第一章 算法分析与 NP-完全问题

§1.1 算法分析

一．程序运行时间的测量

影响程序运行时间的因素：
1.程序的输入的长度
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

二．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。

2.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。

三．记号$\Theta, O, \Omega, o, \omega$

1.记号Θ
设 g(n)是一给定函数，用$\Theta(g(n))$表示函数的集合：
$\Theta(g(n)) = \{ f(n) |$存在正的常数C_1, C_2和n_0使得当$n \geq n_0$有$0 \leq C_1 g(n) \leq f(n) \leq C_2 g(n)\}$。我们写$f(n) = \Theta(g(n))$，表示$f(n) \in \Theta(g(n))$。

例 1：$\frac{1}{2}n^2 - 3n = \Theta(n^2)$。

取$n_0 = 7, C_1 = \frac{1}{14}, C_2 = \frac{1}{2}$

当$n \geq n_0$时，$C_1 n^2 \leq \frac{1}{2}n^2 - 3n \leq C_2 n^2$。

2.记号 O

设 g(n)是一给定函数，用$O(g(n))$表示函数的集合：$O(g(n)) = \{ f(n) \mid$ 存在正的常数 C 和n_0使得当$n \geq n_0$，有$0 \leq f(n) \leq Cg(n)\}$。我们写$f(n) = O(g(n))$, 表示$f(n) \in O(g(n))$。显然有$\Theta(g(n)) \subseteq O(g(n))$。

例 2：同例 1，有$\frac{1}{2}n^2 - 3n = O(n^2)$，同时也有$3n = O(n^2)$,

但是$3n = \Theta(n^2)$不成立。

3.记号Ω

设 g(n)是一给定函数，用$\Omega(g(n))$表示函数的集合：$\Omega(g(n)) = \{f(n) \mid$ 存在正的常数 C 和n_0，使得当$n \geq n_0$有$0 \leq Cg(n) \leq f(n)\}$。

4.记号 o

设 g(n)是一给定函数，用$o(g(n))$表示函数的集合：$o(g(n)) = \{f(n) \mid$ 对任意正的常数 C，存在常数$n_0 > 0$，使得当$n \geq n_0$，有$0 \leq f(n) < Cg(n)\}$.

例 3: $2n = o(n^2)$，但$2n^2 \neq o(n^2)$。

5.记号ω

设 g(n)是一给定函数，用$\omega(g(n))$表示函数的集合：$\omega(g(n)) = \{ f(n) \mid$对于任意正的常数 C，存在常数$n_0 > 0$，使得当$n \geq n_0$，有$0 \leq Cg(n) < f(n)\}$。

例 4：$\frac{1}{2}n^2 = \omega(n)$，但$\frac{1}{2}n^2 \neq \omega(n^2)$。

四．运行时间增长率的比较

例 5：有两个算法运行时间分别为$\Theta(n^2)$和$\Theta(n^3)$是否$\Theta(n^2)$的算法比$\Theta(n^3)$的算法好？

设$T_1(n) = 100n^2$, $T_2(n) = 5n^3$,

$$\frac{T_2(n)}{T_1(n)} = \frac{5n^3}{100n^2} = \frac{n}{20}$$

当$n < 20$时，算法 2 比算法 1 运行得快。当 n 充分大时，

$$\frac{T_2(n)}{T_1(n)} \longrightarrow \infty$$

故当 n 充分大时，算法 1 比算法 2 快。

例 6：有四个算法运行时间增长率如下：（见图 1.1）

§1.2 算法分析技术

一．计算程序运行时间

1.加法规则：

如果$T_1(n)$和$T_2(n)$分别是两段程序P_1和P_2的运行时间，$T_1(n)=O(f(n))$，$T_2(n)=O(g(n))$，那么程序段P_1后跟P_2的运行时间为$T_1(n)+T_2(n)$，时间复杂度为$O(\max(f(n),g(n)))$。

这因为存在正的常数C_1, C_2, n_1和n_2，使得当$n \geq n_1$时，$T_1(n) \leq C_1 f(n)$；当$n \geq n_2$时，$T_2(n) \leq C_2 g(n)$。令$n_0=\max(n_1,n_2)$，$C=C_1+C_2$，当$n \geq n_0$时，$T_1(n)+T_2(n) \leq C_1 f(n)+C_2 g(n) \leq (C_1+C_2)\max(f(n),g(n))$
$\leq C\max(f(n),g(n))$。

2.乘法规则：

如果$T_1(n)$和$T_2(n)$为$O(f(n))$和$O(g(n))$，那么$T_1(n)T_2(n)$为$O(f(n)g(n))$。乘法规则主要用于循环结构的时间分析。

*$O(Cf(n))=O(f(n))$，其中$C>0$。

二．例子：

```
PROCEDURE Bubblesort(VAR A : array[1..n]of integer);
        VAR
          i, j, temp : integer;
       BEGIN
   FOR i := 1 TO n-1 DO
      FOR j := n DOWNTO i+1 DO
        IF A[j-1] > A[j] THEN
              BEGIN
           temp := A[j-1];
           A[j-1] := A[j];
           A[j] := temp;
              END
        END;
```

算法分析：

(4), (5), (6)为 O(1)+O(1)+O(1) = O(max(1, 1, 1)) = O(1);

(3)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)执行循环控制条件为 O(1)，内部 O(1)，共循环 n-i 次，由乘法规则 $O((n-i)\times 1)=O(n-i)$;

(1)将各次循环的时间加起来：

$$O(\sum_{i=1}^{n-1}(n-i)=O\left(\frac{n(n-1)}{2}\right)=O\left(\frac{n^2}{2}-\frac{n}{2}\right)=O(n^2)。$$

§1.3 确定图灵机

一．确定图灵机

1.构成：确定型单带图灵机(DTM)由有限状态控制器，读写头和一条带组成：这条带由标有$\cdots,-2,-1,0,1,2,3,\cdots$的带方格的双向无穷序列构成。（见图 1.2）

2.DTM 定义：

一个 DTM(程序)包括：

(1)有穷的带符号集Γ，包括输入符号子集$\Sigma \subset \Gamma$和一个特殊的空白符$\#\in \Gamma-\Sigma$；

(2)有穷状态集合 Q，包括一个特殊的初始状态q_0和两个特殊的停机状态q_Y和q_N；

(3)转移函数：

$$\delta:(Q\backslash\{q_Y,q_N\})\times\Gamma\to Q\times\Gamma\times\{l,r\}$$

设Σ^*是由 Σ 中符号组成的长度大于等于 0 的任意串的集合。

DTM 的输入$x\in\Sigma^*$放在从 1 到|x|的带方格中。初始时，所有其它方格中放空白符#。DTM 初始处于q_0状态，读写头指向带方格 1.

$\delta(q,s)=(q',s',\Delta)$表示当前有穷状态控制器处于状态 q，读头所指方格上符号为 s。这时状态改为 q'，读头所指的方格写上 s'，再根据Δ决定读写头左移($\Delta=l$)或右移($\Delta=r$)一格。

运行到某一步，当$q=q_Y$时，DTM 给出答案“是”并停机；当$q=q_N$时，DTM 给出答案“否”并停机。

3.例子：给出一个 DTM

$$\Gamma=\{0,1,\#\},\Sigma=\{0,1\},Q=\{q_0,q_1,q_2,q_3,q_Y,q_N\}$$

δ 函数用下表给出：$\delta(q,s)$：

q	0	1	#
q_0	$(q_0,0,r)$	$(q_0,1,r)$	$(q_1,\#,l)$
q_1	$(q_2,\#,l)$	$(q_3,\#,l)$	$(q_N,\#,l)$
q_2	$(q_Y,\#,l)$	$(q_N,\#,l)$	$(q_N,\#,l)$
q_3	$(q_N,\#,l)$	$(q_N,\#,l)$	$(q_N,\#,l)$

设$\alpha,\beta\in\Gamma^*,a\in\Gamma$，我们定义 DTM 的格局($\alpha qa\beta$)表示当前图灵机处于 q 状态，带上的符号串为$\alpha a\beta$，当前读写头指向 a 所在的这个方格。

上例中 DTM 对于输入 10100 的计算过程可表示如下：

$$(q_010100)\Rightarrow(1q_00100)\Rightarrow(10q_0100)\Rightarrow(101q_000)\Rightarrow(1010q_00)$$
$$\Rightarrow(10100q_0\#)\Rightarrow(1010q_10\#)\Rightarrow(101q_20\#\#)\Rightarrow(10q_Y1\#\#\#)$$

4.DTM 接受的语言

$\forall x\in\Sigma^*$，DTM M接受 x 当且仅当输入 x 时，M 停机在状态q_Y。

M 识别的语言L_M为：$L_M=\{x\mid x\in\Sigma^*$且 M 接受 x$\}$。

在上例中：$L_M=\{x\mid x\in\{0,1\}^*$且 x 最后两位为 00$\}$。

DTM 输入 x 后，有三种可能：
(1)停机在q_Y状态；
(2)停机在q_N状态；
(3)永远不停机。
*对应算法的 DTM 程序对任何输入，它都停机。

二．判定问题

1.定义：由于任何事物都可以用 0 和 1 编码，一个判定问题可以描述如下：已知$L \subseteq \{0,1\}^*$, 对于 $x \in \{0,1\}^*$, 若 $x \in L$, 则给出答案“是”；若
$x \notin L$, 则给出答案“否”。
2.例子：
给定一个正整数 N，问是否有正整数 m 使 N = 4m?
在标准编码方案下，整数 N 用二进制数表示。一个整数可被 4 整除，当且仅当它的二进制数最后两位是 00。故上例中的 DTM 可解本判定问题。

三．P 问题类

1.图灵机的时间复杂性
确定型图灵机对输入 x 的时间复杂性指的是，从开始到停机为止的运行步数。
T_M：$Z^+ \to Z^+$
$T_M(n) = \max\{m \mid$ 对 $x \in \Sigma^*, |x| = n$, M 对输入 x 计算需要时间 $m\}$
如果存在多项式 p，使得$\forall n \in Z^+$, 有$T_M(n) \le p(n)$，则图灵机(程序)
M 叫做多项式时间的图灵机(程序)。
2.P 问题类
P = {L | 有多项式时间的 DTM 程序 M 使得$L = L_M$}。

§1.4 非确定型图灵机

一．非确定图灵机的定义

一个非确定型图灵机 NDTM M 包括：
(1) Γ, Σ 与 DTM 相同
(2) Q, q_0, q_Y, q_N与 DTM 相同
(3) 转移函数
δ: $(Q\backslash\{q_Y, q_N\})\times\Gamma \to 2^{Q\times\Gamma\times\{l,r\}}$
*解释δ的多值性，一次可做多种运算。

二．非确定图灵机的另一种定义

NDTM 也可以看作除了多一个“猜想模块”外，其它与 DTM 相同。这个“猜想模块”带

有“猜想头”，可对带写入“猜想”。

计算分为两个阶段，一是猜想阶段，一是检验阶段。对任一输入，NDTM 首先给出(无穷多种猜想中的)一个猜想，然后按 DTM 的动作验证该猜想是否答案。检验阶段若停在q_Y状态，则回答“是”。检验阶段也可能停在q_N状态或不停机。

三．NP 问题类

NDTM 对给定的输入字符串 x 有无穷多种可能的计算，对Γ^*中每一个可能的猜想字符串有一个计算。如果这些计算中至少有一个是接受的计算(停在q_Y状态)，则称这个 NDTM 接受 x。

NDTM 识别的语言为：$L_M = \{x \mid x \in \Sigma^* \text{且 M 接受 } x\}$

NDTM M 接受字符串 x 所需的时间定义为：M 关于 x 的所有接受计算中猜想阶段和检验阶段直到进入停机状态q_Y时为止所需的步数的最小值。

T_M：$Z^+ \to Z^+$

$T_M(n) = \max\{1, m \mid \forall x \in L_M, |x| = n, M \text{ 接受 } x \text{ 的时间为 } m\}$。

若存在多项式 p，使得$\forall n \in Z^+$有$T_M(n) \le p(n)$，那么这个 NDTM M是一个多项式时间的 NDTM。

NP = {L| 存在多项式时间的 NDTM M, 使得$L_M = L$}。

*NP 问题类是多项式时间可验证解的问题类。

§1.5 P 问题与 NP 问题的关系

一．P 与 NP 的关系

1. $P \subseteq NP$

确定图灵机是特殊的非确定图灵机，故$P \subseteq NP$。

2.用确定图灵机解决 NP 问题

定理 1.1：如果$\pi \in NP$，那么存在一个多项式 p 使得π能用时间复杂性为$O(2^{p(n)})$的确定型算法解决。

证明：设非确定图灵机字母表为Γ，且$|\Gamma| = k$。对输入 x，其长度为$|x| = n$。若非确定型图灵机在不超过 q(n)步内对 x 给出肯定的判断，则 NDTM 给出的猜想长度不超过 q(n)。由于猜想每位的字母有 k 种可能，故猜想的全体不超过$k^{q(n)}$个。每个猜想在$q(n)$步内检验完毕，故全体检验完毕的时间复杂度以$q(n)k^{q(n)}$为上界。适当选取多项式 p，则$q(n)k^{q(n)}$为$O(2^{p(n)})$。

*人们普遍认为$P \neq NP$, 即有 (见图 1.3)

*但以上结论无法证明

二．P,NP 与 co-NP 的关系

1.co-NP 的定义

由 NP 类的定义，我们不知$L \in NP$是否蕴含 L 的补集$\bar{L} \in NP$。定义 $co-NP = \{L \mid \bar{L} \in NP\}$。

2. $P, NP, co-NP$关系的几种可能性

(见图 1.4)

注意有以下结论：
1. $P \subseteq co-NP$
2.如果$NP \neq co-NP$, 则 $P \neq NP$
*人们认为情形(a)可能性最小，(d)可能性最大。

三．多项式变换和 NP 完全性

1.多项式变换

从语言$L_1 \subseteq \Sigma_1^*$到语言$L_2 \subseteq \Sigma_2^*$的多项式变换是满足下述两个条件的函数$f: \Sigma_1^* \to \Sigma_2^*$：
(1)存在计算 f 的多项式 DTM;
(2)$\forall x \in \Sigma_1^*, x \in L_1$当且仅当 $f(x) \in L_2$。
如果存在一个从L_1到L_2的多项式变换，则记作$L_1 \propto L_2$。

2.多项式变换的几个性质

定理 1.2：如果$L_1 \propto L_2$，那么$L_2 \in P$蕴含$L_1 \in P$。

证明：设Σ_1和Σ_2分别是L_1和L_2的字母表，$f: \Sigma_1^* \to \Sigma_2^*$是从$L_1$到$L_2$的多项式变换。设$M_f$为计算 f 的多项式时间的 DTM，$M_2$为识别$L_2$的多项式时间的 DTM。构造识别$L_1$的多项式时间的 DTM M_1如下：对输入$x \in \Sigma_1^*, M_1$首先按M_f方法计算$f(x) \in \Sigma_2^*$，然后按M_2方法确定是否$f(x) \in L_2$。因为$x \in L_1$当且仅当 $f(x) \in L_2$，故M_1是识别L_1的 DTM。如果p_f和p_2是限制M_f和M_2运行时间的多项式函数，那么 $|f(x)| \leq p_f(|x|)$，M_1的运行时间为$O(p_f(|x|) + p_2(p_f(|x|)))$，它不超过$|x|$的某个多项式。证毕。

定理 1.3：若$L_1 \propto L_2$且$L_2 \propto L_3$，那么$L_1 \propto L_3$。

证：设Σ_1, Σ_2和Σ_3分别是语言L_1, L_2和L_3的字母表，$f_1: \Sigma_1^* \to \Sigma_2^*$是从$L_1$到$L_2$的多项式变换，$f_2: \Sigma_2^* \to \Sigma_3^*$是从$L_2$到$L_3$的多项式变换。$f: \Sigma_1^* \to \Sigma_3^*$定义为：$\forall x \in \Sigma_1^*, f(x) = f_2(f_1(x))$，那么，f 是从$L_1$到$L_3$的多项式变换。显然，$f(x) \in L_3$当且仅当$x \in L_1$。可以用类似定理 1.2 的证明，证明可用多项式时间的 DTM 计算 f。

3.多项式等价性

若$L_1 \propto L_2$且$L_2 \propto L_1$，则称L_1与L_2是多项式等价的。
多项式等价性是一个等价关系。P 类是其中的一个等价类。

4.NP 完全类(NPC)

如果语言$L \in NP$，并且对所有其它语言$L' \in NP$,有 $L' \propto L$，则 L 称为是$NP-$完全的。
*如果判定问题$\pi \in NP$，并且对所有其它判定问题$\pi' \in NP$,有 $\pi' \propto \pi$，则π 是NP-完全的。
NP-完全问题是 NP 中最难的问题。
*NP 与 P 的关系图可描述如图 1.5
*如果$P \neq NP$，则 NP\P≠ NPC。

5.NP-完全问题的证明方法

定理 1.4：如果L_1和L_2属于 NP，L_1是 NP-完全的，且$L_1 \propto L_2$，那么L_2也是 NP-完全的。

证明：因为$L_2 \in NP$，只要证：$\forall L' \in NP, L' \propto L_2$即可。由于$L_1$是
NP-完全，$L' \propto L_1$，且$L_1 \propto L_2$。由∝的传递性，$L' \propto L_2$。
*定理 1.4 给出了证明一个问题是 NP-完全的方法：
(1) $\pi \in NP$
(2) 某个已知的 NP-完全问题π'可多项式变换到π。