计算复杂性
Computational Complexity

刘显敏
liuxianmin@hit.edu.cn

海量数据
计算研究中心
HARBIN INSTITUTE OF TECHNOLOGY

计算理论—2023年春

什么是计算复杂性?

- 可计算性理论 ⇒ 问题是否可判定，是否存在算法
- 然而，解决该问题的**最优**的图灵机(算法)也许会需要**天文级别的计算资源**(如时间、空间等)

计算复杂性理论 ⇒ 问题是否存在高效算法

- 计算复杂性理论关注问题是否实际可解(tractable)
- 实际可解不是指算法存在，而是高效算法存在!

如何度量计算复杂性
算法的复杂性→问题的复杂性

如何度量图灵机算法的复杂性?

回顾 为了判定语言$A = \{0^n1^n | k \geq 0\}$，图灵机$M_1$如下

➤扫描带，如果0出现在1的右边，拒绝

➤如果带上仍然有0或1，重复下列过程

- 扫描带一遍，将一个0和一个1删除
- 如果带上只剩下0或者1，拒绝
- 否则，接受

时间代价 | **空间代价** | **通信代价** | **...**

如何度量C语言程序的复杂性?

算法的复杂性→问题的复杂性

算法的复杂性度量

最坏(worst-case)情况复杂性(以时间代价为例)

①图灵机在单个输入$x \in \Sigma^*$上运行时间$f_1: \Sigma^* \to \mathbb{N}$

②图灵机在所有长度为n的输入$x \in \Sigma^n$上最长运行时间

$$f_2(n) = \max_{x \in \Sigma^n} f_1(x)$$

定义 计算(时间)复杂性, computational (time) complexity

令M是在所有输入上均停机的图灵机(即算法)，M的计算(时间)复杂性表示为函数$f: \mathbb{N} \to \mathbb{N}$，这里$f(n)$是算法$M$在所有长度为$n$的输入上计算所需的**最大计算资源量(最大计算步数)**。

如果M的时间复杂性是$f(n)$ ，那么我们说M的运行时间是$f(n)$ ，M是一个$f(n)$时间图灵机。

算法的复杂性度量

渐进分析方法(Asymptotic Analysis)

定义 O记号
令$f,g:\mathbb{N}\to\mathbb{R}^+$是两个函数，我们说$f(n)=O\big(g(n)\big)$，如果有正整数$c$和$n_0$使得对任意$n\ge n_0$有$f(n)\le c\cdot g(n)$

- $f(n)=O\big(g(n)\big)$，称$g(n)$是$f(n)$的上界(upper bound)
- 更准确，$g(n)$是$f(n)$的一个渐进上界(asymptotic upper bound)，即忽略掉常系数影响

例
- $5n^3+2n^2+22n+6=O(n^3)$
- $\log_b n=\frac{\log_2 n}{\log_2 b}=O(\log n),(b\ge 2)$
- $2^{10n^2+7n-6}=2^{O(n^2)}$
- $n^c\ (c>0)$是多项式阶，2^{n^ϵ}是指数阶

算法的复杂性度量

渐进分析方法(Asymptotic Analysis)

定义 o记号
令$f,g:\mathbb{N}\to\mathbb{R}^+$是两个函数，我们说$f(n)=o\big(g(n)\big)$，如果有$\lim\limits_{n\to\infty}\frac{f(n)}{g(n)}=0$，即对任意实数$c>0$存在整数$n_0$使得对所有$n\ge n_0$有$f(n)<c\cdot g(n)$

例
- $\sqrt{n}=o(n)$
- $n=o(n\log\log n)$
- $n\log\log n=o(n\log n)$

算法的复杂性度量

M_1的时间代价分析
- 第1阶段，扫描确认输入是0*1*形式，需要n步，然后，将带头移到左端，需要n步，共计$2n=O(n)$步
- 第2、3阶段，机器反复左右扫描带，每次扫描都会删除一个0和1，每次扫描使用$O(n)$步，最多需$n/2$次扫描，共计$\left(\frac{n}{2}\right)\times O(n)\ =\ O(n^2)$步
- 第4阶段，扫描确认接受还是拒绝，共需$O(n)$步

时间代价共计$O(n)+O(n^2)+O(n)=O(n^2)$

算法的复杂性度量

渐进分析方法(Asymptotic Analysis)

定义 Ω记号
令$f,g:\mathbb{N}\to\mathbb{R}^+$是两个函数，我们说$f(n)=\Omega\big(g(n)\big)$，如果有正整数$c$和$n_0$使得对所有$n\ge n_0$有$f(n)\ge c\cdot g(n)$

定义 Θ记号
令$f,g:\mathbb{N}\to\mathbb{R}^+$是两个函数，我们说$f(n)=\Theta\big(g(n)\big)$，若同时有$f(n)=O\big(g(n)\big)$和$f(n)=\Omega\big(g(n)\big)$，即存在正整数$c_1,c_2$和$n_0$使得$\forall n\ge n_0$有$c_1\cdot g(n)\le f(n)\le c_2\cdot g(n)$

如何理解符号$2^{\Theta(\log_2 n)}$

①$f(n)\in\Theta(\log_2 n)$即$c_1\cdot\log_2 n\le\ f(n)\ \le c_2\cdot\log_2 n$

②$2^{c_1\log_2 n}\le 2^{\Theta(\log_2 n)}\le 2^{c_2\log_2 n}$,即$n^{c_1}\le 2^{\Theta(\log_2 n)}\le n^{c_2}$,即$n^{\Theta(1)}$

确定图灵机的计算复杂性

定义 图灵机的空间复杂性, Space Complexity
给定k带图灵机M(1条输入带，$k-1$条存储带)，若对每个长度为n的输入，M在每条存储带上都至多扫描$S(n)$个单元格，那么M是**$S(n)$空间代价图灵机**，M的空间复杂性(space complexity)为$S(n)$。

定义 图灵机的时间复杂性, Time Complexity
给定k带图灵机M，若对每个长度为n的输入，M至多移动$T(n)$次，那么M是**$T(n)$时间代价图灵机**，M的时间复杂性(time complexity)为$T(n)$。

时间复杂性定义中，若使用单带或者固定带数的图灵机，那么某些可以在$T(n)$时间求解的问题则无法描述

不确定图灵机的计算复杂性

定义 不确定图灵机空间复杂性, nondeterministic space
给定k带不确定图灵机M(1条输入带，$k-1$条存储带)，若对每个长度为n的输入，M的任一可能计算在每条存储带至多扫描$S(n)$个单元格，那么M是**$S(n)$空间代价不确定图灵机**，M的空间复杂性(space complexity)为$S(n)$。

定义 不确定图灵机时间复杂性, nondeterministic time
给定k带不确定图灵机M，若对每个长度为n的输入，M的任一可能计算至多移动$T(n)$次，那么M是**$T(n)$时间代价不确定图灵机**，M的时间复杂性(time complexity)为$T(n)$。

算法的复杂性→**问题的复杂性**

时间复杂性类

将可判定问题根据求解图灵机(算法)的时间上界进行分类

定义 确定时间复杂性(问题)类
所有可被时间复杂性为$T(n)$的确定图灵机判定的问题构成的类被称作$T(n)$-确定时间复杂性类(deterministic time complexity class)，也记作$\boldsymbol{DTIME(T(n))}$时间复杂性类。

定义 不确定时间复杂性(问题)类
所有可被时间复杂性为$T(n)$的不确定图灵机判定的问题构成的类被称作$T(n)$-不确定时间复杂性类(nondeterministic time complexity class)，也记作$\boldsymbol{NTIME(T(n))}$时间复杂性类。

空间复杂性类

将可判定问题根据求解图灵机(算法)的空间上界进行分类

定义 确定空间复杂性(问题)类
所有可被空间复杂性为$S(n)$的确定图灵机判定的问题构成的类被称作$S(n)$-确定空间复杂性类(deterministic space complexity class)，也记作$\boldsymbol{DSPACE(S(n))}$空间复杂性类。

定义 不确定空间复杂性(问题)类
所有可被空间复杂性为$S(n)$的不确定图灵机判定的问题构成的类被称作$S(n)$-不确定空间复杂性类(nondeterministic space complexity class)，也记作$\boldsymbol{NSPACE(S(n))}$空间复杂性类。

复杂性类

- 令Φ是某一函数族，DTIME(Φ)、NTIME(Φ)、DSPACE(Φ)和NSPACE(Φ)均是响应的**复杂性类**

$$DTIME(\Phi) = \cup_{T(n)\in\Phi} DTIME(T(n))$$
$$DSPACE(\Phi) = \cup_{T(n)\in\Phi} DSPACE(T(n))$$
$$NTIME(\Phi) = \cup_{T(n)\in\Phi} NTIME(T(n))$$
$$NSPACE(\Phi) = \cup_{T(n)\in\Phi} NSPACE(T(n))$$

- 令poly表示非负系数多项式函数族，即$O(n^{O(1)})$

$$\boldsymbol{P = DTIME(poly)}$$
$$\boldsymbol{NP = NTIME(poly)}$$
$$\boldsymbol{PSPACE = DSPACE(poly)}$$
$$\boldsymbol{NPSPACE = NSPACE(poly)}$$

复杂性类

其他重要的复杂性类

$$LOGSPACE = DSPACE(\log n)$$
$$NLOGSPACE = NSPACE(\log n)$$
$$EXP = \bigcup_{c>0} DTIME(2^{cn})$$
$$NEXP = \bigcup_{c>0} NTIME(2^{cn})$$
$$EXPSPACE = \bigcup_{c>0} DSPACE(2^{cn})$$
$$NEXPSPACE = \bigcup_{c>0} NSPACE(2^{cn})$$

计算复杂性理论

☞问题的复杂性分类

☞复杂性类间的关系

计算模型与复杂性类

一个重要问题：计算资源模型相关(model-depedent)

例如，$L=\{0^k1^k|k\geq 0\}$

➤在单带图灵机上，时间代价$O(n\log n)$

➤在双带图灵机上，时间代价$O(n)$

用单带图灵机，与多带图灵机或RAM等模型有多大**差别**

- 若在单带图灵机上，$L\in TIME(f(n))$，那么在其它模型上有$L\in TIME(g(n))$，满足$g(n)=O\left(poly\left(f(n)\right)\right)$
- 不同计算模型上，特定计算问题L的时间复杂性是多项式相关(polynomial related)

带压缩定理

- 图灵机的状态和带符号的数量可以任意大
- 将多个符号编码为一个，计算空间可以压缩

定理 带压缩(Tape Compression)定理I
如果语言L被$S(n)$空间代价的k-带图灵机判定，那么对任意$c>0$，存在$c\cdot S(n)$空间代价的图灵机判定L。

证明概要 不失一般性，令$c<1$，r满足$rc\geq 1$，新图灵机每个单元存储M的r个相邻单元内容(带压缩)

定理 带压缩(Tape Compression)定理II
对于任意$c>0$，

$$DSPACE\big(S(n)\big)=DSPACE\big(c\cdot S(n)\big)$$
$$NSPACE\big(S(n)\big)=NSPACE\big(c\cdot S(n)\big)$$

空间复杂性类间的关系

不同带数对应的空间复杂性类之间的归约

定理 单带存储定理
若语言L被k-带图灵机($k-1$条存储带)以空间代价$S(n)$判定，存在2-带图灵机(1条存储带)以空间代价$S(n)$判定L。

证明概要 利用多track技术模拟

- 以后提及空间复杂性，图灵机是2带图灵机
- 如果$S(n)\geq n$，不失一般性，图灵机为单带图灵机

线性加速定理

定义 limitation of the least upper bound
$\sup_{n\to\infty}f(n)$是$n\to\infty$时序列$f(n),f(n+1),f(n+2),...$的最小上界极限。

定义 limitation of the greatest lower bound
$\inf_{n\to\infty}f(n)$是$n\to\infty$时序列$f(n),f(n+1),f(n+2),...$的最大下界极限。

- 令$f(n)=1/n$(n为偶数)，$f(n)=n$(n为奇数)，则对应的极限$\sup_{n\to\infty}f(n)=\infty$，$\inf_{n\to\infty}f(n)=0$

定理 线性加速(linear speedup)定理I
当$\inf_{n\to\infty}T(n)/n=\infty$且$c>0$时，语言$L$被$k$带$T(n)$时间图灵机判定($k>1$)，则存在$k$带$c\cdot T(n)$时间图灵机判定$L$，即$DTIME\big(T(n)\big)=DTIME\big(c\cdot T(n)\big)$。

线性加速定理

如果$\inf_{n\to\infty}T(n)/n=\infty$不成立，但满足$\inf_{n\to\infty}T(n)/n=c$

定理 线性加速(linear speedup)定理II
当$\inf_{n\to\infty}T(n)/n=c$且$c>0,\epsilon>0$时，语言$L$被$k(k>1)$带$c\cdot n$时间图灵机判定，则存在$k$带$(1+\epsilon)\cdot n$时间图灵机判定$L$。如果$T(n)=c\cdot n(c>1)$，则有

$$DTIME\big(T(n)\big)=DTIME\big((1+\epsilon)\cdot n\big)$$

定理 线性加速(linear speedup)定理III
①如果$\inf_{n\to\infty}T(n)/n=\infty$，对于任意$c>0$，有

$$NTIME\big(T(n)\big)=NTIME\big(c\cdot T(n)\big)$$

②如果$T(n)=c\cdot n$(c是常数)，对于任意$\epsilon>0$，有

$$NTIME\big(T(n)\big)=NTIME\big((1+\epsilon)n\big)$$

时间复杂性间的关系

定理 单带vs多带
若L被k带图灵机以$T(n)$时间判定，那么存在单带图灵机以$(T(n))^2$时间判定L；如果L属于$NTIME\big(T(n)\big)$，那么L可以被单带图灵机以$(T(n))^2$时间判定。

定理 双带vs多带
若L被k带图灵机以$T(n)$时间判定，那么存在2带图灵机以$T(n)\log T(n)$时间判定L；若L属于$NTIME(T(n))$，那么L可以被2带不确定图灵机以$T(n)\log T(n)$时间判定。

真实计算机的抽象模型、标准编程语言编制的程序以及其它合理的计算模型均可以被基本的图灵机以至多多项式时间的额外代价模拟

时间复杂性类-P类

多项式时间复杂性类

P类用来定义高效计算(efficient computing, tractable)

定义 多项式时间复杂性类, P类

多项式时间复杂性类(P类)由所有可以被确定图灵机在多项式时间内判定的问题组成，具体地

$$P = \bigcup_{p \text{ is a poly}} DTIME(p(n)) = \bigcup_{k\geq 0} DTIME(n^k)$$

- P类问题在计算复杂性理论中的角色类似于可判定问题在可计算理论中的角色
  - P类模型无关(reasonable models)、可扩展、易于合成
- P类是否有局限性?
  - 包含了太多实际难解的时间复杂性类，如n^{10000}
  - 无法描述很多易解的时间复杂性类，如average complexity

多项式时间复杂性类

- 所有问题都在P内吗?
  - 显然不是，$P \subseteq$可判定问题，不可判定问题存在
- 所有判定的问题都在P内吗?
  - 也不是

定理 $\text{deci}\setminus\text{DTIME}(t(n)) \neq \emptyset$

对任一可计算函数t，存在可判定问题L，L不在复杂性类$\text{DTIME}(t(n))$内，因此，存在可判定问题L，$L \notin P$。

证明 令$L_{\text{AccT}} = \{\langle M\rangle | M$在$\leq t(|\langle M\rangle|)$步数内接受$\langle M\rangle\}$

①显然，L_{AccT} 是可判定的

- 给定$\langle M\rangle$，模拟M在$\langle M\rangle$上的运行，模拟$t(|\langle M\rangle|)$步

②可证L_{AccT}无法被任意图灵机在$\leq t(|\langle M\rangle|)$步内判定

- 利用对角线技术

多项式时间复杂性类

L_{AccT}无法被图灵机在$\leq t(|\langle M\rangle|)$步内判定

- 令$n = \langle M\rangle$, 假设 L_{AccT}被图灵机M_1在$t(n)$步内判定
- 存在图灵机M_0在$t(n)$步内判定$\overline{L_{AccT}}$，使得
  - 如果$\langle M\rangle \in \overline{L_{AccT}}$，$M_0$在$t(n)$步内接受$\langle M\rangle$
  - 如果$\langle M\rangle \in L_{AccT}$，$M_0$在$t(n)$步内拒绝$\langle M\rangle$
- $\langle M\rangle \in \overline{L_{AccT}}$当且仅当$M_0$在$t(n)$步内接受$\langle M\rangle$
- 依$\overline{L_{AccT}}$定义,$\langle M\rangle \in \overline{L_{AccT}} \Leftrightarrow M$不在$t(n)$步内接受$\langle M\rangle$

判断$\langle M_0\rangle$与$\overline{L_{AccT}}$关系，构造矛盾

- $\langle M_0\rangle \in \overline{L_{AccT}} \Rightarrow M_0$不在$t(n)$步接受$\langle M_0\rangle \Rightarrow \langle M_0\rangle \in L_{\text{AccT}}$
- $\langle M_0\rangle \in L_{AccT} \Rightarrow M_0$在$t(n)$步内接受$\langle M_0\rangle \Rightarrow \langle M_0\rangle \in \overline{L_{AccT}}$

P类 vs NP类

P和NP类

定义 不确定多项式时间复杂性类, NP类

不确定多项式时间复杂性类(P类)由所有可以被不确定图灵机在多项式时间内判定的问题组成，具体地

$$P = \bigcup_{p \text{ is a poly}} NTIME(p(n)) = \bigcup_{k\geq 0} NTIME(n^k)$$

- $P = \{L|$确定型图灵机在多项式时间判定$L\}$
- $NP = \{L|$不确定型图灵机在多项式时间判定$L\}$

定义 NP的另一种定义

语言$L \in NP$当且仅当存在一个L的多项式时间**检测器**(verifier) V，满足条件：$x \in L$当且仅当$\exists c, |c| \leq poly(|x|)$使得$V(x, c) = 1$。

- 显然，$P \subseteq NP$
- 然而，并不知道$P = NP$还是$P \neq NP$

让我们来看一些问题

如何将问题编码

- 前述理论研究对象是语言，即字符串的集合
- cyclicity问题—图G是否有长度> 2的圈(circle)
  - 将图编码为有穷字母表上的字符串, 进而编码问题
  - $G=(V,E)$，其中V为顶点集合，E为边的集合，无向图编码为$\langle G\rangle=((1,2,3),(1,2),(2,3))$
  - $L_{cyclicity}=\{\langle G\rangle \mid G$包含长度$>2$的圈$\}$
- clique问题—图G是否有大小至少为k的团(clique)
  - $L_{clique}=\{\langle G,k\rangle \mid G$包含大小为$k$的团$\}$
- partition问题—正整数集合S能否分成相等的两部分
  - $L_{partition}=\{\langle S\rangle \mid \exists A\subset S, \sum_{x\in A}x=\sum_{x\in S\setminus A}x\}$
- path问题—图G中顶点s和t间是否有长度$\le k$的路径
  - $L_{path}=\{\langle G,s,t,k\rangle \mid s$和$t$间有一条长度不大于$k$的路径$\}$

不同形式的问题

	判定问题	优化问题	计算问题

这些问题的定义是**不同的**

这些问题的计算难度是**有关系的**

是否可以高效求解

- cyclicity问题—图G是否有长度> 2的圈(circle)

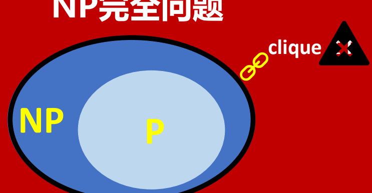
目前，仍然不知道是否$P\neq NP$！

如果假设$P\neq NP$，可以证明问题的难解性

是否有高效算法?
- 不限制时间代价的算法 ---- 是否可判定?
- 多项式时间代价的算法 ---- 是否$\in P$?
- 指数时间代价的算法 ---- 对应复杂性类?
- 上述问题均$\in NP$?

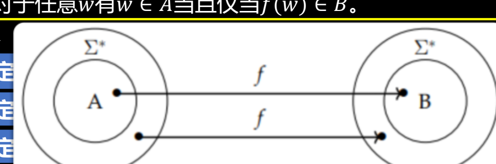
如何证明$\in NP$

NP完全问题



多项式时间归约

定义 多项式时间归约(Polynomial-Time Reducibility)
问题$A\subseteq\Sigma^*$可以多项式时间归约到$B\subseteq\Sigma^*$，记作$A\le_p B$，当且仅当存在一个多项式时间可计算函数$f:\Sigma^*\to\Sigma^*$使得对于任意w有$w\in A$当且仅当$f(w)\in B$。



定理 若$A\in P$且B非平凡$(B\neq\emptyset, B\neq\Sigma^*)$，则$A\le_p B$。

定理 若$A\le_p B$且$B\in NP$则$A\in NP$，若$A\le_p B$且$A\notin NP$则$B\notin NP$。

NP完全问题

直观想法 $\le_p$可以构建NP类内问题的联系

➢ **NP中最难问题是至少和每一个NP问题一样难的问题**

定义 NP完全问题(NP-Completeness)
问题B是NP完全问题iff.❶$B \in NP$❷$\forall A \in NP, A \le_p B$。

定义 NP难问题(NP-Hard)
问题B是NP难问题iff. $\forall A \in NP$有$A \le_p B$。

Cook定理 存在一个问题(SAT问题)是NP完全问题。

定理 如果A和B均是NP完全问题，那么$A \le_p B$。

定理 如果A是NP完全问题且$A \in P$，那么$P = NP$。

NP完全问题

定理 P vs NP与NP完全问题
下列说法是等价的：
➢ $P = NP$
➢ 每一个NP完全问题都属于P类
➢ 某个NP完全问题属于P类

➢ 绝大多数计算机科学家相信$P \ne NP$
➢ NP完全问题直观上为什么困难？
➢ 很多人尝试设计NP完全问题多项式时间算法失败
➢ 当然这并不能说明多项式时间算法不存在
➢ 目前只能证明NP完全性来说明问题的难度

Cook定理——第一个NP完全问题

库克定理

SAT = $\{\langle\varphi\rangle \mid \varphi$是可满足的布尔表达式$\}$

定义 布尔表达式(Boolean Formula)可以归纳地定义如下：
❶布尔变量(variable)及其否定是布尔表达式, 如x和$\neg x$；
❷布尔操作(boolean operators)合成表达式是布尔表达式
- 与($\wedge$)操作: 若φ_1和φ_2是布尔表达式，$\varphi_1 \wedge \varphi_2$也是；
- 或($\vee$)操作: 若φ_1和φ_2是布尔表达式，$\varphi_1 \vee \varphi_2$也是；
- 非($\neg$)操作: 若φ是布尔表达式，$\neg\varphi$也是。

• 文字(literal)—正文字x，负文字$\neg x$；

定义 赋值(assignment) 令布尔表达式φ所有变量表示为$\mathrm{var}(\varphi)$，φ的赋值τ是一个形如$\mathrm{var}(\varphi) \to \{0,1\}$的函数，$\tau(\varphi)$表示表达式$\varphi$在赋值$\tau$下的取值。

定义 可满足的布尔表达式(Satisfiable Boolean Formula)
布尔表达式φ是可满足的当且仅当存在τ使得$\tau(\varphi) = 1$。

库克定理

$$\varphi = x \vee ((y \vee z) \wedge (\neg y \vee \neg z)) \vee \neg(x \vee z)$$

➢ $\tau_1: x = 1, y = 0, z = 0 \quad \Rightarrow \quad \tau_1(\varphi) = ?$
➢ $\tau_2: x = 0, y = 0, z = 1 \quad \Rightarrow \quad \tau_2(\varphi) = ?$

$$\varphi' = (y \vee z) \wedge \neg(x \vee z)$$

➢ $\tau_3: x = 1, y = 0, z = 0 \quad \Rightarrow \quad \tau_3(\varphi') = ?$
➢ $\tau_4: x = 0, y = 0, z = 0 \quad \Rightarrow \quad \tau_4(\varphi') = ?$
➢ $\tau_5: x = 0, y = 1, z = 0 \quad \Rightarrow \quad \tau_5(\varphi') = ?$

$$\varphi'' = (x \wedge \neg x)$$

➢ 不可满足，不属于SAT

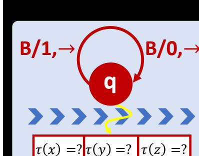
定理 库克定理, Cook Theorem
SAT问题是NP完全问题。

库克定理

Cook定理 ❶SAT∈NP ❷$\forall A$∈NP, $A \le_p$SAT

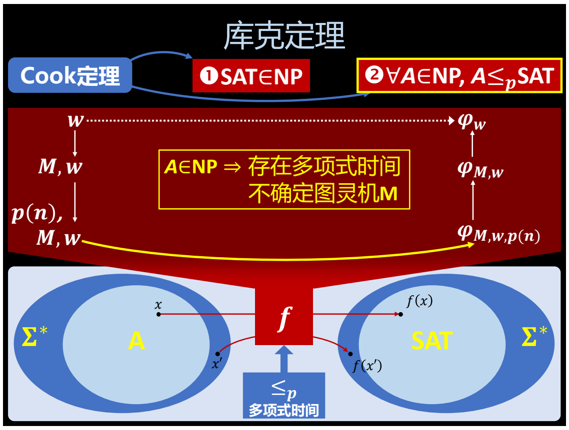
$\varphi = (y \vee z) \wedge \neg(x \vee z)$ → **SAT-Solver算法** → res

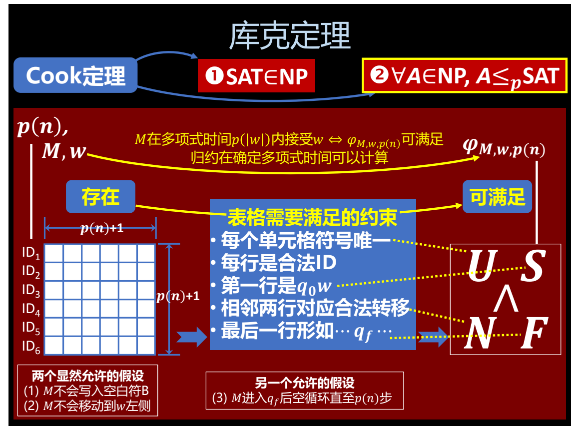
(1) 扫描φ, 收集变量列表$\mathrm{var}(\varphi)$;
(2) 为$\mathrm{var}(\varphi)$**猜测**一个赋值τ;
(3) 计算$res = \tau(\varphi)$;



$\tau: x = 0$
$y = 1$
$z = 0$

SAT-Solver
非确定算法 + 多项式时间





有代表性的NP完全问题

CNF-SAT问题

- 子句(Clause), 即文字的或

  $\neg x_1 \vee x_2 \vee \neg x_4 \qquad x_1 \qquad \neg x_3$
- 合取式(CNF, conjunctive normal form), 即子句的与

  $(\neg x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge (x_3)$
- 3CNF, φ是CNF且每个子句刚好有3个文字
- **CNF-SAT** = $\{\langle\varphi\rangle \mid \varphi$是可满足的CNF$\}$
- **3SAT** = $\{\langle\varphi\rangle \mid \varphi$是可满足的3CNF$\}$

定理 CNF-SAT是NP-完全问题。

定理 3SAT是NP-完全问题。

CNF-SAT问题

定理 CNF-SAT是NP-完全问题。

❶CNF-SAT∈NP

❷∀A∈NP, $A \le_p$CNF-SAT

显然可以接着使用**SAT-Solver算法**解决该问题

SAT-Solver算法

(1) 扫描φ, 收集变量列表var(φ);
(2) 为var(φ)**猜测**一个赋值τ;
(3) 计算$res = \tau(\varphi)$;

CNF-SAT问题

定理 CNF-SAT是NP-完全问题。

❶CNF-SAT∈NP　　**❷∀A∈NP, $A\leq_p$CNF-SAT**

SAT是NP-完全问题

∀A∈NP, $A\leq_p$SAT

SAT $\leq_p$ CNF-SAT?

CNF-SAT问题

定理 CNF-SAT是NP-完全问题。

❶CNF-SAT∈NP　　**❷∀A∈NP, $A\leq_p$CNF-SAT**

SAT　?　CNF-SAT

φ　　φ'

数

不需要φ和φ'等价

只需φ可满足 ⇔ φ'可满足

CNF-SAT问题

定理 CNF-SAT是NP-完全问题。

❶CNF-SAT∈NP　　**❷∀A∈NP, $A\leq_p$CNF-SAT**

先看两个概念

- **变量改名**

x 改为 x', x'', x_1, y

$x\wedge y$ 改为 $x\wedge y', x\wedge y_1, x\wedge z, y\wedge z$

$x\wedge\neg y$ 改为 $x\wedge\neg y_1, x_1\wedge\neg z$

$x\wedge\neg y$ 改为 $x\wedge\neg x$?

- **扩展赋值**
  - 赋值τ_2是τ_1的扩展：τ_1,τ_2分别为变量集合V_1, V_2赋值，满足$V_1\subseteq V_2$，并且对任意$x\in V_1$有$\tau_1(x)=\tau_2(x)$

CNF-SAT问题

定理 CNF-SAT是NP-完全问题。

❶CNF-SAT∈NP　　**❷∀A∈NP, $A\leq_p$CNF-SAT**

第一步 将φ变

$\neg(E_1\vee E$

❶ 如果$\tau(E)=1$，则存在τ的扩展τ'使得$\tau'(F_1\wedge\cdots\wedge F_k)=1$;

❷ 如果$\tau'(F_1\wedge\cdots\wedge F_k)=1$，则$\tau'(E)=\tau(E)=1$;

第二步 将表达式E转换为若干**子句**$F_1,\dots,F_k$，其中$k\leq|E|$，$\{F_i\}$中包含E的变量以及至多额外$|E|$个变量，使得

E可满足 ⇔ $F_1\wedge\cdots\wedge F_k$可满足，且赋值存在扩展关系

基本情况$|E|=1$

显然: 令$F_1=E$

针对表达式长度$|E|$进行归纳证明

假设$|E|<n$时可转换

归纳证明

$|E|=n$时也可以转换

CNF-SAT问题

定理 CNF-SAT是NP-完全问题。

❶CNF-SAT∈NP　　**❷∀A∈NP, $A\leq_p$CNF-SAT**

第一步 将φ变为非操作¬只与基本变量共同出现的形式E

第二步 将表达式E转换为若干**子句**$F_1,\dots,F_k$，其中$k\leq|E|$，$\{F_i\}$中包含E的变量以及至多额外$|E|$个变量，使得

E可满足 ⇔ $F_1\wedge\cdots\wedge F_k$可满足，且赋值存在扩展关系

当$E=E_1\wedge E_2$时

由归纳假设可知

E_1存在$\{G_1,\dots,G_k\}$

E_2存在$\{H_1,\dots,H_k\}$

合并两组子句并将冲突变量改名

当$E=E_1\vee E_2$时

由归纳假设可知

E_1存在$\{G_1,\dots,G_k\}$

E_2存在$\{H_1,\dots,H_k\}$

将冲突变量改名，构造子句$\{G_i\vee y, H_j\vee\neg y\}$

验证多项式时间　**验证正确性**

假设$|E|<n$时可转换

归纳证明

$|E|=n$时也可以转换

CNF-SAT问题

定理 CNF-SAT是NP-完全问题。

❶CNF-SAT∈NP　　**❷∀A∈NP, $A\leq_p$CNF-SAT**

一个例子

$$\varphi=\neg(\neg(x_1\vee x_2)\wedge(\neg x_1\vee x_2))$$

3SAT问题

定理 3SAT是NP-完全问题。

❶3SAT∈NP **显然**

❷$\forall A\in NP,\ A\le_p 3SAT$

3SAT问题

定理 3SAT是NP-完全问题。

❶3SAT∈NP

❷$\forall A\in NP,\ A\le_p 3SAT$

把CNF-SAT问题规约到3SAT问题
即证明CNF-SAT $\le_p$ 3SAT

3SAT问题

定理 3SAT是NP-完全问题。

❶3SAT∈NP

❷$\forall A\in NP,\ A\le_p 3SAT$

CNF-SAT → 3SAT

$\varphi = e_1\wedge e_2\wedge\cdots\wedge e_k$ → $\sigma = e'_1\wedge e'_2\wedge\cdots\wedge e'_l$

转换满足条件：有一个赋值τ满足φ
当且仅当存在τ的扩展赋值τ'使得σ可满足

3SAT问题

定理 3SAT是NP-完全问题。

❶3SAT∈NP

❷$\forall A\in NP,\ A\le_p 3SAT$

$e_i = x$ → $e'_i = (x\vee u\vee v)\wedge(x\vee\neg u\vee v)\wedge(x\vee u\vee\neg v)\wedge(x\vee\neg u\vee\neg v)$

$e_i = x\vee y$ → $e'_i = (x\vee y\vee u)\wedge(x\vee y\vee\neg u)$

3SAT问题

定理 3SAT是NP-完全问题。

❶3SAT∈NP

❷$\forall A\in NP,\ A\le_p 3SAT$

$e_i = x\vee y\vee z$ → $e'_i = x\vee y\vee z$

$e_i = x_1\vee x_2\vee\cdots\vee x_m\ (m>3)$

↓

$$e'_i = (x_1\vee x_2\vee y_1)\wedge(x_3\vee\neg y_1\vee y_2)\wedge(x_4\vee\neg y_2\vee y_3)\wedge\cdots\wedge(x_{m-2}\vee\neg y_{m-4}\vee y_{m-3})\wedge(x_{m-1}\vee x_m\vee\neg y_{m-3})$$

其它SAT问题

定义 NAE-SAT(Not-All-Equal SAT)问题
给定一个SAT实例F，是否有一个赋值τ满足F，并且每个子句中至少一个文字为真且至少一个文字为假。

定理 NAE-SAT和NAE-3SAT问题是NP完全问题。

定义 MAX-SAT问题
给定子句集合F和整数k，是否存在赋值τ至少满足F中的k个子句。

定理 MAX-SAT问题是NP完全问题。

定理 MAX2SAT问题是NP完全问题。

独立集问题

定义 **独立集, independent set, 简称IS**

给定图G和整数k，是否存在一个大小为k的独立集I使得I中任意一对顶点间都没有边。

定理 独立集问题是NP完全问题。

❶IS∈NP ❷∀A∈NP, $A\le_p$IS

IS-Solver算法

(1) 扫描图G的顶点集V,**不确定地**选取一个大小为k的顶点集合I;
(2) 扫描图G的边集E，判定集合I是否为一个图G的独立集;
(3) 如果是，输出yes;否则，输出no.

独立集问题

定理 独立集问题是NP-完全问题。

❶IS∈NP ❷∀A∈NP, $A\le_p$IS

往证3SAT问题可多项式时间归约为IS问题

构造归约

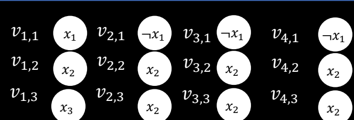
证明多项式时间以及正确性

独立集问题

假设3SAT问题的输入是$\varphi = e_1 \wedge e_2 \wedge \cdots \wedge e_m$

从φ开始构造一个有$3m$个顶点的图$G = (V, E)$:
❶$V = \{v_{i,j} | v_{i,j}$是e_i中第j个变量$, 1 \le i \le m, 1 \le j \le 3\}$

例如：$\varphi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_4) \wedge (\neg x_2 \vee x_3 \vee x_5) \wedge (\neg x_3 \vee \neg x_4 \vee \neg x_5)$

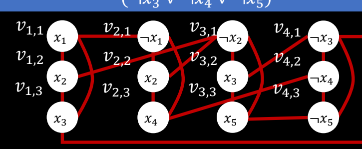


独立集问题

❷构造边集E:
任何大小为m的独立集对应满足φ的一个赋值
A.每个$\{v_{i,1}, v_{i,2}, v_{i,3}\}$中每对顶点都有一条边;
B.如果两个顶点表示x和$\neg x$，则在节点之间加一条边;

显然，这个构造过程可以在多项式时间内完成

例如：$\varphi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_4) \wedge (\neg x_2 \vee x_3 \vee x_5) \wedge (\neg x_3 \vee \neg x_4 \vee \neg x_5)$



正确性

团问题

定义 **团问题, clique**

给定图G和整数k，是否存在一个大小为k的团C使得C中任意一对顶点间都有边。

定理 团问题是NP-完全问题。

❶CLIQUE∈NP ❷∀A∈NP, $A\le_p$CLIQUE

CLIQUE-Solver算法

(1) 扫描图G的顶点集V,**不确定地**选取一个大小为k的顶点集合I;
(2) 扫描图G的边集E判定集合I是否为一个图G的团;
(3) 如果是，输出yes;否则，输出no

团问题

定理 团问题是NP-完全问题。

❶CLIQUE∈NP ❷∀A∈NP, $A\le_p$CLIQUE

往证独立集问题可多项式时间归约为团问题

图G的独立集I中顶点一定在G的补图G'中形成一个团C，反之亦然

顶点覆盖问题

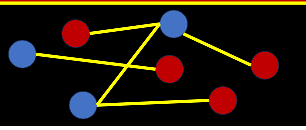
定义 **顶点覆盖, vertex cover, 简称VC**
给定图G和整数k，是否存在一个大小为k的顶点覆盖I使得G中任意一条边都至少有一个顶点在I中。

定理 顶点覆盖问题是NP-完全问题。

$\mathrm{IS} \le_p \mathrm{VC}$

❶VC∈NP ❷∀A∈NP, $A \le_p$ VC

图G中I是独立集，则$V \setminus I$是顶点覆盖，反之亦然



红色顶点为独立集
蓝色顶点为覆盖集

支配集问题

定义 **支配集, dominating set, 简称DS**
给定图G和整数k，是否存在一个大小至多为k的支配集S使得$V \setminus S$中任意一顶点都与S中某个顶点相邻。

定理 支配集问题是NP完全问题。

定义 **边支配集, edge dominating set, 简称EDS**
给定图G和整数k，是否存在一个大小至多为k的边支配集E'使得$E \setminus E'$中任意一边都与E'中某条边相邻。

定理 边支配集问题是NP完全问题。

哈密顿路相关问题

定义 **有向哈密顿路径问题, DHAMPATH**
G是一个有向图，s和t是两个不同的顶点，G中是否存在一条从s到t且经过所有G中顶点恰好一次的路径。

定理 有向哈密顿路径问题是NP-完全问题。

定理 无向哈密顿路径问题是NP-完全问题。

定理 哈密顿环路问题是NP-完全问题。

定义 **旅行商问题, traveling salesman problem, 简称TSP**
给定输入G, c和k，其中$G = (V, E)$是一个完全图，$c: E \to N$是一个边上的赋值函数，k是一个正整数，问是否存在G中的一个环恰好访问每个顶点一次，环的代价$\le k$。

定理 旅行商问题是NP-完全问题。

三维匹配问题

定义 **三维匹配, three dimension matching, 简称3DM**
给定大小均为n的集合X，Y，Z，三个集合彼此不相交，$T \subseteq X \times Y \times Z$，是否存在一个匹配$M \subseteq T$，使得$M$中任意3元组均不与其它元组相交。

定理 三维匹配问题是NP-完全问题。

- 2DM问题？
- 二分图的最大匹配问题

子集和问题

定义 **子集和问题, SUBSET-SUM**
给定S和t，其中S是一个正整数的多重集合，t是一个整数，问是否有S的一个子集合使得子集合中元素和恰好为t。

定理 SUBSET-SUM是NP-完全问题。

划分问题

定义 **划分问题, PARTITION**
给定S，S是一个正整数的多重集合，问是否可以将S划分为两部分S_1和S_2，使得S_1与S_2的元素和相同。

定理 划分问题是NP-完全问题。

处理器调度问题

定义 处理器调度,Multi-Processor Scheduling,简称MPS
给定S，m，D，其中S是一个正整数的多重集合，表示需要执行的任务的运行时间，$m \in N$ 是处理器的个数，$D \in N$ 是截止时间，问是否存在一个S的划分$S = S_1 \cup S_2 \cup \cdots \cup S_m$，对每个$i$有$\mathrm{sum}(S_i) \le D$。

定理 处理器调度问题是NP-完全问题。

NP和CoNP

NP和CoNP

定义 给定语言L，CoL表示语言L的补，即CoL= Σ^*-L。

定理 P = CoP 。

- NP=CoNP?
- 猜想：NPC∉CoNP；$\overline{NPC} \notin$ NP

例

$\overline{SAT}$ = USAT；猜测 USAT ∉ NP；
令NAUT表示永真的表达式；NAUT∈CoNP

定理 NP=CoNP当且仅当存在$\overline{NPC}$问题A使得A∈NP。

空间复杂性

空间复杂性

定义 PS类 确定图灵机在多项式空间代价判定的语言。

定义 NPS类 不确定图灵机在多项式空间代价判定的语言。

定理 空间 ⇒ 时间(算法)
如果M是多项式空间图灵机, 空间代价为$p(n)$, 存在常数c, 使得如果M接受w那么一定在$c^{1+p(n)}$时间内接受。

定理 空间 ⇒ 时间(问题)
假设$L \in PS$，存在多项式$q(n)$和常数$c > 1$，M接受L且时间代价至多为$c^{q(n)}$。

定理 萨维奇定理, Savitch Theorem
PS = NPS。

空间复杂性

定义 多项式空间完全问题, PSPACE-Complete
问题A是PSPACE-complete的，如果(1) $A \in$PSPACE，并且(2)对每一个PSPACE中的问题B都有$B \le_p A$。

定理 如果问题A是PSPACE-complete的，那么
- 若A∈P，那么P = PS;
- 若A∈NP，那么 NP = PS。

QBF问题

定义 带量词的布尔表达式, QBF

带量词的布尔表达式(quantified boolean formula, 简称QBF)可以归纳地定义如下:

- 0, 1, x都是QBF，其中x是变量，且是自由的(free)
- 如果E_1和E_2都是QBF，那么如下公式都是QBF

$$(E_1), \neg(E_1), (E_1) \wedge (E_2), (E_1) \vee (E_2)$$

其中，变量x是自由或受限的取决于E_1和E_2中x的类型

- 如果E是QBF, 那么$\forall x(E)$和$\exists x(E)$都是QBF, 且x为受限变量, 其它变量取决于E

定理 无自由变量的QBF表达式取值为1或者0。

定理 QBF问题(判定无自由变量的QBF表达式φ是否为1)是PS-完全问题。

公式博弈问题

定义 公式博弈, formula game, 简称FG

给定形如$g_F = \exists x_1 \forall x_2 \exists x_3 ... Q x_k [F]$的QBF, 其中$F$是不包含量词的布尔表达式,

- 博弈游戏有两个玩家A和E;
- 玩家A为∀限定的变量选取值, E为∃限定的变量选取值;
- 玩家E的目标是证明g_F=1, A的目标是证明g_F=0;
- 博弈顺序同g_F中量词顺序相同, F=1则E胜, 否则A胜。

定义 公式博弈问题, FG问题

给定公式g_F, 判断在对应博弈中玩家E是否有必胜策略。

定理 FG问题是PS-完全问题。

广义地理学博弈问题

定义 广义地理学博弈, generalized geography game

给定有向图G, 从start节点出发，玩家I和II分别轮流选取下一跳节点, 直到一方无路可走。

定义 广义地理学博弈问题, GG问题

给定有向图G, 判断在对应博弈中玩家I是否有必胜策略。

定理 GG问题是PS-完全问题。

P完全问题

P完全问题

定义 P完全问题, P-completeness

问题A是P-完全问题，如果(1)$A \in$P，并且(2)对任意$B \in$P有$B \le_L A$, 其中$\le_L$表示对数空间归约。

定义 L_{CFE}问题

给定上下文无关文法G, 判断是否$L(G) = \emptyset$。

定理 L_{CFE}问题是P-完全问题。

对数空间复杂性类

对数空间复杂性类

定义 **L(LOGSPACE)类** DTM在对数空间代价判定的语言。

定义 **NL(NLOGSPACE)类** NTM在对数空间代价判定的语言。

定义 **PATH问题** 给定图G及两个顶点s,t，判断是否有一条从s到t的路径。

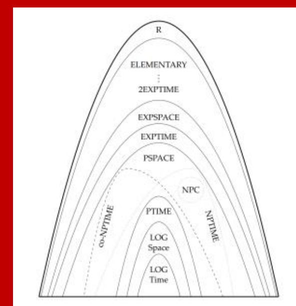
定理 PATH$\in$ NL。

定理 PATH是NL完全问题。

定理 PATH$\in$ CoNL。

定理 NL = CoNL。

复杂性类



☛ 可近似理论