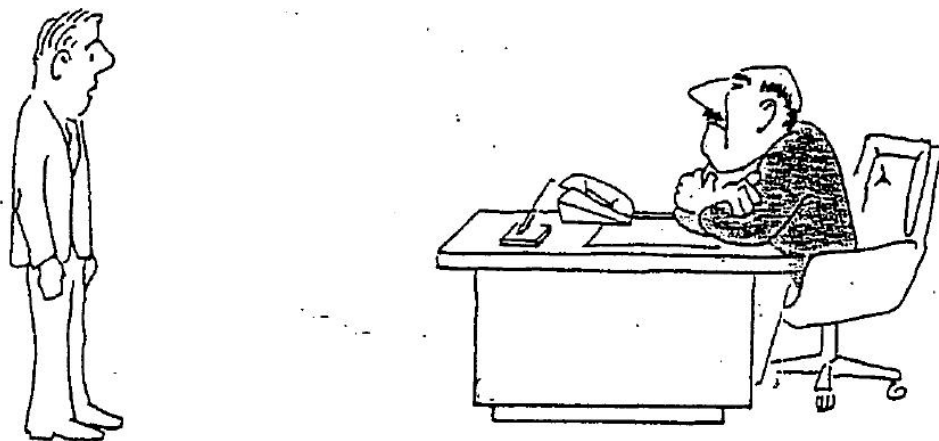


第11讲 NP完全问题 (上)

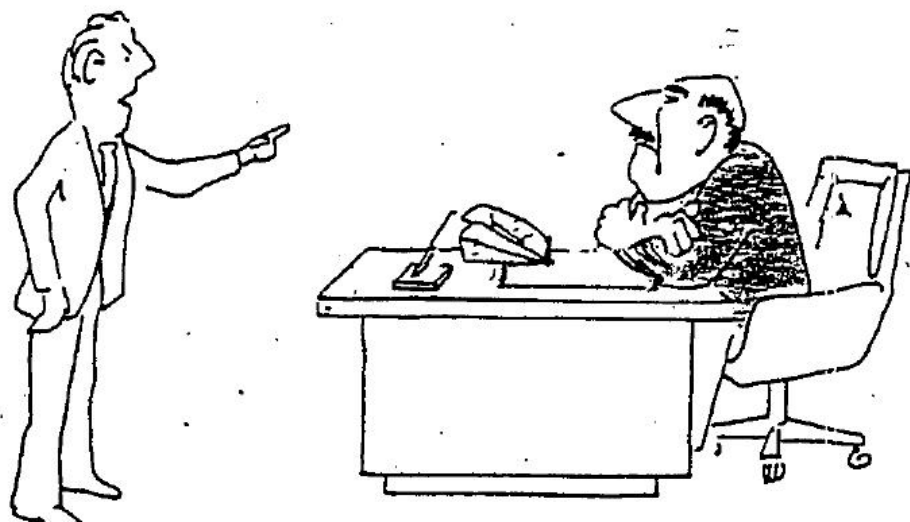
罗国杰

gluo@pku.edu.cn

2025年春季学期

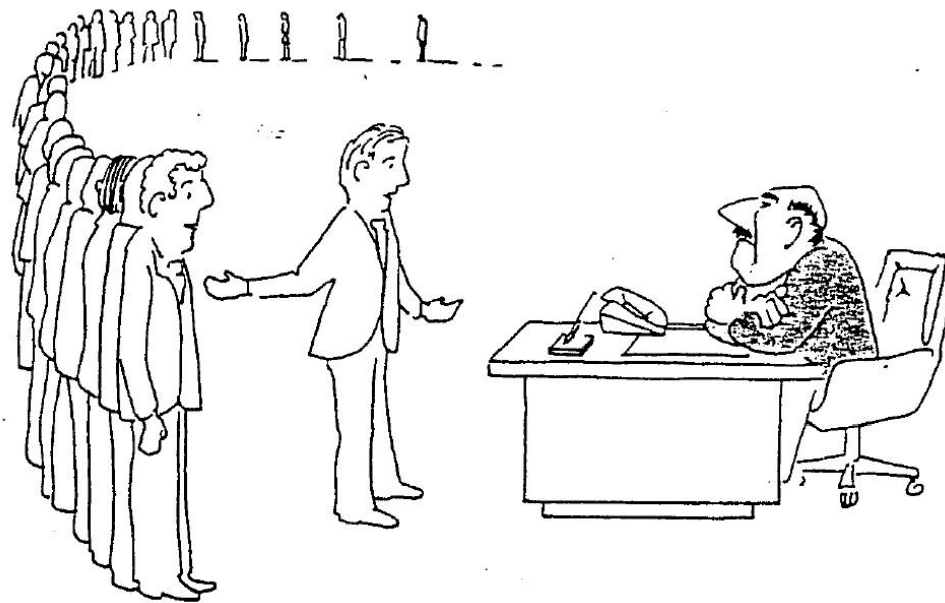


I can't find an efficient algorithm, I guess I'm just too dumb.



I can't find an efficient algorithm, because no such algorithm is possible.

NP 完全问题



I can't find an efficient algorithm, but neither can all these famous people.

主要内容

- P类与NP类
 - ▶ 易解的问题与难解的问题
 - ▶ 判定问题
 - ▶ NP类
- 多项式时间变换与NP完全性
 - ▶ 多项式时间变换
 - ▶ NP完全性
 - ▶ Cook-Levin定理——第一个NP完全问题
- NP完全问题
 - ▶ 最大可满足性与三元可满足性、等等

P类与NP类

- 易解的问题与难解的问题
- 评价算法好坏的重要标准——运行时间
 - ▶ 快速排序算法 $O(n\log n)$
 - ▶ Dijkstra算法 $O(n^2)$
 - ▶ 最大团问题的回溯法 $O(2^n)$
- 用一台每秒10亿次 (1 GOPS) 的计算机，运行快速排序算法，给10万个数据排序，运算量约为 $10^5 \times \log_2 10^5 \approx 1.7 \times 10^6$ ，仅需 $1.7 \times 10^6 / 10^9 = 1.7 \times 10^{-3}$ 秒。

什么是好算法?

- Dijkstra算法求解1万个顶点的图的单源最短路径问题
 - ▶ 运算量约为 $(10^4)^2 = 10^8$, 约需 $10^8/10^9 = 0.1$ 秒.
- 回溯法解100个顶点的图的最大团问题
 - ▶ 运算量为 $100 \times 2^{100} \approx 1.8 \times 10^{32}$, 需要 $1.8 \times 10^{32}/10^9 = 1.8 \times 10^{21}$ 秒
 - ▶ $= 5.7 \times 10^{15}$ 年, 即5千7百万亿年!
- 再从另外一个角度来看——1分钟能解多大的问题。1分钟60秒, 这台计算机可做 6×10^{10} 次运算: 用快速排序算法可给 2×10^9 (20亿) 个数据排序, 用Dijkstra算法可解 2.4×10^5 (24万) 个顶点的图的单源最短路径问题; 而用回溯法一天只能解41个顶点的图的最大团问题

算法的时间复杂度

- 函数 f 和 g 是**多项式相关的**：如果存在多项式 p 和 q 使得，对任意的 $n \in \mathbb{N}$ ， $f(n) \leq p(g(n))$ 和 $g(n) \leq q(f(n))$ 。
例如 $n \log n$ 与 n^2 ， $n^2 + 2n + 5$ 与 n^{10} 都是多项式相关的，
 $\log n$ 与 n ， n^5 与 2^n 不是多项式相关的。
- 问题 Π 的实例 I 的**规模**： I 的二进制编码的长度，记作 $|I|$ 。
- **定义** 如果存在函数 $f: \mathbb{N} \rightarrow \mathbb{N}$ 使得，对任意的规模为 $|I|=n$ 的实例 I ，算法 A 对 I 的运算在 $f(n)$ 步内停止，则称算法 A 的**时间复杂度**为 $f(n)$ 。
- **多项式时间算法**：以多项式为时间复杂度。
- **易解的问题**：有多项式时间算法的问题。
- **难解的问题**：不存在多项式时间算法的问题。

什么是问题？

抽象判定问题

- 判定问题：答案只有两个——是、否。
- 判定问题 $\Pi = \langle D_\Pi, Y_\Pi \rangle$
 - 其中 D_Π 是实例集合， $Y_\Pi \subseteq D_\Pi$ 是所有答案为 “Yes” 的实例。
- 哈密顿回路 (HC)：任给无向图 G ，问 G 有哈密顿回路吗？
- 货郎问题 (TSP)：任给 n 个城市，城市 i 与城市 j 之间的正整数距离 $d(i,j)$, $i \neq j$, $1 \leq i, j \leq n$ ，以及正整数 D ，问有一条每一个城市恰好经过一次最后回到出发点且长度不超过 D 的巡回路线吗？即，是否存在 $1, 2, \dots, n$ 的排列 σ 使得

$$\sum_{i=1}^{n-1} d(\sigma(i), \sigma(i+1)) + d(\sigma(n), \sigma(1)) \leq D?$$

组合优化问题与判定问题

- 0-1背包：任给 n 件物品和一个背包，物品 i 的重量为 w_i ，价值为 v_i ， $1 \leq i \leq n$ ，以及背包的重量限制 B 和价值目标 K ，其中 w_i, v_i, B, K 均为正整数，问能在背包中装入总价值不少于 K 且总重量不超过 B 的物品？即，是否存在子集 $T \subseteq \{1, 2, \dots, n\}$ 使得

$$\sum_{i \in T} w_i \leq B \quad \text{且} \quad \sum_{i \in T} v_i \geq K?$$

- 组合优化问题与判定问题的对应
 - 如果组合优化问题有多项式时间算法，则对应的判定问题也有多项式时间算法
 - 如果判定问题是难解的，则对应的组合优化问题也是难解的

组合优化问题与判定问题

► 组合优化问题 Π^* 由3部分组成:

- (1) 实例集 D_{Π^*}
- (2) $\forall I \in D_{\Pi^*}$, 有一个有穷非空集 $S(I)$, 其元素称作 I 的可行解.
- (3) $\forall s \in S(I)$, 有一个正整数 $c(s)$, 称作 s 的值.

► 如果 $s^* \in S(I)$, 对所有的 $s \in S(I)$, 当 Π^* 是最小 (大) 化问题, 且

$$c(s^*) \leq c(s) \quad (c(s^*) \geq c(s))$$

则称 s^* 是 I 的最优解, $c(s^*)$ 是 I 的最优值, 记作 $\text{OPT}(I)$ 。

► Π^* 对应的判定问题 $\Pi = \langle D_{\Pi}, Y_{\Pi} \rangle$ 定义如下:

- $D_{\Pi} = \{(I, K) \mid I \in D_{\Pi^*}, K \in \mathbb{Z}^*\}$, 其中 \mathbb{Z}^* 是非负整数集合。
- 当 Π^* 是最小化问题时, $Y_{\Pi} = \{(I, K) \mid \text{OPT}(I) \leq K\}$;
- 当 Π^* 是最大化问题时, $Y_{\Pi} = \{(I, K) \mid \text{OPT}(I) \geq K\}$ 。

具体判定问题


- 具体判定问题 $\Pi = \langle D_\Pi, Y_\Pi \rangle$ 的实例 $I \in D_\Pi$ 拥有编码 $e(I)$ 和编码长度 $|e(I)|$
- **编码** $e: D_\Pi \rightarrow \Sigma^*$, 将实例 I 映射为字母表 Σ 上字符串的“合理”编码
 - ▶ “合理的”是指在编码中不故意使用许多冗余的字符
 - ▶ 例如, 一进制编码长度 $|e(k)| \in O(k)$ 不合理; 二进制或以上编码长度 $|e(k)| \in O(\log k)$ 合理
- 时间复杂度是关于 $|e(I)|$ 的函数, 通常表成计算对象的某些自然参数的函数
 - ▶ 如图的顶点数或顶点数与边数的函数
 - ▶ **当采用合理的编码时, 输入的规模通常是多项式相关的**
 - 例如, 设实例 I 是一个无向简单图 $G = \langle V, E \rangle$, 包含4个顶点 $V = \{a, b, c, d\}$ 和5条边 $E = \{(a, b), (a, d), (b, c), (b, d), (c, d)\}$
 - 若用邻接矩阵表示, 则编码 $e_1 = 0101/1011/0101/1110/$, 长度为 $4 \times (4+1) = 20$
 - 若用关联矩阵表示, 则编码 $e_2 = 11000/10110/00101/01011/$, 长度为 $4 \times (5+1) = 24$
 - 设 G 有 n 个顶点 m 条边, 则用邻接矩阵时 $|I| = n \cdot (n+1)$, 用关联矩阵时 $|I| = n \cdot (m+1)$, 两者是多项式相关的。

什么是算法？

希尔伯特第十问题

- 1900年希尔伯特第十问题：是否存在 “a process according to which it [the solution to the problem] can be determined by a finite number of operations” 解丢番图方程
 - ▶ ~ 整系数多项式方程 $f(x_1, x_2, \dots, x_n) = 0$ 是否存在整数解
- 1930年代丘奇（Alonzo Church）研究 λ -calculus，1941年用 λ -calculus 定义算法
- 1936年图灵（Alan Turing）用图灵机（Turing Machine, TM）定义算法
- 丘奇-图灵论题：任何可计算问题都是 λ /TM 可计算的
- 图灵证明了希尔伯特第十问题不可判定；停机问题也是不可判定

Alonzo Church & Alan Turing




Mathematics Genealogy Project

- Home
- Search
- Extrema
- About MGP
- Links
- FAQs
- Posters
- Submit Data
- Contact
- Donate

A service of the [NDSU Department of Mathematics](#), in association with the [American Mathematical Society](#).

Alan Mathison Turing

[Biography MathSciNet](#)

Ph.D. [Princeton University](#) 1938 

Dissertation: *Systems of Logic Based on Ordinals*

Advisor: [Alonzo Church](#)

Students:
Click [here](#) to see the students listed in chronological order.

Name	School	Year	Descendants
Gandy, Robin	University of Cambridge	1953	204
Worsley, Beatrice	University of Cambridge	1954	

According to our current on-line database, Alan Turing has 2 [students](#) and 206 [descendants](#).

We welcome any additional information.

If you have additional information or corrections regarding this mathematician, please use the [update form](#). To submit students of this mathematician, please use the [new data form](#), noting this mathematician's MGP ID of 8014 for the advisor ID.

Search About MGP ▼ Links FAQs Posters Submit Data Contact

The Mathematics Genealogy Project is in need of funds to help pay for student help and other associated costs. If you would like to contribute, please [donate online](#) using credit card or bank transfer or mail your tax-deductible contribution to:

Mathematics Genealogy Project
Department of Mathematics
North Dakota State University
P. O. Box 6050
Fargo, North Dakota 58108-6050

算法即图灵机

- 判定问题 $\Pi = \langle D_\Pi, Y_\Pi \rangle$, 其中 D_Π 和 Y_Π 均为编码后的实例集合
- 字母表 Σ 上的语言是 Σ^* 的子集, $Y_\Pi \subseteq \Sigma^*$ 是 Σ 上的 (形式化) 语言
 - ▶ 其中 $\Sigma^* = \{\epsilon\} \cup \Sigma \cup \Sigma^2 \cup \Sigma^3 \cup \dots$
- 图灵机 TM 是识别形式化语言的计算模型
 - ▶ (略) 三部分: 无穷长的存储带、能移动的读写头、以及状态控制单元
 - ▶ 给定输入字符串, TM 或接受、或拒绝、或死循环
 - ▶ 如果 TM 判定的语言 $L(TM) = Y_\Pi$, 称 TM 为求解 Π 的 “算法”

大部分判定问题不可计算

- ▶ 停机问题：给定计算机算法，判断其是否对任意输入都能停机
 - ▶ 答案是“YES”或“NO”的判定问题
 - ▶ 不存在求解停机问题的算法（对任意输入，算法在有限时间返回正确结果）

▶ 大部分判定问题不可计算

算法 \approx 二进制串 \approx 自然数 $\in \mathbb{N}$

判定问题 = 输入二进制串、输出 $\{1, 0\} = \{\text{YES}, \text{NO}\}$ 的函数

\approx 二进制串的无穷序列 \approx 实数 $\in \mathbb{R}$

$|\mathbb{N}| \ll |\mathbb{R}|$ ：不存在自然数至实数的一一映射（ \mathbb{R} 不可数）

\Rightarrow 问题太多、算法不够用

每个算法只解决一个问题

\Rightarrow 几乎所有问题都是不可计算的

几个等价的计算模型

- 计算模型 (model of computation)
 - ▶ 图灵机、多带图灵机、随机存储机 (RAM)、非确定性图灵机等
- 各种计算模型在“可计算性理论”意义下等价
- 前三种模型在“计算复杂度理论”意义下等价
 - ▶ 最坏运行时间、平均运行时间
 - ▶ 渐近上界、渐近下界、渐近确界
 - ▶ 复杂度分类

随机存储器 (Random Access Machine, RAM) 模型

- RAM 是拥有无穷大寄存器阵列、每个寄存器可存储任意大整数的计算模型
- RAM 能“计算”函数 $\eta: D_{\mathbb{N}} \rightarrow \{0, 1\}$
- 输入 $I = (i_1, i_2, \dots, i_N)$, 长度 $|I| = \sum_k |b(i_k)|$
- 程序 $P = (s_1, s_2, \dots, s_m)$; 程序计数器 κ , 每个指令 s_i 后自动加 1
- 状态 $C = (\kappa, R)$, 其中 $R = \{(j_1, r_{j_1}), (j_2, r_{j_2}), \dots, (j_m, r_{j_m})\}$ 为非零寄存器序号与值的有穷集合
- 初始状态 $\{1, \emptyset\}$, 终止状态 $\{0, R\}$, 其中 $(0, \eta(I)) \in R$
- 如果步数 $k \leq T(|I|)$, 称运行时间为 $T(n)$

Instruction	Operand	Semantics
READ	j	$r_0 := i_j$
READ	$\uparrow j$	$r_0 := i_{r_j}$
STORE	j	$r_j := r_0$
STORE	$\uparrow j$	$r_{r_j} := r_0$
LOAD	x	$r_0 := x$
ADD	x	$r_0 := r_0 + x$
SUB	x	$r_0 := r_0 - x$
HALF		$r_0 := \lfloor \frac{r_0}{2} \rfloor$
JUMP	j	$\kappa := j$
JPOS	j	if $r_0 > 0$ then $\kappa := j$
JZERO	j	if $r_0 = 0$ then $\kappa := j$
JNEG	j	if $r_0 < 0$ then $\kappa := j$
HALT		$\kappa := 0$

RAM 机 vs 图灵机

- 定理：如果语言 L 用图灵机在 $O(f(n))$ 时间内判定，则存在 RAM 程序在 $O(f(n))$ 时间内计算函数 $\eta_L: \Sigma^* \rightarrow \{0,1\}$ ，其中 $\eta_L(s) = 1 \Leftrightarrow s \in L$
 - ▶ 证明思路：用 RAM 程序模拟图灵机
- 定理：设 P 是在 $f(n) \geq n$ 时间内计算 η 函数的 RAM 程序，则存在 7 带图灵机在 $O(f^3(n))$ 时间内判定语言 L_η
 - ▶ 证明思路：用 7 带图灵机模拟 RAM 程序
 - ▶ 7 条带：1) 输入、2) 寄存器阵列、3) 程序计数器、4) 当前寄存器地址、5) 算术运算第一个操作数、6) 第二个操作数、7) 算术运算结果
- 定理：设 $f(n) \geq n$ ，则每个 $f(n)$ 时间的多带图灵机都有等价的 $O(f^2(n))$ 时间图灵机
 - ▶ 证明思路：在单带上用分隔符模拟多带，添加特殊字符标记每条虚拟带的读写头，某条虚拟带的空间不够时转移其他虚拟带的位置来申请空间

P 类

➤ 几种模型的多项式等价性

模拟 →	单带图灵机	多带图灵机	RAM 程序
单带图灵机	-	$O(f^2(n))$	$O(f^6(n))$
多带图灵机	$O(f(n))$	-	$O(f^3(n))$
RAM 程序	$O(f(n))$	$O(f(n))$	-

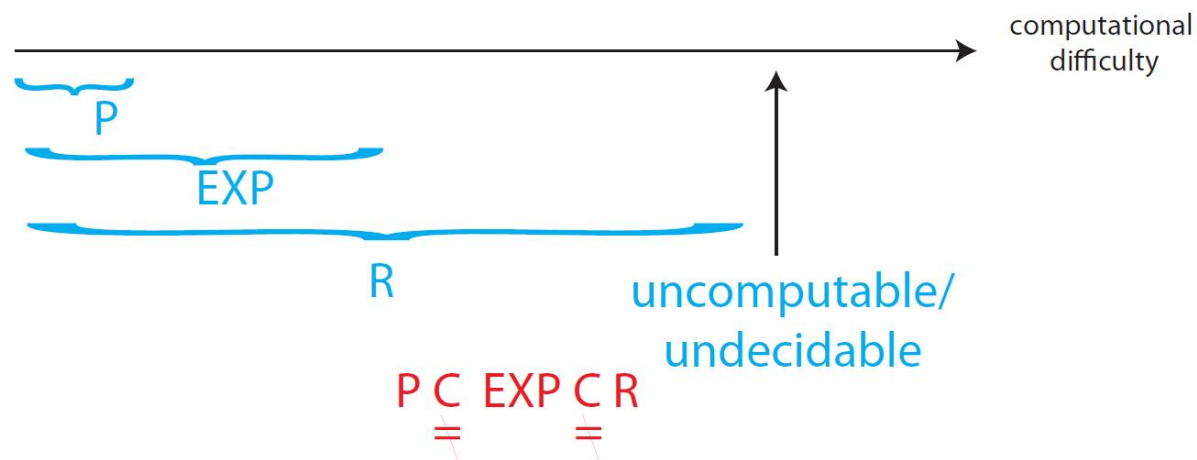
➤ 图灵机模型 $P_{TM} = \cup_{k=0}^{\infty} \text{TIME}(n^k)$

➤ RAM 模型 $P_{RAM} = \{ L : L \text{ 能被多项式时间的 RAM 算法判定} \}$

➤ $P_{TM} = P_{RAM}$

计算复杂度分类

- $P = \{ \text{能在多项式时间 } (n^c) \text{ 内求解的问题} \}$
- $EXP = \{ \text{能在指数时间 } (2^{n^c}) \text{ 内求解的问题} \}$
- $R = \{ \text{能在有限时间内求解的问题} \}$ “recursive” [Turing 1936; Church 1941]



图片来源: Demaine, Devadas, “MIT 6.006: Introduction to Algorithms,” Fall 2011

► Complexity Zoo

► https://complexityzoo.uwaterloo.ca/Complexity_Zoo

► 例子

- 图的负回路检测 $\in P$
- $n \times n$ 象棋 $\in EXP$ 但 $\notin P$
 - 当前局面谁能胜出
- 俄罗斯方块 $\in EXP$ 但 $\in? P$
 - 给定开局和方块次序, 能否存活

易解的问题与难解的问题

- **易解的问题**：如排序、最小生成树、单源最短路径等
- **已证明的难解问题**：一类是不可计算的，即根本不存在求解的算法，如希尔伯特第十问题——丢番图方程是否有整数解。另一类是有算法，但至少需要指数时间，或指数空间，甚至更多的时间或更大的空间。如带幂运算的正则表达式的全体性，即任给字母表 A 上的带幂运算的正则表达式 R ，问： $\langle R \rangle = A^*$ ？这个问题至少需要指数空间。
- **既没有找到多项式时间算法、又没能证明是难解的问题**：如哈密顿回路问题、货郎问题、背包问题等

NP 类

- **定义** 所有多项式时间可解的判定问题组成的问题类称作 **P 类**。
- **定义** 多项式时间可验证
 - ▶ 设判定问题 $\Pi = \langle D, Y \rangle$, 存在**多项式时间的验证算法** V , 使得
 - ▶ 如果 $I \in Y$, 则存在**证据** W 在多项式时间内返回 $V(I, W) = \text{YES}$
 - ▶ 如果 $I \notin Y$, 则对任意证据 W 在多项式时间内返回 $V(I, W) = \text{NO}$
- 由所有多项式时间可验证的判定问题组成的问题类称作 **NP 类**
 - ▶ 注: 将上述定义的 $I \in Y$ 和 $I \notin Y$ 交换、YES 和 NO 交换, 则得到 co-NP 类的定义
- **例如**, HC、TSP、0-1 背包 $\in \text{NP}$
- **定理**: $P \subseteq \text{NP}$ 。
- **问题**: $P = \text{NP}$?

多项式时间变换与NP完全性

- 如何比较两个问题的难度?
- **定义** 设判定问题 $\Pi_1 = \langle D_1, Y_1 \rangle$, $\Pi_2 = \langle D_2, Y_2 \rangle$ 。如果函数 $f: D_1 \rightarrow D_2$ 满足条件:
 - ▶ (1) f 是多项式时间可计算的,
 - ▶ (2) 对所有的 $I \in D_1$, $I \in Y_1 \Leftrightarrow f(I) \in Y_2$ 。
- 则称 f 是 Π_1 到 Π_2 的**多项式时间变换**。
- 如果存在 Π_1 到 Π_2 的多项式时间变换, 则称 Π_1 **可多项式时间变换到** Π_2 , 记作 $\Pi_1 \leq_p \Pi_2$ 。

多项式时间变换的例子

► 例 $HC \leq_p TSP$

► 证 对 HC 的每一个实例 I : 无向图 $G = \langle V, E \rangle$, TSP 对应的实例 $f(I)$ 为: 城市集 V , 任意两个不同的城市 u 和 v 之间的距离

$$d(u, v) = \begin{cases} 1, & \text{若 } (u, v) \in E, \\ 2, & \text{否则,} \end{cases}$$

► 以及界限 $D = |V|$

多项式时间变换的例子

- **最小生成树**: 任给连通的无向赋权图 $G=\langle V,E,W\rangle$ 以及正整数 B , 其中权 $W: E\rightarrow\mathbb{Z}^+$, 问有权不超过 B 的生成树吗?
- **最大生成树**: 任给连通的无向赋权图 $G=\langle V,E,W\rangle$ 以及正整数 D , 其中权 $W: E\rightarrow\mathbb{Z}^+$, 问 G 有权不小于 D 的生成树吗?
- **例** 最大生成树 \leq_p 最小生成树
- 证 任给最大生成树的实例 I : 连通的无向赋权图 $G=\langle V,E,W\rangle$ 和正整数 D , 最小生成树的对应实例 $f(I)$: 图 $G'=\langle V,E,W'\rangle$ 和正整数 $B=(n-1)M-D$, 其中 $n=|V|$, $M=\max\{W(e)|e\in E\}+1$, $W'(e)=M-W(e)$ 。如果存在 G 的生成树 T , 使得 $\sum_{e\in T} W(e) \geq D$, 则

$$\sum_{e\in T} W'(e) = (n-1)M - \sum_{e\in T} W(e) \leq (n-1)M - D = B.$$

- 反之亦然。

\leq_p 的性质

- **定理** \leq_p 具有传递性。即，设 $\Pi_1 \leq_p \Pi_2$, $\Pi_2 \leq_p \Pi_3$, 则 $\Pi_1 \leq_p \Pi_3$ 。
- 证 设 $\Pi_i = \langle D_i, Y_i \rangle$, $i=1,2,3$, f 和 g 是 Π_1 到 Π_2 和 Π_2 到 Π_3 的多项式时间变换。对每一个 $I \in D_1$, 令 $h(I) = g(f(I))$ 。
- 计算 f 和 g 的时间上界分别为多项式 p 和 q , 不妨设 p 和 q 是单调递增的。计算 h 的步数不超过 $p(|I|) + q(|f(I)|)$ 。输出作为合理的指令, 一步只能输出长度不超过固定值 k 的字符串, 因而 $|f(I)| \leq kp(|I|)$ 。于是, $p(|I|) + q(|f(I)|) \leq p(|I|) + q(kp(|I|))$, 得证 h 是多项式时间可计算的。
- 又, 对每一个 $I \in D_1$, $I \in Y_1 \Leftrightarrow f(I) \in Y_2 \Leftrightarrow h(I) = g(f(I)) \in Y_3$,
- 得证 h 是 Π_1 到 Π_3 的多项式时间变换。

\leq_p 的性质

- **定理** 设 $\Pi_1 \leq_p \Pi_2$, 则 $\Pi_2 \in P$ 蕴涵 $\Pi_1 \in P$ 。
- **证** 设 $\Pi_1 = \langle D_1, Y_1 \rangle$, $\Pi_2 = \langle D_2, Y_2 \rangle$, f 是 Π_1 到 Π_2 的多项式时间变换, A 是计算 f 的多项式时间算法。又设 B 是 Π_2 的多项式时间算法。如下构造 Π_1 的算法 C : 对每一个 $I \in D_1$, 首先应用 A 得到 $f(I)$, 然后对 $f(I)$ 应用 B , C 输出 “Yes” 当且仅当 B 输出 “Yes”。
- **推论** 设 $\Pi_1 \leq_p \Pi_2$, 则 Π_1 是难解的蕴涵 Π_2 是难解的。
- **例**: 由最小生成树 $\in P$, 得知最大生成树 $\in P$ 。
- **例**: 如果 TSP $\in P$, 则 HC $\in P$ 。反过来, 如果 HC 是难解的, 则 TSP 也是难解的。

NP 完全性

- **定义** 如果对所有的 $\Pi' \in \text{NP}$, $\Pi' \leq_p \Pi$, 则称 Π 是 NP-hard 的。如果 Π 是 NP-hard 的且 $\Pi \in \text{NP}$, 则称 Π 是 NP-complete (NPC) 的。
- 笼统地说, NPC 问题是 NP 中最难的问题。
- **定理** 如果存在 NP-hard 的问题 $\Pi \in \text{P}$, 则 $\text{P} = \text{NP}$ 。
- **推论** 假设 $\text{P} \neq \text{NP}$, 那么, 如果 Π 是 NP-hard 的, 则 $\Pi \notin \text{P}$ 。
- **定理** 如果存在 NP-hard 的问题 Π' 使得 $\Pi' \leq_p \Pi$, 则 Π 是 NP-hard 的。
- **推论** 如果 $\Pi \in \text{NP}$ 并且存在 NPC 问题 Π' 使得 $\Pi' \leq_p \Pi$, 则 Π 是 NPC 的。

证明 NP 完全性的 “捷径”

- 为了证明 Π 是 NP 完全的，只需要做两件事：
- (1) 证明 $\Pi \in \text{NP}$;
- (2) 找到一个已知的 NP 完全问题 Π' ，并证明 $\Pi' \leq_p \Pi$ 。

Cook-Levin 定理

► **合式公式**是由变元、逻辑运算符以及圆括号按照一定的规则组成的表达式。

- ▶ 变元和它的否定称作**文字**。
- ▶ 有限个文字的析取称作**简单析取式**。
- ▶ 有限个简单析取式的合取称作**合取范式**。
- ▶ 给定每一个变元的真假值称作一个赋值。
- ▶ 如果赋值 t 使得合式公式 F 为真，则称 t 是 F 的**成真赋值**。
- ▶ 如果 F 存在成真赋值，则称 F 是**可满足的**。

► 例如

- ▶ $F_1 = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge \neg x_2$ 是一个合取范式。
- ▶ 令 $t(x_1)=1, t(x_2)=0, t(x_3)=1$ 是 F_1 的成真赋值， F_1 是可满足的。
- ▶ $F_2 = (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge x_2 \wedge \neg x_3$ 不是可满足的。

Cook-Levin 定理

- **可满足性** (SAT): 任给一个合取范式 F , 问 F 是可满足的吗?
- **定理** (Cook-Levin 定理) SAT 是 NP 完全的。
- **定理** $P=NP$ 的充分必要条件是存在 NP 完全问题 $\Pi \in P$ 。

Cook-Levin 定理

► 定理 SAT 是 NP 完全的。

► 证明思路

► “给定 NP 问题 $\Pi = \langle D, Y \rangle$, $I \in Y$? ” \Leftrightarrow “是否存在证据 W ” \Leftrightarrow “circuit SAT”

► 第一个 \Leftrightarrow : 是否存在证据 W , 使多项式时间的验证算法 $V(I, W) = \text{YES}$

► 第二个 \Leftrightarrow : 是否存在 W 满足布尔电路 $C(I, W) = 1$

- 构造计算 V 的多项式大小的布尔电路 $C(I, W)$

- 虽然 $P \subseteq P/\text{poly}$, 但构造不是显然的 (略)

► circuit SAT (布尔电路可满足性) \leq_p SAT (合取范式可满足性)

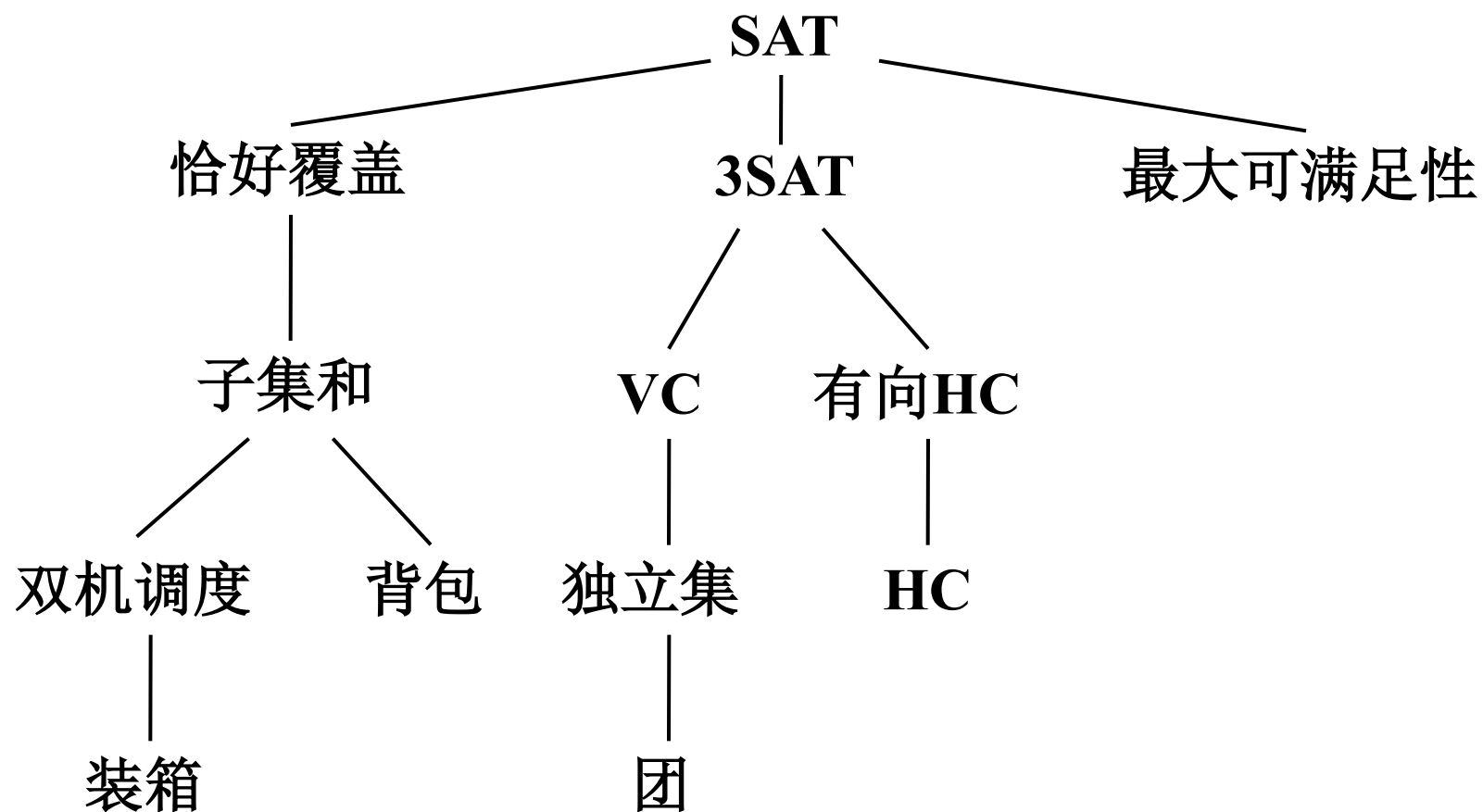
- OR 门: $a \vee b \Rightarrow (a \vee b \vee \neg z_i) \wedge (z_i \vee \neg a) \wedge (z_i \vee \neg b)$

- NOT 门: $\neg a \Rightarrow (a \vee z_i) \wedge (\neg a \vee \neg z_i)$

最大可满足性与三元可满足性

- **最大可满足性** (MAX-SAT): 任给关于变元 x_1, x_2, \dots, x_n 的简单析取式 C_1, C_2, \dots, C_m 及正整数 K , 问存在关于变元 x_1, x_2, \dots, x_n 的赋值使得 C_1, C_2, \dots, C_m 中至少有 K 个为真吗?
- 设判定问题 $\Pi = \langle D, Y \rangle$, $\Pi' = \langle D', Y' \rangle$, 如果 $D' \subseteq D$, $Y' = D' \cap Y$, 则 Π' 是 Π 的特殊情况, 称作 Π 的**子问题**。
- **例如**
 - ▶ “给定一个平面图 G , 问 G 是哈密顿图吗?” 是 HC 的子问题。
 - ▶ SAT 是 MAX-SAT 的子问题: 取 $K=m$ 。

几个NP完全问题



MAX-SAT

- **限制法**：如果已知 Π 的某个子问题 Π' 是 NP 难的，则 Π 也是 NP 难的——一般情况不会比特殊情况容易。容易把 Π' 多项式时间变换到 Π ：只需把 Π' 的实例 I 看作 Π 特殊情况的实例，即可得到 Π 对应的实例。
- **定理** MAX-SAT 是 NP 完全的。
- 证 MAX-SAT 的非确定性多项式时间算法：猜想一个赋值，检查是否有 K 个简单析取式满足。
- 要证 $\text{SAT} \leq_p \text{MAX-SAT}$ 。任给 SAT 的实例 I ：关于变元 x_1, x_2, \dots, x_n 的合取范式 $F = C_1 \wedge C_2 \wedge \dots \wedge C_m$ ，其中 C_1, C_2, \dots, C_m 是简单析取式，对应的 MAX-SAT 的实例 $f(I)$ ：简单析取式 C_1, C_2, \dots, C_m 和正整数 $K=m$ 。

3SAT

- **3元合取范式**: 每一个简单析取式恰好有3个文字的合取范式。
- **三元可满足性 (3SAT)**: 任给一个3元合取范式 F , 问 F 是可满足的吗?
- **定理** 3SAT是NP完全的.
- **证** 显然 $3SAT \in NP$ 。
- 要证 $SAT \leq_p 3SAT$ 。任给一个合取范式 F , 要构造对应的 3 元合取范式 $F' = f(F)$, 使得 F 是可满足的当且仅当 F' 是可满足的。
- 设 $F = C_1 \wedge C_2 \wedge \dots \wedge C_m$, 对应的 $F' = F'_1 \wedge F'_2 \wedge \dots \wedge F'_m$, F'_j 是对应 C_j 的合取范式, 并且 C_j 是可满足的当且仅当 F'_j 是可满足的。

证明

(1) (有1个文字的子句) $C_j = z_1$ 。引入两个新变元 y_{j1} 和 y_{j2} , 令

$$F_j' = (z_1 \vee y_{j1} \vee y_{j2}) \wedge (z_1 \vee \neg y_{j1} \vee y_{j2}) \wedge (z_1 \vee y_{j1} \vee \neg y_{j2}) \wedge (z_1 \vee \neg y_{j1} \vee \neg y_{j2})$$

(2) (有2个文字的子句) $C_j = z_1 \vee z_2$ 。引入一个新变元 y_j , 令 $F_j' = (z_1 \vee z_2 \vee y_j) \wedge (z_1 \vee z_2 \vee \neg y_j)$

(3) (有3个文字的子句) $C_j = z_1 \vee z_2 \vee z_3$ 。令 $F_j' = C_j$ 。

(4) (有4+个文字的子句) $C_j = z_1 \vee z_2 \vee \dots \vee z_k$, $k \geq 4$ 。引入 $k-3$ 个新变元 $y_{j1}, y_{j2}, \dots, y_{j(k-3)}$, 令

$$F_j' = (z_1 \vee z_2 \vee y_{j1}) \wedge (\neg y_{j1} \vee z_3 \vee y_{j2}) \wedge (\neg y_{j2} \vee z_4 \vee y_{j3}) \wedge \dots \wedge (\neg y_{j(k-4)} \vee z_{k-2} \vee y_{j(k-3)}) \wedge (\neg y_{j(k-3)} \vee z_{k-1} \vee z_k)$$

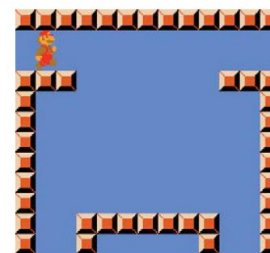
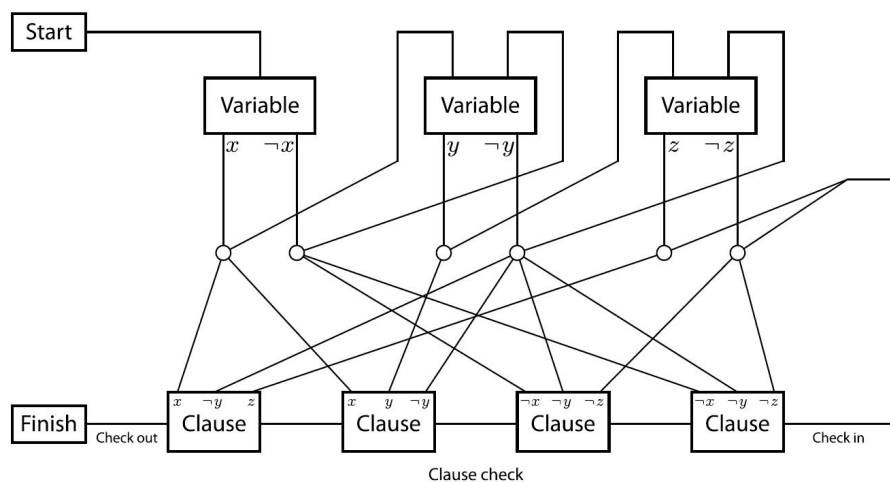
- ▶ 设赋值 t 满足 $C_j = 1$, 则存在 i 使得 $t(z_i) = 1$ 。当 $i=1$ 或 2 时, 令 $t(y_{js}) = 0$ ($1 \leq s \leq k-3$); 当 $i=k-1$ 或 k 时, 令 $t(y_{js}) = 1$ ($1 \leq s \leq k-3$); 当 $3 \leq i \leq k-2$ 时, 令 $t(y_{js}) = 1$ ($1 \leq s \leq i-2$), $t(y_{js}) = 0$ ($i-1 \leq s \leq k-3$)。则有, $t(F_j') = 1$ 。
- ▶ 反之, 设 $t(F_j') = 1$ 。若 $t(y_{j1}) = 0$, 则 $t(z_1 \vee z_2) = 1$; 若 $t(y_{j(k-3)}) = 1$, 则 $t(z_{k-1} \vee z_k) = 1$; 否则必有 s ($1 \leq s \leq k-4$) 使得 $t(y_{js}) = 1$ 且 $t(y_{j(s+1)}) = 0$, 从而 $t(z_{s+2}) = 1$ 。总之, 都有 $t(C_j) = 1$ 。

证明

- F'_j 中简单析取式的个数不超过 C_j 中文字个数的 4 倍，每个简单析取式有 3 个文字，因此可以在 $|F|$ 的多项式时间内构造出 F' 。
- **局部替换法** 要证 $\Pi_1 \leq_p \Pi_2$ 。当 Π_2 是 Π_1 的子问题或两者的结构相似时，往往可以把 Π_1 的实例的每一个子结构替换成对应的 Π_2 实例的子结构。

Super Mario Bros. (SMB) 是 NP-hard 问题

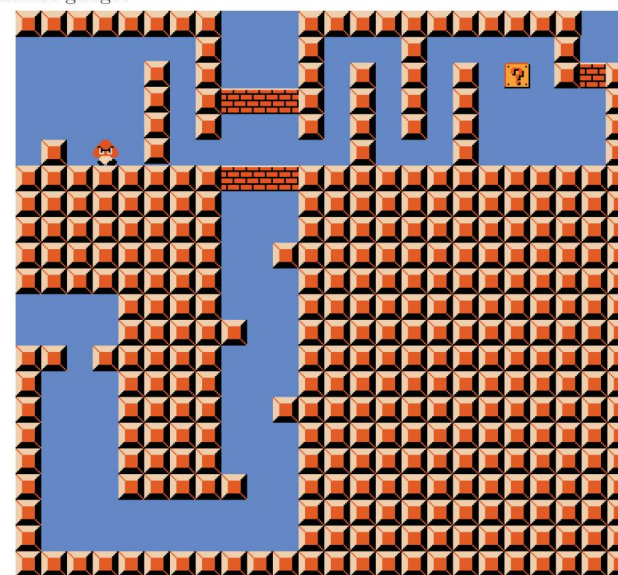
- 判定问题：给定 SMB 的任意关卡，是否存在过关玩法？
- 证明思路： $3SAT \leq_p SMB$ ，**构件法**



(a) Variable gadget



(b) Clause gadget



(c) Crossover gadget



P vs NP Problem



Suppose that you are organizing housing accommodations for a group of four hundred university students. Space is limited and only one hundred of the students will receive places in the dormitory. To complicate matters, the Dean has provided you with a list of pairs of incompatible students, and requested that no pair from this list appear in your final choice. This is an example of what computer scientists call an NP-problem, since it is easy to check if a given choice of one hundred students proposed by a coworker is satisfactory (i.e., no pair taken from your coworker's list also appears on the list from the Dean's office), however the task of generating such a list from scratch seems to be so hard as to be completely impractical. Indeed, the total number of ways of choosing one hundred students from the four hundred applicants is greater than the number of atoms in the known universe! Thus no future civilization could ever hope to build a supercomputer capable of solving the problem by brute force; that is, by checking every possible combination of 100 students. However, this apparent difficulty may only reflect the lack of ingenuity of your programmer. In fact, one of the outstanding problems in computer science is determining whether questions exist whose answer can be quickly checked, but which require an impossibly long time to solve by any direct procedure. Problems like the one listed above certainly seem to be of this kind, but so far no one has managed to prove that any of them really are so hard as they appear, i.e., that there really is no feasible way to generate an answer with the help of a computer. Stephen Cook and Leonid Levin formulated the P (i.e., easy to find) versus NP (i.e., easy to check) problem independently in 1971.

Image credit: on the left, Stephen Cook by [Jiří Janíček](#) (cropped). [CC BY-SA 3.0](#)

Suppose that you are organizing housing accommodations for a group of four hundred university students. Space is limited and only one hundred of the students will receive places in the dormitory. To complicate matters, the Dean has provided you with a list of pairs of incompatible students, and requested that no pair from this list appear in your final choice. This is an example of what computer scientists call an NP-problem, since

Rules:

[Rules for the Millennium Prizes](#)

Related Documents:

 [Official Problem Description](#)

 [Minesweeper](#)

Related Links:

[Lecture by Vijaya Ramachandran](#)

This problem is: Unsolved

<http://www.claymath.org/millennium-problems/p-vs-np-problem>