NP complete problems

Optimization and Decision

很多经典的难问题都是优化问题，而优化问题往往可以转换成对应的判断问题。

在这里以最大团问题 CLIQUE 为例。团是图的一个完全子图，团的大小为其中顶点的个数

Optimization

优化问题往往是关注某种特殊的结构（e.g.，团，路径，着色……），并希望优化该结构的某种指标（e.g.，团的大小，路径的长度，着色种类）

CLIQUE 问题的 Optimization 版本是：

给定一个无向图 G，求其中最大的团的大小

Decision

对于一个优化问题，往往可定义其对应的判定问题。判定问题关注同样的结构，同样的指标，但是不再关注指标的最大/最小值，而是引入一个参数 k，并问一个 yes/no 问题：是否存在一个结构，其指标与 k 满足某种关系

CLIQUE 问题的 Decision 版本是：

给定一个无向图 G，参数 k，判断图中是否存在大小为 k 的团

判定 or 优化?

一般直觉上认为优化问题比判定问题难，知道优化问题的解可以容易得出判定问题的解，然而知道判定问题的解却不一定能简单得出优化问题的解。

事实上，可以证明优化问题和判定问题难度相差不大，很多情况一个优化问题在多项式时间可解 $\iff$ 其对应判定问题在多项式时间可解

> CLIQUE 优化在多项式时间可解 $\iff$ CLIQUE 判定在多项式时间可解

Typical Decision Problems

图着色：能否将无向图 G K-着色

任务调度：能否调度任务使得超过 DDL 的惩罚不超过 k

背包问题：利润至少为 k

CNF-SAT：对一个 CNF 是否有一个 assignment 使其为真

Hamilton 回路/通路：给定无向图是否存在 Hamilton 通路/回路

旅行商问题

NP-Completeness

目标：根据问题的难度划分档次

这里 P 与 NP 问题默认讨论的是判定问题

Class P

P 即为 polynomially bounded problem

> **定义 15.4** P 问题
>
> 一个问题是 P 问题，如果存在一个关于 n 的多项式 $ploy(n)$，且存在一个解决该问题的算法，满足算法的代价 $f(n)=O(ploy(n))$

P 问题可以内聚为一个问题类，源于多项式运算的封闭性。多项式的加法和乘法和嵌套均是封闭的

P 问题是个很大的范围，一个问题是 P 问题并不代表其有高效的解，然而一个问题不是 P 问题却可以肯定其解一定不高效甚至无法在可接受的时间内求解（e.g.，指数级，阶乘级）

P 问题的属性与分析其所用的计算模型没有关系

Class NP

在定义 Class NP 之前需要先引入 Nondeterministic Algorithm

一个 Nondeterministic Algorithm 解决问题的过程是

1. 随机猜测一个问题的“解”
2. 验证这个解是否满足问题（yes/no）

定义 nondeterministic algorithm 算法的答案为 yes $\iff$ 存在某次运行的输出为 yes，同理，答案为 no 表示对于所有猜测的解，验证的结果均为 no

NP 即为 Nondeterministic P

> **定义 16.1** NP 问题
>
> 定义一个问题是 NP 问题，如果该问题的解在多项式时间内可验证，这里的可验证是指首先非确定地猜测一个问题的解，然后可以在多项式时间内检查这个解是不是问题的解。
>
> 严格的 NP 问题的定义基于非确定性图灵机

NP 问题即指存在一个 polynomially bounded nondeterministic algorithm 的问题

从 P 问题到 NP 问题，算法从 deterministic 到 nondeterministic，关注的点从“解决问题”到“验证问题的解”

NP 问题相比 P 问题更难，但也不是非常难，**至少还可以在多项式的时间内验证其解是否正确**

可证明图着色，CLIQUE 等均是 NP 问题

Relation between P and NP

显然 P $\subseteq$ NP，一个问题的 deterministic algorithm 是其 nondeterministic algorithm 的特例（不根据输入和猜测的解判定 yes/no ，而是直接根据输入判定 yes/no）

直觉上认为 P 是 NP 的真子集，且 P 远小于 NP，但至今也没有证明，也无法证明其不成立

Reduction

引入归约（Reduction）比较不同问题间的难易程度

> **定义 15.3** 问题 P 到 Q 的归约 $P \leqslant Q$
>
> 问题 P 到问题 Q 的归约是一个转换函数 $T(x)$ ，满足
>
> - 能将问题 P 的任意合法输入 x 转换为问题 Q 的合法输入 $T(x)$
> - P 问题对于任意输入 x 的输出为 YES $\iff$ Q 问题对输入 $T(x)$ 的输出为 YES

判定问题的便利：只用转换输入，不用转换输出

当 $T(x)$ 的代价为其输入规模的多项式时，则称归约为**多项式归约**，记为 $P \leqslant_p Q$

引入多项式归约是为了不让归约的代价干扰对于解决问题的代价的衡量，在解决难问题时，多项式规模的归约是一个小量。

多项式归约有以下两个性质

- 如果 $P \leqslant_p Q$ 并且 Q 是 P 问题，那么 P 也是 P 问题（多项式运算的封闭性）
- 归约的传递性：如果 $P \leqslant_p Q, Q \leqslant_p R$ ，则 $P \leqslant_p R$

归约反应了问题之间的难易程度，若 $P \leqslant_p Q$，说明 Q **至少**和 P 一样难，因为如果得到解决 Q 的算法，基于归约可以得到解决 P 的算法，但知道解决 P 的算法却对解决 Q 没有帮助。

> P 把皮球踢给了 Q

NP-hard and NP-complete

为了定义 NP-complete，首先引入 NP-hard 的概念

> **定义 16.2** NP-hard 问题
>
> 一个问题 P 是 NP-hard，如果 $\forall Q \in \text{NP}$，$Q \leqslant_p P$

即一个 NP-hard 问题至少与所有 NP 问题一样难，NP-hard 是没有上界的

> **定义 16.3** NP-complete 问题
>
> 一个问题 P 是 NP-complete，如果 $P \in \text{NP}$，且 P 是 NP-hard

NP-complete 即为 NP 问题中最难的问题。停机问题就是个著名的 NP-complete 问题

如果能证明任意一个 NP-complete 在多项式时间内可解，根据归约可得到所有 NP 问题在多项式时间内可解，即证明了 NP = P

Procedure for NP-Completeness

如何证明问题 Q 是 NPC?

- 已知 P 是 NPC
- 将 P 多项式归约到 Q
- 根据多项式归约的传递性，$\forall X \in \text{NP}$，$X \leqslant_p Q$，Q 为 NP-hard
- 证明 $Q \in \text{NP}$ 即可

需要一个最初的 NPC 问题 P 作为种子。第一个被证明 NPC 的问题是 SAT 问题

可以证明 CLIQUE 为 NPC，将 3-SAT 归约到 CLIQUE