上界：算法课
下界：比乎一元所知

5 Complexity Theory

We study the computational resources needed to solve a problem, such as time, space, randomness, parallelism. We study the relationship among problems, whether problem X is harder (not easier than) problem Y.

P: poly-time solvable

NP: poly-time verifiable (given a witness)

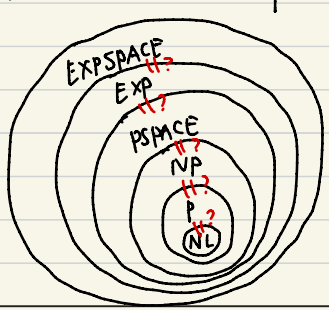
PSPACE: poly-space solvable

NC: polylog-time solvable using poly. number of processors.

BPP: poly-time solvable by randomized algorithms (with bounded error)

Main open problems:

1) $P \stackrel{?}{=} NP$
2) $P \stackrel{?}{=} NC$
3) $P \stackrel{?}{=} BPP$
4) Tradeoff between time and space



PCEXP 对角线法

NL⊂PSPACE⊂EXPSPACE

↳对数空间

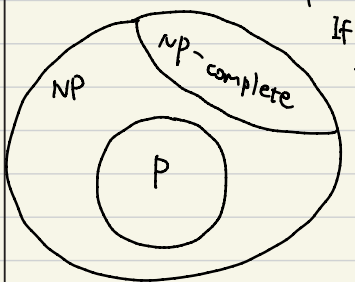
对时间/空间，分别可说⊂

we don't know!

Can we multiply two $n \times n$ matrices using $10000 \cdot n^2$ operations?

Can we factor an n-bit integer using $10000 \cdot n$ operations?

Can we solve Hamilton path in $O(n)$ time?



If a single complete problem is solvable, then the whole complexity class is solvable.

5.1 Reductions

Roughly speaking, a reduction is an algorithm for transforming one problem into another.

Denote by $A \leq B$ if A is reducible to B. // A is easier than B

Examples of reductions

1. Many-one reduction

$$L_1 \leq_m L_2$$

2. Turing reduction $L_1 \leq_T L_2$ 多次调用

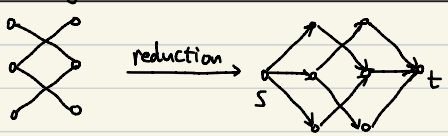
3. Poly-time reduction (Karp reduction) $L_1 \leq_p L_2$.

4. Poly-time Turing reduction (Cook reduction) $L_1 \leq_T^p L_2$.

Example:

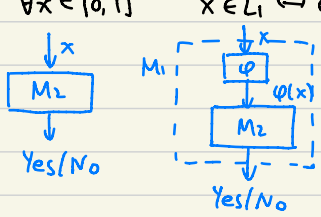
1) Turing reduction $a \times b = \dfrac{(a+b)^2 - a^2 - b^2}{2}$ 调用了3次平方

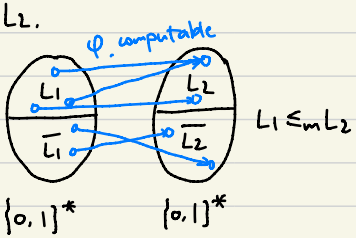
2) many-one reduction



Def 5.1 (Many-one reduction) Let $L_1, L_2 \subseteq \{0,1\}^*$. Say L_1 is <u>many-one reducible</u> to L_2, denoted $L_1 \leq_m L_2$, if there is a computable $\varphi : \{0,1\}^* \to \{0,1\}^*$ such that

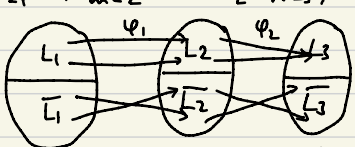
$\forall x \in \{0,1\}^* \quad x \in L_1 \Leftrightarrow \varphi(x) \in L_2$.





Prop 5.2

1. For every language $L \subseteq \{0,1\}^*$, $L \leq_m L$.
2. If $L_1 \leq_m L_2$, then $\overline{L_1} \leq_m \overline{L_2}$
3. If $L_1 \leq_m L_2$ and $L_2 \leq_m L_3$, then $L_1 \leq_m L_3$.



$\varphi_1, \varphi_2 : \{0,1\}^* \to \{0,1\}^*$

$\varphi_2(\varphi_1(x))$

4. Let $L_1 \leq_m L_2$. If L_2 is decidable, then L_1 is decidable. Conversely, if L_1 is undecidable, then L_2 is undecidable.

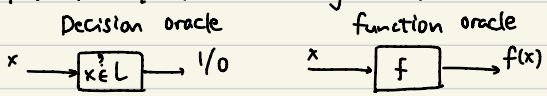
Def 5.3 (Karp reduction) A poly-time many-one reduction, also called Karp reduction, is a many-one reduction where φ is poly-time computable.

Def 5.4 (Oracle TM) A TM with an oracle for language L is a TM M^L with an additional oracle tape and two extra

states: query state q_{query} and response state $q_{response}$.
The machine operates like a standard TM, except that when it enters q_{query}, it performs the following in a single step:

1. The string z that is written on the oracle tape is erased.
2. If $z \in L$, symbol 1 is written on the leftmost of the oracle tape; If $z \notin L$, 0 is written.
3. The oracle tape head is moved to the leftmost cell.
4. The machine transitions to $q_{response}$.

Similarly, one can define a TM equipped with an oracle for a function $f: \{0,1\}^* \to \{0,1\}^*$ denoted by M^f.



Def 5.5 Let $L_1, L_2 \subseteq \{0,1\}^*$. L_1 is Turing reducible to L_2, denoted by $L_1 \leq_T L_2$, if there is an oracle TM, with an oracle access to L_2, that decides L_1.

Prop 5.6 1) For any $L \subseteq \{0,1\}^*$, $L \leq_T \bar{L}$.
2) For any two decidable languages L_1, L_2, $L_1 \leq_T L_2$.
3) If $L_1 \leq_T L_2$ and $L_2 \leq_T L_3$, then $L_1 \leq_T L_3$.
4) If $L_1 \leq_m L_2$, then $L_1 \leq_T L_2$.

Def 5.7 (Cook reduction) Let $L_1, L_2 \in \{0,1\}^*$. L_1 is cook reducible to L_2 if there is a poly-time oracle TM M^{L_2} that decides L_1.

5.2 Complexity class P

Def 5.8 Let $T: \mathbb{N} \to \mathbb{N}$. $L \subseteq \{0,1\}^*$ is in $DTIME(T(n))$ iff. there exists a multitape TM that runs in time $O(T(n))$ and decides L.

Def 5.9 $P = DTIME(n) \cup DTIME(n^2) \cup \cdots = DTIME(n^{O(1)})$.

Closure property of polynomials. Let $f(n), g(n)$ be polynomials. Then,

1) $f(n)+g(n)$ is a polynomial
2) $f(n) \cdot g(n)$ is a polynomial
3) $f(g(n))$ is a polynomial

In complexity, efficient = poly-time

In cryptography, a scheme is secure if any poly-time adversary cannot break it.

BFS, DFS: $O(|V|+|E|) = O(n)$ n: input bit length

String matching: $O(n)$

Disjoint set $O(n\alpha(n)) = \tilde{O}(n)$ $\alpha(n)$: inverse Ackerman function

Shortest path Dijkstra $O((V+E)\log V) = \tilde{O}(n)$ 忽略一个 poly-log term

$O(E+V\log V)$ → Heap → Fibonacci heap

Minimum Spanning Tree:

Prim's algorithm $O((V+E)\log V)$

$O(E+V\log V) = \tilde{O}(n)$

Chazelle's algorithm: $O(n\alpha(n))$

Maximum flow

Edmonds Karp $O(VE^2) = O(n^{2.5})$ → 稠密图

Push-relabel $O(V^2\sqrt{E})$

CKLPGS 22 $O(E^{1+o(1)}) = \tilde{O}(n)$

linear programming

Simplex method $O(2^n)$ worst case

ellipsoid method $O(m^4 L)$ m: #vars L: # constraints

interior-point method $O(m^{3.5} L)$

Vaidya (1989) $O(m^{2.5})$

Cohen-Lee-Song 2019 randomized interior-point method

$\tilde{O}(m^\omega \cdot \log(\frac{1}{\delta}))$ $\delta \in (0,1)$, accuracy

ω: matrix mult exponent, $\omega \leq 2.3$

猜測 Conjecture $\omega = 2 + O(1)$

deterministic version (Brand 2019)

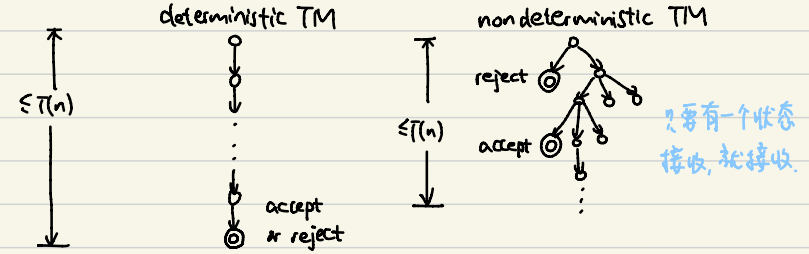
5.3 Complexity class NP

N nondeterministic

P polynomial time

At any point in computation, a nondeterministic TM may continue along several possible transitions, that is,

$\delta: Q \times \Gamma \longrightarrow \mathcal{P}(Q \times \Gamma \times \{L, R\})$.



Def 5.10 (Nondeterministic TM) A nondeterministic TM is a 7-tuple $N = (\Sigma, \Gamma, Q, \delta, q_0, q_{accept}, q_{reject})$, where

1. Q ··· 2. Σ ··· 3. Γ ···

4. $\delta: Q \times \Gamma \longrightarrow \mathcal{P}(Q \times \Gamma \times \{L, R\})$ is the transition function.

5. q_0 ··· 6. q_{accept} ··· 7. q_{reject} ···

The machine N accepts x if there is at least one computation path that reaches an accept state. We say that N runs in time T(n) if, for every input $x \in \{0,1\}^n$ and every sequence of nondeterministic choices, M halts in T(n) within T(n) steps.

Thm 5.11 Every nondeterministic TM has an equivalent deterministic TM, ignoring its runtime.

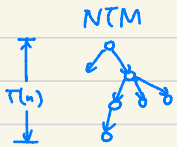
Proof (sketch)

Simulate using BFS

(don't use DFS in case of looping forever)

Only considering computability: DTM = NTM

but in terms of complexity . . . ✗



Def 5.12 (Binary-choice NTM) A binary-choice NTM is a 8-tuple $N = (Q, \Sigma, \Gamma, \delta_0, \delta_1, q_0, q_{accept}, q_{reject})$, where

$\delta_0, \delta_1 : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ ~~probabilistic~~

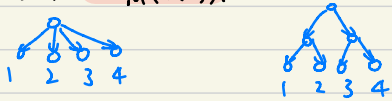
At each step, M non-deterministically chooses to apply either δ_0 or δ_1.

直觉很灵敏的人，每次都选对（不是概率地选）

Lem 5.13 Let $L \subseteq \{0,1\}^*$. If L can be decided by an NTM in time $T(n)$, then L can be decided by a binary-choice NTM in time $O_M(T(n))$.

2个选和多个选只是一个乘性常数

Proof.



At each step, NTM N has $\leq |Q| \times |\Gamma| \times 2$ possible transitions.

$\delta : Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{L, R\})$

This can be simulated in $\lceil \log_2(|Q| \times |\Gamma| \times 2) \rceil = O_M(1)$ steps by a binary-choice NTM.

Thus the total runtime is $O_M(T(n))$.

□

Def 5.14 Let $T : \mathbb{N} \to \mathbb{N}$. A language $L \subseteq \{0,1\}^*$ belongs to

$NTIME(T(n))$ if there is a multitape NTM that decides L in time $O(T(n))$.

$DTIME(T(n)) \subseteq NTIME(T(n))$ // 因为NTM更强大

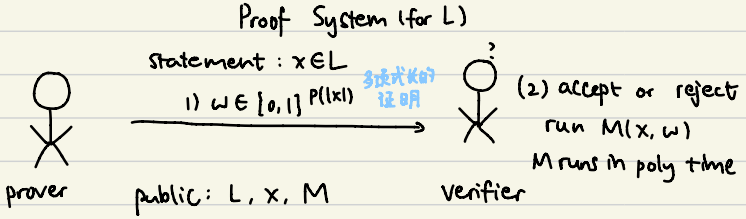
Def 5.15 $NP = \bigcup_{c \geq 1} NTIME(n^c) = NTIME(n^{O(1)})$

Two views of NP:

P: problems that can be solved by DTM in poly time.

NP: problems that can be solved by NTM in poly time

NP: problems that can be verified by DTM in poly time



Thm 5.16 Let $L \subseteq \{0,1\}^*$. L is in NP iff. there is a polytime TM M and polynomial $p(n)$ such that for any $x \in \{0,1\}^*$,

1) (complete) If $x \in L$, then there exists a certificate $w \in \{0,1\}^{p(|x|)}$ such that $M(x, w) = 1$.

2) (Sound) If $x \notin L$, then for all $w \in \{0,1\}^{p(|x|)}$, $M(x, w) = 0$

Probalistic checkable proof

PCP $[r(n), q(n)]$

只要读这些bit，就可以较大概率判断证明是否正确

(PCP Theorem 1998) $NP = PCP[O(\log n), O(1)]$

Verifiable computation: $y = f(x)$, π proof "$y = f(x)$"

zero-knowledge proof: $(\exists w)(y = f(x, w))$

Thm 5.16

Proof. For one direction, letting $L \in NP$, we prove there exists a poly time TM M and polynomial $p(n)$: satisfying these conditions. Let $L \in NTIME(n^{c_1})$, where $c_1 > 0$. So there is a binary-choice NTM N that decides L in time $c_2 \cdot n^{c_1}$. Let $p(n) = c_2 \cdot n^{c_1}$, and let $w \in \{0,1\}^{p(n)}$, where $w_i \in \{0,1\}$ indicates which transition function to apply at step i. The verifier determines whether N accepts x given w. (只要有一条路走下去即可)

需要一个DTM：每次告诉NTM 2选1时选哪个

For the other direction, suppose there is a TM M and polynomial p satisfying the above conditions, we construct an NTM N that decides L.

On input x, NTM N does the following:

1) non-deterministically guess $w \in \{0,1\}^{p(|x|)}$.

2) simulate M on input (x, w). Accept if M accepts.

N runs in poly time and decides L.

Thus, $L \in NP$

□

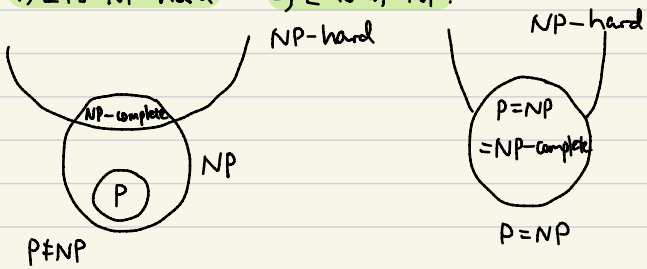
$\leq_p$：多项式归约

Def 5.17 (NP hard) Language $L \subseteq \{0,1\}^*$ is NP-hard if, for every $K \in NP$, we have $K \leq_p L$. ($K \leq_T^p L$) 相同吗？不知道

Open: If we replace $K \leq_p L$ by $K \leq_T^p L$, are they the same?

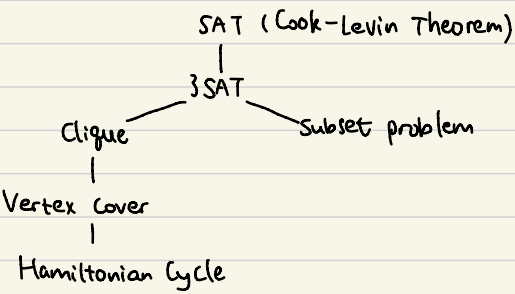
Def 5.18 (NP complete) L is NP-complete if:

1) L is NP-hard 2) L is in NP.



$(\forall L \in NP)$

$(L \leq_p SAT)$



P versus NP problem asks whether the two classes are the same.

If P=NP, all cryptography systems are insecure.

$c = Enc(m,k)$

Given m, c, solve k.

$(\exists k')(c = Enc(m, 0\|1\|k'))$

?

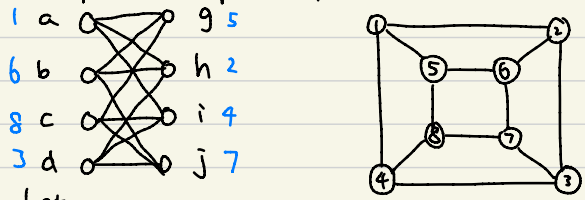
$k_{pub} = Gen(k_{pri})$

Given k_{pub}, compute k_{pri}.

$(\exists w)(k_{pub} = Gen(0||w))$. → 猜算1位是0.

Examples of NP problems:

1. Graph isomorphism problem 图同构



Let

$Graph-Iso = \{\langle G,H\rangle : G \text{ and } H \text{ are isomorphic}\}$

An isomorphism of graphs G and H is bijection $f: V(h) \to V(H)$ such that $(u,v) \in E(G)$ iff. $(f(u), f(v)) \in E(H)$

$Graph-Iso \in NP$

Let w be the bijection. Verifier checks if w is an isomorphism.

(Babai 2016?) $Graph-Iso \in DTIME(2^{((\log n)^{O(1)})})$

2. Clique problem

Let $Clique = \{\langle G,k\rangle : G \text{ contains } K_k \text{ subgraph}\}$. 完全图

$Clique \in NP$

certificate: a set of k vertices

verifier: if these k vertices form K_k.

3. Traveling salesman problem (NP hard)

Given a weighted directed graph on n vertices, together with $n(n-1)$ distances d_{ij} ...

each pair of vertices, and a number k, decide if there is a closed "tour" that visits each vertex only once and has total length $\leq k$.

欧拉路 $\in P$.

4. Hamiltonian cycle (NP complete)

Given a directed graph, decide if there is a closed cycle that visits each vertex exactly once. 证书: 回路

5. Factoring

Given integers N and $L, R \leq N$. decide if N has a divisor in $[L, R]$

Factoring $\in NP$

Certificate: $d \in [L, u]$

verifier: checks if $d|n$.

conjecture 猜想

Conj. Factoring $\in NP$-complete assuming $P \neq NP$.

Factoring $\in BQP$. (对量子计算机, 简单) 检查是否质数 $\in P$.

*A major open problem in cryptography: NP-hard based encryption

6. 0/1 integer programming

7. Subset sum problem

classical | quantum

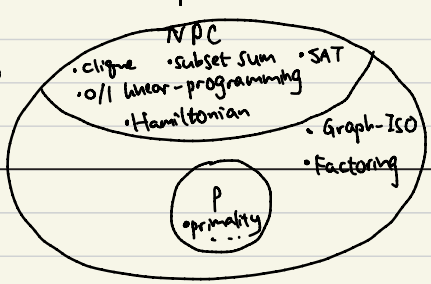
P | BQP

Bounded-error | quantum

NP | QMA

$P \subseteq BQP$

$NP \nsubseteq BQP$



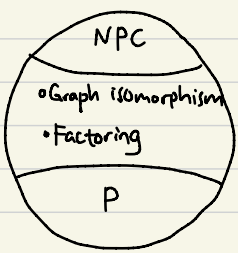
NP

1) problems that can be solved by NTM in poly time
2) problems that can be verified by TM in poly time

$x \overset{?}{\in} L$

if $x \in L$, $\exists w \in \{0,1\}^{p(|x|)}$ s.t. $M(x,w)=1$.

if $x \notin L$, $\forall w \in \{0,1\}^{p(|x|)}$ $M(x,w)=0$.



Def 5.23 $EXP = \bigcup_{c \geq 1} DTIME(2^{n^c})$

$= DTIME(2^{n^{O(1)}})$

Prop 5.24 $P \subseteq NP \subseteq EXP$

Proof ($P \subseteq NP$) Let $p(n)=0$ and $w=\varepsilon$.

Verifier: decide if $x \in L$. ⊛

($NP \subseteq EXP$) Let $L \in NP$. Then $\exists$ polynomial $p: \mathbb{N} \to \mathbb{N}$ and poly-time verifier M. Construct a TM that decides L as follows:

枚举所有可能的证书

on input $x \in \{0,1\}^*$, enumerate all $w \in \{0,1\}^{p(|x|)}$ and check if $M(x,w)=1$. Time: $2^{p(n)}$ $poly(n+p(n)) = 2^{n^{O(1)}}$

→ M的时间

Thus, $L \in EXP$.

5.4 Cook-Levin theorem

SAT (Boolean satisfiability problem)

variable. A variable can take value true or false

literal. is a variable or its negation. eg. x, $\neg x$, y, $\neg y$

clause. is an OR (disjunction) of one or more literals

eg. $x \vee y \vee z$ $x \vee \neg y$