Theorical Computer Science

Week 1

TCS
- upper bound → algorithm $O(n^2) \to O(n\log n)$
- lower bound → hardness $\Omega(n) \to \Omega(n\log n)$ comparison-base

definition: problem, computing model, computation
computability, tractability 老师说, randomness

important: definition, theorem
- statement
- proof idea
- proof

P, NP

一、Problem

1. Given a weighted graph G, what is the MST? ↑ delete an edge, check whether the weight increases or not (in MST / not in) $O(\hat{E})$
   - what is the weight of its MST? ← binary search k $O(\log \hat{O})$
   - and an integer k, does G have a ST with weight at most k?

decision problem ← P

2. Given a string w, is $w \in L$? ← Given a string w, is $w \in L$?
   $L = \{\text{encode} \langle G, k\rangle : G \text{ is a graph that has a ST with weight at most } k\}$
   Language ← $L = \{\text{encoding of yes-instance of } P\}$ ← L

3. An alphabet is a *finite* set of symbols.
   $\Sigma = \{0,1\}$ $\Sigma = \{A, B, X, Y\}$ $\Sigma = \{\}$ $\Sigma = \{\Delta, \square, O, \times\}$

(1) A string over Σ is a *finite* sequence of symbols from Σ.

(2) The length of a string w is symbols. $|w|$

(3) empty string ε, $|\varepsilon| = 0$

(4) concatenation: $u = 010$, $v = 110$, $uv = 010110$

(5) exponentiation: $w^i = \underbrace{w \cdots w}_{i \text{ times}}$ $w^0 = \varepsilon$, $w^2 = w \cdot w$

(6) reversal: $w = 0011$, $w^R = 1100$

4. A language over Σ is a set of strings over Σ

(1) concatenation: $AB = \{uv : u \in A \text{ and } v \in B\}$ $A \cdot \{\varepsilon\} = A$

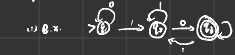
(2) exponentiation: $A^i = \underbrace{A \cdots A}_{i \text{ times}}$, $A^0 = \{\varepsilon\}$, $A^{i+1} = A \cdot A^i$

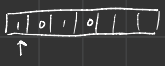
(3) $A^+ = \bigcup_{k \geq 1} A^i$, $A^* = \bigcup_{i \geq 0} A^i = A^+ \cup \{\varepsilon\}$

(4) Σ^i: a set of string of length i consisting of symbols in Σ
Σ^+, Σ^*

(5) reversal: $A^R = \{w^R : w \in A\}$

5. finite automata

(a) Ex:  → state diagram



>○ : initial state (unique)

◎ : final state (several)

if stop at final state, accept the sequence

$(q_1, 1010)$ $(q_2, 010)$ $(q_3, 10)$

$(q_2, 0)$ (q_3, ε)

(b) def: A finite automaton $M = (K, \Sigma, \delta, s, F)$

K: a finite set of states

Σ: input alphabet

$s \in K$: initial state

$F \subseteq K$: a set of final states

δ: $K \times \Sigma \to K$ transition function

(current state, symbol, next state)

A configuration is an element of $K \times \Sigma^*$ (current state, remaining input)

(c) $(q,w) \vdash_M (q',w')$ if $w = aw'$ and $\delta(q,a) = q'$, for some $a \in \Sigma$

(yields in one step)

$(q,w) \vdash_M^* (q',w')$ if $(q,w) = (q',w')$ or $(q,w) \vdash_M \cdots \vdash_M (q',w')$

feed w on M $\begin{cases} (s,w) \vdash_M^* (q,\varepsilon),\ q \in F & \Rightarrow M \text{ accepts } w \\ (s,w) \vdash_M^* (q,\varepsilon),\ q \notin F & \Rightarrow M \text{ rejects } w \end{cases}$

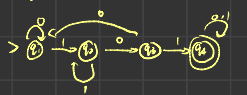
M accepts a language L if $\begin{cases} M \text{ accepts all strings in } L \\ M \text{ rejects all strings not in } L \end{cases}$ (unique)

$L(M) = \{w : M \text{ accepts } w\}$

example: >○ 0,1 $L(M) = \emptyset$

>◎ 0,1 $L(M) = \{0,1\}^*$

$\{w \in \{0,1\}^* : w$ contains 101 as a substring$\}$



(d) A language is regular if it is accepted by some finite automaton.

(e) Theorem: If A and B are regular, so is $A \cup B$.

Idea: 

Proof: $\exists M_A = (K_A, \Sigma, \delta_A, s_A, F_A)$, $M_B = (K_B, \Sigma, \delta_B, s_B, F_B)$

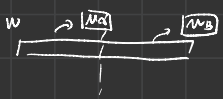
$M_U = (K_U, \Sigma, \delta_U, s_U, F_U)$,

$K_U = K_A \times K_B$, $s_U = (s_A, s_B)$, $F_U = (F_A \times K_B) \cup (K_A \times F_B)$

$\delta_U((q_A, q_B), a) = (\delta_A(q_A, a), \delta_B(q_B, a))$

for any $(q_A, q_B) \in K_A \times K_B$ and any $a \in \Sigma$

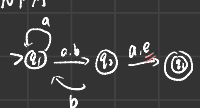
(f) Theorem: If A and B are regular, so is AB



Non-determinism

Week 2

1. NFA



(1) next state is not unique

(2) ε-transition

$\Delta = \{(q_1, a, q_1), (q_1, a, q_2), (q_1, b, q_2) \cdots (q_2, \varepsilon, q_3)\}$

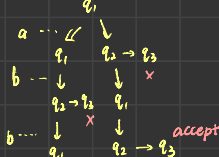
(q_1, abb)

$\vdash_M (q_1, bb)$ $\vdash_M (q_2, b)$ $\vdash_M (q_3, \varepsilon)$ ×

$\vdash_M (q_2, bb)$ ×

$\vdash_M (q_1, b)$ $\vdash_M (q_2, \varepsilon)$ $\vdash_M (q_3, \varepsilon)$ ✓

like a tree.



(3) A NFA is a 5-tuple $(K, \Sigma, \Delta, s, F)$ (transition relation)

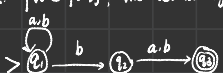
$\Delta \subseteq K \times (\Sigma \cup \{\varepsilon\}) \times K$ (current state, symbol, next state)

M on input w $\begin{cases} \text{reject} & \text{otherwise} \\ \text{accept} & \text{if } (s,w) \vdash_M^* (q,\varepsilon) \text{ for some } q \in F \end{cases}$

M accepts a language L ($L(M)$) if $\begin{cases} \text{accepts every } w \in L \\ \text{reject every } w \notin L \end{cases}$

Magic: always make the right guess

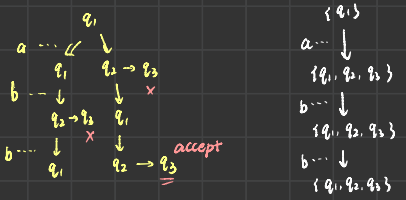
(2) E.x. $\{w \in \{a,b\}^* :$ the second symbol from the end of w is $b\}$



(3) Theorem: given NFA N $\Rightarrow$ DFA M, $L(M) = L(N)$

$\Leftarrow$ → this way is obvious

Idea: $\Rightarrow$ DFA M simulates "tree-like" computation of NFA N



NFA $N = (K, \Sigma, \Delta, S, F)$

DFA $M = (K', \Sigma, \delta, S', F')$

$K' = 2^K = \{Q : Q \subseteq K\}$

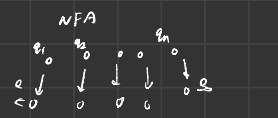
$F' = \{Q \subseteq K : Q \cap F \neq \emptyset\}$

$\forall q \in K$, $E(q) = \{p \in K : (q, e) \vdash_N^* (p, e)\}$

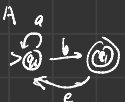
$S' = E(S)$

$\delta(Q, a) = \bigcup_{q \in Q} \bigcup_{p : (q,a,p) \in \Delta} E(p)$

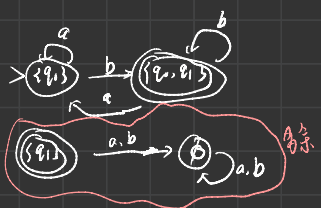
$L(M) = L(N)$



Example: NFA



DFA

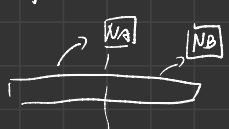


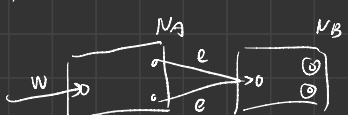
(4) Theorem:

A language is regular if and only if it is accepted by some NFA

(5) Theorem:

If A and B are regular, so is AB.





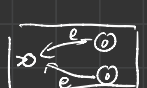
$N_A = (K_A, \Sigma, \Delta_A, S_A, F_A)$ $N_B = (K_B, \Sigma, \Delta_B, S_B, F_B)$

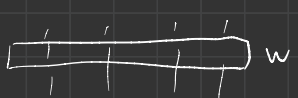
$N_0 = (K_0, \Sigma, \Delta_0, S_0, F_0)$

$K_0 = K_A \cup K_B$ $S_0 = S_A$, $F_0 = F_B$

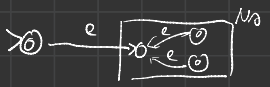
$\Delta_0 = \Delta_A \cup \Delta_B \cup \{(q, e, S_B) : q \in F_A\}$

(6) Theorem: If A is regular, so is A^+. $A^+ = A^1 \cup A^2 \cup A^3 \cdots = \{w_1 w_2 \cdots w_k : w_i \in A, k \geq 1\}$





(7) Theorem: If A is regular, so is A^*



2. Regular Expression (REX)

(1) $R = a(a \cup b)^* b$

$L(R) = \{w \in \{a,b\}^* \mid w \text{ starts with a and ends with b}\}$

(2) Atomic: ϕ, $L(\phi) = \phi$, e

$a \in \Sigma$, $L(a) = \{a\}$

e表示空串，ϕ表示不包含任何字符串

Precedence: $^* > \circ > \cup$

$ab^* \cup ba^* = (a(b^*)) \cup (b(a^*))$

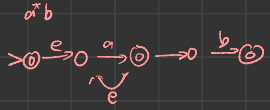
Composite: $(\cup, \circ, ^*)$

$R_1 \cup R_2$, $L(R_1 \cup R_2) = L(R_1) \cup L(R_2)$

$R_1 R_2$, $L(R_1 R_2) = L(R_1) L(R_2)$

R^*, $L(R^*) = (L(R))^*$

example: a^*b



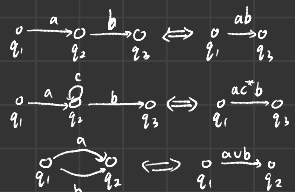
(3) $\{w \in \{a,b\}^* \mid w \text{ has two occurences of } ab\}$

$(a \cup b)^* a (a \cup b)^* a (a \cup b)^*$

Week 3

1. NFA → REX

Idea: state elimination

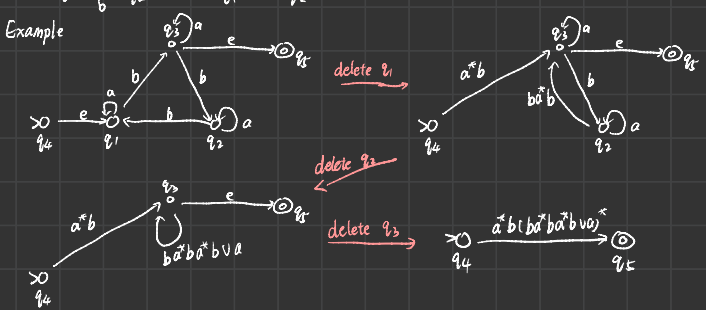


Simplify M so that

(1) no arc enters the initial state

(2) unique final state, and no arc leaves it.

Example



example: $R^0_{55} = \phi^* \quad \{e\}$

$R^0_{54} = \phi^* \quad \{e\}$

$R^0_{33} = \phi^* \cup \{a\} \quad \{e, a\}$

Proof: Dynamic Programming

Input: Simplified NFA $M = (K, \Sigma, \Delta, s, F)$

1. $K = \{q_1, \cdots q_n\}$, $s = q_{n-1}$, $F = \{q_n\}$
2. $\forall p. \forall a. (p, a, q_{n-1}) \notin \Delta$
3. $\forall p. \forall a. (q_n, a, p) \notin \Delta$

Subproblem:

for $i,j,k \in \{1, \cdots, n\}$, let R^k_{ij} be a REX. s.t.

$L(R^k_{ij}) = \{w \in \Sigma^* \mid (q_i, w) \vdash^*_M (q_j, e) \text{ with no intermediate state having index} > k\}$

Goal:

$R_{i\cup n}^{n-2}$

Base case $k=0$

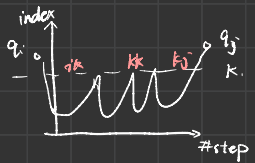
if $i=j$: $R_{ij}^0 = \phi^* \cup a_1 \cup a_2 \cdots$, $(q_i, a_n, q_j) \in \Delta$

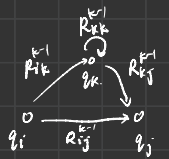
if $i\neq j$: $R_{ij}^0 = a_1 \cup a_2 \cup \cdots$, $(q_i, a_n, q_j) \in \Delta$

Recurrence

$\{R_{ij}^{k-1}\}_{i,j} \rightarrow \{R_{ij}^{k}\}_{i,j}$

$R_{ij}^k = R_{ij}^{k-1} \cup R_{ik}^{k-1} (R_{kk}^{k-1})^* R_{kj}^{k-1}$





Conclusion: To prove that L is regular:

DFA, NFA, REX

closure property: $\cup, \cap, \circ, \bar{A}, *$

2. Pumping Theorem

Pumping length

Let L be regular language. There exists an integer $p\geq 1$ such that $\forall w\in L$ which $|w|\geq p$ can be divided into three parts $w=xyz$, satisfying:

(1) $|y|>0$

(2) for any $i\geq 0$, $xy^iz\in L$

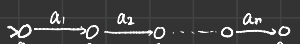
(3) $|xy|\leq p$

example: ab^*a, $p=3$

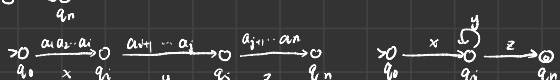
Proof: if L is finite, $p = \max_{w\in L}(|w|+1)$

Assume L is infinite, $\exists$ DFA M accepts L. $p=$ # states of M

$\forall w\in L$ with $|w|\geq p$, $w=a_1a_2\cdots a_n$, $n\geq p$



$\exists\, 0\leq i<j\leq p$, $q_i=q_j$



(1) $1<j \Rightarrow |y|>0$

(2) $\forall t\geq 0$, $xy^tz\in L$

(3) $j\leq p \Rightarrow |xy|\leq p$

3, Prove $\{0^n1^n : n\geq 0\}$ is not regular

Proof: Assume it is regular, let p be the pumping length

consider $w=0^p1^p$. By pumping theorem, $w=xyz$, s.t.

(1) $|y|>0$ (2) $xy^iz\in L$, for any $i\geq 0$ (3) $|xy|\leq p$

From (1)(3), $y=0^k$, for some $k\geq 1$, then $xy^0z = 0^{p-k}1^p \notin L$

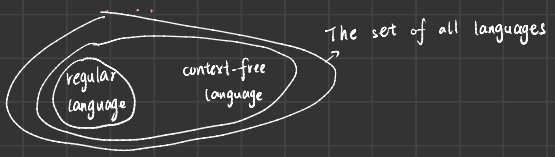
contradictory with (2)

4. Prove $\{w\in\{0,1\}^* : w$ has equal number of 0's and 1's$\}$ is not regular

assume L is regular $\Rightarrow$ $L\cap 0^*1^*$ is regular $\Rightarrow$ $\{0^n1^n : n\geq 0\}$

Chapter 2

5.



6. context-free grammar (CFG)

S: start symbol S, A: non-terminals

$S \to aSb$ rules a, b, c: terminals

$S \to A$ ↓ 左邊 non-terminal

$A \to c$

$A \to e$

$S \to aSb \to aaSbb \to aaAbb \to aacbb$

A context-free grammar $G=(V, \Sigma, S, R)$

V: a finite set of symbols

$\Sigma \subseteq V$: the set of terminals

$V-\Sigma$: the set of non-terminals

$S \in V-\Sigma$: start symbol

$R \subseteq (V-\Sigma) \times V^*$: a finite set of rule $(A, w) \in R$. $A \to w$

(1) for any $x, y, u \in V^*$, for any $A \in V-\Sigma$ $xAy \Rightarrow xuy$ if $(A, w) \in R$

derive in one step

(2) for any $w, u \in V^*$, $w \Rightarrow^* u$ if $w = u$ or $w \Rightarrow \cdots \Rightarrow \cdots \Rightarrow u$

terminal ↑ ↓ a derivation from w to u of length n

(3) G generates $w \in \Sigma^*$ if $S \Rightarrow^* w$

$L(G) = \{w \in \Sigma^* : G \text{ generates } w\}$, G generates $L(G)$

example: (1) $\{0^n 1^n, n \geq 0\}$

$S \to 0S1 \mid e$

(2) $\{w \in \{a,b\}^*, w = w^R\}$

$S \to aSa$
$S \to bSb$
$S \to e$
$S \to a$
$S \to b$

$\Rightarrow S \to aSa \mid bSb \mid e \mid a \mid b$

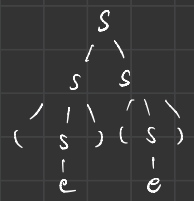
Week 4

1. $S \to SS$
   $S \to (S)$
   $S \to e$

   $S \Rightarrow SS \Rightarrow (S)S \Rightarrow ()S \Rightarrow ()(S) \Rightarrow ()()$ → leftmost derivation

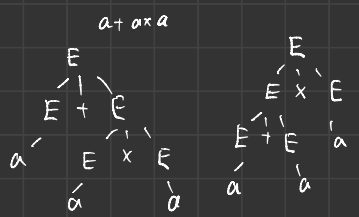
   $S \Rightarrow SS \Rightarrow S(S) \Rightarrow S() \Rightarrow (S)() \Rightarrow ()()$ → rightmost derivation

Parse tree:



( ) ( ) : the yield of the parse tree

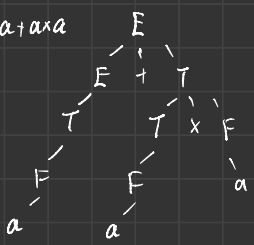
2. $E \to E+E$
   $E \to E \times E$
   $E \to (E)$
   $E \to a$



→ two different

ambiguous ① grammar 多数的
② language 多数的

3. $E \to E+T$
   $E \to T$
   $T \to T \times F$
   $T \to F$
   $F \to (E)$
   $F \to a$

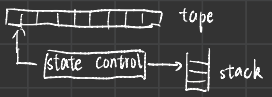


E: Expression
T: term
F: factor

4. $\{a^i b^j c^k : i=j \text{ or } j=k\}$ 所有空或此语言的CFG都有歧义
   ↳ inherent ambiguous

5. Pushdown Automata $\rightleftharpoons$ Context-free language

   PDA = NFA + stack



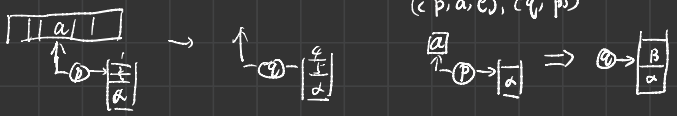
A PDA is 6-tuple $P = (K, \Sigma, T, \Delta, s, F)$

- K: a finite set of state
- Σ: tape alphabet
- T: stack alphabet
- s: initial state
- F: a set of final state
- Δ: transition relation, a finite subset of

$$(K \times (\Sigma \cup \{e\}) \times T^*) \times (K \times T^*)$$

current state | current symbol | the top part of stack | next state | pop now, push a string onto the stack

example: $(\langle p, a, 123\rangle, (q, 45))$



6. A configuration of PDA P is a member of $K \times \Sigma^* \times T^*$ (stack content)

$(p, x, \alpha) \vdash_P (q, y, \beta)$ if $\exists\ [(p, a, \tau), (q, \gamma)] \in \Delta$ st $x = ay$, $\alpha = \gamma\tau$ and $\beta = \gamma\tau$ for some $\tau \in T^*$

(yields in one step)

$(p, x, \alpha) \vdash_P^* (q, y, \beta)$ if $(p, x, \alpha) = (q, y, \beta)$ or $(p, x, \alpha) \vdash_P \cdots \vdash_P (q, y, \beta)$

P accepts a string $w \in \Sigma^*$ if $(s, w, e) \vdash_P^* (q, e, e)$ (① ② ③) for some $q \in F$

$L(P) = \{w \in \Sigma^* : P \text{ accepts } w\}$, P accepts $L(P)$

7. E.x. $L = \{w \in \{0,1\}^* : \#0\text{s} = \#1\text{s in } w\}$

Idea: When read a 0:

if stack is empty or the top is 0:

push(0)

else // top is 1

pop(1)

When read a 1:

Proof: Version 1:

$K = \{s, q, f\}$, $s = s$, $F = \{f\}$

$\Sigma = \{0,1\}$, $T = \{0,1,\$\}$

$\Delta = \{((s, e, e), (q, \$)), ((q, 0, \$), (q, 0\$)), ((q, 0, 0), (q, 00)), ((q, 0, 1), (q, e))$

(同理 …… …… , $((q, e, \$), (f, e))\}$

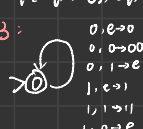
Version 2:

$K = \{q\}$, $s = q$, $F = \{q\}$ 利用 PDA 可以猜 like NFA

$\Sigma = T = \{0,1\}$

$\Delta = \{((q, 0, e), (q, 0)), ((q, 0, 0), (q, 00)), ((q, 0, 1), (q, e))$ …… $\}$

Version 3:



Week 5

1. CFG ⟶ PDA

$S \to aSb \quad S \to e$

for input a a b b

Idea:

1. in stack, non-deterministically generate a string according CFG
2. compare it to input
3. accept it if they match

Given a CFG, $G=(V,\Sigma,R,S)$, construct a PDA, $P=(K,\Sigma,T,\Delta,s,F)$, such that $L(P)=L(G)$

$K=\{s,f\}$, $F=\{f\}$, $T=V$,

$\Delta=\{((s,e,e),(f,S))$,

$((f,e,A),(f,u))$, $\forall(A,u)\in R$

$((f,a,a),(f,e))$, $\forall a\in\Sigma\}$

2. PDA → CFG

↓ simple PDA ↑

1) Def.

A PDA $P=(K,\Sigma,T,\Delta,s,F)$ is simple if

a) $|F|=1$

b) for each transition $((p,a,\alpha),(q,\beta))\in\Delta$, either $\alpha=e$ and $|\beta|\le 1$ // push a symbol

or $|\alpha|=1$ and $\beta=e$ // pop a symbol

2) PDA → simple PDA

① if $|F|=0$, $L(P)$ 为空, PDA → CFG

② if $|F|>1$, create a new state f'

for each $q\in F$

create a transition $((q,e,e),(f',e))$

③ 2.3.1 goal: $\alpha=e$ or $\beta=e$

for each $((p,a,\alpha),(q,\beta))$ with $\alpha\neq e$ and $\beta\neq e$

create a state r

replace it with

$((p,a,\alpha),(r,e))$ and $((r,e,e),(q,\beta))$

2.3.2 for each $((p,a,\alpha),(q,e))$ with $\alpha=C_1\cdots C_k$

create $k-1$ new states $r_1,\cdots r_{k-1}$

replace it with

$((p,a,C_1),(r_1,e))$ $((r_1,e,C_2),(r_2,e))$

⋮

$((r_{k-1},e,C_k),(q,e))$

2.3.3 for each $((p,a,e),(q,\beta))$ with $\beta=C_1\cdots C_k$

注意: push 顺序由 $C_k, C_{k-1}\cdots$ 到 C_1

2.3.4 for each $((p,a,e),(q,e))\in\Delta$

create a new state r

let $b\in T$

replace it with

$((p,a,e),(r,b))$ $((r,e,b),(q,e))$

3) simple PDA → CFG

Given a simple PDA $P=(K,\Sigma,\Gamma,\Delta,s,\{f\})$

construct a CFG $G=(V,\Sigma,S,R)$

subproblems

Non-terminal: $\{A_{pq}$: for any $p,q\in K\}$

$A_{pq} \Rightarrow^* w\in\Sigma^*$ if and only if $(p,w,e)\vdash_P^* (q,e,e)$

so $S=A_{sf}$

$A_{pp}\to e \quad \forall p\in K$

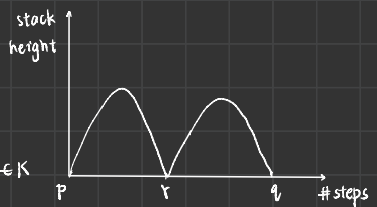
for any $p,q\in K$

$A_{pq}\to A_{pr}A_{rq}$ for any $r\in K$ ← 有可能遇到的步

如

$S\to A$
$A\to a$
$S\to B$

$L(G)=\{a\}$
B之后就不行了

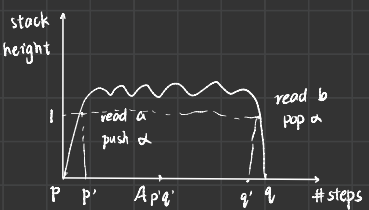


$A_{pq}\to aA_{p'q'}b$

for any p',q',a,b satisfying

$((p,a,e),(p',\alpha))\in\Delta$

$((q',b,\alpha),(q,e))\in\Delta$



3. 推论: prove context-free 可用 PDA / CFG

4. $\{w\in\{a,b\}^*$: #a's = 2#b's in $w\}$

$2\#b's-\#a's=?$

stack → unary counter

+: +1
−: −1

$PDA=(K,\Sigma,\Gamma,\Delta,S,F)$

$K=\{s\}$, $F=\{s\}$, $\Sigma=\{a,b\}$, $\Gamma=\{+,-\}$

$\Delta=\{((s,b,e),(s,++)),((s,b,-),(s,+)),((s,b,--),(s,e)),$
$((s,a,e),(s,-)),((s,a,+),(s,e))\}$

5. Theorem:

Every regular language is context-free. 反之, context-free 的不一定 regular

simply, NFA → PDA

harder: DFA → CFG $M=(K,\Sigma,\delta,s,F)$, $G=(V,\Sigma,s,R)$ ($s=q_0$)

$\forall q_i\in K \Rightarrow Q_i\in(V-\Sigma)$

$Q_i\Rightarrow^* w$ iff $(q_i,w)\vdash_M^* (q,e)$ for $q\in F$

$S=Q_0$

$\forall q_i\in F$, $Q_i\to e$ base case

for any $q_i\in K$, if $\delta(q_i,a)=q_j$, $Q_i\to aQ_j$

6. The family of context-free languages is closed under $\cup$ $\circ$ $*$

but not closed under $\cap$, $-$

A, B context-free

$G_A = (V_A, \Sigma, S_A, R_A)$, $G_B (V_B, \Sigma, S_B, R_B)$

(1) $A \cup B$ $\quad S \to S_A \mid S_B$

(2) $A \circ B$ $\quad S \to S_A S_B$

(3) A^* $\quad \begin{cases} S \to e \\ S \to SS_A \end{cases}$

(4) $A = \{a^i b^j c^k : i=j\}$

$B = \{a^i b^j c^k : j=k\}$

$A \cap B = \{a^i b^j c^k : i=j=k\}$ ← not context-free

(5) $A \cap B = \overline{\overline{A} \cup \overline{B}}$

Week 6

1. Pumping Theorem for CFL

Let L be a context-free language. There exists an integer $p>0$, such that any $w \in L$ with $|w| \geq p$ can be devided into five pieces $w = uvxyz$ satisfying:

(1) $uv^i x y^i z \in L$ for any $i \geq 0$

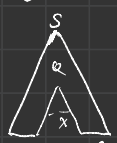
(2) $|v| + |y| > 0$

(3) $|vxy| \leq p$

example: $\{a^n b^n : n \geq 0\}$: $p=2$ $\quad$ eaebe

Idea:

$S \Rightarrow^* uAz \Rightarrow^* uvAyz \Rightarrow^* uvxyz$



$\to uv^0xy^0z \in L$

$\to uv^2xy^2z \in L$

类推也，都成立

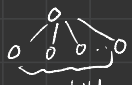
Proof:

does not contain $A \to A$

$\exists$ a CFG, $G = (V, \Sigma, S, R)$ generate L

distinct non-terminals $= |V - \Sigma|$

required tree height $\geq |V-\Sigma| + 2$ → contain leaf (terminal)



fanout $\leq \max_b \{|u| : (A, u) \in R\}$

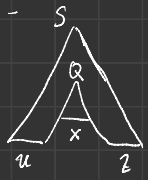
pick the smallest parse tree that yields w

So, let $p = b^{|V-\Sigma|+1}$. If a tree of n nodes has fanout $\geq b$

$\Rightarrow$ height $\geq \log_b n + 1$

longest path has at least $|V-\Sigma|+1$ non-terminals

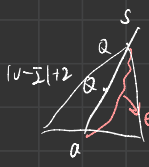
$\Rightarrow$ some non-terminal appears at least twice then repeat the graph above



If $|v|+|y|=0$, then there exists smaller tree

contradictory!

So $|v|+|y|>0$



pick the lowest Q that appears twice on the path

$|v-\Sigma|+2$

every non-terminal appears at most once

$\because$ 최대 $b^{|v-\Sigma|+1} \le p$

2. $\{a^n b^n c^n, n \ge 0\}$ is not context-free.

Let p be the pumping length given by the pumping theorem.

Pick $w = a^p b^p c^p \in L$

By pumping theorem, $w = uvxyz$. st

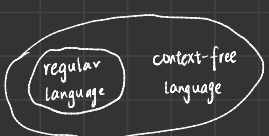
(1) $uv^i xy^i z \in L$, for any $i \ge 0$ (2) $|v|+|y|>0$ (3) $|vxy| \le p$

From (3), v and y contain either a or c, not both

$+(2) \Rightarrow uv^0xy^0z \notin L$, contradicting with (1)

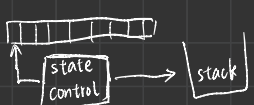
Short Review

DFA ↘
RE x ↔ regular language ↔ CFG ↔ context-free language
NFA ↗ ↑↓ PDA



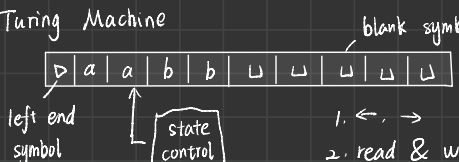
finite automaton

PDA



---

3. Turing Machine



A Turing Machine is a 5-tuple $M = (K, \Sigma, \delta, s, H)$

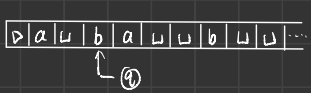
- K: a finite set of states
- Σ: an alphabet (containing ▷ and ⊔)
- s: initial state
- $H = \{y, n\} \subseteq K$. halting states
- $\delta: (K-H) \times \Sigma \to K \times (\Sigma \cup \{\leftarrow, \rightarrow\})$

current state / next state / write / moving

satisfying: (1) for any $q \in K-H$, $\delta(q, \triangleright) = (p, \rightarrow)$ for some $p \in K$

(2) for any $q \in K-H$, for any $a \in \Sigma$,
if $\delta(q,a) = (p,b)$, then $b \neq \triangleright$ → 即 left-end symbol 不能被改寫

Configuration:

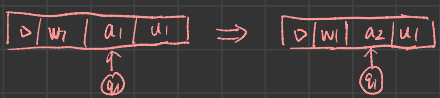


$(q, \triangleright a \underline{\sqcup} b, a \sqcup b)$ or $(q, \triangleright a \sqcup \underline{b} a \sqcup \sqcup b)$

A configuration is a member of $K \times \triangleright\{\Sigma - \{\triangleright\}\}^* \times (\{\Sigma - \{\triangleright\}\}^* (\Sigma - \{\triangleright, \sqcup\}) \cup \{\epsilon\})$

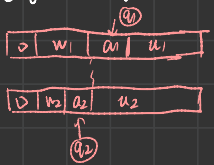
We say $(q_1, \triangleright w_1 \underline{a_1} u_1) \vdash_M (q_2, \triangleright w_2 \underline{a_2} u_2)$ if

writing (1) $\delta(q_1, a_1) = (q_2, a_2)$ and $a_2 \in \Sigma - \{\triangleright\}$ and $w_2 = w_1$ and $u_2 = u_1$



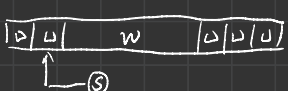
moving left ↓ right 向右移

(2) $\delta(q_1, a_1) = (q_2, \leftarrow)$ $w_1 = w_2 a_2$, $u_2 = a_1 u_1$



(3) M halts whenever it reaches a halting configuration.

Given an input w initial configuration $(s, \triangleright \underline{\sqcup} w)$



M accepts the input w if $(s, \triangleright \underline{\sqcup} w) \vdash_M^* (y, \triangleright \sqcup \underline{a} w)$

M rejects the input w if $(s, \triangleright \underline{\sqcup} w) \vdash_M^* (n, \triangleright \sqcup \underline{a} w)$

accept / reject / looping

Given a TM M, $L(M) = \{w \in \{\Sigma - \{\triangleright, \sqcup\}\}^* : M \text{ accepts } w\}$

M decides $L(M)$ if
- accept if $w \in L(M)$
- reject if $w \notin L(M)$

recursive language

semidecides $L(M)$ if
- accept if $w \in L(M)$
- reject or looping if $w \notin L(M)$

→ recursively enumerable language

4. TM decides $\{a^n b^n c^n : n \geq 0\}$

(1) check if it is the form $a^* b^* c^*$ 確保為 a, b, c

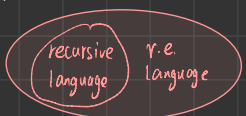
(2)

Week 7

1. Given M, $L(M) = \{w : M \text{ accepts } w\}$ → M semidecides $L(M)$

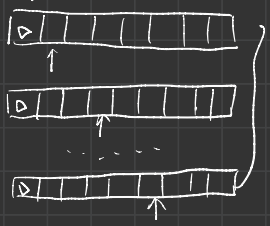
If M halts on every time, then M decides $L(M)$

Every recursive language must be recursively enumerable.



Extensions of TM

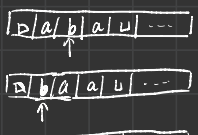
1. multiple tapes



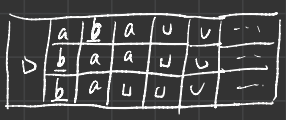
$\delta : (K-H) \times \Sigma^k \to K \times ((\Sigma - \{\triangleright\}) \cup \{\leftarrow, \rightarrow\})$

和原先的表达能力相同

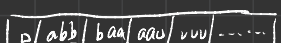
Idea: 3-tape



single-tape



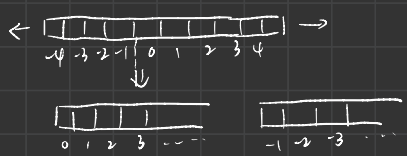
即



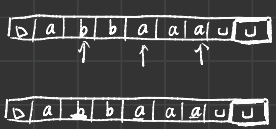
$|\Sigma_1| = (2|\Sigma_2|)^3$

↳ 2为有下划线或无下划线

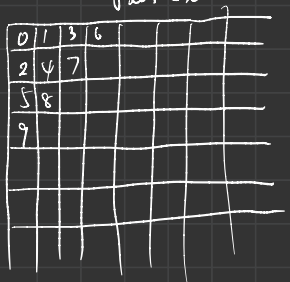
2. Two-way infinite tape



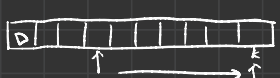
3. Multiple head



4. 2D-tape 可数无穷大



5. random access



6. Non-deterministic TM (NTM): A NTM is a 5-tuple $(K, \Sigma, \Delta, s, H)$

- K
- Σ
- s
- H
- Δ: a *finite* subset of $((K-H) \times \Sigma) \times (K \times ((\Sigma - \{\triangleright\}) \cup \{\leftarrow, \rightarrow\}))$

DTM

NTM

M decides L if 只能有有限个分支，不能写成 every branch halts

(1) for any w, there is an integer N, depending on w and M such that every branch halts with N steps

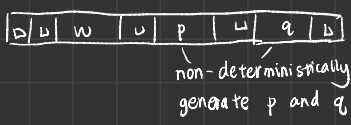
(2) if $w \in L$, $\exists$ some branch accepts w.
if $w \notin L$, every branch rejects w.

M semidecides L if for any input w.

(1) if $w \in L$, $\exists$ some branch accepts w.

(2) if $w \notin L$, no branch accepts w.

example: Let $C = \{100, 110, 1000, \dots\}$ 为所有合数的集合



if $w = pq$, accept ; otherwise, reject

7. Theorem: Every NTM can be simulated by a DTM.

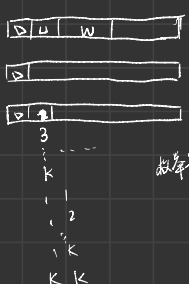
Sketch: a NTM N semidecides L → a DTM M semidecides L

given input w, $w \in L$ iff some branch accepts w (some node is in state y)

DFS × (looping)

BFS √

3-tape DTM:



做第每一层选择第几个指令

8. Church-Turing Thesis

Intuition of Algorithm is equivalent to Turing Machine.

9. Fact1: Any finite set can be encoded

$\{a_1, \dots, a_n\} \Rightarrow a00\,000\,|\,a\,000\,|0 \cdots a\,|\underbrace{\quad}_{\log n}$

Fact 2: Any finite tuple whose elements are finite sets can be encoded

$(A, B, C, D) \Rightarrow a000\,|\,a00|0 \cdots a \cdots b \cdots c \cdots d \cdots$

E.x. $G = (V, E) \rightarrow$ "G" $\quad M = (K, \Sigma, \delta, s, H) \rightarrow$ "M"

10. $L = \{\text{"}G\text{"} : G \text{ is a connected graph}\}$

M = on input "G"

0. if the input is illegal, reject (by default)
1. select a node of G and mark it.
2. repeat the following
3. for each marked node
4. mark all of its neighbors
5. if all the nodes are marked
   accept
6. else
   reject

11. Problems

(1) R_1: $A_{DFA} = \{\text{"}D\text{"}\,\text{"}w\text{"} : D \text{ is a DFA that accepts } w\}$

M_{R_1} = on input "D" "w"

1. run D on w
2. if D accepts w
   accepts "D" "w"
3. else D rejects w
   rejects "D" "w"

A_{NFA} → A_{DFA}

"N" "w" → "D" "w"

1. "N" "w" $\in A_{NFA}$ if and only if "D" "w" $\in A_{DFA}$
2. the function is computable

(2) R_2: $A_{NFA} = \{\text{"}N\text{"}\,\text{"}w\text{"} : N \text{ is a NFA that accepts } w\}$

M_{R_2} = on input "N" "w"

1. convert N to an equivalent DFA D
2. run M_{R_1} on "D" "w"
3. return the result of M_{R_1}

(3) R_3: $A_{REx} = \{\text{"}R\text{"}\,\text{"}w\text{"} : R \text{ is a regular expression with } w \in L(R)\}$

M_{R_3} = on input "R" "w"

1. convert R into an equivalent NFA N
2. run M_{R_2} on "R" "w"
3. return the result of M_{R_2}

(4) R_4: $E_{DFA} = \{\text{"}D\text{"} : D \text{ is a DFA with } L(D) = \emptyset\}$

到不了 final

M_{R_4} = on input "D" >0

1. run DFS on the state diagram of D
2. 若有 path 到 reject, 否則 accept

(5) R_5: $EQ_{DFA} = \{\text{"}D_1\text{"}\,\text{"}D_2\text{"} : D_1 \text{ and } D_2 \text{ are two DFAs with } L(D_1) = L(D_2)\}$

Symmetric different

$A \oplus B = (A \cup B) - (A \cap B)$

1. $A = B$ if and only if $A \oplus B = \emptyset$

2. $A \oplus B = (A \cup B) \cap (\overline{A \cap B})$

$= (A \cup B) \cap (\bar{A} \cup \bar{B})$

M_{R5} : on input "D_1" "D_2"

1. construct a DFA D^* with $L(D^*) = L(D_1) \oplus L(D_2)$
2. run M_{R4} on "D^*"

Week 8

1. $A_{DFA} = \{$"D" "w" : D is a DFA that accepts w$\}$

D is a DFA. $L(D) = \{w$: D accepts w$\}$ ↑ 判断："看一个给定的 D

$w \in L(D) \Leftrightarrow$ D accepts w $\Leftrightarrow$ "D" "w" $\in A_{DFA}$

A_{DFA} is recursive $\Leftrightarrow$ $L(D)$ is recursive.

C1 $A_{CFG} = \{$"G" "w" : G is a CFG that generates w$\}$

A CFG is in chomsky norm form (CNF) if every rule is one of the following forms:

1. $S \to e$
2. $A \to BC$ B, $C \in V - \{S\}$ 可以证明所有 CFG 都可以被表示为 CNF
3. $A \to a$

generate $|w| = n \geq 1$, we need $2n-1$ times (n-1 次 2, n 次 3) in total, $|R|^{2n-1}$ strings

M_{C1} : on input "G" "w"

1. convert G to CNF, $G' = (V, \Sigma, R', S)$
2. enumerate all derivation of length at most $|R'|^{2n-1}$
3. accept if any of these derivation generates w and reject otherwise

C2 $A_{PDA} = \{$"P" "w" : P is a PDA that accepts w$\}$

M_{C2} = on input "P" "w"

1. convert P into an equivalent CFG G
2. run M_{C1} on "G" "w"
3. return the result of M_{C1}

C3 $E_{CFG} = \{$"G" : G is a CFG with $L(G) = \phi\}$

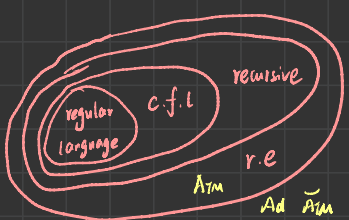
$S \to A$
$A \to aB$
$B \to b$
$B \to Ca$
$C \to e$

M_{C3} = on input "G"

1. mark terminals and e
2. while there is a rule with all symbols on its right are marked but the symbol on the left is not marked : (at most $|R|$ loops)
   mark the symbol on the left
3. if S is marked
   reject
4. else
   accept

C4 $E_{PDA} = \{$"P" : P is a PDA with $L(P) = \phi\}$

transfer P into corresponding CFG



2. countable

A set A is countable if it is finite or $\exists$ bijection $f: A \to \mathbb{N}$. uncountable otherwise

(1) Lemma

A set A is countable if and only if $\exists$ injection $f: A \to \mathbb{N}$ ($a \neq a', f(a) \neq f(a')$)

Proof

"$\Rightarrow$" obviously

"$\Leftarrow$" injection: $f: A \rightarrowtail \mathbb{N}$, need to find bijection $g: A \rightarrowtail \mathbb{N}$

sort A in increasing order of $f(\cdot)$

$g(a)$ = rank of a

(2) Corollary

Any subset A' of a countable set A is countable

Proof

$\exists$ injection $f: A \to \mathbb{N} \Rightarrow \exists$ injection $f': A' \to \mathbb{N} \Rightarrow A'$ is countable

(3) Lemma

Any language is countable.

Proof

Let Σ be an alphabet, we only need to prove Σ^* is countable

Assume $\Sigma = \{0, 1\}$ $\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000 \cdots\}$

map to their ranks $\quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad \cdots\cdots$

(4) Corollary

$\{M : M$ is a TM$\}$ is countable.

Proof

encode: $\{\langle M \rangle : M$ is a TM$\}$. The corresponding language is countable.

so the original statement is true.

$\exists$ language is not recursively enumerable

(5) Lemma

Let Σ be an non-empty alphabet

Let $\mathcal{L}$ be the set of languages over Σ

$\mathcal{L}$ is uncountable.

Proof

Diagonalization

Suppose that $\mathcal{L}$ is countable. $L_1, L_2, L_3 \cdots$

Since Σ^* is countable, the strings in Σ^* can be labeled as

$s_1, s_2, s_3, \cdots\cdots$

$D = \{s_i : s_i \notin L_i\}$

for $i = 1, 2, 3 \cdots\cdots$

if $s_i \notin L_i$

add s_i to D

$s_i \in D$ if and only if $s_i \notin L_i \ \forall i \Rightarrow D \neq L_i, \forall i$

But $D \in \mathcal{L}$, contradictory!

existing a table

	s_1	s_2	s_3	s_4	$\cdots\cdots$
L_1	1	0	0	1	
L_2	1	0	1	0	
L_3	1	1	0	1	
L_4	0	1	1	0	
D	0	1	1	1	

对角 $s_i \in L_i$

取对角线，若 L_i 中有 s_i，则 s_i 不在 D 中，否则在

选择的 D 不与前面所有 L_i 相同，则不可数.

3. $A_{TM} = \{"M""w" : M \text{ is a TM that accepts } w\}$

Theorem:

A_{TM} is recursive enumerable

Proof: → universal turing machine

U = on input $"M""w"$

1. run M on w
2. accept $"M""w"$ if M accepts w
   reject if M rejects w

U accepts $"M""w"$ if and only if M accepts w
if and only if $"M""w" \in A_{TM}$

Theorem

A_{TM} is not recursive.

Proof

$A_d = \{"M" : M \text{ is a TM that does not accept } "M"\}$

① if A_{TM} is recursive, A_d is recursive. $\Rightarrow$ A_{TM} is not recursive.

② A_d is not recursively enumerable.

① if A_{TM} is recursive $\Rightarrow \exists M_1$ decides A_{TM}

M^* = on input $"M"$

1. run M_1 to check whether M accepts $"M"$
2. If M accepts $"M"$
   reject
3. else // M rejects $"M"$ or looping on $"M"$
   accept

M^* decides A_d $\Rightarrow$ A_d is recursive

② A_d is not recursively enumerable

Suppose that A_d is r.e. $\Rightarrow \exists$ TM D semidecides A_d

D on input $"M"$ = { accept if $"M" \in A_d$ (M rejects $"M"$ or looping on $"M"$)
reject or looping if $"M" \notin A_d$ (M accepts $"M"$)

D on input $"D"$ = { accept if D rejects $"D"$ or looping on $"D"$
reject or looping if D accepts $"D"$

contradictory ! $\Rightarrow$ $\overline{A_{TM}}$ 也不可判定

Theorem

If L and $\bar{L}$ is enumerable recursive, then L is recursive.

Proof

$\exists M$ semidecides L, $\exists M'$ semidecides $\bar{L}$

M^* = on input w,

1. run M and M' on w in parallel
2. if M accepts w
   accept w
3. if M' accepts w
   reject w

A_{TM} is not recursive

but is recursively enumerable

$\Rightarrow \bar{A}_{TM}$ is not recursively enumerable

Week 9

1. $H_{TM} = \{"M" "w" : M \text{ is a TM that halts on } w\}$

   $A_{TM} = \{"M" "w" : M \text{ is a TM that accepts } w\}$ → not recursive

(归约) A_{TM}

"M" "w"

M accepts w if and only if

M^* halts on w

H_{TM}

M^* = on input x

1. run M on x
2. if M accepts x
   M^* accepts x
3. if M rejects x
   M^* looping
4. if M is looping
   M^* is looping

→ 可以不写

Suppose that H_{TM} is recursive. $\exists M_H$ decides H_{TM}.

M_A = on input "M" "w" → 相当于 input

相当于问题求 problem 的答案

1. construct M^x using M

M^x = on input x

1. run M on x
2. if M accepts x
   M^* accepts x
3. if M rejects x
   M^* looping
4. if M is looping
   M^* is looping

2. run M_H on "M^x" "w"
3. if M_H accepts "M^x" "w":
   (then we know M accepts w)
   accept "M" "w"
4. else M_H reject "M^x" "w"
   reject "M" "w"

M_A decides A_{TM}. 矛盾.

L = the set of all re language containing e

2. $L_1 = \{\langle M\rangle : M \text{ is a TM that accepts } e\}$ 不可判定

A_{TM}	L_1
$\langle M\rangle\langle w\rangle$	M^*

We need M accepts w if only if M^* accepts e

M^* = on input x

1. run M on w
2. if M accepts w
   accepts x
3. if M rejects w
   reject x

归约：一种映射

if M accepts w, then M^* accepts all inputs, including e

if M rejects or loops on w, then M^* doesn't accept any input

Suppose that L_1 is recursive, $\exists M_1$ decides L_1

M_A = on input $\langle M\rangle\langle w\rangle$:

1. construct M^* using $\langle M\rangle\langle w\rangle$
2. run M_1 on M^*
3. if M_1 rejects $\langle M^*\rangle$
   accepts $\langle M\rangle\langle w\rangle$
4. if M_1 accepts $\langle M^*\rangle$:
   reject $\langle M\rangle\langle w\rangle$

M_A decides A_{TM}, wrong

3. $L_2 = \{\langle M\rangle : M \text{ is a TM that accepts all strings}\}$ 不可判定

A_{TM}	L_2
$\langle M\rangle\langle w\rangle$	$\langle M^*\rangle$

M accepts w if and only if M^* accepts all the strings.

之前与上一题类似。

4. $L_3 = \{\langle M_1\rangle\langle M_2\rangle : M_1 \text{ and } M_2 \text{ are the TMs that accept the same set of strings}\}$

Suppose L_3 is recursive. $\exists M_3$ decides L_3

use M_3 to construct M_2 decides L_2

M_2 = on input $\langle M\rangle$

1. construct M_E as follows

M_E = on input x

1. accept

2. run M_3 on "M" "M_E" // check whether M and M_E accepts the same set of inputs
3. if M_3 accepts "M" "M_E"
   accept "M"
4. if M_3 rejects "M" "M_E"
   reject "M"

We know M_2 decides L_2, so contradictory. 不可判定

5. $E_1 = \{$"M" : M is a TM with $L(M)$ is regular$\}$

A_{TM} : "M" "w"

E_1 : "M^x"

M^x = on input x
1. run M on w
2. if M accepts w
3. run U on x
4. accepts x if U accepts x
5. rejects x if U reject x
6. if M reject w
7. reject x

universal TM
U semidecides A_{TM}
$L(U) = A_{TM}$

U = on input "M" "w"
1. run M on w
2. if M accepts w
   U accepts "M" "w"
3. else
   U rejects "M" "w"

M does not accept w, $L(M^x) = \emptyset$ regular / context-free / recursive
M accepts w, $L(M^x) = A_{TM}$ not regular / not context-free / not recursive

$E_2 = \{$"M" : M is a TM with $L(M)$ is context-free$\}$

$E_3 = \{$"M" : M is a TM with $L(M)$ is recursive$\}$

都有为何形式：Given a TM M, is $L(M) \in \mathcal{L}$?

$\mathcal{L}$ is a proper, non-empty subset of the class of recursive enumerable language

6. $R = \{$"M" : M is a TM with $L(M) \in \mathcal{L}\}$ Rice's Theorem

Proof: Assume $\phi \notin \mathcal{L}$ (if $\phi \in \mathcal{L}$, $\mathcal{L} = \bar{\mathcal{L}}$)
Let $A \in \mathcal{L}$, $\exists M_A$ semidecides A

A_{TM} : "M" "w"

R : "M^x"

M^x = on input x
1. run M on w
2. if M accepts w

3. run M_A on x
4. if M_A accepts x
5. accept x
6. if M_A rejects x
7. reject x
8. if M rejects w
9. reject x

$L(M^*)$ { if M does not accept w, $L(M^*) = \emptyset \notin \mathcal{L}$
if M accepts w, $L(M^*) = L(M_A) = A \in \mathcal{L}$

如果可判定 R，那也可以判定 A_{TM}，但 A_{TM} 不可判定.

矛盾.

Week 10

1. Given a TM M, does $L(M)$ have property P?

equiv $\mathcal{L}(P)$ = the set of all r.e language satisfying P

Given a TM M, is $L(M) \in \mathcal{L}(P)$?

if non-empty, proper subset then undecidable

DFA		PDA		TM	
A_{DFA}	decidable	A_{PDA}	decidable	A_{TM}	undecidable
E_{DFA}		E_{PDA}		E_{TM}	
EQ_{DFA}		ALL_{PDA}	undecidable	EQ_{TM}	
ALL_{DFA}				HT_M	
				ALL_{TM}	

$ALL_{PDA} = \{ \text{"P"} : P \text{ is a PDA with } L(P) = \Sigma^* \}$ is not recursive

$\Updownarrow$

$NOTALL_{PDA} = \{ \text{"P"} : P \text{ is a PDA with } L(P) \neq \Sigma^* \}$ is not recursive

H_{TM} "M" "w"

$NOTALL_{PDA}$ "P"

We want M halts on w if and only if $L(P) \neq \Sigma^*$

$\Downarrow$

$(s, \Delta \underline{u}w) \vdash_M \cdots \vdash_M (h, \Delta u \underline{a} v)$

等价地表示成：$\Delta usw \vdash_M \cdots \vdash_M \Delta uahv$ → state 放在读写头之后

即 $\Delta usw \# \cdots \# \cdots \# \cdots \# \Delta uahv$ 于是相连出现一个 string

↳ computing history of M that halts on w

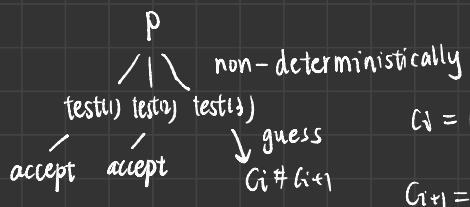
P accepts all the strings that are not computing history of M that halts on w.

$C_1 \# C_2 \cdots \# C_k$ C_i is a configuration

(1) C_1 is not in the form of $\square u s w$

(2) C_k is not in the form of $\square u a h v$

(3) $\exists i$, $C_i \nvdash C_{i+1}$



$C_i = \square a_1 a_2 \cdots \boxed{a_j q a_{j+1}} \cdots a_n$

移动一步，只有这里的部分可能发生变化

$C_{i+1} = \square a_1 a_2 \cdots a_{j+1} \boxed{\quad} a_{j+2} \cdots a_n$

只有三个字母，可以建立恰好的 rule 来判断是否符合条件

判断框外的部分一样：用 PDA 的 stack

细节：考虑到入栈和出栈顺序，我们可以将偶数次的串反转

即 $C_{2k}' = C_{2k}^R$

IF NOTALL PDA is recursive, then H_{TM} is recursive. Contradiction

2. Reduction

Let A and B be two languages over Σ. A reduction from A to B is a computable function $f: \Sigma^* \to \Sigma^*$, such that for any $x \in \Sigma^*$

$x \in A$ if and only if $f(x) \in B$

Lemma

Suppose that $\exists$ reduction from A to B

$A \leq B$

(1) if B is recursive, so is A.

B is recursive $\Rightarrow \exists M_B$ decides B

$\exists f: \Sigma \to \Sigma^*$ such that

$x \in A$ if and only if $f(x) \in B$

M_A = on input x

1. compute $f(x)$
2. run M_B on $f(x)$
3. return the result of B

(2) if A is not recursive, nor is B

Week 10

1. a TM M prints "M" on its tape

A→B | "B" "A"

A: write "B" to the tape

B: write "A" to the tape and swap it with "B"

function $q: \Sigma^* \to \Sigma^*$ such that $q(w) =$ "M_w"

Given any string w

1. construct M_w
2. return "M_w"

↳ M_w = on any input

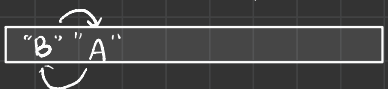
1. print w on the tape

B = on input w

1. compute q(w)
2. write it to the tape and swap it with w

A = on any input

1. write "B" to the tape



get "A" "B"

2. Recursive Theorem

For any TM T, there is a TM R such that for any string w, the computation of T on "R" w is equivalent to that R on w.

T = on input "M" w

do something

R = on input w

1. run T on "R" w

如何写如下代码：

M = on input x

1. obtain its own coding "M"

Legal

T

R

A→B→T

A writes "B" "T"

B writes "A" and reorder the content of tape

3. $L(R)=\{\langle R\rangle\}$

R = on input x
1. obtain its own coding $\langle R\rangle$
2. if $\langle R\rangle == x$
3. accept
4. else
5. reject

T = on input $\langle M\rangle x$
1. if $\langle M\rangle == x$
2. accept
3. else
4. reject

4. A_{TM} is not recursive.

$\exists$ TM M_A decides A_{TM}

M^* = on input x
1. obtain $\langle M^*\rangle$
2. run M_A on $\langle M^*\rangle x$
3. if M_A accepts $\langle M^*\rangle x$
4. rejects x
5. else
6. accepts x

M_A的判断总是错误的，因此不存在这样的

那么A_{TM}不可判定

5. 哥德尔不完备性定理：

$$V(x,t)=\begin{cases}1 & \text{if } t \text{ is a valid for } x\\ 0 & \text{otherwise}\end{cases}$$

a statement x

a proof t

let Y be a language (the set of true statement)

A proof system for Y is a TM V such that:

1) effectiveness, for $x,y\in\Sigma^*$, V either accepts or rejects (x,y)

2) soundness, for $x\notin Y$, $V(x,y)=0$ for all y

V is complete if for any $x\in Y$, there is some y such that $V(x,y)=1$ ↗ provable

Theorem:

Some language Y does not have a complete proof system.

Lemma:

A has a complete proof system if and only if A is recursively enumerable.

$\Rightarrow$) $\exists$ complete proof system V

M = on input x
1. for $y\in\Sigma^x$ in increasing order of length
2. if $V(x,y)=1$
3. accept x

⇐) $\exists M$ semidecides A

V = on input (x, y)

1. run M on x for y steps
2. accepts (x, y) if M accepts x within y steps

① - 会停机，满足 effectiveness ② $x \in A$，对 $\forall y$，x 都不会被接受，满足 soundness

③ complete 显然满足

So $\bar{A}_{TM}$ has no complete proof system

a proof system V for $\bar{A}_{TM}$

R = on input x

1. obtain "R"
2. for $y \in \Sigma^*$ in increasing order of length
3. if $V("R"x, y) = 1$
4. halt and accept x

If V is complete

R accepts $x \iff \exists y$. s.t. $V("R"x, y) = 1 \iff "R"x \in \bar{A}_{TM}$

6. Enumerator

We say a TM M enumerates a language L if for some state q

$L = \{w : (s, \triangleright\sqcup) \vdash_M^* (q, \triangleright\sqcup w)\}$ → output w → output state

↳ Turing enumerable

Theorem:

A is Turing enumerable if and only if A is recursively enumerable.

Proof:

Assume that A is a infinite set

⇒) Obvious

⇐) $\exists M$ semidecides A

for $s \in \Sigma^*$ in increasing order of length

run M on s

if M accepts s, → 可能会出现 looping

accept s

for $i = 1, 2, 3, 4 \cdots$

for $j = 1 \cdots i$

run M on s_j for i steps

if M accepts s_j

output s_j

7. A TM M is minimal if $|"N"| < |"M"|$ implies $L(N) \neq L(M)$

$MIN_{TM} = \{"M": M \text{ is a minimal TM}\}$

R = on input x
1. obtain "R"
2. enumerate MIN_{TM} until obtain "B" with $|"B"| \neq |"R"|$
3. run B on x

We know $L(R) = L(B)$, but $|"R"| < |"B"|$, contradictory with B is minimal.

So we can't enumerate MIN_{TM}, is not r.e.

Week 11

1. Let M be a standard DTM that halts on every input

The running time of M is a function $f: N \to N$ (input length → # steps)

On any input of length n, M halts within $f(n)$ steps.

2. $DTIME(t(n)) = \{L: L \text{ can be decided by some standard DTM within running time } O(t(n))\}$

① $\{0^k 1^k, k \geq 0\}$:
1) 扫第一趟，确定长度
2) 之后每次，对序列0和1，每隔一个删一个。 DTIME(nlogn)
3) 看0和1是否同时删完

if use two-tape, we only need $O(n)$ steps
1) 一个 tape 0　2) 一个 tape 1

② If k-tape TM in $t(n)$ time, then for single tape
one step → $O(t(n))$ steps (At most $2t(n)$ steps)
$t(n)$ step → $O(t(n)^2)$ steps

3. The Cobhum-Edmond Thesis
Any "reasonable" and "general" deterministic model of computation is polynomically related.

4. $P = \{L: L \text{ can be decided by some standard DTM with poly}(n) \text{ running time}\}$

$P = \bigcup_{k \geq 0} DTIME(n^k)$

5. Theorem
Every context-free language is in P.

Proof:
chomsky norm form

$\begin{cases} S \to e \\ A \to BC \\ A \to a \end{cases}$

$|R|^{2n-1}$

Dynamic Programming

$a_1 a_2 \cdots a_n \qquad S \overset{*}{\Rightarrow} a_1 a_2 \cdots a_n$

subproblems

for $1 \le i \le j \le n$ define

$T[i][j] = \{ A \in V-\Sigma : A \overset{*}{\Rightarrow} a_i a_{i+1} \cdots a_j \}$

Goal: $T[1][n]$

Base case: for $1 \le i \le n$, $T[i][i] = \{A \in V-\Sigma : A \Rightarrow a_i\}$

Recurrence

$a_i \cdots a_k \; a_{k+1} \cdots a_j \qquad A \to BC$

($a_i \cdots a_k$: B, $a_{k+1} \cdots a_j$: C)

for $1 \le i \le j \le n$

$T[i][j] = \bigcup_{k=i}^{j-1} \{ A : A \to BC, B \in T[i][k] \wedge C \in T[k+1][j] \}$

subproblems: n^2

cost per subproblem: $n \cdot |R| \cdot (|V|-|\Sigma|)^2$ ← 判断 B, C 是否在对应集合

total: $n^3 \cdot |R| \cdot (|V|-|\Sigma|)^2 \qquad O(n^3)$

6. SAT (satisfiability) Problem

Let $X = \{x_1, \cdots, x_n\}$ be a set of boolean variable

$x_1, \cdots, x_n, \bar{x}_1, \cdots, \bar{x}_n$: literals

a clause is like $x_1 \cup x_2 \cup \bar{x}_3 \cup \bar{x}_4 \cup x_5$

a boolean formula is like $(x_1 \cup x_2 \cup \bar{x}_3) \wedge (x_3 \cup x_4) \wedge (x_1 \cup \bar{x}_4 \cup x_5)$

A truth assignment of X is $T: X \to \{0,1\}$

T satisfies F if F is true under T.

SAT: given a boolean formula F, is F satisfiable?

$\{$"F": F is a satisfiable boolean formula$\}$

SAT $\in$ P? unknown

7. Let M be a NTM. The running time of M is a function $f: N \to N$.

For any input of length n, every branch of M halts within f(n) steps.

M = on input F

1. non-deterministically generate a truth assignment T
2. if T satisfies F
3. accept
4. else
5. reject

8. NP = { L : L can be decided by some NTM within poly(n) running time }

$P \subseteq NP$ $P = NP$? unknown $P \neq NP$ is widely believed

9. Theorem

$P = NP$ if and only if $SAT \in P$

10. language A — polynumial-time verifiable, DTM V 满足这三个条件的证明系统不总存在 但对部分A成立

(1) for any $x \in \Sigma^*$, any $y \in \Sigma^*$

$V(x;y) = \begin{cases} \text{accept} \\ \text{reject} \end{cases}$ effectiveness

and V has polynomial running time

(2) for any $x \notin A$, $V(x;y) = \text{reject}$ for any $y \in \Sigma^*$ soundness

(3) for any $x \in A$, $\exists y \in \Sigma^*$ (with $|y| \leq \text{poly}(|x|)$), $V(x;y) = \text{accept}$ completeness

11. SAT is polynomial-time verifiable.

V = on input F and T

1. evaluate F using T
2. if T satisfies F
3. accept
4. else
5. reject

12. Theorem:

L is polynomial-time verifiable $\iff L \in NP$

Proof:

$\Rightarrow$) $\exists V$ ----------

M = on input x (decide $x \in L$ in poly(|x|) time)

1. non-deterministically generate a proof y with $|y| \leq \text{poly}(|x|)$
2. run V on x and y
3. if V accepts x and y
4. accept x
5. else
6. reject

$\Leftarrow$) $\exists$ NTM M decides L in poly(n) time

V = on input x and y

1. run M on x following y deterministically
2. if M accepts x
3. accept x and y
4. else

可以验证V符合要求

5. reject x and y

Week 13

1. Theorem: SAT∈P if and only if P=NP ⇔ SAT is NP-complete

2. Polynomial time reduction

computable ← $f: \Sigma^* \to \Sigma^*$ s.t. ② $x \in A$ iff $f(x) \in B$

① Given x, f(x) can be computed in poly(|x|) time

记作 $A \leq_p B$

(1) Lemma: If $A \leq_p B$ and $B \in P$, then $A \in P$

Proof: $x \in A$? → $f(x)$ → $f(x) \in B$?

time: poly(|x|) poly(|f(x)|) time

|f(x)| = poly(|x|) ↓

poly(|f(x)|) = poly(poly(|x|)) = poly(|x|)

(2) A is NP-complete if ① $A \in NP$ ② $\forall B \in NP$, $B \leq_p A$

3. Cook-Levin Theorem: SAT is NP-complete. $x \xrightarrow{f}$ F ← boolean formula

Proof: Let $B \in NP$, then we need to prove $B \leq_p SAT$ $x \in B$ iff F is satisfiable

Cause $B \in NP$, ∃NTM N decides B in $poly(n) \to n^k$ time

$a_1 a_2 \dots a_n \in B \Leftrightarrow (s, \sqcup a_1 a_2 \dots a_n) \vdash_N (\cdots) \vdash_N \cdots \vdash_N (y, \sqcup u \sqcup v)$

用来标标记读写头位置 $\leq n^k$ configurations of length n^k

$\Leftrightarrow \exists \sqcup s a_1 a_2 \dots a_n \vdash_N \cdots \vdash_N u \sqcup a y v \Leftrightarrow F = (x_1 \vee x_2 \vee x_3) \wedge (\cdots) \wedge (\ ) \cdots$

$\Leftrightarrow$ 

每行对应一个 configuration

for $1 \leq i \leq n^k$, $1 \leq j \leq n^k$, $c \in k \cup \Sigma$

x_{ijc} → 第 i 行 j 列是否是 c

① every cell contains a character $c \in k \cup \Sigma$

$\sum_{c \in k \cup \Sigma} x_{ijc} = 1$ ⇐ $\bigwedge_{c, c' \in k \cup \Sigma, c \neq c'} \overline{x_{ijc} \wedge x_{ijc'}}$

$\Leftrightarrow \begin{cases} \sum_{c \in k \cup \Sigma} x_{ijc} \geq 1 \\ \sum_{c \in k \cup \Sigma} x_{ijc} \leq 1 \end{cases}$

② first raw contains initial configurations

$x_{11 \sqcup} \wedge x_{12 \sqcup} \wedge x_{13 s} \wedge x_{14 a_1} \wedge \cdots \wedge$

③ some cell contains y

$\sum_{i,j} x_{ijy} \geq 1 \Leftrightarrow \bigvee_{i,j} x_{ijy}$

④ i $i_1 \cdots a b_1 \cdots$ b_{k-1} b_k b_{k+1} b_{k+2} $b_{k+3} \cdots$ $k \times (2+|\Sigma|)$

$i+1$: $\sqcup b_1 \cdots b_{k-1}$ $\cdots$ b_{k+2} $b_{k+3} \cdots$

b_i	b_{i+1}	b_{i+2}	x_{ij}
c_i	c_{i+1}	c_{i+2}	$x_{i+1,j}$

$(\bigvee_{c \in k \cup \Sigma} (x_{ijc} \wedge x_{(i+1)je})) \wedge (j-1) \wedge (j+2)$

b_{k-2}	b_{k-1}	b_k	q
b_{k-2}	b_{k-1}	?	

$(\ )_{k-2} \wedge (\ )_{k-1} \wedge (x_{ijc} \vee x_{ijc'} \cdots \vee)$

↓ 背后 rule，如 (q, b_k, p, c)，则？可能是 c

b_{k-1}	b_k	q
b_{k-1}	?	?

b_{k-1}	q	b_k
?	?	?

..... 类似、

因此，存在上述的 table 可以用一个 bool 函数表示

同时我们验证的过程也是多项式时间的

这样我们就完成了证明

4. Lemma: Let A be a NP-complete. Let B be language. If

(1) $B \in NP$ (2) $A \leq_p B$

then B is NP-complete.

Proof: If $C \leq_p A$ and $A \leq_p B$, then $C \leq_p B$

$\Rightarrow \forall C \in NP$, $C \leq_p A$ + (2)

$\Rightarrow \forall C \in NP$, $C \leq_p B$ + (1)

$\Rightarrow$ B is NP-complete ▨

5. 3SAT: $(x_1 \vee x_2 \vee \bar{x}_4) \wedge (\cdots \vee \cdots \vee \cdots) \wedge (\cdots \vee \cdots \vee \cdots) \cdots$

each clause has three literals 不能重复

① 3SAT $\in$ NP, simple.

② SAT $\leq_p$ 3SAT SAT 3SAT

( ) ∧ ( ) ∧ ( ) → (∨ ∨) ∧ ⋯ ∧ (∨ ∨)

$x_1 \rightarrow (x_1 \vee y) \wedge (x_1 \vee \bar{y})$ 1→3 记住长度 等数

$x_1 \vee x_2 \rightarrow (x_1 \vee x_2 \vee y) \wedge (x_1 \vee x_2 \vee \bar{y})$

$x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5 \rightarrow (x_1 \vee x_2 \vee y) \wedge (\bar{y} \vee x_3 \vee x_4 \vee x_5)$

k项 → k-3次，生成 k-2 个 ∧，即为 k-2 项，多项式

▨

6. Clique G=(V, E) 团：点点两两之间都有边

{b, c, d} ∨ {a, b, c} ×

Clique = {"G" "k": G has a clique with at least k nodes}

① Clique $\in$ NP 略

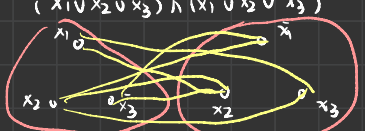
② 3SAT $\leq_p$ Clique 3SAT Clique

$(x_1 \vee x_2 \vee \bar{x}_3) \wedge (\cdots \vee \cdots \vee \cdots) \cdots \wedge (\quad)$ G, k

m clauses m groups of nodes

3m nodes $\leq$ $9m^2$ edges

$(x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3)$



我们只在不同组之间连边，连接所有不冲突的点，

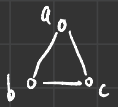
冲突：x_i 与 $\bar{x}_i$ 这种情况

F is satisfiable ⟺ G has a clique at least m nodes

↳ 至少存在m个互不冲突的点 → m个点的团

可满足 ← 直接让这m个点对应变量为真

7. Vertex Cover $G=(V,E)$ 一组顶点，使得所有边至少有一个顶点在其中



{a,b} √ {a} ×

VC = {"G" "k" : G has a vertex cover with at most k nodes}

① VC ∈ NP 略

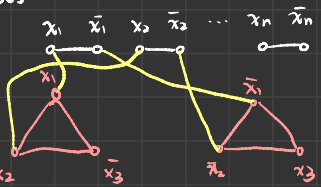
② 3SAT ≤p VC

3SAT: F=( )∧( )∧( )

n variables

m clauses

VC: G k



① 2n nodes

② 3m nodes

m groups

③ 在不同颜色相同名顶点连一条边

覆盖白点和红点，至少需要 n+2m nodes

可以证明：F is satisfiable ⟺ G has a vertex cover with at most k=n+2m nodes

↳ 每个gmup，变量为真选白点，为假选红点

Week 14

1. Let M be a standard DTM. M runs in f(n) space if for any input of length n, M uses at most f(n) tape cells.

(1) PSPACE = {A: A can be decided by some DTM that runs in poly(n) space}

DTM

time f(n) ⇒ space O(f(n))

$2^{O(f(n))}$ ⇐ f(n) always halt

(q, D, ···) configuration

$|K| \cdot f(n) \cdot |\Sigma|^{f(n)} = 2^{O(f(n))}$

↓ 读写头位置

↓ 由于保证停机，因此运算过程不会出现重复的 configuration

So, $P \subseteq PSPACE \subseteq Exp$

$Exp = \{A: A \text{ can be decided by some DTM within } 2^{poly(n)} \text{ time}\}$

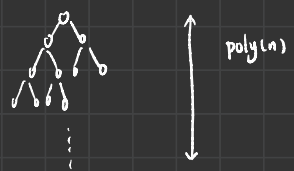
2. Let N be a standard NTM, N runs in $f(n)$ space if for any input of length n, every branch uses at most $f(n)$ tape cells.

$NPSPACE = \{A : A$ can be decided by some NTM in $poly(n)$ space$\}$

$NP \subseteq NPSPACE$ $\quad$ $PSPACE \subseteq NPSPACE$

(1) **$NP \subseteq PSPACE$**

$A \subseteq NP \Rightarrow \exists$ NTM decides A in $poly(n)$ time



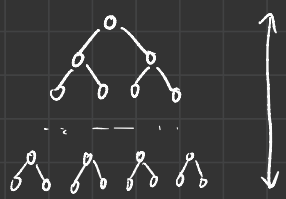
→ 分支数量是有限的

$\Rightarrow \exists$ DTM M. $poly(n) + poly(n) = poly(n)$

(2) **$PSPACE = NPSPACE$**

$A \in NPSPACE$

$\Rightarrow \exists$ NTM N decides A in $poly(n)$ space



DTM 模拟 NTM 还需要存每次的选择

$\Rightarrow$ DTM M decides A in $poly(n)$ space

N has $2^{poly(n)}$ distinct configurations

accept : $\exists C_{initial} \longrightarrow C_{accept}$ in $2^{poly(n)}$ time and $poly(n)$ space

$\Longleftrightarrow \exists C'$, $C_{initial} \longrightarrow C'$ in $2^{poly(n)-1}$ time and $poly(n)$ space

$C' \longrightarrow C_{accept}$ in $2^{poly(n)-1}$ time and $poly(n)$ space

M = on input C_1, C_2, t (N $C_1 \to C_2$ in time t)

```
1. if t=1
2.     if C1=C2 or C1 ⊢N C2
3.         accept
4.     else
5.         reject
6. for each c' using at most poly(n) space
7.     run M on C1, c', t/2
8.     run M on c', C2, t/2
```

9. if both accepts
10. accept
11. reject

S(t) represents the space with input t.

$$\begin{cases} S(1) = poly(n) \\ S(t) = S(t/2) + poly(n) \end{cases} \Rightarrow S(t) = poly(n) \cdot \log t$$

空间可以复用

run M on Cinitial, Caccept, $2^{poly(n)}$

$\Rightarrow poly(n) \cdot poly(n) = poly(n)$ ▨

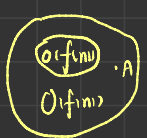
(3) $P \subseteq NP \subseteq PSPACE \in EXP$ We can prove $P \subsetneq EXP$

NPSPACE

3. Hierarchy Theorem

(1) Space: for any $f: \mathbb{N} \to \mathbb{N}$ satisfying certain technical conditions (f(n) can be computed in O(f(n)) space)

there is a language A such that



(1) A can be decided by some DTM in O(f(n)) space

(2) A cannot be decided by any DTM in o(f(n)) space

construct DTM D

(1) D decides some language A in O(f(n)) space.

(2) for any DTM M that runs in o(f(n)) space,

D and M differ on at least one input.

	"M_1"	"M_2"	"M_3"
M_1	+		
M_2		−	
M_3			0
⋮			
D	−	+	+/−

o(f(n)) + 1

D = on input "M"

1. let n = |"M"|
2. compute f(n) (f(n) can be computed in O(f(n)) space)
3. run M on "M"
4. If M accepts "M" within $c^{f(n)}$ steps and o(f(n)) cells
5. reject
6. else
7. accept

D runs in O(f(n)) space and D decides some language.

And we easily know D satisfies (2)

(2) Time: For any $f: \mathbb{N} \to \mathbb{N}$ satisfying f(n) can be computed in O(f(n)) time

there is a language A:

(1) A can be decided by some DTM in $O(f(n))$ time

(2) A cannot be decided by any DTM in $o\left(\frac{f(n)}{\log f(n)}\right)$ time

D = on input "M"

1. let $n = |\text{"M"}|$
2. compute $f(n)$
3. run M on "M" for $f(n)/\log f(n)$ steps ↗ 需要时间更新 counter
4. If M accepts "M" within $f(n)/\log f(n)$ steps
5. reject
6. else
7. accept

Week 15

1. $F(x) = \prod_{i=1}^{d}(x - c_i)$ $\quad G(x) = \sum_{i=0}^{d} c_i x^i$

We need to judge whether $G(x) = F(x)$

直接比: $O\left(\sum_{i=1}^{d} i\right) = O(d^2)$

随机, sample $r \in \{1, \dots, 100d\}$

if $F(r) = G(r)$

yes $\quad O(d)$ time

else

no

"Fail" if and only if $F(x) \neq G_1(x)$ but $F(r) = G(r)$

$\Pr[F(r) = G(r) \mid F(x) \neq G_1(x)]$ $\quad$ define $H(x) = F(x) - G_1(x)$

$= \Pr[H(r) = 0 \mid H(x) = \sum_{i=0}^{d} c_i' x^i$ and $\exists c_i' \neq 0]$

$\leq \frac{1}{100}$ ($H(x)$ 至多 d 个根) $\quad$ in $\overline{RP}$

if $F(x) = G_1(x)$, always correct

if $F(x) \neq G_1(x)$, "fail" probability $\leq \frac{1}{100}$ ↗ 降低错误率: ↑ sample 数 / 重复此算法

Algorithm B:

run A for k times

if $F(x) \neq G_1(x)$ in some execution of A

no

else

yes

"Fail" probability $\leq \left(\frac{1}{100}\right)^k \leq \frac{1}{d}$ $\quad$ if $k = \log d$

2. Given three $n \times n$ matrices A, B, C, whether $A \cdot B = C$

不用随机 $O(n^{2+\delta})$

sample $(r_1, r_2, \cdots r_n) \in \{0,1\}^n$

If $A(B\vec{r}) == C\vec{r}$

yes $O(n^2)$ time

else

no

"fail" probability

$\Pr(AB\vec{r} = C\vec{r} \mid AB \neq C)$ $D = AB - C$ $\begin{pmatrix} d_{11} & d_{12} \cdots & d_{1n} \\ \vdots & & \vdots \\ d_{n1} & d_{n2} \cdots & d_{nn} \end{pmatrix}$

$= \Pr(D\vec{r} = 0 \mid D \neq \vec{0})$ Assume $d_{11} \neq 0$

$\Pr(d_{11}r_1 + d_{12}r_2 + \cdots + d_{1n}r_n = 0 \mid r_2, \cdots, r_n) \leq \frac{1}{2}$ deferred decisions

$\Pr(d_{11}r_1 + d_{12}r_2 + \cdots + d_{1n}r_n = 0)$

$= \sum_{(r_2 \cdots r_n)} \Pr(d_{11}r_1 + d_{12}r_2 + \cdots + d_{1n}r_n = 0 \mid r_2 \cdots r_n) \cdot \Pr(r_2 \cdots r_n)$

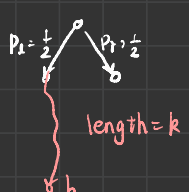
$\leq \frac{1}{2} \cdot \sum_{(r_2 \cdots r_n)} \Pr(r_2 \cdots r_n) \leq \frac{1}{2}$

repeat $\log n$ time $\Rightarrow$ fail probability $\to \frac{1}{n}$

3. Probabilistic Turing Machine

Non-determinstic TM

每个状态可选择的至多只有两个，并以等概率选择任一个状态



$\Pr[b] = 2^{-k}$

$\Pr[M \text{ accepts } w] = \sum_{b \text{ accepts } w} \Pr[b]$

M decides a language A with error probability δ if

(1) for any $w \in A$, $\Pr[M \text{ accepts } w] \geq 1 - \delta$

(2) for any $w \notin A$, $\Pr[M \text{ rejects } w] \geq 1 - \delta$

BPP: the class of languages that can be decided by probalistic polynomial-time TM with error probability at most $1/3$ → can be replaced by any $c < \frac{1}{2}$

多次run取可

Amplication Lemma

Let δ be a constant with $0 < \delta < 1/2$, Let M be a probability poly-time TM

with err prob δ. Then for any poly $p(n)$, there is a prob poly-time TM M' with error probability $2^{-p(n)}$

M'= on input x

1. run M on x independently for $2k$ times
2. If most runs of M accepts x
   accept
3. Else
   reject

$2k$: w wrong, c correct

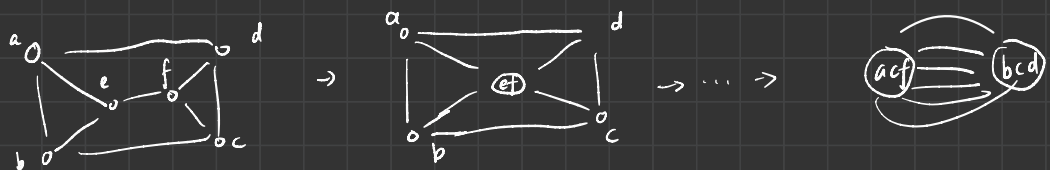
$$\Pr[M' \text{ fails}] = \Pr[M' \text{ fails for } w \text{ times for } w>k]$$
$$= \sum_{w=k+1}^{2k} \Pr[M' \text{ fails for } w \text{ times}]$$
$$= \sum_{w=k+1}^{2k} C_{2k}^{w} \delta^{w}(1-\delta)^{2k-w} \qquad \delta < \tfrac{1}{2},\ w>k$$
$$\le \delta^{k}(1-\delta)^{k}$$
$$\le \delta^{k}(1-\delta)^{k} \cdot \sum_{w=k+1}^{2k} C_{2k}^{w}$$
$$\le \delta^{k}(1-\delta)^{k}\, 2^{2k} \le (4\delta(1-\delta))^{k} < 2^{-p(n)}$$

$$k \cdot \log 4\delta(1-\delta) < -p(n)$$
$$k > \frac{-p(n)}{\log 4\delta(1-\delta)}$$

So $P \subseteq BPP$ $\quad P=BPP$? Unknown $\quad \exists A \in BPP, A \notin P$? Unknown

4. Min-cut 使图不连通，最少需要删多少边

undirected graph, $G=(V,E)$ 假设最小割边集 C^*



合并到只剩下两个顶点，这时剩下的边就是一个割边集（注意到会有重边）

如果每一次合并的边都不在 C^* 中，那么最后剩的就是 C^*

Lemma

If the min-cut of G has at least k edges, then G has at least $\frac{kn}{2}$ edges.

Proof

一个顶点连着的所有边必然构成割边集，则由 min-cut $\ge k$，每个点度数也大于 k

因此一共至少 $\frac{kn}{2}$ 边. ☒

Assume $G_0 = G \to G_1 \to G_2 \to \cdots \to G_{n-2}$

$\Pr[C^* \le E(G_{n-2})]$

$\Pr[C^* \le E(G_1)] \ge 1 - \frac{k}{kn/2} = \frac{n-2}{n}$

$\Pr[C^* \le E(G_2)] = \Pr[C^* \le E(G_2) \mid C^* \le E(G_1)] \cdot \Pr[C^* \le E(G_1)]$ $|E[G_1]| \ge \frac{k(n-1)}{2}$

$\ge \left(1 - \frac{k}{k(n-1)/2}\right) \cdot \frac{n-2}{n} = \frac{n-3}{n-1} \cdot \frac{n-2}{n}$

So,

$\Pr[C^* \le E(G_{n-2})] \ge \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \cdots \cdot \frac{1}{3} = \frac{2}{n(n-1)}$

repeat $n(n-1)\ln n$ times, return best

fail: $\left(1 - \frac{2}{n(n-1)}\right)^{n(n-1)\ln n} \le e^{-2\ln n} \le \frac{1}{n^2}$

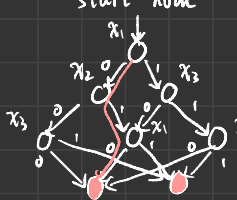
5. another error 单边 对应复杂类: RP

(1) $w \in A$, $\Pr[M \text{ accepts } w] \ge 1 - \delta$

(2) $w \notin A$, $\Pr[M \text{ rejects } w] = 1$

Week 16

1. Read Once Branching Program



query node: black, out-degree = 2

output node: red, out-degree = 0

given $(x_1, x_2, x_3) = (0, 1, 0)$ 根据变量取值走对应边

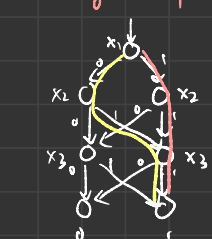
$B(0,1,0) = 0$ $B(1,1,1) = 1$

Read-Once: 到output node的路上没有重复的变量

$B_1 \equiv B_2$ if $x_1, \cdots x_n$, $\forall$ assignment $(x_1 = y_1, x_2 = y_2, \cdots x_n = y_n)$

$B_1(y_1 \cdots y_n) = B_2(y_1, \cdots y_n)$

$EQ_{robp} = \{(B_1, B_2): B_1 \text{ and } B_2 \text{ are read once branching program, such that } B_1 \equiv B_2\}$



$F = (x_1 \wedge x_2 \wedge x_3) \vee (\bar{x_1} \wedge x_2 \wedge x_3) \vee (x_1 \wedge \bar{x_2} \wedge x_3) \vee (\bar{x_1} \wedge \bar{x_2} \wedge x_3)$

$B(1,1,1) = F(1,1,1) = 1$

1. randomly sample an assignment of $x_1, \cdots, x_n$
2. if $B_1(y_1, \cdots y_n) = B_2(y_1, \cdots y_n)$
   return $B_1 == B_2$
3. else
   return $B_1 \neq B_2$

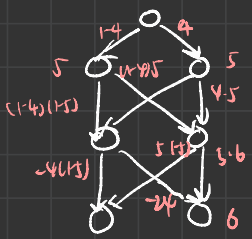
$\Pr[B_1(y_1, \cdots y_n) \neq B_2(y_1, \cdots y_n) = B_1 \neq B_2] = \frac{1}{2^n}$

read-once branching program:	B_1	B_2	$x_1 \vee x_2$	$x_1 + x_2$
boolean function:	F_1	F_2	$x_1 \wedge x_2$	$x_1 x_2$
polynomial	P_1	P_2	$x_1 \wedge \overline{x_2}$	$x_1(1-x_2)$

judge two polynomial with n variables

$P_1(x_1, \cdots x_n)$ of degree d → path长度 $P_2(x_1, \cdots, x_n)$ of degree d

$P(4,5,6) = 6$ $P(4,5,6) = 6$



1. randomly sample $x_1, \cdots x_n$ from field F of k elements
2. If $P_1(x_1, \cdots x_n) = P_2(x_1, \cdots x_n)$
   return $B_1 \equiv B_2$
3. else
   return $B_1 \not\equiv B_2$

$Z_p = \{0, 1, \cdots, p-1\}$, p is a prime

$2 + p - 1 \equiv 1 \pmod p$

$2 \times (p-1) \equiv p-2 \pmod p$

$P_1 \not\equiv P_2$ $P_1(x_1, x_2, \cdots x_n) = P_2(x_1, x_2, \cdots x_n)$

$P^* = P_1 - P_2$ $P^*(x_1, \cdots x_n) = 0$

Lemma

$Pr[P^*(x_1, x_2 \cdots x_n)] \leq \frac{nd}{p}$

Proof

If n=1, trivially holds

Assume $n \leq k$, ····· holds

when $n = k+1$

$P^*(x_1, x_2, \cdots, x_{k+1}) \Rightarrow P'(x_{k+1})$

assume 前k个sample好了

$P'(x_{k+1}) = p_0 + p_1 x_{k+1} + p_2 x_{k+1}^2 + \cdots + p_d x_{k+1}^d$

$p_0, p_1, \cdots p_d \xleftarrow{\text{decide}} x_1, x_2, \cdots x_k$

case 1: $p_0 = p_1 = \cdots = p_d = 0$ $Pr[\ ] \leq \frac{kd}{p}$

case 2: $\exists p_i \neq 0$, $Pr[P'(x_{k+1}) = 0] \leq \frac{d}{p}$

$Pr[\text{failure}] \leq \frac{kd}{p} + \frac{d}{p} = \frac{(k+1)d}{p}$ ▨

如何等概率地 sample:

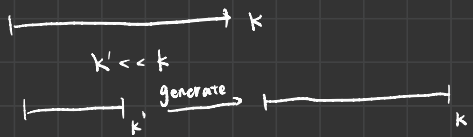
$S = \{1, 2, \cdots m\}$ $m = 2^n$ $b_1 b_2 \cdots b_{\log m}$ 每位都可以抽硬币决定0或1

$p > \frac{1}{2}$ · $\lceil \log m \rceil$ $b_1 b_2 \cdots b_{\lceil \log m \rceil}$ $\Pr[i \text{ is sampled}] = \frac{p}{m}$ for every $i \in S$

$D(I, r)$ If $|r| \le \log |I|$ 按着 n 次 去随机

random tape

BPP = P → 是否可以通过某种算法降低随机化程度



P NP EXP BPP

$P \subseteq BPP \subseteq EXP$

Conjecture

$P \neq NP$ $P = BPP$

Lemma

If $P = NP$, then $BPP = P$