第一章 算法与算法分析

§1.1 从问题到程序

1. 搞清问题

\*要用程序解决问题，首先要搞清我们要解决什么问题。有些问题，比如：维护世界和平，无法用计算机解决。有人说，搞清了问题，问题就解决了一半了。

二． 用形式模形来描述问题

三．设计抽象数据类型

1. 什么是抽象数据类型？

抽象数据类型(ADT)是一个数学模型(通常有一个值的集合)以及定义在该模型上的一组操作。

\*例子：整数集合的集合以及集合的并、交、差运算。

一个群、环、域也是抽象数据类型。

\*面向对象的程序设计。

2. 数据类型、数据结构和抽象数据类型的差别

(1) 数据类型：在程序设计语言中，一个变量的数据类型是该变量所能取的值的集合。

\*例如：整型，布尔型

(2) 数据结构：具有不同数据类型的一组变量按各种形式联系起来。

(3) 抽象数据类型：是一个数学模型以及定义在该模型上的一组操作。

3. 实现抽象数据类型

\*根据实现的程序设计语言提供的数据类型，用各种数据结构组织起来实现。抽象数据类型的操作用程序的过程来实现。

4. 抽象数据类型的特点：

一般性

封装性

四. 设计算法

\*有了问题的数学模型和抽象数据类型后，可以设计解决问题的算法。

1. 什么是算法？

算法是一个有穷的指令序列。每条指令有清楚的含义，并且在有穷的时间内用有穷的动作能完成。一个算法无论接受任何输入都必须在有穷步内停止。

\*算法与程序不同，操作系统永远不停止，因此不是算法。而操作系统里的子任务是用算法实现的。

2. 关于指令

具体地说，在现实计算机上或图灵机上在有穷时间内能完成的才叫指令。

§1.2 程序的运行时间

1. 算法的设计目标
2. 算法应易于理解、编程和调试
3. 算法应尽可能有效地利用计算机的资源，特别地，它应尽可能快地运行。

\*两者矛盾，设计算法时需要权衡。

二. 程序运行时间的测量

影响程序运行时间的因素：

1. 程序输入的长度
2. 编译程序生成目标代码的质量
3. 计算机指令的性质和速度
4. 算法的时间复杂性

三．评价算法运行时间的标准

运行时间作为输入长度的函数T(n)

1. 最坏运行时间

算法对具有长度n的任何输入的最长运行时间

1. 最好运行时间

算法对具有长度n的任何输入的最短运行时间

1. 平均运行时间

即在“平均”输入下，算法的运行时间。通常我们假设给定长度的各种输入概率相同。平均运行时间是在这个假设下，运行时间的数学期望值。

1. 为什么常用最坏运行时间来估计

最坏运行时间是算法运行时间的上界，在实际问题中，算法的运行时间常常达到这个上界。平均运行时间难以计算。假定每一个输入具有相同的概率有时没有意义。平均运行时间常常与最坏运行时间有相同的数量级。

四．记号

1. 记号

设g(n)是一个给定函数，用表示函数的集合：

{存在正的常数和, 使得当, 有

}

我们写, 表示。

例1：

取, 当时，有。

2.记号O

设g(n)是一个给定函数，用表示函数的集合：

{存在正的常数和, 使得当, 有

}

我们写, 表示。

显然有 。

例2：同例1，有。同时，也有。

但是，不成立。

3.记号

设g(n)是一个给定函数，用表示函数的集合：

{存在正的常数和, 使得当, 有

}  
我们写, 表示。

定理1.1：对任意两个函数和, 当且仅当

且。

4.记号o

设g(n)是一个给定函数，用表示函数的集合：

{对任意正的常数c, 存在常数, 使得当, 有}

我们写, 表示。

例3：，但。

5.记号

设g(n)是一个给定函数，用表示函数的集合：

{对任意正的常数c, 存在常数, 使得当

, 有}

我们写，表示。

例4：, 但。

五．运行时间增长率的比较

例5：有两个算法运行时间分别为和，是否的算法比的算法好？

设, 。

当时，算法2比算法1运行得快，当充分大时，

，故当充分大时，算法1比算法2快。

例6：有4个算法，运行时间增长率如下：（见图1.1）

图1.1 图1.2

\*运行时间T(n)为n的多项式的算法称为好算法。

六．设计算法的几个原则：

1. 如果一个程序只用一两次，那么书写和调试所用的时间比程序运行时间大得多，因而算法应易于理解和正确实现。

2. 如果一个程序只对小的输入运行，运行时间增长率比运行时间前面的常数因子显得不重要。可选常数因子小而增长率大的算法。

3. 一个复杂而有效的算法可能不利于维护，有时应选择简单而相对低效的算法。

4. 时间复杂度小的算法，空间复杂度可能很大。

5. 对数值算法而言，精度和稳定性与时间高效一样重要。

§1.3 算法分析技术

一．计算程序运行时间

1. 加法规则：

如果和分别是两个程序段和的运行时间，，，那么程序段后跟的运行时间为

, 时间复杂度为。

这因为存在正的常数和，使得当时，；当时，。令，

, 当时，

2.乘法规则：

如果和为和，那么为

。

乘法规则主要用于循环结构的时间分析。

\*注意有，其中为常数。

二．例子：

PROCEDURE Bubblesort (VAR A : array [1n] of integer);

VAR

i, j, temp : integer;

BEGIN

1. FOR i := 1 TO n-1 DO
2. FOR j := n DOWNTO i+1 DO
3. IF A[j-1] > A[j] THEN

BEGIN

1. temp := A[j-1];
2. A[j-1] := A[j];
3. A[j] := temp;

END

END;

算法分析：

(4), (5), (6)句为。

(3)句取最坏情况，执行条件判断O(1), 语句内部O(1), 结果为O(1)。

(2)句执行循环控制条件为O(1)，内部O(1), 共循环次，

由乘法规则，时间为。

(1)句将各次循环的时间加起来，为

三．过程调用和GOTO语句的分析

\*假设没有递归调用，则把过程调用语句当作一个语句处理，被调用过程的执行时间即是该语句的执行时间。从而我们可以分析调用过程的执行时间。由于没有递归调用，按以上方法，我们最终可以得到主程序的运行时间。

GOTO语句只限于用在循环内某个位置跳出循环，异常终止循环。按最坏情况运行时间分析，可以假设不存在该GOTO语句。

§1.4 递归程序的分析

一．什么是递归程序？

\*一个过程在运行时直接或间接地调用自己，则该过程称为递归程序。

二．递归程序的分析

1. 例子：归并排序

以下过程Merge()将两个已排好序的表和归并成一个排好序的表L。

\*不妨设和中的元素按从小到大的顺序排列。具体做法是，取和中第1个元素，比较它们的大小，不妨设的第1个元素小，将其从中取出，放入L中。再取和中第1个元素比较，取小者放入L中。反复做上述工作，直到或中一个表的元素已取完，然后将另一个表剩下的元素放在L的末尾。

例7： 将以下元素归并排序

图1.3

算法：

FUNCTION Mergesort (L : LIST; n : integer) : LIST;

/\*L是长为n的表，返回排好序的表。设n是2的幂\*/

BEGIN

IF n=1 THEN RETURN(L)

ELSE BEGIN

把L分成两个长为n/2的表L1和L2;

RETURN(Merge(Mergesort(L1, n/2), Mergesort(L2, n/2)));

END

END;

2.列递推关系式

设T(n)是过程Mergesort在最坏情况下的运行时间，那么

\*上式的含义是：当时，该过程只需返回L，故只需要常数时间, 当时，该过程两次递归调用自身，且和的长度为, 故需时间，而Merge过程将和归并成一个排好序的表L，最坏需时间。从而。

3. 几点解释

(1) 只有当n为偶数时，(1.1)式才成立，另外，只有当n为2的幂时，才能得到解。

(2) 对任意n, 我们可以合理地假设，如果，。从而对2的幂解(1.1)式，可以对任意n给出一个合理的估计。

(3) 对奇数n，我们也可以把递归式中改为来解。这样使(1.1)式解起来复杂一些。

作业1：

1. 估计以下程序最坏情况下的时间复杂性。

PROCEDURE Mystery (n : integer);

VAR

i, j, k : integer;

BEGIN

FOR i := 1 TO n-1 DO

FOR j := i+1 TO n DO

FOR k := 1 TO j DO

{一些O(1)时间的语句}

END；

1. 按增长率从小到大将以下函数排序：(a) ; (b) ; (c) ; (d) ; (e) ; (f) ; (g) ; (h) ; (i) ; (j) 17 。