计算机算法设计与分析之
算法概述

易凯

2017 年 3 月 26 日

班　　级 软件 53 班
学　　号 2151601053
邮　　箱 williamyi96@gmail.com
联系电话 13772103675
个人网站 https://williamyi96.github.io
williamyi.tech
实验日期 2017 年 3 月 26 日
提交日期 2017 年 6 月 6 日

目录

1 重点掌握内容 **3**

1.1 算法与程序 3

1.2 算法复杂度分析 3

1.3 NP 完全性理论 4

2 典型题详解 **4**

2.1 算法复杂性分析问题 4

2.2 证明 $n! = o(n^n)$ 5

2.3 由平均计算复杂性推导最坏情况复杂性 5

1 重点掌握内容

1.1 算法与程序

算法 算法是由若干条指令组成的有穷序列。

其具有的特点是：

1. 输入

2. 输出

3. 确定性

4. 有限性

程序 程序就是算法用某种程序设计语言进行的具体实现。其可以不满足算法的性质 4，也就是算法的有限性。

1.2 算法复杂度分析

衡量算法复杂度的指标是在理想抽象计算机上运行某一算法所需资源消耗量的大小。而资源消耗量包含了时间复杂度和空间复杂度两个层面的内容。

其中值得注意的是三种，最坏时间，最好时间和平均时间：

最坏时间 ：$T(N,I) = max_{I\in D_N} T(N,I) = T(N,I^+)$

最好时间 ：$T(N,I) = min_{I\in D_N} T(N,I) = T(N,I^-)$

平均时间 ：$T(N,I) = \sum_{I\in D_N} P(I)T(N,I)$

算法复杂度一般采用渐进性分析的方式进行。其中，渐进符号有 O,Ω,Θ,o,ω 这五种，关于其基础性的定义较为简单，在此不进行赘述。其中带等号的两个中的常量是**存在**，而不带等号的两个中的常量是对于任意给定的常量。

其中有两个 small tricks:

1. $o: f(n) < cg(n) \Rightarrow \frac{f(n)}{g(n)} < c \Rightarrow \frac{f(n)}{g(n)} \to 0$

2. $\omega : cg(n) < f(n) \Rightarrow \frac{f(n)}{g(n)} > c \Rightarrow \frac{f(n)}{f(n)} \to \infty$

1.3 NP 完全性理论

一般来说，在多项式时间内可以求解的问题称之为易解问题，而将需要超过多项式时间才能求解的问题称之为难解问题。

为了研究难解问题的计算复杂性，人们提出了非确定性图灵机计算模型。

值得注意的是，书中探讨的许多问题都是以最优化的形式给出的，而对于每一个最优化问题，都有一个与之对应的判定问题。

所有可以在多项式时间内求解的判定问题构成了 P 类问题。通常情况下，解一个问题要比验证一个问题的一个解困难得多。P 类问题是确定性计算模型之下的易解问题类，而 NP 类问题是非确定性计算模型之下的已验证问题类。所有非确定性多项式时间可解的判定问题构成 NP 类问题。P 与 NP 的关系暂时没有一个很好的解答，目前而言，存在一类被称之为 NP 完全的问题，也就是 NPC 问题，其具有一个很特殊的性质就是如果一个 NP 完成问题能在多项式时间内得到解决，那么 NP 中的每一个问题都可以在多项式时间内求解，也就是 N = NP。

2 典型题详解

2.1 算法复杂性分析问题

试题描述 对于下列给定各组函数 f(n) 和 g(n), 确定 $f(n) = O(g(n))$ 或者 $f(n) = \Omega(g(n))$ 或者 $f(n) = \Theta(g(n))$，并阐述理由。

试题解答 理由的阐述由相关的定义来得到。以下仅根据典型性的例子进行分析：

证明： $f(n) = \Theta g(n)$ $f(n) = logn^2, g(n) = logn + 5$
首先证明 $f(n) = O(g(n))$
当 $c = 2, n_0 = 1$ 时，n > n_0 f(n) < c g(n) 恒成立。
然后证明 $f(n) = \Omega(g(n))$
当 $c = 1, n_0 = 10^5$ 时，对 n > n_0 c g(n) < f(n) 恒成立。

综上所述，$f(n) = \Theta g(n)$

2.2 证明 $n! = o(n^n)$

此处需要使用斯特林似然公式：$n! = \sqrt{2\pi n}(\frac{n}{e})^n(1 + \Theta(\frac{1}{n}))$,
此题用 $o : f(n) < cg(n) \Rightarrow \frac{f(n)}{g(n)} < c \Rightarrow \frac{f(n)}{g(n)} \to 0$ 即可证明。

2.3 由平均计算复杂性推导最坏情况复杂性

题目分析 如果一个算法在平均情况下的计算时间复杂性为 $\Theta(f(n))$, 则该算法在最坏情况下所需的计算时间为 $\Omega(f(n))$。

题目求解 该题目的求解方式是通过将所有对应的合法输入转变为对应的最坏输入情况，通过放缩进行求解：

$$
\begin{aligned}
& T_{avg}(N) \\
&= \textstyle\sum_{I \in D_N} P(I)T(N, I) \\
&\leq \textstyle\sum_{I \in D_N} P(I) max_{I' \in D_N} T(N, I') \\
&= T(N, I+) \textstyle\sum_{I \in D_N} P(I) \\
&= T(N, I+) \\
&= T_{max}N
\end{aligned}
$$

因此有 $T_{max}(N) = \Omega(T_{avg}(N)) = \Omega(\Theta(f(n))) = \Omega(f(n))$。