_{n-1} 与 a_n 之间和 a_n 之后，分别需要移动的元素个数为 n，n-1，…，1，0，则平均插入一个元素需要移动的元素个数为

$$(n+(n-1)+\cdots+1+0)\times\frac{1}{n+1}=\frac{n}{2}$$

若元素插入在 a_i 与 a_{i+1} 之间（$1\leqslant i\leqslant n-1$）的概率为 $\frac{n-i}{n(n+1)/2}$，其中移动的元素个数为

$$(n-i)\times\frac{n-i}{n(n+1)/2}=\frac{(n-i)^2}{n(n+1)/2}$$

则平均每插入一个元素所要移动的元素个数为

$$\sum_{i=1}^{n-1}\frac{(n-i)^2}{n(n+1)/2}=\frac{2n+1}{3}$$

5.【答案要点】

（1）O(1)；（2）O(n)；（3）O(n)；（4）$O(\sqrt{n})$；（5）$O(\log_3 n)$。

本题较为简单，这里只讲解（5）。

设执行 k 次，则有 $3^k+C=n$（其中 C 是起到修正作用的常数），即 $k=\log_3(n-C)$。

$\log_3(n-C)$与 $\log_3 n$ 同数量级，因此复杂度为 $O(\log_3 n)$。

6.【答案要点】

此处循环体里面是 i=i*2，即每循环一次，i 值增加一倍，所以执行次数与 n 之间是以 2 为底的对数关系，故时间复杂度为 $O(\log_2 n)$。

7.【答案要点】

n 与 count 的关系如下：

当 $n=2^0$ 时，count=0；当 $n=2^1$ 时，count=0；当 $n=2^2$ 时，count=0；当 $n=2^3$ 时，count=1；

当 $n=2^4$ 时，count=2，…；当 $n=2^m$ 时，count=m−2。则本算法的时间复杂度为 $O(\log_2 n)$。count=$\log_2 n-2$。

8.【答案要点】

设 $n=2^m$，则

$$\begin{aligned}T(n)=T(2^m)&=2T(2^{m-1})+2^m\\&=2(2T(2^{m-2})+2^{m-1})+2^m=2^2\times T(2^{m-2})+2\times 2^m\\&\vdots\\&=2^m\times T(1)+m\times 2^m\\&=(m+1)2^m\\&=(\log_2 n+1)n\\&=O(n\log_2 n)\end{aligned}$$

9.【答案要点】

order( )函数是一个递归排序过程，设 T(n)是排序 n 个元素所需要的时间。在排序 n 个元素时，算法的计算时间主要花费在递归调用 order( )上。在第一次调用时，处理的元素序列个数为 n−1，也就是对余下的 n−1 个元素进行排序，所需要的计算时间应为 T(n−1)。又因为在其中的循环中，需要 n−1 次比较，所以排序 n 个元素所需要的时间为

$$T(n)=T(n-1)+n-1,\quad n>1$$

可以得到如下方程：

$$\begin{aligned}&T(1)=0\\&T(n)=T(n-1)+n-1,\quad n>1\end{aligned}$$

求解过程为

$$\begin{aligned}T(n)&=[T(n-2)+(n-2)]+(n-1)\\&=[T(n-3)+(n-3)]+(n-2)+(n-1)\\&\vdots\\&=(T(1)+1)+2+\cdots+n-1\\&=0+1+2+\cdots+n-1\\&=n(n-1)/2\\&=O(n^2)\end{aligned}$$

所以 order 函数的时间复杂度为 $O(n^2)$。

10.【答案要点】

（1）从题中可知：$F_0=0$，$F_1=1$，$F_2=1$，$F_3=2$，$F_4=3$，$F_5=5$，$F_6=8$，…

$$\begin{aligned}F_n&=F_{n-1}+F_{n-2}\\&=2F_{n-2}+F_{n-3}(\text{即 }F_3F_{n-2}+F_2F_{n-3})\\&=3F_{n-3}+2F_{n-4}(\text{即 }F_4F_{n-3}+F_3F_{n-4})\\&=5F_{n-4}+3F_{n-5}(\text{即 }F_5F_{n-4}+F_4F_{n-5})\\&\vdots\\&=F_{n-1}F_1+F_{n-2}F_0\end{aligned}$$

归纳起来，在递归计算 F_n 时，需要对较小的 F_{n-1} 计算 1（即 F_2）次，F_{n-2} 计算 2（即 F_3）次，…，F_1 计算 F_n 次，F_0 计算 F_{n-1} 次。

以计算 F_5 为例，其递归调用树如图 1-1 所示。从图 1-1 中可以看到，F_4 计算 1 次，F_3 计

算 2 次，F_2 计算 3 次，F_1 计算 F_5（=5）次，F_0 计算 F_4（=3）次。

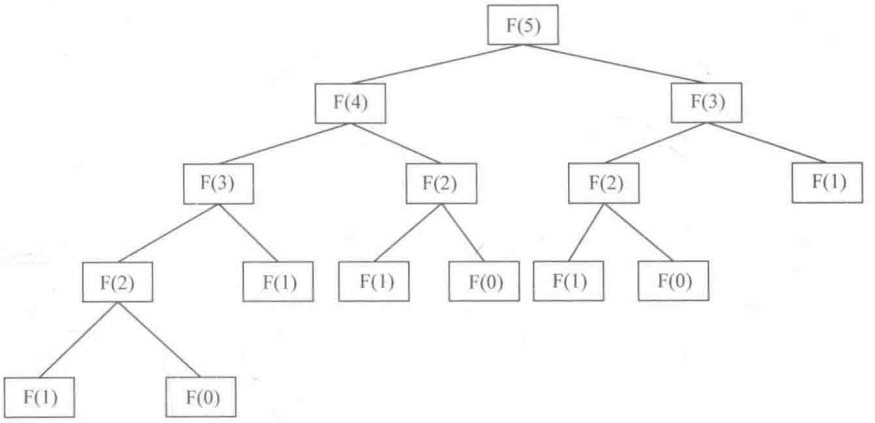

图 1-1　F(5)递归调用树

（2）设计算 F_n 的时间复杂度为 T(n)，有

$$\begin{aligned}T(n)&=T(n-1)+T(n-2)<2T(n-1)\\&<2^2T(n-2)\\&\vdots\\&<2^nT(0)\\&=O(2^n)\end{aligned}$$

又有

$$\begin{aligned}T(n)&=T(n-1)+T(n-2)\\&>2T(n-2)\\&>2^2T(n-4)\\&\vdots\\&>2^{n/2}T(1)\text{（n 为奇数时；或 }2^{n/2}T(0)\text{，n 为偶数时）}\end{aligned}$$

即

$$T(n)>O(2^{n/2})$$

则

$$O(2^{n/2})<T(n)<O(2^n)$$

取最坏情况上限，即 $T(n)=O(2^n)$。

11.【答案要点】

（1）算法如下：

```
假设 datatype 是 typedef 出来的一个 C 语言标准数据类型的别名
int locate(datatype A[],datatype k)
{
    int i=n;
    while(i>=1 && A[i]!=k)
        --i;
    return i;
}
```