第八章

NP完全性理论简介

Theory of NP-completeness

理解图灵机概念和计算模型

理解非确定性图灵机的概念

理解P类与NP类语言的概念

理解NP完全问题的概念

何为“好的”算法？

- Cobham[1964]和Edmonds[1965]首先加以区别。Edmonds把多项式时间算法与“好的”算法等同看待，并猜想某些整数规划问题可能不能用这种“好的”算法求解。
- 一般观点：认为指数时间算法不应该算作“好的”算法。
  - 大多数指数时间算法只是穷举搜索法的变种
  - 多项式时间算法通常只有在对问题的结构有了某些比较深入地了解之后才有可能给出

什么问题“易计算”？“难计算”？

- 只有用多项式时间算法解决的问题才能认为“易计算”的。
- 一般，如果一个问题困难到不可能用多项式时间算法求解，则认为这个问题是“难解的”。

上述按能否用“多项式时间算法解决问题”来理解一个问题是否“易计算”或“难解的”是不深刻的。

有关时间复杂性问题的探讨

- **时间复杂性是一种最坏情况的度量**。时间复杂性为2^n的算法仅仅表示至少有一个规模为n的问题实例需要这么多的运算时间，而大多数问题实例可能需要远比2^n少得多的时间。
- 例如：几个著名的算法

- 线性规划的**单纯形算法**：已经证明具有指数时间复杂性[Klee & Minty，1972]，但是在实际中它计算得很好。
- **背包问题**的**分支限界算法**：虽然也具有指数时间复杂性，但是它是一种非常成功的算法，使得许多人认为背包问题已经很好地解决了。

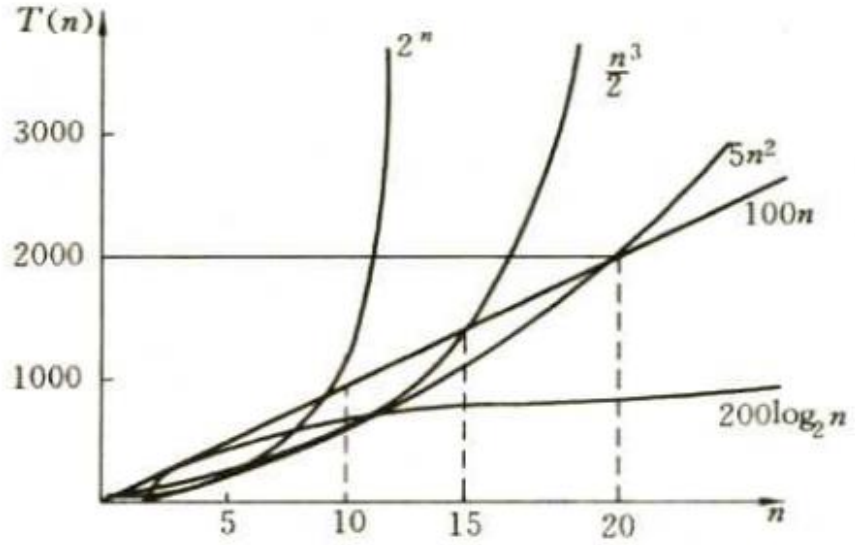
•注：具有指数时间但计算得很好的例子很少。

多项式阶算法(有效算法)：若算法的最坏情形时间复杂度$T(n)=O(n^k)$；**P（Polynomial,多项式）类问题**

指数阶算法：若算法的最坏情形时间复杂度$T(n)=\Omega(a^n)$，$a>1$. 这类算法可认为计算上不可行的算法.

NP（Non-Deterministic Polynomial, 非确定多项式）问题

常见时间复杂度的增长率



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)

$$O(1)<O(\log_2 n)<O(n)<O(n\log_2 n)<O(n^2)<O(n^3)<O(2^n)<O(n!)<O(n^n)$$

探讨（续）

- 一些问题尽管有指数计算时间，但算法很成功。研究人员没有停止继续寻找该类问题的多项式时间算法的努力。
- 疑问：出现上述现象的问题其关键性的性质是什么？

对这些性质的仔细研究可能给出更好的方法，但至今在解释这种成功方面几乎毫无进展，也没有一种方法能够事先预言给定的指数时间算法在实际中能否快速运算。

NP完全理论的基本概念

一. 判定问题和语言

- **判定问题**:判定问题π的全体实例集合记作Dπ, 肯定实例集记作Yπ.

例 TSP问题: 设城市集合v={v1,v2,…vm }, c=(cij) 为代价矩阵,要求找一条周游路径使路径长度最小.

与此相应的判定问题可描述为:

城市集合v={v1,v2,…vm },c=(cij)代价矩阵,对给定限界b,
问: 是否存在周游路线,其路径长度<=b ?

当m和cij的值确定后,就得到问题的一个实例.

- **语言**：设Σ是有穷符号的集合(字母表)，Σ上的所有有限长度字符串的集合记作Σ^*，则$\forall\ L\subseteq\Sigma^*$，称为Σ上的一个语言

例如：　Σ ={ 0,1}，L={ 11^*01^* }是Σ上的一个语言.

- **判定问题与语言的关系**：

判定问题可通过编码将问题的一个实例与一个字符串对应起来，而全部实例和字符串的集合(语言)对应起来.

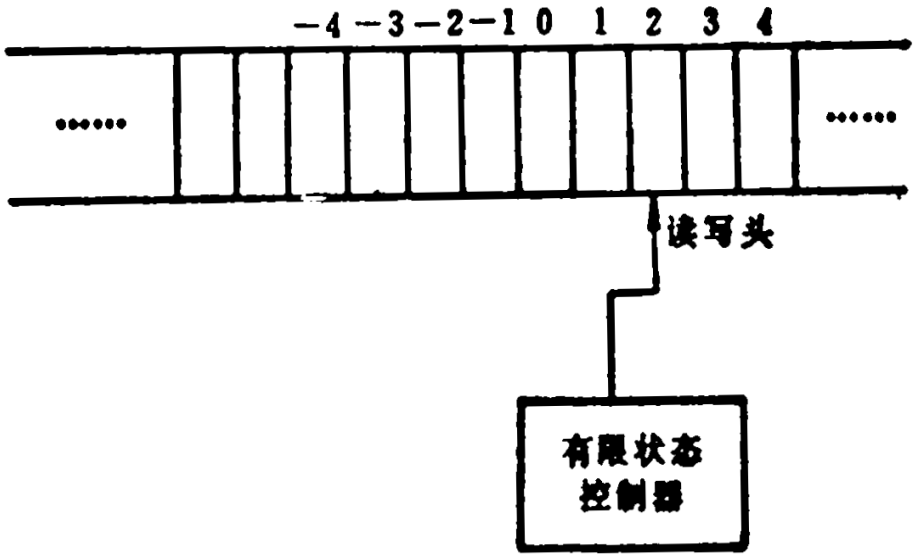
设问题π的编码系统为e，Σ为e所使用的符号集，则与π，e相关联的符号语言为：

$$L(\pi, e)=\{\omega \mid \omega\in\Sigma^*, \Sigma\text{为e的符号集}, \omega\text{是实例在e下的编码}\}$$

二. 确定型图灵机(DTM)和P类问题

DTM可表示为一个七元组:

$(Q, \Sigma, \Gamma, \delta, \_, q_0, q_t)$:



Q: 状态的集合(有限)

Γ: 带符号的集合(有限)

Σ: 输入符号的集合,$\Sigma \subset \Gamma$

_: 空白符, $\_ \in \Gamma - \Sigma$

q0:初始状态, qt: 停机状态(包括接受态q_Y和拒绝态q_N),

$q_0, q_Y, q_N \in Q$.

δ: 状态转移函数,

$\delta(q,a)=(q',(a',d))$ 表示图灵机从q状态,磁头指向a时,转换为q'状态,符号a改为a',并按方向d移动磁头.

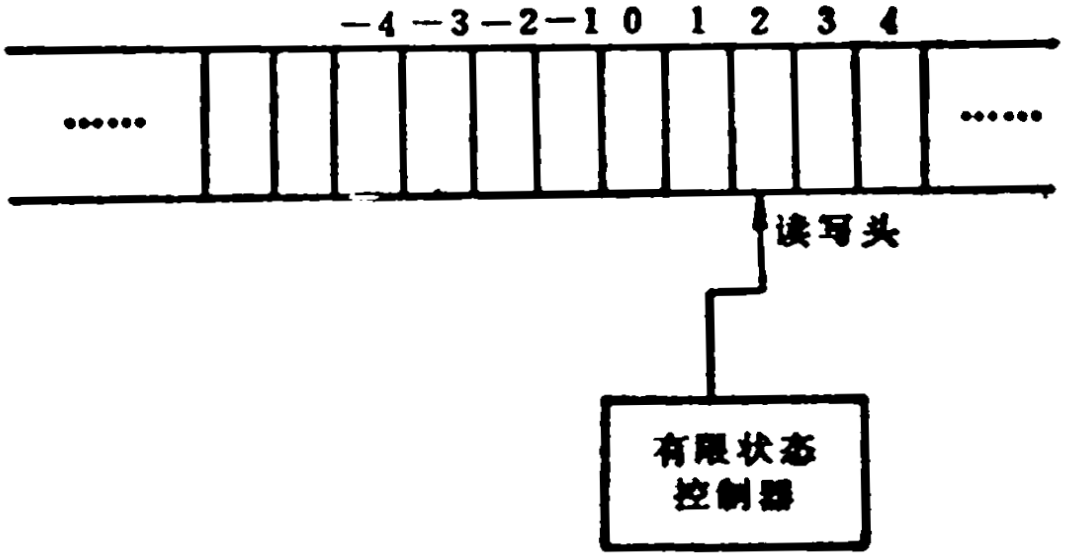
当图灵机处于状态q;磁头指向a时,可依次完成如下的一步计算:

1) 将图灵机的当前状态q改为q';

2) 把当前磁头指向的符号a清除并写上新的带符号a';

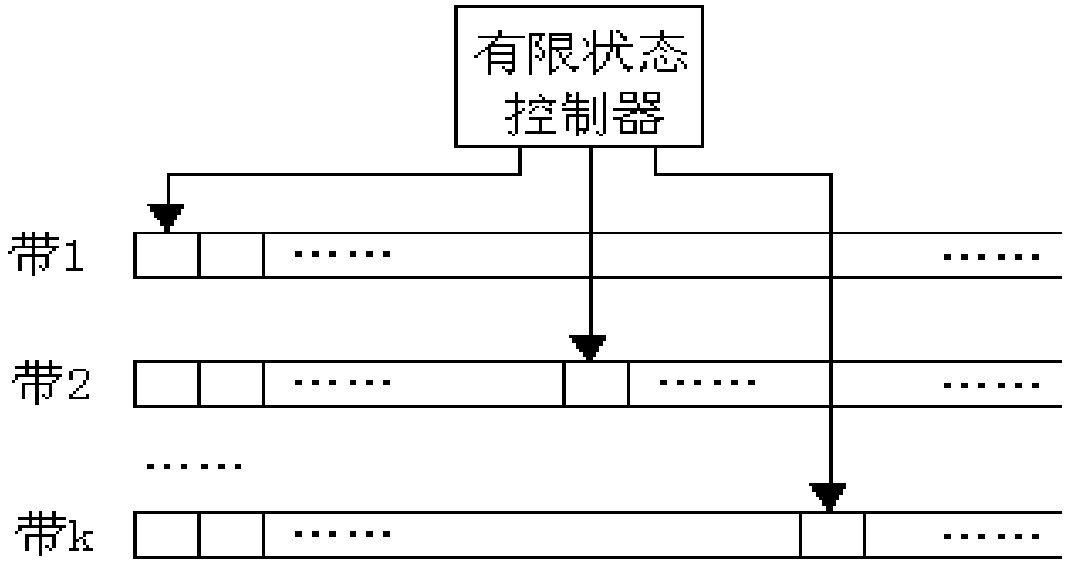
3) 按d指出的方向移动读写头:

d=L:左移一格; d=R:右移一格; d=S 不动.



TM功能很强大

许多TM能够完成复杂的工作。虽然TM只有一条（或k条）连续的带，在每一步也只能完成非常有限的工作。但是，可以设计出TM完成在普通计算机上能够完成的任何计算，尽管可能要慢得多。



结构：有限状态控制器、k条读/写头（双向移动）、k条读写带

DTM与语言识别

当DTM开始计算时，带上放一输入符号串ω，位置1-|ω|，其他为空白，当且仅当图灵机从指定的初始状态q_0开始，经过一系列的计算步后最终进入接受状态q_Y时，称DTM接受了这个ω，一台DTM所能接受的输入符号串全体称作该DTM能识别的语言，记作

$$L_M=\{\ \omega \in \Sigma^* \mid \text{DTM接受}\omega\ \}$$

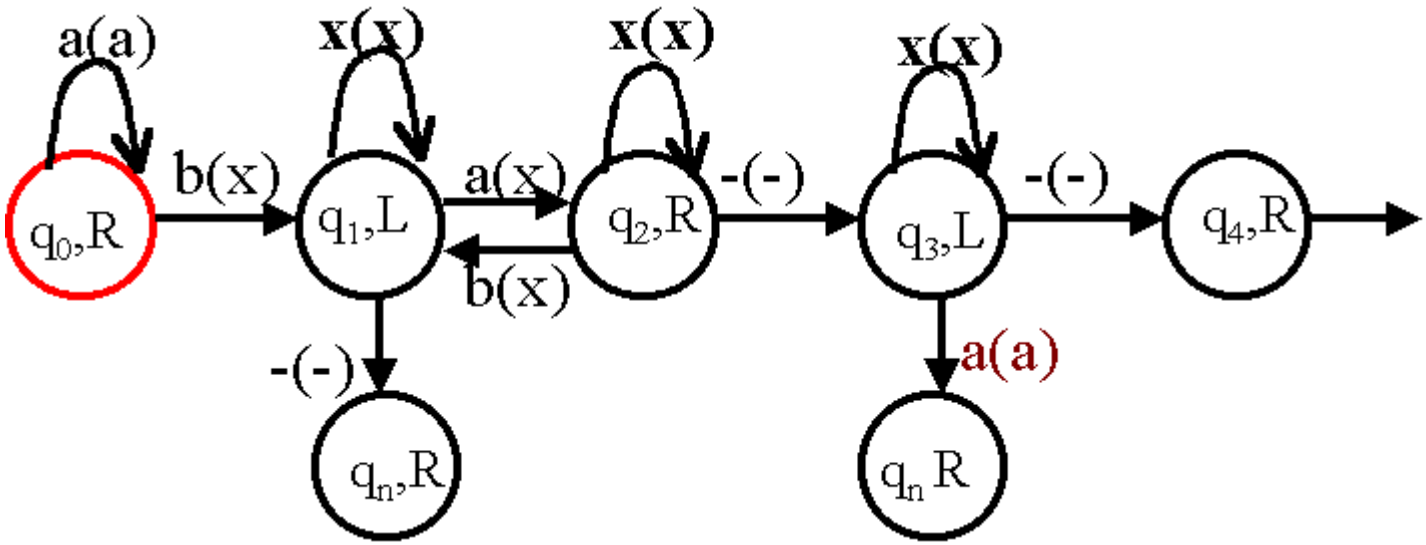
P类语言（问题）：

$$P=\{\ L \mid \exists \text{多项式时间的图灵机}M, \text{使得}L=L_M\ \}$$

例 题 设计一台DTM,它能接受语言 $L=\{a^n b^n \mid n=1,2,\ldots\}$

算法思路: 语言中的串 ω: ab, aabb , aaabbb,...

读写头向右移动,遇到第一个b,将其改为x,再左移到第一个a改为x,反复进行下去,直到整个串变成x…x, DTM接受ω,否则拒绝.该过程的状态转移图为:



$Q=\{q0,\ldots q4, q_N\}$, $\Sigma=\{a,b\}$, $\Gamma=\{a, b\ x, -\}$, $Q_f=\{q_N,q4\}$

$\delta(q0,a)=(q0, a, R)$, $\delta(q0, b)=(q1,x, l)$,$\delta(q1, a)=(q2,x, R)$,...

三.非确定型图灵机(NDTM)和NP类问题

尚无多项式算法的问题可用非确定型图灵机描述.

NDTM:一台非确定型图灵机表示为一个七元组:

$$(Q, \Sigma, \Gamma, \delta, \_, q_0, q_t)$$

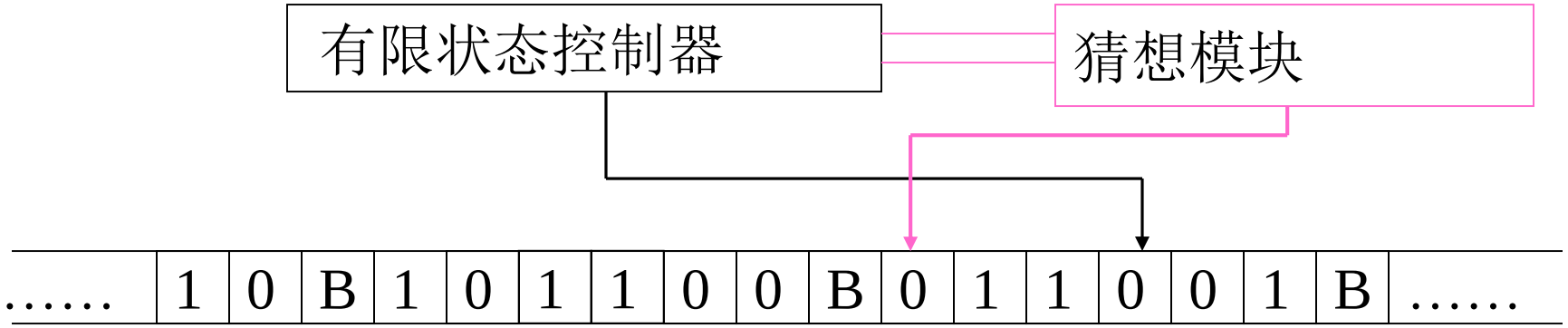
各成分与DTM相同,但δ不再是单值函数,而NDTM对输入串$\omega \in \Sigma^*$的计算过程分为两个阶段:

1.猜想阶段

开始时在带方格的**1-|ω|**处写入输入串,读写头位于方格**1,** 有限状态控制器处于休眠状态. 此时有一猜想模块工作.通过它的只写头的活动给出Γ^*中的任一字符串(一种猜想),然后猜想模块停止活动进入休眠,而有限状态控制器启动进入q_0.

2. 检查阶段

当有限状态控制器处于q_0时，该阶段开始. 此时NDTM的活动与DTM相同，不同的是它对猜想阶段写下的字串进行计算，当有限状态控制器进入q_f时，计算停止，q_r为接受，q_n为拒绝.



NDTM 对给定的输入串ω有无穷多可能的猜想,而对每一个猜想字符串只有一个计算,如果这些计算中至少有一个是可接受计算,称此NDTM接受ω, 此NDTM可识别的语言为: $L_M=\{ \omega \in \Sigma^* \mid M接受\omega \}$

NDTM对输入串ω的所有接受计算中,在猜想阶段和检查阶段直到进入停机状态所需的最小计算步称为M接受ω所需时间$T_M(n)$.如果存在多项式P使得对$\forall n \geq 1$,有$T_M(n) \leq P(n)$,称此NDTM为多项式时间的NDTM.

NP类语言

NP={L|存在多项式时间的NDTM使得LM=L}

如果$L(\pi,e) \in NP$ 则称判定问题属于NP.

四. P与NP类的关系

P类和NP类语言的定义：

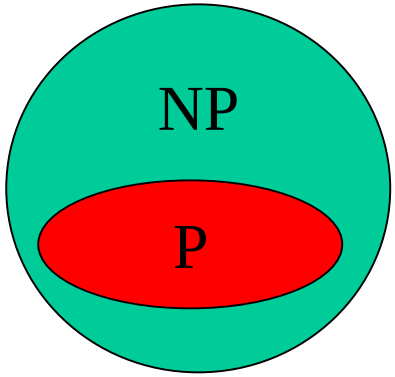
P={L|L是一个能在多项式时间内被一台DTM所接受的语言}

NP={L|L是一个能在多项式时间内被一台NDTM所接受的语言}

世界难题之一　**P=NP?**

显然有$P \subseteq NP$，但尚不能证明$P \neq NP$.

由于一台确定性图灵机可看作是非确定性图灵机的特例，所以可在多项式时间内被确定性图灵机接受的语言也可在多项式时间内被非确定性图灵机接受。故$P \subseteq NP$。



五. 多项式变换和NP完全性

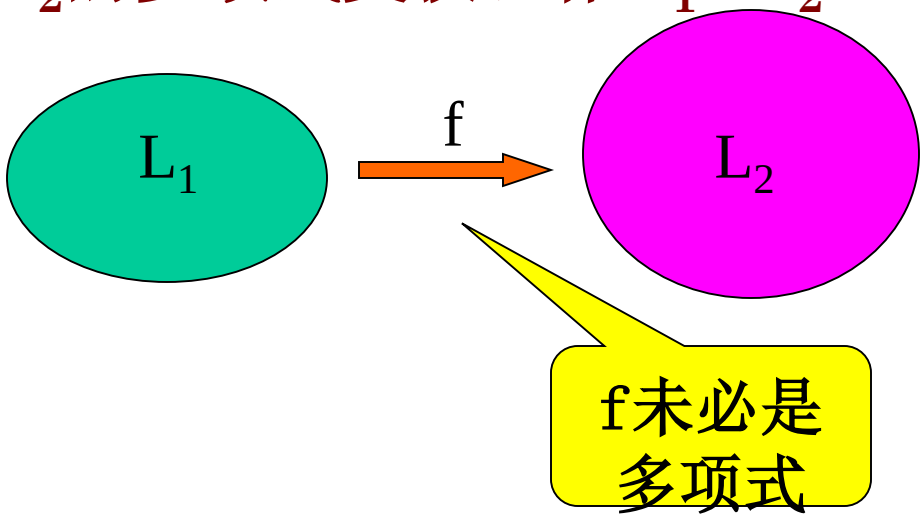
1. 多项式变换

多项式规约:从语言$L_1 \subseteq \Sigma_1^*$到$L_2 \subseteq \Sigma_2^*$的多项式变换是指存在满足下述两个条件的函数f

1) 存在计算f的多项式时间的DTM

2) 任意的$\omega \in \Sigma_1^*$, $\omega \in L_1 \Leftrightarrow f(\omega) \in L_2$

从L_1到L_2的多项式变换记作$L_1 \propto L_2$



2. NP完全性(NPC)

定义：语言L是NP完全的当且仅当

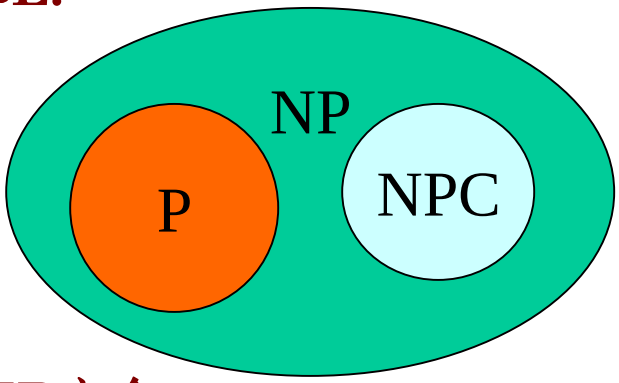
(1)$L \in NP$；

(2)对于所有$L' \in NP$，有$L' \propto_p L$。

定义：语言L是NP难的，如果语言L满足上述性质(2)，但不一定满足性质(1)

定义：所有NP完全语言构成的语言类称为NP完全语言类，记为NPC。

L称为NP完全的,若$L \in NP$ 且 对 $\forall L' \in NP$ 有 $L' \propto L$.



由此得如下性质:

1).若$L_1 \propto L_2$且$L_2 \propto L_3$则$L_1 \propto L_3$

2).若$L_1 \propto L_2$且$L_2 \in P$则$L_1 \in P$

3).若L_1是NP完全的,则$L_1 \in P \Rightarrow P=NP$

4).若$L_1, L_2 \in NP$, $L_1 \in$ NP完全且$L_1 \propto L_2$,则$L_2 \in$ NP完全.

2. NP完全性(NPC)

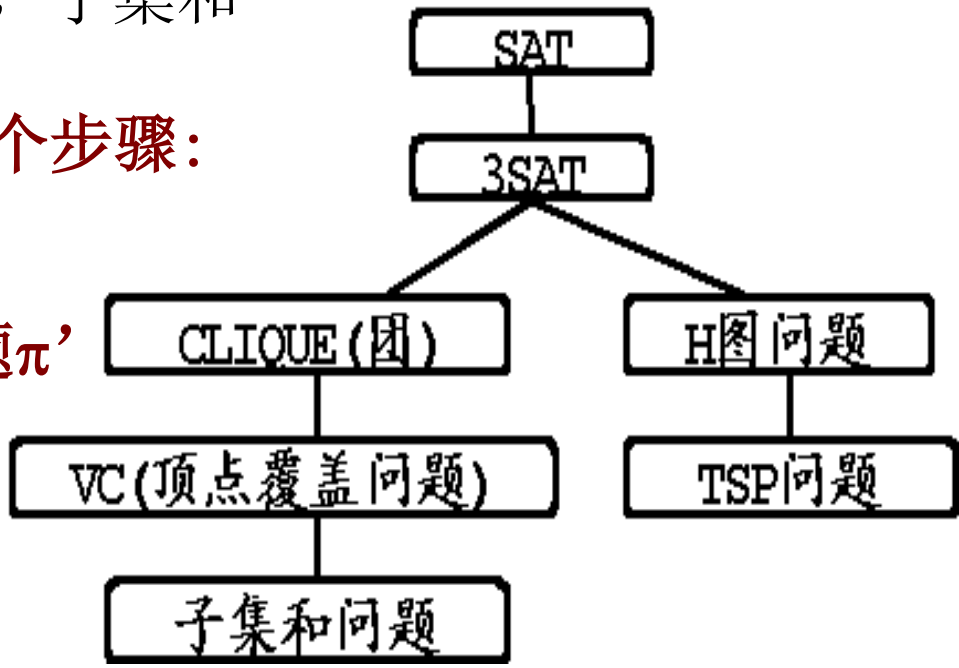
- ✓ 第一个NP完全问题（Cook定理　1971）
  可满足性问题是NP完全问题
- ✓ 六个NP完全问题（Karp 1972)
  3SAT，VC，团，H图，TSP，子集和

证明问题π是NP完全问题包括三个步骤:

1). 证明问题π∈NP

2). 选择一个已知的NP完全问题π'

3). 证明π' ∝π



COOK定理给出了第一个NP完全问题SAT,任何判定问题π,只要能证明π∈NP且SAT∝π,则π∈NPC.所有已知的NPC问题构成了以SAT为根的NP完全问题树.

其它算法…

机器学习五大流派

①符号派：使用符号、规则和逻辑来表征知识和进行逻辑推理，
如：规则、决策树…
②贝叶斯派：获取发生的可能性来进行概率推理，
如：朴素贝叶斯、马尔可夫…
③联结派：使用概率矩阵和加权神经元来动态地识别和归纳模式，
如：神经网络、深度学习..
④进化派：生成变化，然后为特定目标获取其中最优的，
如：遗传算法、蚁群算法、量子粒子群算法…
⑤判别派：根据约束条件来优化函数，
如：支持向量机

计算模式：
分布式计算
并行计算
DNA计算
量子计算
…

派别	起源	擅长算法
符号主义（Symbolists）	逻辑学、哲学	逆演绎算法（Inverse deduction）
联结主义（Connectionists）	神经科学	反向传播算法（Backpropagation）
进化主义（Evolutionaries）	进化生物学	基因编程（Genetic programming）
贝叶斯派（Bayesians）	统计学	概率推理（Probabilistic inference）
Analogizer	心理学	核机器（Kernel machines）

期末复习

复习内容：

1教材基本内容

2课件、练习、作业…

3网上的BAT、华为算法笔试面试题（跟教材所讲算法策略相关，10分左右，算法分析、或代码实现、原题）

考试题型：

1填空题15

2选择题20

3问答题15

4算法分析题25（写出分析过程和最优解、或关键语句填空）

5算法设计实现题25（写出设计方法、思路或步骤，写出核心函数或代码段）