NP完全性问题

2025年4月17日 星期四 20:46

定义 13.1.1（P 类问题（polynominal）） 在多项式时间内可以解决的问题。

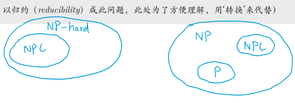
P 类问题通常被认为是比较容易解决的问题，它的时间复杂度通常为多项式时间，如 $O(n)$，$O(n\log n)$，$O(n^3)$ 等等。前面学习的搜索、排序，小数背包问题等都是 P 类问题。

定义 13.1.2（NP 类问题（Nondeterministic Polynominal）） 给定一个证书（*certificate*），可以在多项式时间内验证此证书是否是问题的一个解的问题。

P ⊆ NP

定义 13.1.3（NPC 问题（NP 完全问题）） 是 *NP* 类问题中最难的问题，包含两个条件：

- 是一个 *NP* 问题
- 所有的 *NP* 问题都可以'转换'成此问题（确切的定义是：所有的 *NP* 问题都可以归约（*reducibility*）成此问题，此处为了方便理解，用'转换'来代替）



基本概念：归约

定义 13.1.4（归约） 一个问题（Q_1）可以规约为另外一个问题（Q_2），需满足以下两个条件：

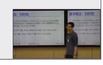
1. 实例对应性：Q_1 的任意一个实例 ϕ，通过函数 f 都可转化成 Q_2 的一个实例 $f(\phi)$，且这个转化函数 f 必须为多项式时间；
2. 输出一致性：规约后的输出和原来的输出一致，即，如 $algo_1$ 是 Q_1 的算法，$algo_2$ 是 Q_2 的算法，则在相同的输入下，$algo_1(Q_1) = algo_2(Q_2)$。

归约具有传递性

基本概念：归约性

如果问题 Q_1 可以归约为问题 Q_2，则记为 $Q_1 \leq_P Q_2$。表示 Q_1 大于 Q_2 的难度不会超过一个多项式时间因子（但通常我们可以根据小于等于号来通俗的认为 Q_1 的难度要小于等于 Q_2）。$Q_1 \leq_P Q_2$，可以推出：

- 如果 Q_1 是 NPC 问题，则 Q_2 必然是 NPC 问题（因 Q_2 不比 Q_1 容易）；
- 如果 Q_2 是 P 问题，则 Q_1 必然是 P 问题（因 Q_1 不比 Q_2 难）；



❑2 合取范式(CNF)的可满足性问题(SAT)

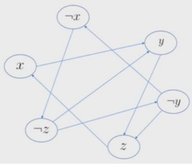
定义 13.2.1（2CNF-SAT） *2CNF* 是一个布尔表达式，其由子句（*clause*）的'与'组成，而每个子句都由 2 个文字（*literal*，文字为一个变量或者变量的'否'）的'或'组成。如 $(\neg x \vee y) \wedge (\neg y \vee z) \wedge (x \vee \neg z) \wedge (z \vee y)$，其中共有 x，y，z 三个变量，这些变量及其'否'为文字，两个文字构成了一个子句。*2CNF* 的可满足性问题是指是否存在一个赋值，使得 *2CNF* 为真。

❑2 合取范式(CNF)到图的转换

设 2CNF-SAT 有 n 个变量，m 个子句。构造图 $G = (V, E)$。图中共 $2n$ 个节点，代表 n 个变量及每个变量的'否'。对于 2CNF 中的每个子句，如 $(x \vee y)$，则图中同时存在从节点 $\neg x$ 到节点 y 的有向边，和从 $\neg y$ 到 x 的有向边。

❑2 合取范式(CNF)到图的转换

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



❑2合取范式(CNF)到图的转换

- 当且仅当 2CNF 中存在子句 (x ∨ y)，图 G 中存在边 ¬x → y (和 边 ¬y → x)

1. 如图 G 中存在边 $x \to y$，则必然会同时存在边 $\neg y \to \neg x$。
2. 如图 G 中包含一条从 x 到 y 的路径，则必然也包含一条从 $\neg y$ 到 $\neg x$ 的路径。

定理 22 *2CNF* 是不可满足的，当且仅当存在变量 x，使得在图 G 中同时存在

- 一条从 x 到 $\neg x$ 的路径，
- 一条从 $\neg x$ 到 x 的路径。

Complexity Class – P class

- **P**: the set of decision problems for which there is a polynomial-time algorithm to **solves** it.
- Here, we say that an algorithm A **solves** problem X if, for all instance I of X, $A(I) =$ YES iff I is a YES instance.
- For example, TOPOLOGICAL SORT problem: $O(m+n)$.

Complexity Class – NP class

- **NP**: the set of decision problems for which the problem instances, where the answer is YES, have *solutions* verifiable in polynomial time.

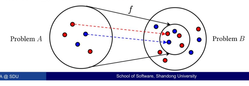
Complexity Class – NP class

- **NP**: the set of decision problems for which the problem instances, where the answer is YES, have *solutions* verifiable in polynomial time.
- An equivalent definition of **NP**: the set of decision problems *solvable* in polynomial time by a non-deterministic Turing machine. This definition is the basis for the abbreviation NP (Nondeterministic, Polynomial time).
- The above two definitions are equivalent because the algorithm based on the Turing machine consists of two phases, the first of which consists of a guess about the solution, which is generated in a non-deterministic way, while the second phase consists of a deterministic algorithm that verifies if the guess is a solution to the problem.

Reduction: Uncovering the connection between two problems

MMA

- POLYNOMIAL-TIME REDUCTION: a procedure f to transform **any** instance α of problem A to some instance $\beta = f(\alpha)$ of problem B with the following characteristics:
  1. **Transformation**: The transformation takes polynomial time;
  2. **Equivalence**: The answers are the same, i.e. the answer for α is YES iff the answer to $\beta = f(\alpha)$ is also YES.
- Denoted as $A \leq_P B$, read as "**A is reducible to B**".



Complexity Class – NP class – 3SAT problem

MMA

3SAT:

INPUT: Given a CNF (Conjunctive Normal Form) $\Phi = C_1 \wedge C_2 ... \wedge C_k$, where each clause consists of exactly tree literals.

OUTPUT: Is there an assignment of all x_i such that all clauses C_j are satisfied?

- If your answer is YES, you just need to provide a **solution**, i.e. an assignment for all n boolean variables.
- **Verifier:** checking whether each clause is satisfied by this assignment.
- It takes poly-time to verify the solution. Thus 3SAT is in **NP class**.
- Example:
  - An instance: $(x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$.
  - Solution: $x_1 =$ TRUE, $x_2 =$ TRUE, $x_3 =$ FALSE.

Complexity Class （复杂类）

- *Complexity classes* are concerned with the rate of growth of the requirement in resources as the input size n increases. The resource in question can either be time, essentially the number of primitive operations on an abstract machine, or (storage) space.
- It is an abstract measurement, and does not give time or space in requirements in terms of seconds or bytes.
- Decision problems are assigned complexity classes based on the fastest known algorithms. Therefore, decision problems may change classes if faster algorithms are discovered.

Reduction: CLIQUE $\leq_P$ VERTEX COVER

MMA

Theorem

CLIQUE $\leq_P$ VERTEX COVER.

Proof

- Given an instance $\langle G = (V, E), k\rangle$ of CLIQUE, we construct a VERTEX COVER instance $\langle \bar{G} = (V, \bar{E}), k'\rangle$. In the following, we show that G has a clique of size k iff $\bar{G}$ has a vertex cover of size $k' = |V| - k$.

***Complement*(补图) of G:**

$\bar{G} = (V, \bar{E})$, where $\bar{E} = \{(u, v) : u, v \in V, u \neq v, \text{ and } (u, v) \notin E\}$

证明 $Y \in NPC$：

1. show that $Y \in NP$
2. choose an NP-complete problem X
3. Prove that $X \leq_P Y$

Prove that set cover is NP-complete

MMA

SET COVER

INPUT: A set $U = \{e_1, e_2, \ldots, e_n\}$ of n elements, a collection of sets $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$, $S_i \subseteq U$ for all i, and an integer k.

OUTPUT: Does there exist a set $\mathcal{C} \subseteq \mathcal{S}$, such that $\bigcup_{S_i \in \mathcal{C}} = U$ and $|\mathcal{C}| \leq k$?

- To prove that SET COVER $\in$ NP-complete:
  - **Step 1.** Show that SET COVER $\in$ NP.
  - **Step 2.** Choose an NP-complete problem, VERTEX COVER.
  - **Step 3.** Prove that VERTEX COVER $\leq_P$ SET COVER.

Theorem

SET COVER is NP-complete.

- **证明一个问题属于NP完全问题的一般过程：**
  1. **证明该问题属于NP**：需要说明对于该问题的任意一个可能的解，能够设计一个算法在多项式时间内验证这个解的正确性。通常就是详细描述如何根据给定的解去检查问题的约束条件等是否满足，以表明可以在多项式时间内完成验证过程。
  2. **选择一个已知的NP完全问题进行归约**：常见的已知NP完全问题如SAT问题、3-SAT问题、团问题（Clique Problem）、顶点覆盖问题（Vertex Cover Problem）等。根据要证明的问题的特点选择合适的已知NP完全问题作为归约的源问题。
  3. **构造归约函数并证明其正确性和多项式时间性质**：
     - **构造归约函数**：设计一个从已知NP完全问题的实例到要证明问题的实例的转换函数（映射），明确如何将已知问题的输入数据按照一定规则转化为要证明问题的输入数据。
     - **证明正确性**：需要论证通过这个归约函数转换后，原已知NP完全问题实例的解和转化后的要证明问题实例的解存在一一对应关系，即原问题有解当且仅当转化后的问题有解。
     - **证明多项式时间性质**：说明构造的归约函数在执行转换过程中，其时间复杂度是关于原问题实例规模的多项式函数，也就是转换操作不会耗费过高的时间，保证整个归约可以在多项式时间内完成。