算法设计与分析

复杂性理论基础

目录

判定问题

判定问题：识别字符串集合 $L \subseteq X$

- X：字符串集合
- x：X 中的一个字符串（每个字符串对应一个实例）
- L：语言（X 中满足某种性质的子集）

[见原文第 3 页图]

任务：判定成员关系——$x \in L$ 是否成立

判定问题

判定问题：识别字符串集合 $L \subseteq X$

- X：字符串集合
- x：X 中的一个字符串（每个字符串对应一个实例）
- L：语言（X 中满足某种性质的子集）

[见原文第 3 页图]

任务：判定成员关系——$x \in L$ 是否成立

例子

- $X = \mathbb{N}$
- L 是素数集合 $= \{2, 3, 5, 7, 11, 13, \ldots\}$
- 判定 x 是否为素数

复杂性理论的动机

我们总是想知道一个给定的问题是否可以被算法高效地求解。

复杂性理论的动机

我们总是想知道一个给定的问题是否可以被算法高效地求解。

1. 精确地建模算法
   - 什么是计算?
   - 什么是可计算的?
2. 精确地定义”高效”的含义

目录

图灵机

1936 年，伦敦数学学会：《论可计算数及其在判定问题中的应用》

[见原文第 6 页图：Alan Turing]

图灵机

图灵机：一种自动机器，包含一条纸带（分成无限多个单元格）、一个控制单元和一个读写头。

[见原文第 7 页图]

图灵机

图灵机：一种自动机器，包含一条纸带（分成无限多个单元格）、一个控制单元和一个读写头。

[见原文第 7 页图]

- 在开始时，纸带的若干单元格中包含输入。其他位置为空。

图灵机

图灵机：一种自动机器，包含一条纸带（分成无限多个单元格）、一个控制单元和一个读写头。

[见原文第 7 页图]

- 在开始时，纸带的若干单元格中包含输入。其他位置为空。
- 在计算过程中，控制单元监视当前状态和头部值，可以执行以下操作：
  1. 擦除旧值并写入新值
  2. 改变当前状态
  3. 将读写头左移或右移

一个例子

[见原文第 8 页图]

一个例子

[见原文第 8 页图]

$a \to b$；左移；当前状态 $q_1 \to q_2$

一个例子

[见原文第 8 页图]

$a \to b$；左移；当前状态 $q_1 \to q_2$

[见原文第 8 页图]

图灵机的直觉

模拟人类解决问题的方式

图灵机的直觉

模拟人类解决问题的方式

- 图灵机有有限数量的状态（记忆）

图灵机的直觉

模拟人类解决问题的方式

- 图灵机有有限数量的状态（记忆）
- 图灵机配有一条纸带，包含无限多个单元格（纸张）

图灵机的直觉

模拟人类解决问题的方式

- 图灵机有有限数量的状态（记忆）
- 图灵机配有一条纸带，包含无限多个单元格（纸张）
- 可以从单元格中扫描符号或将符号打印到单元格（读和写）

形式化定义

定义 1（图灵机）

图灵机由 $(Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$ 组成

- Q：有限状态集合
- Σ：输入字母表
- Γ：工作字母表（包括 $\perp$，$\Sigma \subseteq \Gamma$）
- q_0：Q 的初始状态
- q_{acc}, q_{rej}：Q 的接受状态和拒绝状态
- δ：转移函数

$$\delta : (Q \setminus \{q_{acc}, q_{rej}\}) \times \Gamma \to Q \times \Gamma \times \{L, R\}$$

图灵机的运行时间

定义 2

我们用 $t_M(n)$ 表示图灵机的运行时间，它是图灵机在所有长度为 n 的输入上运行的最大步数。

多项式时间

$$\bigcup_{k\in\mathbb{N}} \mathsf{TIME}(n^k)$$

扩展的 Church-Turing 论题

[见原文第 12 页图：Alonzo Church & Alan Turing]

每个人对高效算法的直觉 = 多项式时间确定性图灵机

非确定性图灵机

非确定性图灵机与标准图灵机相似，但有以下不同：

非确定性图灵机

非确定性图灵机与标准图灵机相似，但有以下不同：

1. 非确定性图灵机可能根据多个可能的转移进行
   - 我们可以假设每个配置导致两个可能的配置

非确定性图灵机

非确定性图灵机与标准图灵机相似，但有以下不同：

1. 非确定性图灵机可能根据多个可能的转移进行
   - 我们可以假设每个配置导致两个可能的配置
2. 非确定性图灵机接受当且仅当至少有一个分支接受

非确定性图灵机

非确定性图灵机与标准图灵机相似，但有以下不同：

1. 非确定性图灵机可能根据多个可能的转移进行
   - 我们可以假设每个配置导致两个可能的配置
2. 非确定性图灵机接受当且仅当至少有一个分支接受

- 非确定性图灵机实际上并不对应任何现实世界的物理模型，它只是一个理论构造。

非确定性图灵机

非确定性图灵机与标准图灵机相似，但有以下不同：

1. 非确定性图灵机可能根据多个可能的转移进行
   - 我们可以假设每个配置导致两个可能的配置
2. 非确定性图灵机接受当且仅当至少有一个分支接受

- 非确定性图灵机实际上并不对应任何现实世界的物理模型，它只是一个理论构造。

非确定性并不能使图灵机识别更多的语言。

- 任何非确定性图灵机都可以被一个图灵机模拟（可能有指数级的时间开销），方法是使用广度优先搜索”并行”尝试非确定性机器的所有分支。

注意事项

关于图灵机的一些重要事实

注意事项

关于图灵机的一些重要事实

1. 任何图灵机都可以编码为 $\{0,1\}^*$。

注意事项

关于图灵机的一些重要事实

1. 任何图灵机都可以编码为 $\{0,1\}^*$。
2. 任何 $\{0,1\}^*$ 都表示某个图灵机。

注意事项

关于图灵机的一些重要事实

1. 任何图灵机都可以编码为 $\{0,1\}^*$。
2. 任何 $\{0,1\}^*$ 都表示某个图灵机。
3. 任何图灵机都有多个 $\{0,1\}^*$ 形式的表示。

注意事项

关于图灵机的一些重要事实

1. 任何图灵机都可以编码为 $\{0,1\}^*$。
2. 任何 $\{0,1\}^*$ 都表示某个图灵机。
3. 任何图灵机都有多个 $\{0,1\}^*$ 形式的表示。

为什么图灵机如此强大?

注意事项

关于图灵机的一些重要事实

1. 任何图灵机都可以编码为 $\{0,1\}^*$。
2. 任何 $\{0,1\}^*$ 都表示某个图灵机。
3. 任何图灵机都有多个 $\{0,1\}^*$ 形式的表示。

为什么图灵机如此强大?

- 图灵机有一条工作纸带（好记性不如烂笔头）

注意事项

关于图灵机的一些重要事实

1. 任何图灵机都可以编码为 $\{0,1\}^*$。
2. 任何 $\{0,1\}^*$ 都表示某个图灵机。
3. 任何图灵机都有多个 $\{0,1\}^*$ 形式的表示。

为什么图灵机如此强大?

- 图灵机有一条工作纸带（好记性不如烂笔头）
- 图灵机本身可以被视为数据！图灵机可以将另一个图灵机作为其输入。

通用图灵机

[见原文第 15 页图]

通用图灵机

[见原文第 15 页图]

通用图灵机

[见原文第 15 页图]

$\|$

[见原文第 15 页图：计算器类比]

目录

目录

时间复杂性层次：P 和 $\mathcal{NP}$

我们已经介绍了

- 判定问题 $L \subseteq X$ 的概念
- （非确定性）图灵机的概念

时间复杂性层次：P 和 $\mathcal{NP}$

我们已经介绍了

- 判定问题 $L \subseteq X$ 的概念
- （非确定性）图灵机的概念

我们说图灵机 M 接受（"1" 表示接受）语言 L，如果：

$$x \in L \Longleftrightarrow M(x) = 1$$

- L 被 M 判定（M 求解 L）

时间复杂性层次：P 和 $\mathcal{NP}$

我们已经介绍了

- 判定问题 $L \subseteq X$ 的概念
- （非确定性）图灵机的概念

我们说图灵机 M 接受（"1" 表示接受）语言 L，如果：

$$x \in L \Longleftrightarrow M(x) = 1$$

- L 被 M 判定（M 求解 L）

接下来，我们介绍两个重要的问题集合，它们由确定性图灵机和非确定性图灵机的时间复杂性来刻画：

P 和 $\mathcal{NP}$

P 复杂性类

定义 3（P 语言）

$L \in$ P 如果存在一个确定性多项式时间图灵机 M 使得

$$M(x) = 1 \Longleftrightarrow x \in L$$

P 复杂性类

定义 3（P 语言）

$L \in \mathsf{P}$ 如果存在一个确定性多项式时间图灵机 M 使得

$$M(x) = 1 \iff x \in L$$

- 多项式时间：最多执行 $p(n)$ 步，其中 $p(\cdot)$ 是某个多项式，n 是输入长度

P 复杂性类

定义 3（P 语言）

$L \in \mathsf{P}$ 如果存在一个确定性多项式时间图灵机 M 使得

$$M(x) = 1 \iff x \in L$$

- 多项式时间：最多执行 $p(n)$ 步，其中 $p(\cdot)$ 是某个多项式，n 是输入长度

$$\mathsf{P} = \bigcup_{k \in \mathbb{N}} \mathsf{TIME}(n^k)$$

P 复杂性类

定义 3（P 语言）

$L \in \mathsf{P}$ 如果存在一个确定性多项式时间图灵机 M 使得

$$M(x) = 1 \iff x \in L$$

- 多项式时间：最多执行 $p(n)$ 步，其中 $p(\cdot)$ 是某个多项式，n 是输入长度

$$\mathsf{P} = \bigcup_{k \in \mathbb{N}} \mathsf{TIME}(n^k)$$

P 语言的例子

- $L =$ {偶数}，M 只需检查最后一位是否为 0
- $L = \mathsf{PRIME}$，M 是 AKS 素性测试算法

$\mathcal{NP}$ 复杂性类

定义 4（$\mathcal{NP}$ 语言 - 传统定义）

$L \in \mathcal{NP}$ 如果存在一个非确定性多项式时间图灵机 M:

$$x \in L \Longleftrightarrow M(x) = 1$$

$$\mathcal{NP} = \bigcup_{k \in \mathbb{N}} \mathsf{NTIME}(n^k)$$

$\mathcal{NP}$ 复杂性类

定义 4（$\mathcal{NP}$ 语言 - 传统定义）

$L \in \mathcal{NP}$ 如果存在一个非确定性多项式时间图灵机 M:

$$x \in L \Longleftrightarrow M(x) = 1$$

$$\mathcal{NP} = \bigcup_{k \in \mathbb{N}} \mathsf{NTIME}(n^k)$$

警告

$\mathcal{NP}$ 表示非确定性多项式时间，而不是非多项式时间!

现代定义

定义 5（$\mathcal{NP}$ 复杂性 - 现代定义）

$L \in \mathcal{NP}$ 如果存在一个确定性多项式时间图灵机 M:

$$x \in L \Longleftrightarrow \exists w \in \{0,1\}^{\mathsf{poly}(n)} \text{ s.t. } M(x,w) = 1$$

现代定义

定义 5（$\mathcal{NP}$ 复杂性 - 现代定义）

$L \in \mathcal{NP}$ 如果存在一个确定性多项式时间图灵机 M:

$$x \in L \Longleftrightarrow \exists w \in \{0,1\}^{\mathsf{poly}(n)} \text{ s.t. } M(x,w) = 1$$

- 如果 $M(x,w) = 1$，w 被称为 $x \in L$ 的" 证据"。可以将 w 视为 $x \in L$ 的可高效验证的" 证书" 或" 证明"。

现代定义

定义 5（$\mathcal{NP}$ 复杂性 - 现代定义）

$L \in \mathcal{NP}$ 如果存在一个确定性多项式时间图灵机 M:

$$x \in L \Longleftrightarrow \exists w \in \{0,1\}^{\mathsf{poly}(n)} \text{ s.t. } M(x,w) = 1$$

- 如果 $M(x,w) = 1$，w 被称为 $x \in L$ 的" 证据"。可以将 w 视为 $x \in L$ 的可高效验证的" 证书" 或" 证明"。
- 我们要求 w 是短的，即 $|w| = \mathsf{poly}(n)$。这是自然的，因为 M 在 n 的多项式时间内运行。

现代定义

定义 5（$\mathcal{NP}$ 复杂性 - 现代定义）

$L \in \mathcal{NP}$ 如果存在一个确定性多项式时间图灵机 M:

$$x \in L \Longleftrightarrow \exists w \in \{0,1\}^{\mathsf{poly}(n)} \text{ s.t. } M(x,w) = 1$$

- 如果 $M(x,w) = 1$，w 被称为 $x \in L$ 的" 证据"。可以将 w 视为 $x \in L$ 的可高效验证的" 证书" 或" 证明"。
- 我们要求 w 是短的，即 $|w| = \mathsf{poly}(n)$。这是自然的，因为 M 在 n 的多项式时间内运行。

$L \in \mathcal{NP}$ 当且仅当 L 中每个元素的成员资格都有短证明。

现代定义

定义 5（$\mathcal{NP}$ 复杂性 - 现代定义）

$L \in \mathcal{NP}$ 如果存在一个确定性多项式时间图灵机 M:

$$x \in L \Longleftrightarrow \exists w \in \{0,1\}^{\mathsf{poly}(n)} \text{ s.t. } M(x,w) = 1$$

- 如果 $M(x,w) = 1$，w 被称为 $x \in L$ 的" 证据"。可以将 w 视为 $x \in L$ 的可高效验证的" 证书" 或" 证明"。
- 我们要求 w 是短的，即 $|w| = \mathsf{poly}(n)$。这是自然的，因为 M 在 n 的多项式时间内运行。

$L \in \mathcal{NP}$ 当且仅当 L 中每个元素的成员资格都有短证明。

传统定义和现代定义的等价性

- 尽管 M 是确定性机器，但它的第二个参数 w 捕获了定义中的非确定性。

$\mathcal{NP}$ 语言的例子 - 合数

$L =$ COMPOSITE（合数）

- 实例 x 是一个整数
- $x \in L$ 的证据 w 是 x 的一个非平凡因子
- M 只需检查 w 是否整除 x，这可以在多项式时间内完成

$\mathcal{NP}$ 语言的例子 - 合数

$L =$ COMPOSITE（合数）

- 实例 x 是一个整数
- $x \in L$ 的证据 w 是 x 的一个非平凡因子
- M 只需检查 w 是否整除 x，这可以在多项式时间内完成

COMPOSITE 的例子

- 实例：15
- 证据：3, 5

$\mathcal{NP}$ 语言的例子 - 合数

$L =$ COMPOSITE（合数）

- 实例 x 是一个整数
- $x \in L$ 的证据 w 是 x 的一个非平凡因子
- M 只需检查 w 是否整除 x，这可以在多项式时间内完成

COMPOSITE 的例子

- 实例：15
- 证据：3, 5

事实上，COMPOSITE 也属于 P（想想为什么?）

$\mathcal{NP}$ 语言的例子 - 合数

$L =$ COMPOSITE（合数）

- 实例 x 是一个整数
- $x \in L$ 的证据 w 是 x 的一个非平凡因子
- M 只需检查 w 是否整除 x，这可以在多项式时间内完成

COMPOSITE 的例子

- 实例：15
- 证据：3, 5

事实上，COMPOSITE 也属于 P（想想为什么?）

P 在补运算下封闭，即 $L \in \mathsf{P} \Rightarrow \bar{L} \in \mathsf{P}$。

- 确定性图灵机可以简单地在最后翻转接受/拒绝输出，这不会改变运行时间。

$\mathcal{NP}$ 语言的例子 - SAT 和 3-SAT

SAT：给定一个合取范式公式 Φ，检查它是否有满足的真值赋值。
3-SAT：每个子句恰好包含 3 个文字的 SAT
证据：布尔变量的真值赋值

$\mathcal{NP}$ 语言的例子 - SAT 和 3-SAT

SAT：给定一个合取范式公式 Φ，检查它是否有满足的真值赋值。
3-SAT：每个子句恰好包含 3 个文字的 SAT
证据：布尔变量的真值赋值

3-SAT 的例子

- 实例 $\Phi = (x_1 \vee x_2 \vee x_3) \wedge (\bar{x_1} \vee x_2 \vee \bar{x_3}) \wedge (x_1 \vee \bar{x_2} \vee x_4)$
- 证据：$x_1 = 1, x_2 = 1, x_3 = 0, x_4 = 0$

$\mathcal{NP}$ 语言的例子 - 哈密顿路径

哈密顿图：给定一个无向图 $G = (V, E)$，是否存在一条访问每个节点的简单路径?

[见原文第 24 页图：哈密顿图（一条路径恰好经过每个顶点一次）]

证据：一条路径
M 检查该路径是否恰好包含 V 中的每个节点一次

P vs. $\mathcal{NP}$

根据定义，P $\subseteq \mathcal{NP}$。因为 $L \in \mathsf{P} \Rightarrow L \in \mathcal{NP}$:

- $M'(x, w)$ 总是可以设 $w = \perp$ 并使用 M 判定 $x \in L$
- 或者，" 短的" M 可以被视为 $x \in L$ 的证据。想想为什么 M 的描述是短的?

P vs. $\mathcal{NP}$

根据定义，P $\subseteq \mathcal{NP}$。因为 $L \in \mathsf{P} \Rightarrow L \in \mathcal{NP}$:

- $M'(x, w)$ 总是可以设 $w = \perp$ 并使用 M 判定 $x \in L$
- 或者，" 短的" M 可以被视为 $x \in L$ 的证据。想想为什么 M 的描述是短的?
- P $=$ 所有实例都可以被高效判定的判定问题集合
- $\mathcal{NP}$ $=$ " 是" 实例可以用短证明被高效判定的判定问题集合

P vs. $\mathcal{NP}$

根据定义，$\mathsf{P} \subseteq \mathcal{NP}$。因为 $L \in \mathsf{P} \Rightarrow L \in \mathcal{NP}$：

- $M'(x, w)$ 总是可以设 $w = \perp$ 并使用 M 判定 $x \in L$
- 或者，" 短的" M 可以被视为 $x \in L$ 的证据。想想为什么 M 的描述是短的?
- $\mathsf{P} =$ 所有实例都可以被高效判定的判定问题集合
- $\mathcal{NP} =$ " 是" 实例可以用短证明被高效判定的判定问题集合

1971 年：Cook, Edmonds, Levin, Yablonski, Gödel
也许是理论计算机科学中最突出的问题：

$$\mathsf{P} \stackrel{?}{=} \mathcal{NP}$$

$\mathrm{P} = \mathcal{NP}$

[见原文第 26 页图]

如果 $\mathrm{P} = \mathcal{NP}$

现代密码学的基础将崩塌!

[见原文第 27 页图]

如果 $\mathrm{P} = \mathcal{NP}$

现代密码学的基础将崩塌!

[见原文第 27 页图]

- 我们所知的密码学可能变得不可能
- 密码学研究人员将失业

如果 $\mathrm{P} = \mathcal{NP}$

现代密码学的基础将崩塌!

[见原文第 27 页图]

- 我们所知的密码学可能变得不可能
- 密码学研究人员将失业

原则上，生活的方方面面都可以被高效且全局地优化 $\cdots$

- $\cdots$ 我们所知的生活将会不同!

$\mathrm{P} = \mathcal{NP}$ 的后果

$\mathrm{P} = \mathcal{NP} \Rightarrow$ 单向函数不存在

$\mathsf{P} = \mathcal{NP}$ 的后果

$$\mathsf{P} = \mathcal{NP} \Rightarrow \text{单向函数不存在}$$

设 $f\colon \{0,1\}^n \to \{0,1\}^m$。为了高效地找到 y 的原像 x，思路是逐位确定 x。
$f(x_1||\cdots||x_n) = y$。

$\mathsf{P} = \mathcal{NP}$ 的后果

$$\mathsf{P} = \mathcal{NP} \Rightarrow \text{单向函数不存在}$$

设 $f\colon \{0,1\}^n \to \{0,1\}^m$。为了高效地找到 y 的原像 x，思路是逐位确定 x。

$f(x_1||\cdots||x_n) = y$。

定义一族语言 $L_i = \{(y,z)|\exists w \text{ s.t. } y = f(z||w)\}$，其中 $z \in \{0,1\}^i$，$w \in \{0,1\}^{n-i}$

- $L_i \in \mathcal{NP}$，因此根据假设也属于 P，我们定义 Invert 算法：

Algorithm 3 Invert(y)

```
z = ε
for i ← 1 to n do
    if (y, z||0) ∈ L_i then
        z = z||0
    else
        z = z||1
    end
end
```

反方向

$$\text{单向函数存在} \Rightarrow \mathrm{P} \neq \mathcal{NP}$$

- 我们有许多候选的单向函数，但它们需要假设。

反方向

$$\text{单向函数存在} \Rightarrow \mathrm{P} \neq \mathcal{NP}$$

- 我们有许多候选的单向函数，但它们需要假设。

警告

单向函数不存在并不意味着 $\mathrm{P} = \mathcal{NP}$

共识

[见原文第 30 页图]

$\mathsf{P} \subset \mathcal{NP}$ 的证据

共识观点：$\mathsf{P} \neq \mathcal{NP}$

$\mathrm{P} \subset \mathcal{NP}$ 的证据

共识观点：$\mathrm{P} \neq \mathcal{NP}$

- 因为 $\mathcal{NP}$ 包含许多被认为不在 P 中的语言，如：3-SAT 和哈密顿图。

$\mathrm{P} \subset \mathcal{NP}$ 的证据

共识观点：$\mathrm{P} \neq \mathcal{NP}$

- 因为 $\mathcal{NP}$ 包含许多被认为不在 P 中的语言，如：3-SAT 和哈密顿图。

问：如何求解一个有 n 个变量的 3-SAT 实例?

$\mathrm{P} \subset \mathcal{NP}$ 的证据

共识观点：$\mathrm{P} \neq \mathcal{NP}$

- 因为 $\mathcal{NP}$ 包含许多被认为不在 P 中的语言，如：3-SAT 和哈密顿图。

问：如何求解一个有 n 个变量的 3-SAT 实例?
穷举搜索：尝试所有 2^n 种真值赋值。

$\mathrm{P} \subset \mathcal{NP}$ 的证据

共识观点：$\mathrm{P} \neq \mathcal{NP}$

- 因为 $\mathcal{NP}$ 包含许多被认为不在 P 中的语言，如：3-SAT 和哈密顿图。

问：如何求解一个有 n 个变量的 3-SAT 实例?
穷举搜索：尝试所有 2^n 种真值赋值。
问：我们能做得更聪明一些吗?

P $\subset \mathcal{NP}$ 的证据

共识观点：P $\neq \mathcal{NP}$

- 因为 $\mathcal{NP}$ 包含许多被认为不在 P 中的语言，如：3-SAT 和哈密顿图。

问：如何求解一个有 n 个变量的 3-SAT 实例?
穷举搜索：尝试所有 2^n 种真值赋值。
问：我们能做得更聪明一些吗?
猜想：3-SAT 没有多项式时间算法

$$\underbrace{\qquad\qquad}_{\text{难解的}}$$

目录

归约的动机

$\mathcal{NP}$ 是许多问题的集合。

如何弄清它们之间的关系?

一个核心方法是寻找归约

多项式时间可归约性

语言 L' 多项式时间可归约到语言 L，或归约到语言 L，记作 $L' \leq_p L$，如果存在一个确定性多项式时间函数 $R : L' \to L$ 使得：

$$x \in L' \iff R(x) \in L$$

多项式时间可归约性

语言 L' 多项式时间可归约到语言 L，或归约到语言 L，记作 $L' \leq_p L$，如果存在一个确定性多项式时间函数 $R : L' \to L$ 使得：

$$x \in L' \iff R(x) \in L$$

R 被称为从 L 到 L' 的多项式时间归约。

多项式时间可归约性

语言 L' 多项式时间可归约到语言 L，或归约到语言 L，记作 $L' \leq_p L$，如果存在一个确定性多项式时间函数 $R : L' \to L$ 使得：

$$x \in L' \iff R(x) \in L$$

R 被称为从 L 到 L' 的多项式时间归约。

$L' \leq_p L$ 意味着 L' 不比 L 更难

多项式时间可归约性

语言 L' 多项式时间可归约到语言 L，或归约到语言 L，记作 $L' \leq_p L$，如果存在一个确定性多项式时间函数 $R : L' \to L$ 使得：

$$x \in L' \iff R(x) \in L$$

R 被称为从 L 到 L' 的多项式时间归约。

$$L' \leq_p L \text{ 意味着 } L' \text{ 不比 } L \text{ 更难}$$

我们应该注意：

- R 的方向
- R 的时间复杂度

$\mathcal{NP}$-困难

定义 6（$\mathcal{NP}$-困难）

L 被称为 $\mathcal{NP}$-困难的，如果对于每个 $\mathcal{NP}$ 语言 L'，存在一个确定性多项式时间算法（一个归约）R:

$$x \in L' \iff R(x) \in L$$

$\mathcal{NP}$-困难

定义 6（$\mathcal{NP}$-困难）

L 被称为 $\mathcal{NP}$-困难的，如果对于每个 $\mathcal{NP}$ 语言 L'，存在一个确定性多项式时间算法（一个归约）R:

$$x \in L' \iff R(x) \in L$$

- 我们可以理解为 $\mathcal{NP}$ 中的语言不比 $\mathcal{NP}$-困难中的语言更难。

$\mathcal{NP}$-困难

定义 6（$\mathcal{NP}$-困难）

L 被称为 $\mathcal{NP}$-困难的，如果对于每个 $\mathcal{NP}$ 语言 L'，存在一个确定性多项式时间算法（一个归约）R:

$$x \in L' \iff R(x) \in L$$

- 我们可以理解为 $\mathcal{NP}$ 中的语言不比 $\mathcal{NP}$-困难中的语言更难。

事实：$\mathcal{NP}$-困难中的语言可能不在 $\mathcal{NP}$ 中。

$\mathcal{NP}$ 完全

定义 7（$\mathcal{NP}$ 完全）

L 是 $\mathcal{NP}$ 完全的，如果它是 $\mathcal{NP}$-困难的，并且它本身在 $\mathcal{NP}$ 中。

$\mathcal{NP}$ 完全

定义 7（$\mathcal{NP}$ 完全）

L 是 $\mathcal{NP}$ 完全的，如果它是 $\mathcal{NP}$-困难的，并且它本身在 $\mathcal{NP}$ 中。

定义直觉：$\mathcal{NP}$ 完全代表 $\mathcal{NP}$ 中最难的问题集合。

$\mathcal{NP}$ 完全

定义 7（$\mathcal{NP}$ 完全）

L 是 $\mathcal{NP}$ 完全的，如果它是 $\mathcal{NP}$-困难的，并且它本身在 $\mathcal{NP}$ 中。

定义直觉：$\mathcal{NP}$ 完全代表 $\mathcal{NP}$ 中最难的问题集合。

- 如果我们为 $\mathcal{NP}$ 完全中的任何问题找到一个高效算法，我们就可以求解 $\mathcal{NP}$ 中的所有问题。

$\mathcal{NP}$ 完全的意义

定理 (定理 8)

假设 $Y \in \mathcal{NP}$完全，则 $Y \in \mathrm{P} \Longleftrightarrow \mathrm{P} = \mathcal{NP}$。

$\mathcal{NP}$ 完全的意义

定理 (定理 8)

假设 $Y \in \mathcal{NP}$完全，则 $Y \in \mathsf{P} \iff \mathsf{P} = \mathcal{NP}$。

$\Leftarrow$: $Y \in \mathcal{NP}$ 完全，因此当然 $Y \in \mathcal{NP}$。现在假设 $\mathsf{P} = \mathcal{NP}$，我们有 $Y \in \mathsf{P}$。

$\mathcal{NP}$ 完全的意义

定理 (定理 8)

假设 $Y \in \mathcal{NP}$完全，则 $Y \in \mathsf{P} \iff \mathsf{P} = \mathcal{NP}$。

$\Leftarrow$: $Y \in \mathcal{NP}$ 完全，因此当然 $Y \in \mathcal{NP}$。现在假设 $\mathsf{P} = \mathcal{NP}$，我们有 $Y \in \mathsf{P}$。
$\Rightarrow$: $\forall X \in \mathcal{NP}$，$X \leq_p Y$，因为 $Y \in \mathcal{NP}$ 完全。现在假设 $Y \in \mathsf{P}$，我们进一步有 $\mathcal{NP} \subseteq \mathsf{P}$。
我们已经知道 $\mathsf{P} \subseteq \mathcal{NP}$，因此 $\mathsf{P} = \mathcal{NP}$。

$\mathcal{NP}$ 完全的意义

定理 (定理 8)

假设 $Y \in \mathcal{NP}$完全，则 $Y \in \mathsf{P} \iff \mathsf{P} = \mathcal{NP}$。

$\Leftarrow$: $Y \in \mathcal{NP}$ 完全，因此当然 $Y \in \mathcal{NP}$。现在假设 $\mathsf{P} = \mathcal{NP}$，我们有 $Y \in \mathsf{P}$。
$\Rightarrow$: $\forall X \in \mathcal{NP}$，$X \leq_p Y$，因为 $Y \in \mathcal{NP}$ 完全。现在假设 $Y \in \mathsf{P}$，我们进一步有 $\mathcal{NP} \subseteq \mathsf{P}$。
我们已经知道 $\mathsf{P} \subseteq \mathcal{NP}$，因此 $\mathsf{P} = \mathcal{NP}$。

- 这个定理本质上说明了 $\mathsf{P} \cap \mathcal{NPC}$ 非空当且仅当 $\mathsf{P} = \mathcal{NP}$。

重新审视 P vs. $\mathcal{NP}$

压倒性的共识（仍然是）：$\mathrm{P} \neq \mathcal{NP}$。

重新审视 P vs. $\mathcal{NP}$

压倒性的共识（仍然是）：$\mathrm{P} \neq \mathcal{NP}$。

[见原文第 38 页图：$\mathrm{P} \neq \mathcal{NP}$ 和 $\mathrm{P} = \mathcal{NP}$]

重新审视 P vs. $\mathcal{NP}$

压倒性的共识（仍然是）：$\mathrm{P} \neq \mathcal{NP}$。

[见原文第 38 页图：$\mathrm{P} \neq \mathcal{NP}$ 和 $\mathrm{P} = \mathcal{NP}$]

为什么我们相信 $\mathrm{P} \neq \mathcal{NP}$？因为某些问题看起来明显更难。

[见原文第 39 页图]

Cook-Levin 定理

是否存在" 自然的"$\mathcal{NP}$ 完全问题?

Cook-Levin 定理

是否存在" 自然的"$\mathcal{NP}$ 完全问题?

[见原文第 40 页图：Stephen Cook & Leonid Levin]

Cook-Levin 定理

是否存在" 自然的"$\mathcal{NP}$ 完全问题?

[见原文第 40 页图：Stephen Cook & Leonid Levin]

Cook-Levin 定理证明了 SAT $\in \mathcal{NPC}$。通过 Cook-Levin 归约，我们可以将 $\mathcal{NP}$ 中的任何问题归约到 SAT。

Cook-Levin 定理

是否存在" 自然的"$\mathcal{NP}$ 完全问题?

[见原文第 40 页图：Stephen Cook & Leonid Levin]

Cook-Levin 定理证明了 SAT $\in \mathcal{NPC}$。通过 Cook-Levin 归约，我们可以将 $\mathcal{NP}$ 中的任何问题归约到 SAT。

- $\mathcal{NP}$ 完全的其他例子包括图 3-着色、图哈密顿性等。

目录

co-$\mathcal{NP}$ 的动机

$\mathcal{NP}$ 的不对称性：我们只需要" 是" 实例的短证书。

$$x \in L \iff \exists w \in \{0,1\}^{\mathsf{poly}(|x|)} \text{ s.t. } M(x,w) = 1$$
$$x \notin L \iff \forall w \in \{0,1\}^{\mathsf{poly}(|x|)} \text{ s.t. } M(x,w) = 0$$

co-$\mathcal{NP}$ 的动机

$\mathcal{NP}$ 的不对称性：我们只需要" 是" 实例的短证书。

$$x \in L \Longleftrightarrow \exists w \in \{0,1\}^{\mathsf{poly}(|x|)} \text{ s.t. } M(x,w) = 1$$
$$x \notin L \Longleftrightarrow \forall w \in \{0,1\}^{\mathsf{poly}(|x|)} \text{ s.t. } M(x,w) = 0$$

SAT vs. UN-SAT

- 可以通过指定一个赋值来证明一个 CNF 公式是可满足的
- 如何证明一个公式是不可满足的?

co-$\mathcal{NP}$ 的动机

$\mathcal{NP}$ 的不对称性：我们只需要" 是" 实例的短证书。

$$x \in L \iff \exists w \in \{0,1\}^{\mathsf{poly}(|x|)} \text{ s.t. } M(x,w) = 1$$
$$x \notin L \iff \forall w \in \{0,1\}^{\mathsf{poly}(|x|)} \text{ s.t. } M(x,w) = 0$$

SAT vs. UN-SAT

- 可以通过指定一个赋值来证明一个 CNF 公式是可满足的
- 如何证明一个公式是不可满足的?

HAM-CYCLE vs. NO-HAM-CYCLE

- 可以通过指定一条路径来证明一个图是哈密顿的
- 如何证明一个图不是哈密顿的?

co-$\mathcal{NP}$

问：如何分类 UN-SAT 和 NO-HAM-CYCLE?

co-$\mathcal{NP}$

问：如何分类 UN-SAT 和 NO-HAM-CYCLE?

- SAT $\equiv_p$ UN-SAT，HAM-CYCLE $\equiv_p$ NO-HAM-CYCLE，但 SAT 和 HAM-CYCLE 都不知道在 P 中。

co-$\mathcal{NP}$

问：如何分类 UN-SAT 和 NO-HAM-CYCLE?

- SAT $\equiv_p$ UN-SAT，HAM-CYCLE $\equiv_p$ NO-HAM-CYCLE，但 SAT 和 HAM-CYCLE 都不知道在 P 中。
- SAT $\in \mathcal{NPC}$ 且 HAM-CYCLE $\in \mathcal{NPC}$，但 UN-SAT 和 NO-HAM-CYCLE 都不知道在 $\mathcal{NP}$ 中。

co-$\mathcal{NP}$

问：如何分类 UN-SAT 和 NO-HAM-CYCLE?

- SAT $\equiv_p$ UN-SAT，HAM-CYCLE $\equiv_p$ NO-HAM-CYCLE，但 SAT 和 HAM-CYCLE 都不知道在 P 中。
- SAT $\in \mathcal{NPC}$ 且 HAM-CYCLE $\in \mathcal{NPC}$，但 UN-SAT 和 NO-HAM-CYCLE 都不知道在 $\mathcal{NP}$ 中。

给定一个判定问题 $L \subseteq X$，它的补 $\bar{L} \subseteq X$ 是将" 是" 和" 否" 实例交换的同一问题。

co-$\mathcal{NP}$

问：如何分类 UN-SAT 和 NO-HAM-CYCLE?

- SAT $\equiv_p$ UN-SAT，HAM-CYCLE $\equiv_p$ NO-HAM-CYCLE，但 SAT 和 HAM-CYCLE 都不知道在 P 中。
- SAT $\in \mathcal{NPC}$ 且 HAM-CYCLE $\in \mathcal{NPC}$，但 UN-SAT 和 NO-HAM-CYCLE 都不知道在 $\mathcal{NP}$ 中。

给定一个判定问题 $L \subseteq X$，它的补 $\bar{L} \subseteq X$ 是将" 是" 和" 否" 实例交换的同一问题。

补问题的例子

- $L = \{4, 6, 8, 9, 10, 12, 14, 15, \ldots\}$，$\bar{L} = \{2, 3, 5, 7, 11, 13, 17, 23, 29, \ldots\}$

co-$\mathcal{NP}$

问：如何分类 UN-SAT 和 NO-HAM-CYCLE?

- SAT $\equiv_p$ UN-SAT，HAM-CYCLE $\equiv_p$ NO-HAM-CYCLE，但 SAT 和 HAM-CYCLE 都不知道在 P 中。
- SAT $\in \mathcal{NPC}$ 且 HAM-CYCLE $\in \mathcal{NPC}$，但 UN-SAT 和 NO-HAM-CYCLE 都不知道在 $\mathcal{NP}$ 中。

给定一个判定问题 $L \subseteq X$，它的补 $\bar{L} \subseteq X$ 是将" 是" 和" 否" 实例交换的同一问题。

补问题的例子

- $L = \{4, 6, 8, 9, 10, 12, 14, 15, \ldots\}$，$\bar{L} = \{2, 3, 5, 7, 11, 13, 17, 23, 29, \ldots\}$

定义 9（co-$\mathcal{NP}$）

$\mathcal{NP}$ 中判定问题的补。

co-$\mathcal{NP}$

问：如何分类 UN-SAT 和 NO-HAM-CYCLE?

- SAT $\equiv_p$ UN-SAT，HAM-CYCLE $\equiv_p$ NO-HAM-CYCLE，但 SAT 和 HAM-CYCLE 都不知道在 P 中。
- SAT $\in \mathcal{NPC}$ 且 HAM-CYCLE $\in \mathcal{NPC}$，但 UN-SAT 和 NO-HAM-CYCLE 都不知道在 $\mathcal{NP}$ 中。

给定一个判定问题 $L \subseteq X$，它的补 $\bar{L} \subseteq X$ 是将" 是" 和" 否" 实例交换的同一问题。

补问题的例子

- $L = \{4, 6, 8, 9, 10, 12, 14, 15, \ldots\}$，$\bar{L} = \{2, 3, 5, 7, 11, 13, 17, 23, 29, \ldots\}$

定义 9（co-$\mathcal{NP}$）

$\mathcal{NP}$ 中判定问题的补。

例子：UN-SAT, NO-HAM-CYCLE, 和 PRIMES。

$\mathcal{NP}$ vs. co-$\mathcal{NP}$

基本的开放问题：$\mathcal{NP} = \text{co-}\mathcal{NP}$ 吗?

$\mathcal{NP}$ vs. co-$\mathcal{NP}$

基本的开放问题：$\mathcal{NP} = \text{co-}\mathcal{NP}$ 吗?

- " 是" 实例有简洁证书当且仅当" 否" 实例有吗?

$\mathcal{NP}$ vs. co-$\mathcal{NP}$

基本的开放问题：$\mathcal{NP} = \text{co-}\mathcal{NP}$ 吗?

- " 是" 实例有简洁证书当且仅当" 否" 实例有吗?
- 共识观点：否。

$\mathcal{NP}$ vs. co-$\mathcal{NP}$

基本的开放问题：$\mathcal{NP} = \text{co-}\mathcal{NP}$ 吗?

- " 是" 实例有简洁证书当且仅当" 否" 实例有吗?
- 共识观点：否。

定理 (定理 10)

如果 $\mathcal{NP} \neq \text{co-}\mathcal{NP}$，则 $\mathrm{P} \neq \mathcal{NP}$。

$\mathcal{NP}$ vs. co-$\mathcal{NP}$

基本的开放问题：$\mathcal{NP} = \text{co-}\mathcal{NP}$ 吗?

- " 是" 实例有简洁证书当且仅当" 否" 实例有吗?
- 共识观点：否。

定理 (定理 10)

如果 $\mathcal{NP} \neq \text{co-}\mathcal{NP}$，则 $\text{P} \neq \mathcal{NP}$。

证明思路

- P 在补运算下封闭。
- 如果 $\text{P} = \mathcal{NP}$，则 $\mathcal{NP}$ 在补运算下封闭。换句话说，$\mathcal{NP} = \text{co-}\mathcal{NP}$。
- 这是定理的逆否命题。

$\mathcal{NP} \cap$ co-$\mathcal{NP}$

良好刻画：[Edmonds 1965] $\mathcal{NP} \cap$ co-$\mathcal{NP}$

$\mathcal{NP} \cap$ co-$\mathcal{NP}$

良好刻画：[Edmonds 1965] $\mathcal{NP} \cap$ co-$\mathcal{NP}$
如果问题 $X \in \mathcal{NP} \cap$ co-$\mathcal{NP}$，则：

- 对于" 是" 实例，有简洁证书
- 对于" 否" 实例，有简洁取消资格者

$\mathcal{NP} \cap$ co-$\mathcal{NP}$

良好刻画：[Edmonds 1965] $\mathcal{NP} \cap$ co-$\mathcal{NP}$
如果问题 $X \in \mathcal{NP} \cap$ co-$\mathcal{NP}$，则：

- 对于" 是" 实例，有简洁证书
- 对于" 否" 实例，有简洁取消资格者

为推理问题提供了概念上的杠杆。

目录

随机化算法的动机

图灵机建模确定性算法。
图灵机似乎没有捕捉到现实的一个方面——在计算过程中做出随机选择的能力

- 大多数编程语言提供内置的随机数生成器

考虑可以抛硬币的算法是有意义的，即使用随机比特源。这类算法已经被隐式研究了很长时间。

- 通过取小样本来估计大样本的事实
- 模拟本身是概率性的现实世界系统，如核裂变和股票市场
- 微分方程

概率图灵机

概率多项式时间图灵机建模概率算法。

[见原文第 48 页图]

概率图灵机 vs. 非确定性图灵机

非确定性图灵机是具有两个转移函数的图灵机。概率图灵机在语法上类似。
区别在于我们如何解释图灵机的工作。

- 在概率图灵机中，每个转移以概率 $1/2$ 进行，运行时间 t 的计算在所有计算的图中产生 2^t 个分支，每个分支以概率 $1/2^t$ 进行。$\Pr[M(x) = 1]$ 就是以 M 输出 1 结束的分支的比例。
- 在非确定性图灵机中，$M(x) = 1$ 当且仅当存在一个输出 1 的分支

在概念层面上，概率图灵机和非确定性图灵机非常不同

- 概率图灵机像图灵机一样，与非确定性图灵机不同，旨在建模现实的计算设备。

目录

有界错误概率多项式时间

定义 11（BPP 复杂性）

$L \in$ BPP 当且仅当存在一个概率多项式时间图灵机 M 使得：

$$\forall x \in L : \Pr[M(x) = 1] \geq \frac{1}{2} + \epsilon$$

$$\forall x \notin L : \Pr[M(x) = 1] \leq \frac{1}{2} - \epsilon$$

直觉：M 正确的可能性大于错误的可能性。

有界错误概率多项式时间

定义 11（BPP 复杂性）

$L \in$ BPP 当且仅当存在一个概率多项式时间图灵机 M 使得：

$$\forall x \in L : \Pr[M(x) = 1] \geq \frac{1}{2} + \epsilon$$

$$\forall x \notin L : \Pr[M(x) = 1] \leq \frac{1}{2} - \epsilon$$

直觉：M 正确的可能性大于错误的可能性。

关键性质

- 有界错误：M 必须明显更可能正确而不是错误，即错误与随机猜测 $1/2$ 相差一个明显的间隙。
- 错误降低：通过重复然后取多数投票，错误可以降低到 2^{-n}，只需多项式开销。
- 实际意义：BPP 捕获了实践中使用的高效随机化算法（例如，AKS 之前的素性测试）。

错误降低

在实践中，错误概率 $1/2 - \epsilon$（间隙是 ϵ）可能不可接受。

错误降低

在实践中，错误概率 $1/2 - \epsilon$（间隙是 ϵ）可能不可接受。
然而，间隙 ϵ 的选择可以是任意的，集合 BPP 将保持不变。

- 它可以是 $(0, 1/2)$ 之间的任何常数。
- 它甚至不必是常数：ϵ 可以小到 n^{-c}，其中 c 是任何正常数，n 是输入长度。

错误降低

在实践中，错误概率 $1/2-\epsilon$（间隙是 ϵ）可能不可接受。
然而，间隙 ϵ 的选择可以是任意的，集合 BPP 将保持不变。

- 它可以是 $(0,1/2)$ 之间的任何常数。
- 它甚至不必是常数：ϵ 可以小到 n^{-c}，其中 c 是任何正常数，n 是输入长度。

方法是独立运行 M 若干次，然后取多数投票。Chernoff 界保证多数运行错误的概率呈指数下降。

Chernoff 界（下尾）：设 $\{X_i\}$ 是独立的 0-1 随机变量，使得 $\Pr[X_i=b]\geq 1/2+\epsilon$，$X=\sum_{i=1}^{m}X_i$。

$$\Pr[X\neq b]\leq \frac{1}{e^{\epsilon^2 m/2}}$$

设 $m\geq n^{2c}+1$ 确保错误概率在 n 中可忽略。

为什么可靠性放大不起作用?

在零知识证明文献中，我们关心的错误是可靠性错误：

$$\epsilon = \Pr[\langle P, V \rangle(x) : x \notin L]$$

我们可以通过以下方式将 ϵ 降低到 ϵ^n：

- 顺序/并行运行协议 n 次
- 当且仅当所有重复都接受时才接受

这种方法对 BPP 不起作用，因为在零知识证明中完备性错误通常为 0，而在 BPP 中错误是双边的。

- 判定算法不知道输入实例属于" 是" 还是" 否"。它必须应用相同的策略。

概率多项式时间

定义 12（PP 复杂性）

$L \in$ PP 当且仅当存在一个概率多项式时间图灵机 M 使得：

$$\forall x \in L : \Pr[M(x) = 1] \geq 1/2$$
$$\forall x \notin L : \Pr[M(x) = 1] \leq 1/2$$

概率多项式时间

定义 12（PP 复杂性）

$L \in$ PP 当且仅当存在一个概率多项式时间图灵机 M 使得：

$$\forall x \in L : \Pr[M(x) = 1] \geq 1/2$$
$$\forall x \notin L : \Pr[M(x) = 1] \leq 1/2$$

关键性质

- 无界错误：接受概率可能超过 $1/2$ 一个指数级小的量。
- 无法错误降低：与 $1/2$ 的间隙可能太小而无法高效降低。
- 极其强大：PP 包含许多困难的类。

直觉：M 勉强更可能正确而不是错误。

为什么多数投票有效?

回想 $X = X_1 + \cdots + X_m$ 是二项分布。当 ϵ 明显时，经过多项式次重复后，$x \in L$ 和 $x \notin L$ 导致的两个分布在很大程度上分离。否则，它们混合在一起；很难找到分割线。

[见原文第 55 页图]

这也解释了对于 PP，如果 ϵ 可忽略地接近 0，多数投票不起作用。例如：

$$x \in L \Rightarrow \Pr[M(x) = 1] \geq \frac{1}{2} + \frac{1}{2^n}$$
$$x \notin L \Rightarrow \Pr[M(x) = 1] \leq \frac{1}{2} - \frac{1}{2^n}$$

关系和能力

包含关系

$$\text{BPP} \subseteq \text{PP}$$

- 任何有界错误算法也是无界错误算法。
- 这个包含关系被认为是严格的。

n^{-c} vs 2^{-n} 之间的差异在数值上看起来很小，但在计算上是巨大的：

- BPP 需要明显的概率优势，并建模现实的随机化计算
- PP 允许超过猜测的无穷小优势，使其更强大，更接近计数复杂性而非高效计算。

关于 BPP

BPP 是最大的实用问题类之一，因为 BPP 中的问题有高效的概率算法，可以在真实的现代机器上快速运行。

关于 BPP

BPP 是最大的实用问题类之一，因为 BPP 中的问题有高效的概率算法，可以在真实的现代机器上快速运行。

- 显然，BPP $\supseteq$ P，因为确定性机器是概率机器的特例。

关于 BPP

BPP 是最大的实用问题类之一，因为 BPP 中的问题有高效的概率算法，可以在真实的现代机器上快速运行。

- 显然，BPP $\supseteq$ P，因为确定性机器是概率机器的特例。
- 许多问题已知在 BPP 中但不知道在 P 中。这类问题的数量正在减少，人们猜测 P $=$ BPP。

关于 BPP

BPP 是最大的实用问题类之一，因为 BPP 中的问题有高效的概率算法，可以在真实的现代机器上快速运行。

- 显然，BPP $\supseteq$ P，因为确定性机器是概率机器的特例。
- 许多问题已知在 BPP 中但不知道在 P 中。这类问题的数量正在减少，人们猜测 P $=$ BPP。

长期以来，已知在 BPP 中但不知道在 P 中的最著名问题之一是 PRIME。

关于 BPP

BPP 是最大的实用问题类之一，因为 BPP 中的问题有高效的概率算法，可以在真实的现代机器上快速运行。

- 显然，BPP $\supseteq$ P，因为确定性机器是概率机器的特例。
- 许多问题已知在 BPP 中但不知道在 P 中。这类问题的数量正在减少，人们猜测 P $=$ BPP。

长期以来，已知在 BPP 中但不知道在 P 中的最著名问题之一是 PRIME。[AKS04] 给出了 PRIME 的确定性多项式时间算法，从而证明它在 P 中。

Gödel 奖和 Fulkerson 奖

单边和零边错误

ZPP：概率多项式时间图灵机总是返回正确的”是”或”否”答案，或以低概率停止，即对于每个输入，运行时间在期望上是多项式的

[见原文第 58 页图]

- BPP：蒙特卡洛算法（概率性的）在严格多项式运行时间内很可能正确
- ZPP：拉斯维加斯算法（概率性的）在期望多项式运行时间内总是正确

BPP 与其他概率复杂性类的关系

BQP（有界错误量子多项式时间）：可由量子图灵机在多项式时间内以有界错误求解的判定问题类

- 它是 BPP 的量子类比

BPP 的限制

共识：$\mathsf{P} \subseteq \mathsf{ZPP} = \mathsf{RP} \cap \text{co-RP} \subseteq \mathsf{BPP} \subseteq \mathcal{NP}$

$\mathsf{P} \subseteq \mathsf{BPP}$

- 一个在 BPP 中但仍不知道在 P 中的重要问题例子是多项式恒等测试——当你可以访问多项式对于任何给定输入的值，但不能访问系数时，确定多项式是否恒等于零多项式。

$\mathsf{BPP} \subseteq \mathcal{NP}$

- Adleman 定理：$\mathsf{BPP} \subseteq \mathsf{P/poly}$（多项式大小的布尔电路）
- Karp-Levin 定理：$\mathcal{NP} \subseteq \mathsf{P/poly} \Rightarrow \mathsf{PH} = \Sigma_2^P$

因此，$\mathcal{NP} \subseteq \mathsf{BPP}$ 将意味着 PH 坍缩，这不太可能是真的。换句话说，$\mathcal{NPC}$ 问题不存在有界错误概率算法。

目录

PSPACE

P：可由确定性图灵机在多项式时间内求解的判定问题。
PSPACE：可由确定性图灵机在多项式空间内求解的判定问题。

PSPACE 例子

- 二进制计数器。从 0 数到 $2^n - 1$。
- 算法。使用 n 位里程表。

观察 P $\subseteq$ PSPACE

- 这是因为多项式时间算法只能消耗多项式空间。

$\mathcal{NP}$ 和 PSPACE 的关系

声明：3-SAT $\in$ PSPACE。

声明的证明

- 使用计数器枚举所有 2^n 种可能的真值赋值。
- 检查每个赋值是否满足所有子句。

定理：$\mathcal{NP} \subseteq$ PSPACE

- 考虑任意问题 $Y \in \mathcal{NP}$。
- 由于 $Y \leq_p$ 3-SAT，存在在多项式时间加上对 3-SAT 黑盒的多项式次调用中求解 Y 的算法。
- 因此，Y 可以由确定性图灵机在多项式空间内求解。

容易验证 co-$\mathcal{NP} \subseteq$ PSPACE。

3-SAT 的优点

直觉上，使用多项式空间，确定性图灵机可以简单地枚举所有证据然后检查。为什么我们要借助 3-SAT?

3-SAT 的优点

直觉上，使用多项式空间，确定性图灵机可以简单地枚举所有证据然后检查。为什么我们要借助 3-SAT?
原因是对于某些 $\mathcal{NP}$ 语言，我们只知道 $|w|$ 的渐近界。

3-SAT 的优点

直觉上，使用多项式空间，确定性图灵机可以简单地枚举所有证据然后检查。为什么我们要借助 3-SAT?
原因是对于某些 $\mathcal{NP}$ 语言，我们只知道 $|w|$ 的渐近界。
但对于 3-SAT，给定一个实例 x，我们知道其证据的确切长度。

目录

判定与搜索

到目前为止，我们只考虑判定问题，即成员资格问题

判定与搜索

到目前为止，我们只考虑判定问题，即成员资格问题
更常见的是，我们想找到给定实例的解，即相应的证据。

- 这类问题本质上是搜索问题。

判定与搜索

到目前为止，我们只考虑判定问题，即成员资格问题
更常见的是，我们想找到给定实例的解，即相应的证据。

- 这类问题本质上是搜索问题。

问：判定问题是否足够有表达力？或者与其搜索版本一样难？

判定与搜索

到目前为止，我们只考虑判定问题，即成员资格问题
更常见的是，我们想找到给定实例的解，即相应的证据。

- 这类问题本质上是搜索问题。

问：判定问题是否足够有表达力？或者与其搜索版本一样难？
答：对于 $\mathcal{NP}$ 语言。是的！

3-SAT 的例子

3-SAT（判定）：判定给定公式 ψ 是否可满足

3-SAT 的例子

3-SAT（判定）：判定给定公式 ψ 是否可满足
3-SAT（搜索）：如果给定公式 ψ 可满足，找到它的一个赋值

3-SAT 的例子

3-SAT（判定）：判定给定公式 ψ 是否可满足
3-SAT（搜索）：如果给定公式 ψ 可满足，找到它的一个赋值

定理 (定理 13)

如果存在 *3-SAT* 的多项式时间算法，则存在找到其赋值的多项式时间算法。

3-SAT 的例子

3-SAT（判定）：判定给定公式 ψ 是否可满足
3-SAT（搜索）：如果给定公式 ψ 可满足，找到它的一个赋值

定理 (定理 13)

如果存在 *3-SAT* 的多项式时间算法，则存在找到其赋值的多项式时间算法。

- 定义一个 $\mathcal{NP}$ 语言
$$L = \{(\phi, r) : \exists w \text{ s.t. } \phi(w) = 1 \wedge w < r\}$$
- 给定 L 的黑盒访问（最多 $\log |W|$ 次），可以使用二分搜索找到 w。

3-SAT 的例子

3-SAT（判定）：判定给定公式 ψ 是否可满足
3-SAT（搜索）：如果给定公式 ψ 可满足，找到它的一个赋值

定理 (定理 13)

如果存在 *3-SAT* 的多项式时间算法，则存在找到其赋值的多项式时间算法。

- 定义一个 $\mathcal{NP}$ 语言

$$L = \{(\phi, r) : \exists w \text{ s.t. } \phi(w) = 1 \wedge w < r\}$$

- 给定 L 的黑盒访问（最多 $\log |W|$ 次），可以使用二分搜索找到 w。

结论：判定 3-SAT $\approx_{\text{hard}}$ 搜索 3-SAT

密码学问题的情况

应用 Cook-Levin 归约，所有 $\mathcal{NP}$ 语言都享有相同的性质。

密码学问题的情况

应用 Cook-Levin 归约，所有 $\mathcal{NP}$ 语言都享有相同的性质。
问：密码学问题呢？对于一个搜索问题（例如因子分解），我们能把它转化为判定问题吗？

密码学问题的情况

应用 Cook-Levin 归约，所有 $\mathcal{NP}$ 语言都享有相同的性质。
问：密码学问题呢？对于一个搜索问题（例如因子分解），我们能把它转化为判定问题吗？因子分解呢？（找到 N 的唯一因子分解）

密码学问题的情况

应用 Cook-Levin 归约，所有 $\mathcal{NP}$ 语言都享有相同的性质。
问：密码学问题呢？对于一个搜索问题（例如因子分解），我们能把它转化为判定问题吗？因子分解呢？（找到 N 的唯一因子分解）

- 为了考虑因子分解的计算复杂性，我们需要制作一个语言版本。
- 定义 Factor $= \{(N, r) : \exists s \text{ s.t. } 1 < s < r \wedge s | N\}$。$(m, r) \in$ FACTOR 当且仅当 r 大于 m 的最小素因子。

密码学问题的情况

应用 Cook-Levin 归约，所有 $\mathcal{NP}$ 语言都享有相同的性质。
问：密码学问题呢？对于一个搜索问题（例如因子分解），我们能把它转化为判定问题吗？因子分解呢？（找到 N 的唯一因子分解）

- 为了考虑因子分解的计算复杂性，我们需要制作一个语言版本。
- 定义 Factor $= \{(N, r) : \exists s \text{ s.t. } 1 < s < r \wedge s | N\}$。$(m, r) \in$ FACTOR 当且仅当 r 大于 m 的最小素因子。
- 给定 FACTOR 的黑盒，可以使用二分搜索找到 m 的一个素因子。

目录

由密码学问题定义的语言

基于 DDH 的例子，关于 (G, p, g_1, g_2)

- $X = G \times G$
- $L_{g_1,g_2} = \{(g_1^w, g_2^w)\}$，其中 $w \in \mathbb{Z}_p$

由密码学问题定义的语言

基于 DDH 的例子，关于 (G, p, g_1, g_2)

- $X = G \times G$
- $L_{g_1,g_2} = \{(g_1^w, g_2^w)\}$，其中 $w \in \mathbb{Z}_p$

基于 PRG 的例子：$\{0,1\}^n \to \{0,1\}^{2n}$

- $X = \{0,1\}^{2n}$
- $L = \{\mathsf{PRG}(s) : s \in \{0,1\}^n\}$

由密码学问题定义的语言

基于 DDH 的例子，关于 (G, p, g_1, g_2)

- $X = G \times G$
- $L_{g_1,g_2} = \{(g_1^w, g_2^w)\}$，其中 $w \in \mathbb{Z}_p$

基于 PRG 的例子：$\{0,1\}^n \to \{0,1\}^{2n}$

- $X = \{0,1\}^{2n}$
- $L = \{\mathsf{PRG}(s) : s \in \{0,1\}^n\}$

它们都在 $\mathcal{NP}$ 中，但既不在 P（甚至 BPP）中也不在 $\mathcal{NPC}$ 中。

由密码学问题定义的语言

基于 DDH 的例子，关于 (G, p, g_1, g_2)

- $X = G \times G$
- $L_{g_1,g_2} = \{(g_1^w, g_2^w)\}$，其中 $w \in \mathbb{Z}_p$

基于 PRG 的例子：$\{0,1\}^n \to \{0,1\}^{2n}$

- $X = \{0,1\}^{2n}$
- $L = \{\mathsf{PRG}(s) : s \in \{0,1\}^n\}$

它们都在 $\mathcal{NP}$ 中，但既不在 P（甚至 BPP）中也不在 $\mathcal{NPC}$ 中。
官方答案：位于 $\mathcal{NP}$-中间 $= \mathcal{NP} \setminus (\mathsf{P} \cup \mathcal{NPC})$

由密码学问题定义的语言

基于 DDH 的例子，关于 (G, p, g_1, g_2)

- $X = G \times G$
- $L_{g_1,g_2} = \{(g_1^w, g_2^w)\}$，其中 $w \in \mathbb{Z}_p$

基于 PRG 的例子：$\{0,1\}^n \to \{0,1\}^{2n}$

- $X = \{0,1\}^{2n}$
- $L = \{\mathsf{PRG}(s) : s \in \{0,1\}^n\}$

它们都在 $\mathcal{NP}$ 中，但既不在 P（甚至 BPP）中也不在 $\mathcal{NPC}$ 中。
官方答案：位于 $\mathcal{NP}$-中间 $= \mathcal{NP} \setminus (\mathsf{P} \cup \mathcal{NPC})$

备注

密码学问题假设平均情况复杂性（对于 $x \xleftarrow{R} X$），而计算机科学中的问题考虑最坏情况复杂性。

密码学需要什么样的问题?

$\mathcal{NP}$-困难性对密码学是不够的

密码学需要什么样的问题?

$\mathcal{NP}$-困难性对密码学是不够的

- 密码学假设 $\mathcal{NP}$ 中存在平均情况难解的问题。

密码学需要什么样的问题?

$\mathcal{NP}$-困难性对密码学是不够的

- 密码学假设 $\mathcal{NP}$ 中存在平均情况难解的问题。
- $\mathcal{NP}$-困难性只给我们关于极限复杂性的信息。

密码学需要什么样的问题?

$\mathcal{NP}$-困难性对密码学是不够的

- 密码学假设 $\mathcal{NP}$ 中存在平均情况难解的问题。
- $\mathcal{NP}$-困难性只给我们关于极限复杂性的信息。
- $\mathcal{NP}$-困难性考虑最坏情况时间。即使 $\mathcal{NPC}$ 问题在最坏情况下是困难的（$\mathrm{P} \neq \mathcal{NP}$），它们在平均情况下仍然可能是可高效求解的。

密码学需要什么样的问题?

$\mathcal{NP}$-困难性对密码学是不够的

- 密码学假设 $\mathcal{NP}$ 中存在平均情况难解的问题。
- $\mathcal{NP}$-困难性只给我们关于极限复杂性的信息。
- $\mathcal{NP}$-困难性考虑最坏情况时间。即使 $\mathcal{NPC}$ 问题在最坏情况下是困难的（$\mathrm{P} \neq \mathcal{NP}$），它们在平均情况下仍然可能是可高效求解的。

备注

使用 $\mathrm{P} \neq \mathcal{NP}$ 假设证明 $\mathcal{NP}$ 中存在平均情况困难问题是一个重大开放问题。

密码学期望什么?

最坏情况 = 平均情况

- 最坏情况假设比平均情况假设更弱

密码学期望什么?

最坏情况 = 平均情况

- 最坏情况假设比平均情况假设更弱

抵抗量子攻击

- 一些实际感兴趣的问题（例如，因子分解，离散对数）已知在 BQP 中，但被怀疑在 P 之外。

密码学期望什么？

最坏情况 = 平均情况

- 最坏情况假设比平均情况假设更弱

抵抗量子攻击

- 一些实际感兴趣的问题（例如，因子分解，离散对数）已知在 BQP 中，但被怀疑在 P 之外。

领先候选者：格问题（两者兼具）

参考文献


Manindra Agrawal, Neeraj Kayal, and Nitin Saxena.
Primes is in P.
Annals of Mathematics, 160(2):781–793, 2004.