14 - NP Completeness

归约

多项式时间归约（Polynomial Time Reduction）

- 给定一个问题 A，我们希望解决它
- 我们可以将问题 A 归约为解决另一个问题 B（利用 B 来解决 A）
- 假设我们有一个算法 ALG^B 可以解决问题 B，并且其成本为 $O(1)$
- 我们可以构造一个算法 ALG^A 来解决问题 A，该算法使用 ALG^B 作为 subroutine
- 如果 ALG^A 是一个多项式时间算法，那么这就是一个多项式时间归约

值得注意的是，归约并不等于等价，A 归约到 B 意味着我们可以利用 B 的解法来解决 A

多项式时间归约（Polynomial Time Reduction）的表示法：

$$A \leq^P B$$

这表示 A 可以在多项式时间内归约到问题 B

如何使用归约（reductions）来处理问题

- Positive（正面）：假设我想解决问题 A，并且我知道如何在多项式时间内解决问题 B
  - 我可以尝试设计一个多项式时间归约 $A \leq^P B$，这将给我们一个在多项式时间内解决问题 A 的算法
- Contrapositive（反面）：假设存在一个问题 A，对于它不太可能存在一个多项式时间算法来解决它
  - 如果我设计出一个多项式时间归约 $A \leq^P B$，那么也不太可能存在一个多项式时间算法来解决问题 B
  - B 至少和 A 一样难以解决，因为如果我能解决B，那么我们也能解决 A

图灵归约

$$A \leq_T B$$

图灵归约是指通过（多项式次）调用一个解决问题 B 的算法（称为oracle）来解决问题 A 的归约方法

Many-One 归约

$$A \leq_m B$$

直接将问题 A 的每个实例转换成一个等价的 B 的问题实例，这样如果能解决 B，就能解决 A

- 如果 z 是问题 A 的实例 I 的解，那么 z' 是问题 B 的实例 $f(I)$ 的解
- 如果不是，那么不是
- 等价地，如果 z' 是问题 B 的实例 $f(I)$ 的解，那么 z 是问题 A 的实例 I 的解

例子 Bipartite Matching $\leq_m$ Maximum Flow

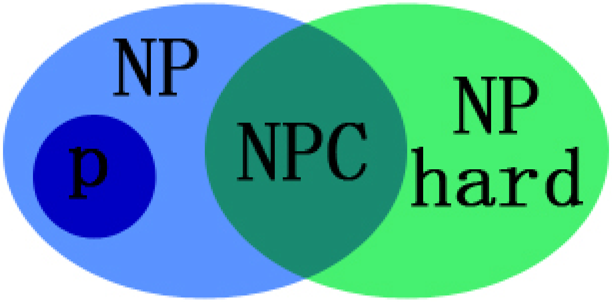
- 从匹配到流
  - 假设在 G 上有一个值为 k 的流 f，那么在 G^f 上有一个大小为 k 的匹配 M

- 从流到匹配
  - 假设在 G^f 上有一个值为 k 的流 f，那么在 G 上有一个大小为 k 的匹配 M

Technically Speaking:

- 问题 A：是否存在一个大小至少为 k 的二分匹配?
- 问题 B：是否存在一个值至少为 k 的流?
- 最大二分图匹配和最大流都是优化问题（optimisation problems），即寻找在给定条件下最优解的问题
- 归约使用对应的决策问题。决策问题是判断某个条件是否成立的问题，而不是求最优解

P、NP、NP-C、NP-Hard



P

类 P 是计算复杂性理论中的一个基本类，包含所有存在已知多项式时间算法的问题

NP

表示：非确定性多项式时间（non-deterministic polynomial time）
即，可以在多项式时间内由非确定性图灵机（non-deterministic Turing machine）解决的问题

- 如果给定一个解，可以在多项式时间内验证它是否确实是该问题的解
- 这些问题是“有效可验证的”（efficiently verifiable）

不需要在多项式时间内解决一个问题，只需要在多项式时间内验证一个答案

子集和问题

问题是：

问题描述:

- **给定**：我们有一组包含 n 个物品的集合 $\{1, 2, \ldots, n\}$。
- **权重**：每个物品 i 有一个非负整数权重 w_i。
- **限制**：我们给定一个整数上限 W。

目标:

- **选择一个物品的子集** S，使得所有被选物品的总重量 $\sum_{i \in S} w_i \leq W$，并且 $\sum_{i \in S} w_i$ 被最大化。

其等价的决策版本是：

问题描述：

- **给定**：我们有一组包含 n 个物品的集合 $\{1, 2, \ldots, n\}$。
- **权重**：每个物品 i 有一个非负整数权重 w_i。
- **限制**：我们给定一个整数上限 W。

目标：

- **决策目标**：决定是否存在一个物品的子集 S，使得这些物品的总重量 $\sum_{i \in S} w_i = W$。

子集和问题是 NP 问题

NP-Hard

一个问题 B 是 NP-Hard 的，当且仅当对于 NP 问题中的每一个问题 A，都有 $A \leq^P B$。即，NP 中的每一个问题都可以在多项式时间内归约到 B （如果 NP 中的每一个问题都可以多项式时间归约到 B，这就说明 B 至少和 NP 中最难的问题一样难）

即

- 问题 A_i 是一个 NP 问题
- 所有的问题 A_i 都可以归约到问题 B

证明 NP-Hardness 的挑战：
看起来我们必须从NP中的每一个问题 A 构造出一个归约到 B，这似乎是非常不实际的

NP-Complete

如果一个问题 A 是 NP-C 的，那么 NP 中的每一个问题都可以有效地归约到 A
因为 NP-C 问题既在 NP 类中（解可以在多项式时间内验证），又是 NP-hard 的（NP中的每个问题都可以在多项式时间内归约到它）

证明问题 B 的 NP-H 性：
看起来我们需要从NP中的每一个问题 A_i 构造出一个到 B 的归约，但实际上，

- 只需要从一个已知的 NP-C 问题 A 构造归约到 B
- 任何其他问题 A_i 到 B 的归约都可以通过 A 来实现（即通过已知的 A 到 B 的归约）

3-SAT

这是一种布尔可满足性问题（Boolean Satisfiability Problem, SAT）的特例

3-SAT 问题射击一个合取范式（CNF）公式，公式包含 m 个子句（clauses）和 k 个文字（literals），其公式形式为

$$\varphi = (x_1 \vee \neg x_5 \vee x_3) \wedge (x_2 \vee x_6 \neg x_5) \wedge \cdots \wedge (x_3 \vee x_8 \vee x_{12})$$

每个 clause 都包含三个 literal

真值赋予（Truth Assignment）：对每个变量 x_i 赋予一个 $\{0, 1\}$ 中的值
可满足赋值（Satisfying Assignment）：一种真值赋值，使公式 φ 的值为 1（真）
3-SAT 的计算问题（Computational Problem 3-SAT）：判断输入公式 φ 是否存在一个可满足赋值

3-SAT 问题是 NP-C 的

这说明：

- 3-SAT 问题是 NP 的
- 3-SAT 问题是 NP-Hard 的

同时：

- 第一个被证明为 NP-C 的问题是 SAT 问题，而它可以规约为 3-SAT 问题
- 许多教科书从电路 SAT 问题开始，这是定义在布尔电路上的 SAT 问题，包含与（AND）、或（OR）、非（NOT）等布尔门

证明 NP-C

假设我们有一个问题 A，并且我们想证明它是 NP-C 的

- 首先，证明 A 在 NP 中
  - 给定一个解，可以在多项式时间内验证该解是否正确
- 然后，证明 A 是 NP-Hard 的
  - 从某个已知的 NP-C 问题 P 构造一个多项式时间归约

但是实际上：

- 首先，证明 A 在 NP 中
  - 给定一个解，可以在多项式时间内验证该解是否正确
- 然后，证明 A 是 NP-Hard 的
  - 从某个已知的 NP-Hard 问题 P 构造一个多项式时间归约