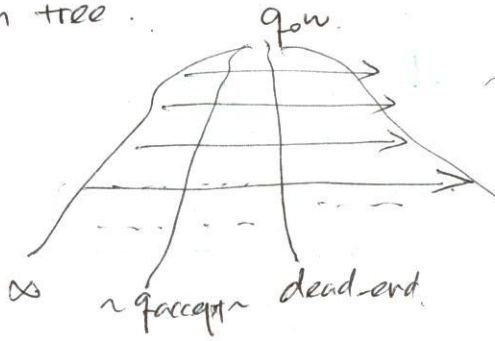
HW. bcabc. Give the computation tree on this

Nondeterminism can be used to make choice in search process.

A DTM can be regarded as a special case of NTM where $|\delta(q_i, a)| \le 1$ for all q_i, a. The computation tree on any w has just one branch.
$\delta(q_i, a) = \{\}$ corresponds to a transition to q_{reject}

Every NTM can be simulated by a DTM. (模擬 DTM computation tree 3 config)

(Idea only.) The simulating DTM performs a breath-first search of the computation tree.



If it finds an accepting branch, it accepts input w.
If it finds all branches dead-end, it rejects. (" ").
O.w., the tree has no accepting branch and has at least one infinite branch — the breadth-first search runs forever.
and w is neither accepted nor rejected

⚠ Theorem: Let p be any transition sequence of a decider DTM or any branch of any computation tree of a decider NTM. Then all configurations of p are distinct.

Proof. Suppose p has a repeated configuration α. This means p extends infinitely by repeating the segment p_α, forming an infinite path, a contradiction to the assumption that the given machine is a decider.

p_α 根據兩個的定義.

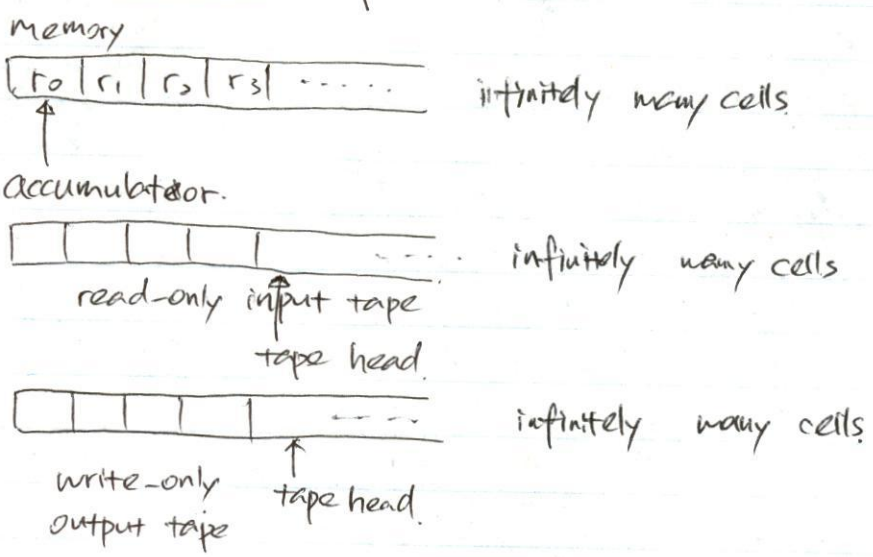
⇒ This theorem can be used to establish an upper bound on the length of p.

9/8

Two reasons Turing machine are hard to program

- No concept of memory address — a program needs to rely on tape symbol patterns on tape to access the desired cell
- No concept of instructions — operations must be done by δ

网页中 Random Access Machines 的解释



知道区别！

Load =5	$r_0 \leftarrow 5$
Load 5	$r_0 \leftarrow r_5$
Add =5	$r_0 \leftarrow r_0 + 5$
Add 5	$r_0 \leftarrow r_0 + r_5$

store 总是从 r0 来，先从 r0 load 再 store

Simulation of DTMs by RAMs

Let the given DTM be $(Q, \Sigma, T, \delta, q_1, q_{accept}, q_{reject})$

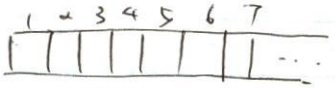
where $Q = \{q_1, \dots, q_n\}$ $T = \{a_0, a_1, \dots a_m\}$, $a_0 = \sqcup$

Represent

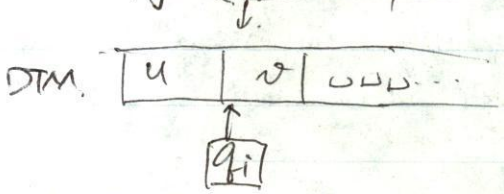
- $q_1, \dots q_n$ by $1, \dots n$
- $a_0, \dots a_m$ by $0, \dots, m$ $\sqcup$ is represent (encoded by 0)

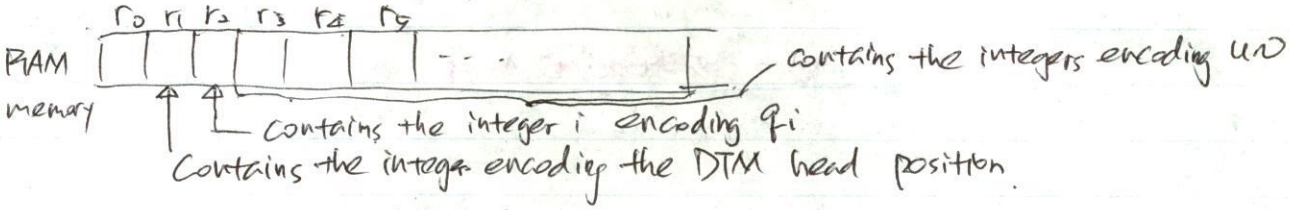
$\sqcup$

assign virtual index 1, 2, 3, ... to the DTM tape cells



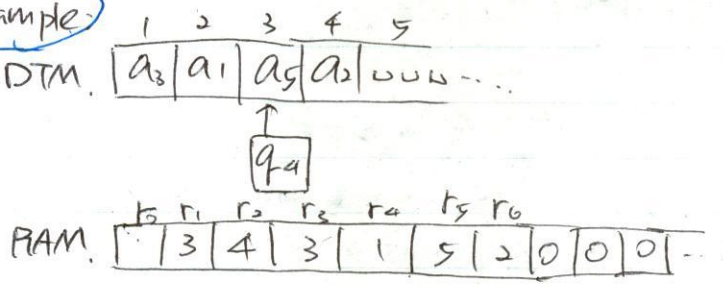
Simulate a DTM configuration $u q_i v$ by the following RAM memory configuration:





Example:

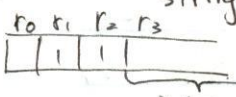
(virtual index)



Simulate δ by a RAM program implementing the following algorithm

Let $\delta(q_i, a_j) = (q_i', a_j', L/R)$

Set the RAM Memory to the configuration representing $q_1 w$ // w is the input string to DTM.



while (state ≠ q_{accept} and state ≠ q_{reject})

{ if (state = q_1 and tape symbol read = a_0) change state to q_1' and tape symbol to a_0' and move the head to L or R.

else if (state = q_1 and ——— = a_i) ——— a_1'

else if (state = q_1 and ——— = a_m) ——— a_m'

else if (state = q_2 and ——— = a_0) ——— q_2' ——— a_0'

⋮ ⋮ ⋮

else if (... q_2 ——— a_m) ——— q_2' ——— a_m'

⋮

enumerate all combinations of $\{q_1, \cdots, q_n\}$ and $\{a_0, \cdots a_m\}$ except q_{accept} and q_{reject}.

}

ideas for code above.

The current control state is in r_2

The current tape symbol being read is in r_{r_1+3}.

- move tape head to P_i" implemented by $r_1 \leftarrow r_1 + 1$
  L $r_1 \leftarrow r_1 - 1$ if $r_1 \geq 2$
  do nothing if $r_1 = 1$

Simulation of RAM by 5-tape DTMs

(We saw how to simulate any 5-tape DTMs by 1-tape DTMs).

Concept:

Representation of integers. Any radix notation will do, but let's use unary notation.

integer	unary notation
-3	-000
-2	-00
-1	-0
0	ϵ
1	0
2	00

$0^k = \overbrace{0 \cdots 0}^{k}$ to represent $k > 0$

ϵ, the empty string to represent the integer 0

$-0^k = - \overbrace{0 \cdots 0}^{k}$ to represent $-k < 0$

Tape 1 simulates the part of the RAM memory cells that have ever been accessed by the program by a linear list of (index, value) pairs. Example:

r_5 containing 3. — represented by (00000, 000)

Let $r_{k_1}, \cdots r_{k_m}$ be the memory cells that have been accessed by the RAM program, $k_1, \cdots k_m$ need not be stored.

Represent r_{k_j} by the pair (i_{k_j}, C_{k_j}) where i_{k_j} is the unary encoding of k_j, C_{k_j} is the unary encoding of the content of r_{k_j}

DTM tape.

#	#	i_{k_1}	#	C_{k_1}	#	#	i_{k_2}	#	C_{k_2}	#	#	...	#	#	i_{k_m}	#	C_{k_m}	#	#	␣ ␣ ...

Tape 2 simulates r_0

unary encoding of the integer contained in r_0.

Tape 3 simulates the input tape of RAM

unary encoding of x_1	#	unary encoding of x_2	#	...	#	unary ... x_k	#	␣ ␣ ...

Tape 4 simulates the output tape in unary encoding

Tape 5 is used for scratch work.

9/13

Simulation of RAM by 5-tape DTMs

Tape 1 simulates the memory $r_1, r_2 \cdots$

#	#	i_{k_1}	#	c_{k_1}	#	#	- - -	#	#	i_{k_m}	#	c_{k_m}	#	#	⊔ ⊔ ⋯

r_{k_j} represented by (i_{k_j}, c_{k_j})

tape 2 simulates r_0

Simulation of instructions

Let $I_1, I_2, \cdots I_m$ be a RAM instruction stream.

q_1 $\mathcal{L}_1$: I_1
q_2 $\mathcal{L}_2$: I_2
q_3 $\mathcal{L}_3$: I_3
⋮ ⋮ ⋮
q_m $\mathcal{L}_m$: I_m

$\mathcal{L}_i$ (program)

Assign a "virtual label" in front of each instruction.

Use control state q_i to represent $\mathcal{L}_i$

Whenever the RAM jumps to label $\mathcal{L}_i$, the simulating DTM goes to control state q_i. We also need additional control states to simulate individual instructions.

Load=n $r_0 \leftarrow n$.

replace the contents of Tape 2 by unary code of n.

Load n $r_0 \leftarrow r_n$.

linear search for entry (i_n, c_n) on Tape 1.
if found, replace the contents of Tape 2 by c_n.
if not found, r_n has never been accessed and has 0, so replace the contents of Tape 2 by ϵ (unary code of 0)

Store n $r_n \leftarrow r_0$

let c_0 be the contents of Tape 2
linear search for entry (i_n, c_n) on Tape 1.
If found, replace c_n by c_0. If $c_0 \neq c_n$, shift the contents of Tape 1 after entry $(i_n c_n)$ to left or right.
If not found, put (i_n, c_0) after the last "##" on Tape 1.

Add =n $r_0 \leftarrow r_0 + n$

add n to the contents of Tape 2.

Tape 2: 000 ⊔⊔⊔

add =5 ↓ add →

000 00000 ⊔⊔⊔ ⊔⊔⊔⊔⊔

Add n. $r_0 \leftarrow r_0 + r_n$

linear search for (i_n, c_n) on Tape 1.

if found, add c_n to the contents of tape 2.

if not found, do noting

Similarly for sub, Mult, Div

Jump L_i transit. to q_i representing L_i

greater than 0 Jgtz L_i if $r_0 > 0$, jump to q_i

Jzero L_i if $r_0 = 0$, " "

---

fact 9. n 指的是 growth rate.

eg: $2^n = 3^{O(n)}$ but

$2^n = o(3^n)$ and

$2^n \neq O(3^n)$

刚预错误 fact 8. $(\log_a n)^k$.

Time Complexity Chapter 7.

Let M be any 1-tape DTM decider (halts on all input strings)

Let $D_n = \{w \in \Sigma^* \mid |w| = n\}$, for all $n \geq 0$.

For each $w \in \Sigma^*$, $[s(w)$ = the length of the transition sequence of M on w, measured by the # of one-step transition $\vdash]$

The worst-time complexity function of M is

$$W_M(n) = \max\{s(w) \mid w \in D_n\}$$

One way to define the expected (average) time complexity function of M.

Let $p(w)$ = probability that an input w is presented to M for computation, where $\sum_{w \in D_n} p(w) = 1$, for each $n \geq 0$

$$E_M(n) = \sum_{w \in D_n} p(w) \cdot s(w)$$

$|\Sigma| = 3$ input symbols

$|\Sigma|^0 = 1$ $\in$

$|\Sigma| = 3$

$|\Sigma|^2 = 3^2 = 9$

$|\Sigma|^3 = 3^3 = 27$

⋮

Example of $p(w)$: equi-probability distribution

For each $w \in D_n$, $p(w) = \frac{1}{|D_n|} = \frac{1}{|\Sigma|^n}$.

A DTM to decide $\{0^k 1^k \mid k \geq 0\}$.

1. Scan the tape from left to right and reject if any 0 is founded immediately after a 1.
   Otherwise the input is $0^{k_1}1^{k_2}$. Now check if $k_1 = k_2$.
   $O(n)$
   $O(n)$ → Scan the head back to left most position
2. Repeat the following while at least one 0 or one 1 remain on the tape:
   Scan the tape from left to right, crossing off leftmost 0 and 1 by X $O(n)$
   0 0 0 0 1 1 1 1 ␣ ␣
   X 0 0 0 X 1 1 1 ␣ ␣
   X X 0 0 X X 1 1 ␣ ␣
   # of repetitions = $O(\frac{n}{2})$ as a single scan crosses off two symbols
   So, step 2 takes $O(n) \times O(\frac{n}{2}) = O(n^2)$
3. If 0's remain on the tape after all 1's have been crossed off or vice versa, reject. O.w. accept. $O(n)$

∴ Grand Total: $O(n) + O(n) + O(n^2) + O(n) = O(n^2)$.

9/15

Example 3.9 | About assignment!

The transition to q_{reject} are omitted in the diagram.
When a transition dead ends implicity transits to q_{reject}.

~~TIME $t(n)$~~ Let $t: \mathbb{N} \to \mathbb{R}^+$

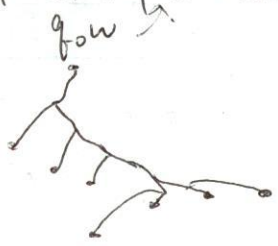
- TIME$(t(n)) = \{L(M) \mid M$ is a 1-tape decider DTM with $W_M(n) = O(t(n))\}$
  Languages (set of subset of string)
  called the deterministic time complexity class of $t(n)$

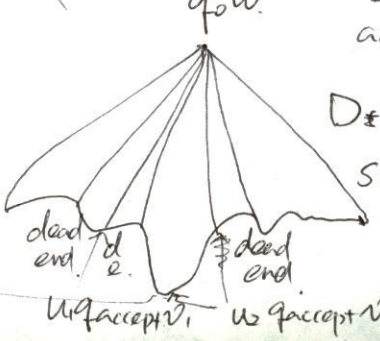
TIME$(n) = \{$ with $W_M(n) = O(n)\}$

TIME$(n^3) = \{$ with $W_M(n) = O(n^3)\}$

Worst-case time complexity functions of 1-tape decider NTMs.

Let M be a 1-tape decider NTM. For each input string w, the computation tree of M on w is a finite tree. (no infinite-length branches, does not contain infinite # of branches)





Each branch halts in an accepting configuration or dead end

$D_{=n} = \{w \in \Sigma^* \mid |w| = n\}, n \geq 0$

$s(w) =$ the length of the longest branch(es) of the computation tree on w

$s(w)$ as measured by the # of one-step transitions

$W_M(n) = \max\{s(w) \mid w \in D_n\}$

$NTIME(t(n)) = \{L(M) \mid M \text{ is a 1 tape NTM with } W_M(n) = O(t(n))\}$

called the nondeterministic time complexity class of $t(n)$.

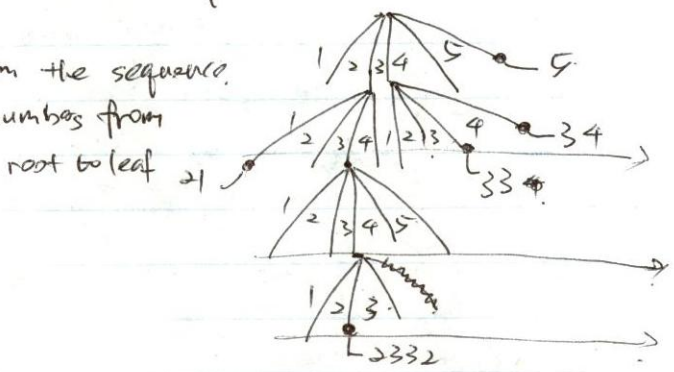
Theorem 3.16 in the book. Every nondeterministic NTM has an equivalent deterministic DTM.

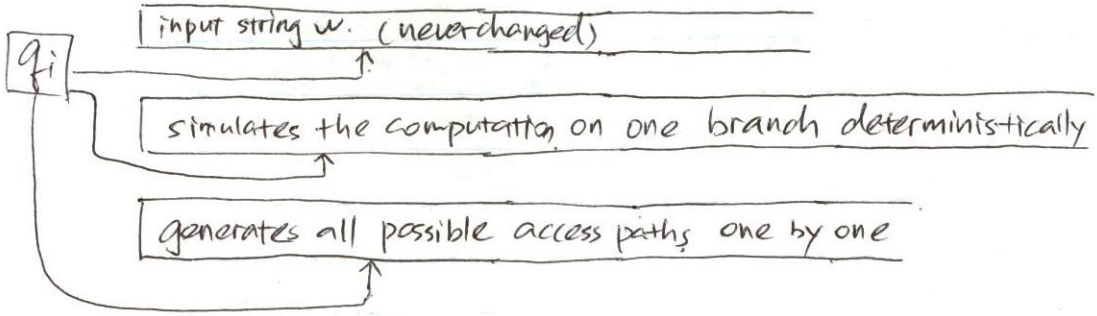
Any 1-tape NTM (M) can be simulating by a 1-tape DTM (M')

First, ~~M' perf~~ construct a 3-tape DTM, M'', performing a breadth-first search of the computation trees of M. Then convert M'' to 1-tape M'.

Use access paths (called "address" in the book) of the computation trees.

Form the sequence of numbers from the root to leaf

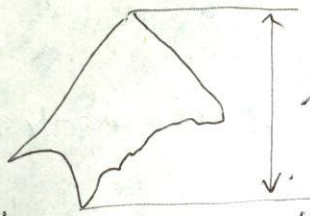




Theorem Let N be any decider NTM with $W_N(n) = t(n) \geq n$, for the NTM. Then there exists a DTM, M, that decides $L(N)$ with $W_M(n) = 2^{O(t(n))}$

Proof As per Theorem 3.16, a 3-tape DTM, M', simulating N is constructed. Recall that, for any configuration $uqav$, $\delta_N(q,a) = \{(q_1, a_1, d_1), \ldots (q_k a_k, d_k)\}$. Let b be the maximum value of k. Then any computation has at most b children. We assume $b \geq 2$. (If $b=1$, N is effectively a DTM, so take the simulation DTM to be a 1-tape DTM that sends all dead-end transitions to rejecting configurations. Then $W_M(n) \leq W_N(n) + 1$

$= 2^{O(W_N(n))}$



So, the theorem trivially holds)

The total # of nodes in the tree ≤ the total # of nodes in the complete tree of height $t(n) = W_M(n) = b^0 + b^1 + \cdots + b^{t(n)} = \frac{b^{t(n)+1} - 1}{b-1}$ (This uses $b \geq 2$)

(nodes n)

↑ has exactly b children

So, $W_{M'}(n) \leq$ (total # of nodes) · (time to visit each node)

$$\leq \frac{b^{t(n)+1} - 1}{b-1} \cdot O(t(n))$$

$$\leq \frac{b^{t(n)+1} - 1}{b-1} \cdot c\,t(n) \text{ for some constant } c$$

$$\leq \frac{c}{b-1} \cdot b^{t(n)+1} \cdot t(n)$$

$$= a \cdot b^{t(n)+1} \cdot t(n), \quad a = \frac{c}{b-1}$$

$$= 2^{\lg(a \cdot b^{t(n)+1} \cdot t(n))}$$

$$= 2^{\lg a + \lg b^{t(n)+1} + \lg t(n)}$$

$$= 2^{(\text{constant})(t(n)+1)(\lg b) + \lg t(n)}$$

$$= 2^{O(t(n))}$$

Now, M' is a 3-tape DTM. Conversion to an equivalent 1-tape DTM, M, at most squares the ~~worst~~ case runtime (Theorem 7.8 in the book,

So, $W_M(n) \leq (2^{O(t(n))})^2 = 2^{2 \cdot O(t(n))} = 2^{O(t(n))}$ which needs assumption $t(n) \geq n$)

Definition

$P = \bigcup_k \text{TIME}(n^k)$
$k \geq 0$, k is an integer

The class of all languages decidable by a DTM whose worst-case runtime is a polynomial.

We could define NP as

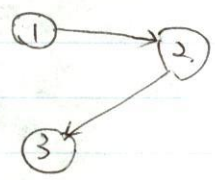
$\bigcup_k \text{NTIME}(n^k)$
$k \geq 0$, k is an integer

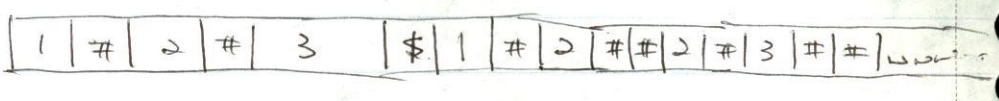
But we will define NP by verification Turing machines instead and later prove that it is equal to

Motivating example
Hamiltonian Path Problem

visit. each node exactly once.

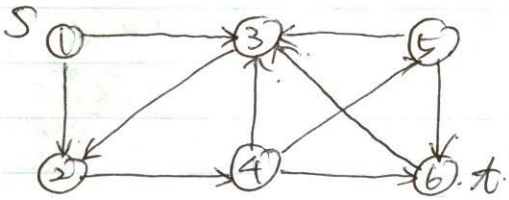
Input: a directed graph G and nodes s, t in G
Output: accept if G has a Hamilton path from s to t. reject o.w.
Encoding method: Nodes by integers 1, ..., m.
Edges by ordered pairs (i, j) of two integers



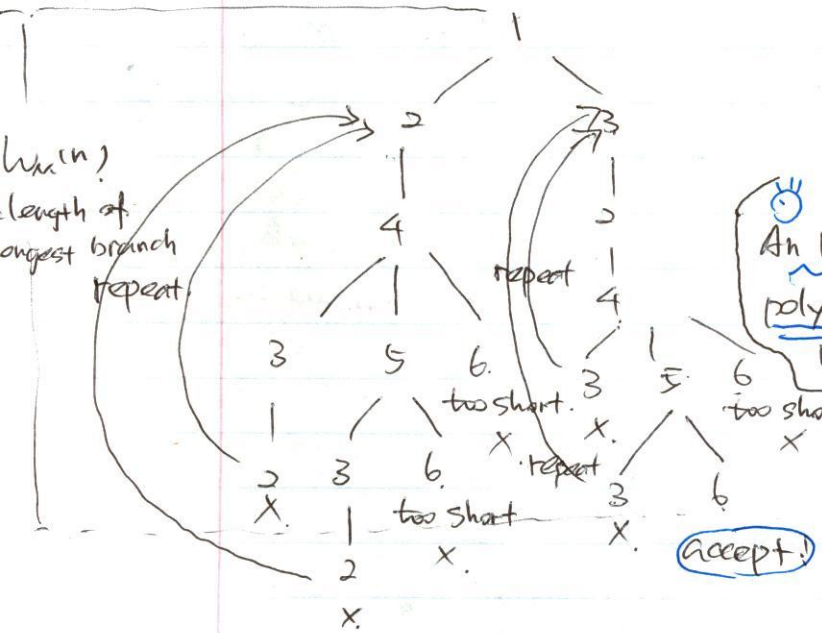


9/20

Encode G, s, t by input strings of Turing machines.
~~Ham~~ Let $\langle G, s, t \rangle$ be the string encoding a given G, s, t.
HAMPATH = $\{\langle G, s, t \rangle \mid G$ has a Ham path from s to $t\}$
↑ (The language encoding the Ham path problem)



search tree.
Has to begin with s and ending with t.
Each Ham path must have the form $(s, n_2, n_3, n_4, n_5, t)$ $n_i \neq 1$, $n_i \neq 6$
$n_2 n_3 n_4 n_5$ must be all distinct.
must be a permutation of 1, ..., 6 starting with 1 and ending with 6



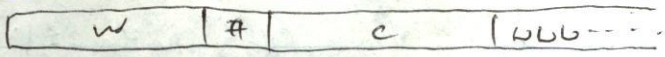
$h_M(n)$ = length of longest branch

An NTM can find a Ham path in polynomial time in the # of nodes by utilizing nondeterministic choices to adjacent nodes

No DTM has been found to decide this polynomial-time in worst case

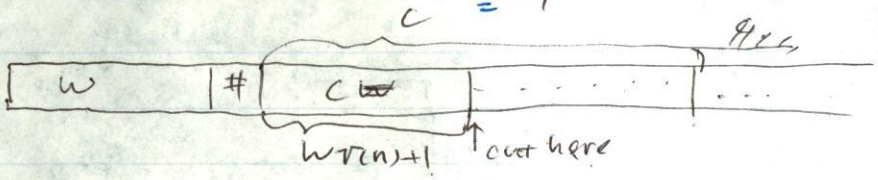
Verification version of Ham path problem
Additional ~~path~~ input: a path P in the given G ← proposed certificate
accept if P is a Ham path from s to t ← if verified, P is a certificate for G having a Ham path
reject o.w.
This can be checked in polynomial time

A Verifier is a 1-tape decider DTM that takes 2-component inputs $\langle w, c \rangle$



w is the normal input (e.g. encoding $\langle G, s, t \rangle$ for Hampath problem)
c is string encoding the proposed certificate (e.g. encoding a path p for Ham path problem)
The worst-case time complexity function is defined as for 1-tape DTMs, except that $n = |w|$ ignoring c

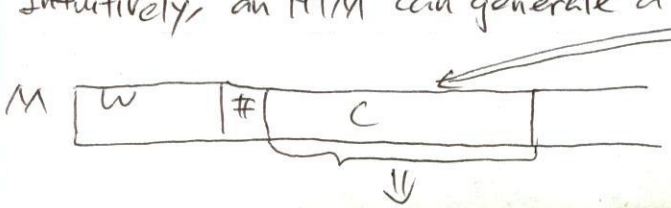
If the verifier accepts $\langle w, c \rangle$, w is said to be verified by a certificate c.
The language verified by a verifier V is
$L(V) = \{w \mid \exists c \text{ s.t. } V \text{ accepts } \langle w, c \rangle\}$,
w is rejected by V if $\nexists c$ s.t. V accepts $\langle w, c \rangle$ (that is, for $\forall c$, V rejects $\langle w, c \rangle$)
(for all possible strings)
Let $W_V(n)$ be the worst-case time complexity function of V; $n = |w|$
The length of c can be limited by $|c| \leq W_V(n) + 1$ since V can only access at most $W_V(n)+1$ symbols on tape in $W_V(n)$ steps.



Wether V accepts/rejects $\langle w, c \rangle$ is determined by the initial $W_V(n)+1$ symbols of c.
From now on, we assume that $|c| \leq W_V(n)$, dropping "+1" as it does not affect asymptotic analysis

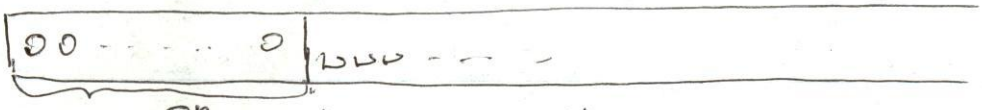
Theorem $V \to M$ (verifiers can be simulated by NTMs)
Let $L = L(V)$ for a verifier V, and assume for $\forall n$, $W_V(n) \leq f(n)$ for some $f(n)$ s.t. $n = |w|$ and the values of $f(n)$ in unary notation can be computed in $O(g(n))$ worst-case time. Then L is decided by an NTM, M, with $W_M(n) = O(f(n)) + O(g(n))$.

Intuitively, an NTM can generate a proposed certificate c nondeterministically



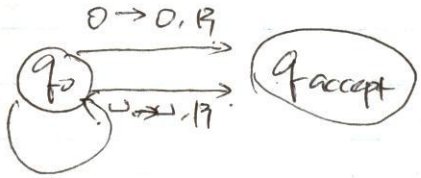
⇓ feed this to the given V.

NTM string generation module.



0^n n in unary notation.

The following NTM writes $x \in \{a_1, \ldots, a_k\}^*$ with $0 \le |x| \le n$



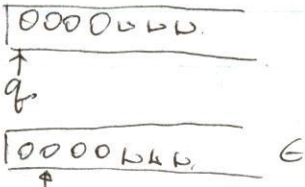
k nondeterministic choices on "0"

$0 \to a_1, R$

$0 \to a_2, R$

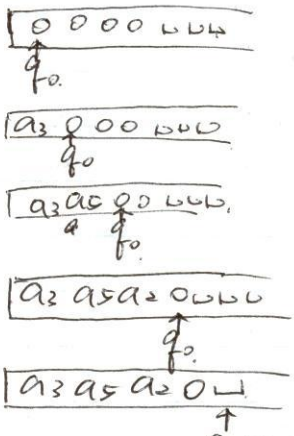
$\vdots$

$0 \to a_k, R$



$\in$ generated

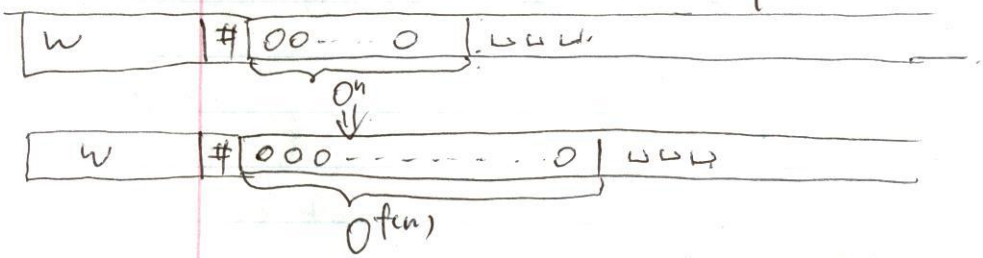
(deadend, no zero.



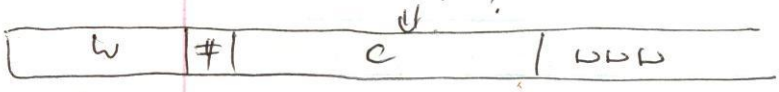
generated $a_3 a_5 a_2$

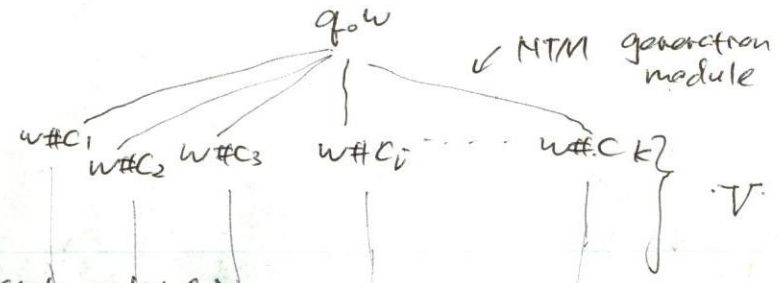
Proof Let V be the given verifier. Define the simulating NTM as follows

step1 Compute $n = |w|$ and the value of $f(n)$ in unary notation and write them on tape.



Using the NTM string generation module, generate a proposed certificate c s.t. $|c| \le f(n)$





step2 Run V on $\langle w, c \rangle$

step3 M accepts w if V accepts $\langle w, c \rangle$

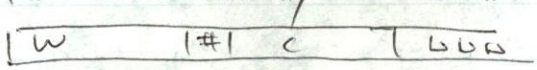
M halts by dead end if V rejects $\langle w, c \rangle$

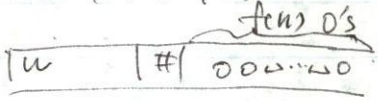
9/22

Theorem. $\forall M$ $L_0 + L = L(V)$ for a verifier V and assum for $\forall n$, $W_V(n) \le f(n)$ for some $f(n)$ such that $n = |w|$ and the value of $f(n)$ in unary notation can be computed deterministically in $O(g(n))$ worst-case time. Then L is decided by an NTM, M, with $W_M(n) = O(f(n)) + O(g(n))$

step1 Compute $n = |w|$ and the value of $f(n)$ is unary notation

Nondeterministically construct a string c s.t. $|c| \le f(n)$





step2 Run V on $\langle w, c \rangle$

step3 同上 (雙步上)

We prove ~~that~~ $L(V) = L(M)$

→ $L(V) \subseteq L(M)$, Suppose $w \in L(V)$. By def of V, $\exists$ c s.t. V accept $\langle w, c \rangle$ Since $|c| \le W_V(n) \le f(n)$. step1 of M nondeterministically generate c.

In step2, V accept $\langle w, c \rangle$ In step3, M accept w, hence $w \in L(M)$

→ $L(M) \subseteq L(V)$: Suppose $w \in L(M)$. Then M accepts w, hence V accept $\langle w, c \rangle$ for some c generate in step1. So $\exists$ c st. V accept $\langle w, c \rangle$, hence $w \in L(V)$.

Analysis of $W_M(n)$

step1 By assumption, $n = |w|$ and the value of $f(n)$ can be computed in $O(g(n))$ time. Nondeterministically generating c can be done in $O(f(n))$ time since $|c| \le f(n)$.

In total, step1 takes $O(g(n)) + O(f(n))$ time

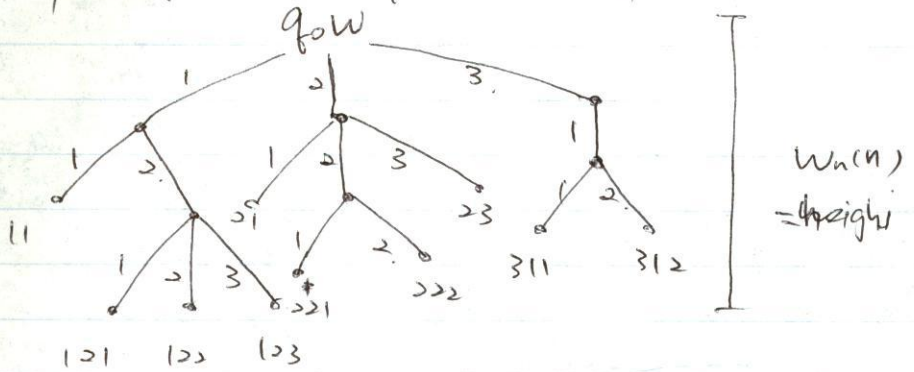
step2 Takes $W_V(n) \le f(n)$ time. Obviously $O(f(n))$ time

step3 constant.

Grand Total = $O(g(n)) + O(f(n)) + O(f(n)) + \text{constant}$

= $O(g(n)) + O(f(n))$

NTM's can be simulated by verifiers. A verifier simulates a given NTM by letting c encode one branch of the computation tree. Encoding by ~~acception~~ access path (address).



Recall $\delta_M(q,a) = \{(q_1,a_1,d_1) \cdots (q_k,a_k,d_k)\}$ Sequentially number the k transitions by $1 \cdots k$. Let b = the max value of k. Encode each branch by a sequence of integers $n_1 \cdots n_d$. $1 \le n_i \le b$, where the branch chooses the n_i-th transition in the i-step

Observations:

(1) $d \le W_M(n)$, $n = |w|$ height.

(2) Each n_i, $1 \le n_i \le b$, can be encoded by a string of at most a fixed length a. Hence, $n_1 \cdots n_d$ can be encoded by a string of length $\le a \cdot d \le a \cdot W_M(n)$ for some constant a.

Theorem. N to V Let $L = L(M)$ for a 1-tape decider NTM M. Then $L = L(V)$ for a verifier V with $W_V(n) \ge O(W_M(n) \times \max(n, W_M(n)))$ assuming that for all proposed certificate c, $|c| \le a \cdot W_M(n)$.

(time to simulate one step of M.)

This assumption is reasonable ∵ Observation (2)

Proof Defines the simulating verifier as following

Given input $\langle w, c \rangle$

step 1. Deterministically simulates M on input w, following the branch $n_1 \cdots n_d$ encoded in C. If n_i is encountered s.t. n_i-th choice does not exist, reject w. E.g. 4 → 3

If the simulation does not lead to q_{accept} within d steps, reject. Eg. d is too short 12, 31 etc. The sequence lead to a deadend

step 2 V accepts $\langle w, c \rangle$ if the simulation leads to q_{accept}. i.e. M accepts w.

V rejects $\langle w, c \rangle$ if w is rejected in steps.

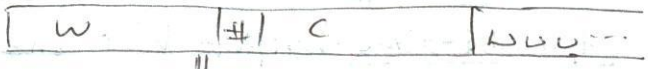
We prove $L(M) = L(V)$

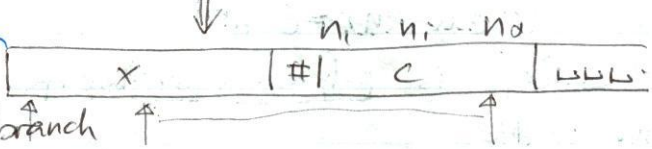
→ Suppose $w \in L(M)$ Then, M accepts w. So there is at least one branch of the computation of M leading to qaccept. Let c be $n_1 \ldots n_d$ encoding this branch's access path. Given $\langle w, c \rangle$, V simulates this branch leading to qaccept V then accepts $\langle w, c \rangle$ So $\exists c$ s.t. V accepts $\langle w, c \rangle$ hence $w \in L(V)$.

→ Conversely suppose $w \in L(V)$. By definition of V, this means $\exists c$. s.t. V accepts $\langle w, c \rangle$.
By def of step 2. the simulation of M on w by the branch encoded in c leads to qaccept, hence $w \in L(M)$.

Analysis of $W_V(n)$

Initial configuration.





V preserve c as it needs to follow the encoded branch

Everytime V needs to move the head to the position of #, it moves "#c" one position to right before head

$n = |w| \le |x| \le \max(n, W_V(n))$. $W_V(n)$ maybe a sublinear function like $O(\lg n)$ or even constant.

step1 In ith step, V needs to extract n_i out of c. This can be done in $O(|x|) + O(|c|) \le$
$O(\max(n, W_V(n))) + O(a \cdot W_V(n))$
$= O(\max(n, W_V(n)) + O(W_V(n))$
$= O(\max(n, W_V(n)))$

simulating of M.

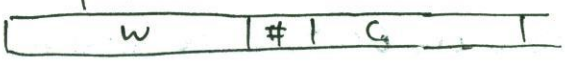
→ Note $|c| \le a \cdot W_V(n)$

x is the string M would write in following the branch encoded in $c = n_1 \ldots n_d$

Therom MoV proof continued...

9/27

Analysis of $W_V(n)$



$n_1 \ldots n_i \ldots n_d$ encodes an branch

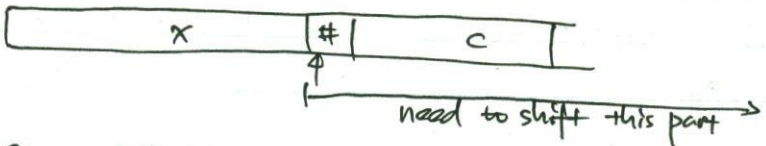
$n = |w| \le |x| \le \max(n, W_M(n))$

x is the string M would write on tape in following the branch encoded inc

step1 $O(\max(n, W_M(n)))$ to extract n_i out of c.

Also, in a single step, M may move the tape head one position to the right

of the rightmost symbol of x



Since $|c| \le a \cdot W_M(n)$, this shifting can be done in $O(W_M(n))$ steps.

In total, step 1 requires at most

$$\underbrace{W_M(n)}_{\text{total steps \# of M's steps}} \times \underbrace{(O(\max(n, W_M(n))) + O(W_M(n)))}_{\text{the time to simulate one step of M}} =$$

$$W_M(n) \times (O(\max(n, W_M(n)))) = O(W_M(n) \times \max(n, W_M(n)))$$

Step 2 constant time

A polynomial-time verifier V is one with $W_V(n) = O(n^k)$ for some $k \ge 0$

Hence $|c| \le W_V(n) = O(n^k)$

- A language is polynomial verifiable if it is verified by a polynomial-time verifier.

Def. $NP = \{L(V) \mid V \text{ is a polynomial-time verifier}\}$

Theorem $NP = \bigcup_{k \ge 0} NTIME(n^k) = \{L(M) \mid M \text{ is a 1-tape decider NTM with } W_M(n) = O(n^k)\}$

Horner's Algorithm

Proof Suppose $L \in NP$ Then $L = L(V)$ with $W_V(n) = O(n^k)$. Let $W_V(n) \le c\,n^k$ for some constant c.

In theorem V to M, take $f(n) = c \cdot n^k$, which known to be computable in polynomial time $O(g(n))$, $g(n)$ is some polynomial.

L is then decided by an NTM, M, with $W_M(n) = O(f(n)) + O(g(n))$

$= O(n^k) + O(g(n))$, which is polynomial bounded.

So, $L \in \bigcup_{k \ge 0} NTIME(n^k)$

Conversely suppose $L \in \bigcup_{k \ge 0} NTIME(n^k)$ Then $L = L(M)$, with $W_M(n) = O(n^k)$. By Theorem N to V, $L = L(V)$ with $W_V(n) = O(W_M(n) \times \max(n, W_M(n))) = O(O(n^k) \times \max(n, O(n^k)))$, polynomially bounded.

Theorem $P \subseteq NP$

Proof Every 1-tape DTM is a special case of 1-tape NTM

Outputs of NP problems take the form:

accept if $\exists$ certain object / structure (set, graph, number, mapping, etc) satisfying a set of constrains

reject o.w.

The proposed certificate is an object/structure
The verifier checks if the constrains are satisfied

SAT Satisfiability Problem

Input: a Boolean formula consisting of variables, ¬ (negation), ∨ (OR), ∧ (and).
$(x_1 \vee \bar{x}) \wedge (x_2 \vee \bar{x}_3) \wedge (x_1 \wedge x_3) \vee (x_4 \wedge \bar{x}_1)$

Output: accept if ∃ 0/1 assignment to the formula's variables (structure) that evaluate to 1 (constraint).
reject o.w.

SAT ∈ NP The proposed certificate is a 0/1 assignment to the vars. The verifier checks if it satisfies the input Boolean formula. This can be done in poly time.

SUBSET-SUM

A multiset (also called a bag) unordered list
can contain multiple copies of an element {1, 1, 1, 2, 2, 33, 5, 4}

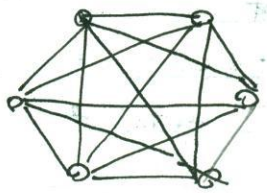
Input: A multiset $S = \{x_1, \dots x_k\}$ of integers x_i and an integer t

Output: accept if ∃ sub-multiset $\{y_1, \dots y_e\} \subseteq S$ (structure) s.t $y_1 + y_2 \dots + y_e = t$ (constraint)
reject o.w.

SUBSET-SUM ∈ NP The proposed certificate is a sub-multiset $\{y_1, \dots y_e\} \subseteq S$
The verifier checks if $y_1 + \dots + y_e = t$ This can be done in poly time.

CLIQUE undirected graphs

A complete graph is one where each pair of vertices are adjacent by an edge



A clique in a graph is a complete subgraph
A k-clique ~~~~~ with k vertices

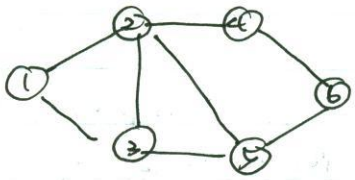
Input: Undirected graph G and an integer $k \geq 1$
output: accept if G has a k-clique
reject o.w

CLIQUE ∈ NP The proposed certificate is a subgraph of G.
The verifier checks if the subgraph is complete and has k vertices

VERTEX – COVER

Undirected graphs

A vertex cover of Graph is a set C of vertices s.t. each edge of G is incident on a vertex in C



$C = \{1, 4, 5, 2\}$

A k-vertex cover is one with exactly k vertices

Input: Undirected graph G and an integer $k \geq 1$

Output: accept if G has a k-vertex cover

reject o.w

VERTEX-COVER $\in$ NP. The proposed certificate is a set of vertices of G

The verifier checks if it is a k-vertex cover

Next class will be on Oct. 13. # Assignment 2 will be posted on Oct. 3.

Reductions

$\Pi_1 \rightarrow \Pi_2$

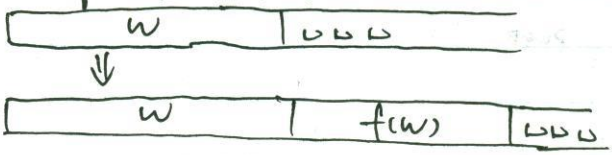
Each instance of Π_1 is mapped to an instance of Π_2.

A solution for S_2 can be used to obtain a solution of S_1

DTMs to compute functions over languages

Let Σ be an alphabet, a finite set of symbols without $\sqcup$

A function $f: \Sigma^* \rightarrow \Sigma^*$ is a computable function if there exists a DTM, M, which given any input $w \in \Sigma^*$, writes f(w) on tape and halts in q_{accept}. (q_{reject} is not used)



M is said to compute f

The worst case complexity function $W_M(n)$ is defined as the same for decider DTMs

Decider DTMs can be simulated by computing DTMs as a special case of computing DTMs.

Include special symbols 'a' and 'r' in Σ to represent accept and reject

decider — computing M

$w \to [M'] \to q_{accept}$ ← writes 'a' and enters q_{accept}

$\to q_{reject}$ ← " r_1 " " " q_{accept}.

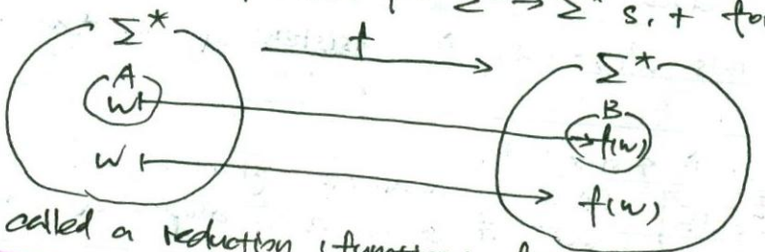
If $w_M(n) = O(n^k)$, f is polynomial-time computable. called equivalent condition.

Reductions

Let $A, B \subseteq \Sigma^*$ A is (mapping) reducible to B, written $A \leq_m B$,

if $\exists$ computable function $f: \Sigma^* \to \Sigma^*$ s.t for $\forall w \in \Sigma^*$, $w \in A \Leftrightarrow f(w) \in B$

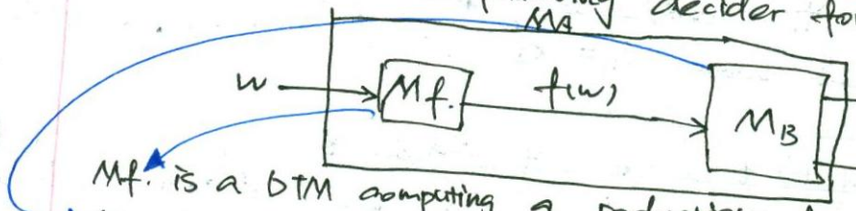
this implies $w \notin A \Leftrightarrow f(w) \notin B$



f is called a reduction (function) from A to B.

Theorem 1 If $A \leq_m B$ and B is decidable, then A is decidable.

Proof Construct the following decider for A:



M_f is a DTM computing a reduction function f from A to B.

M_B is a decider DTM for B.

We denote M_A by $M_f ; M_B$

If a reduction ~~from~~ function f from A to B is polynomial-time computable, A is said to be polynomial-time reducible to B. written $A \leq_p B$

Notation

$f: A \leq_p B$ f is a polynomial-time reduction function from A to B.

$M_f: A \leq_p B$ M_f is a DTM computing a polynomial-time reduction function f from A to B

A literal is x_i or $\bar{x}_i$ where x_i is a Boolean var.

A clause is a disjunction of literals $l_1 \vee l_2 \vee \dots \vee l_m$ where each l_i is a literal.

A conjunctive normal form, cnf-formula, is a conjunction of clauses $c_1 \wedge c_2 \wedge \dots \wedge c_n$ where each c_i is a clause

$\underbrace{(x_1 \vee \bar{x}_2 \vee x_3 \vee x_4)}_{c_1} \wedge \underbrace{(\bar{x}_1 \vee \bar{x}_2 \vee x_4)}_{c_2} \wedge \underbrace{(x_1 \vee \bar{x}_3 \vee \bar{x}_4 \vee x_5)}_{c_3} \wedge \dots$

A 3-cnf-formula is a cnf-formula where each clause c_i has exactly 3 literals.

接前 $(x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_3 \vee x_4) \wedge (x_1 \vee \bar{x}_3 \vee \bar{x}_4) \wedge \cdots$

3SAT is a satisfiability problem for 3-cnf-formulas.

Literal x_i and $\bar{x}_i$ are said to be inconsistent. $x_i \wedge \bar{x}_i$ is not satisfied.

O.w., two literals l_1, l_2 are said to be consistent. $l_1 \wedge l_2$ is satisfied

A set of literals $l_1, \cdots l_m$ is consistent if any two of them is consistent.

$l_1 \wedge \cdots \wedge l_m$ is satisfied.

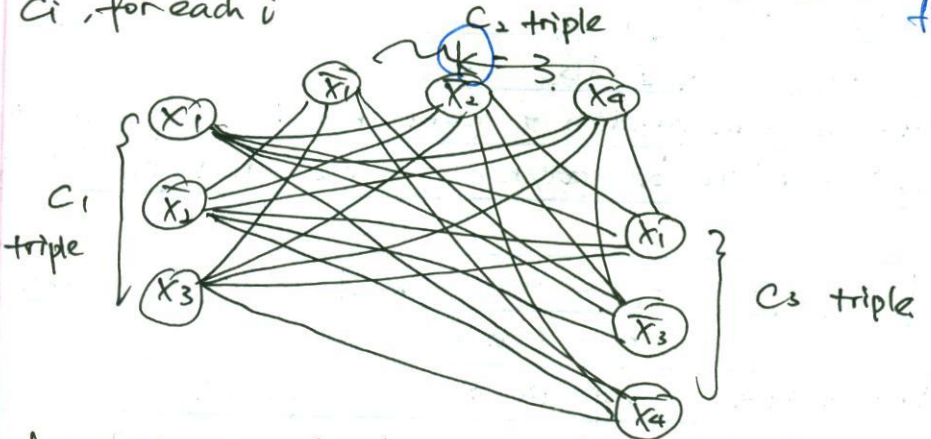
3SAT $\leq_p$ CLIQUE Given an undirected graph G and an integer k, decide if G has a k-clique a complete subgraph with k nodes

We show a poly-time reduction function f : 3-cnf-formula $\varphi \mapsto (G, k)$ s.t φ is satisfied iff G has a k-clique.

Let $\varphi = C_1 \wedge \cdots \wedge C_k$ where $C_i = l_i^1 \vee l_i^2 \vee l_i^3$, $1 \leq i \leq k$. each l_i is a literal.

f sets k in (G, k) to be the # of clauses k in φ

f creates a triple of nodes labeled by $l_i^1\ l_i^2\ l_i^3$ (fixed 3) from the clause C_i, for each i



f creates an edge between l_i^p and l_j^q if,

(1) $i \neq j$ (l_i^p and l_j^q are in different clause/triples)

(2) l_i^p and l_j^q are consistent ($l_i^p \wedge l_j^q$ is satisfiable)

Proof of equivalence condition

Suppose φ is satisfiable by an assignment A

Then A makes each clause C_i, $1 \leq i \leq k$, true.

So, A make at least one literal l_i^p of C_i true

Let $C = \{l_1^{p_1}, l_2^{p_2}, \cdots l_k^{p_k}\}$ be the set of k nodes labeled by $l_1^{p_1}, l_2^{p_2}, \cdots l_k^{p_k}$ that are true under A.

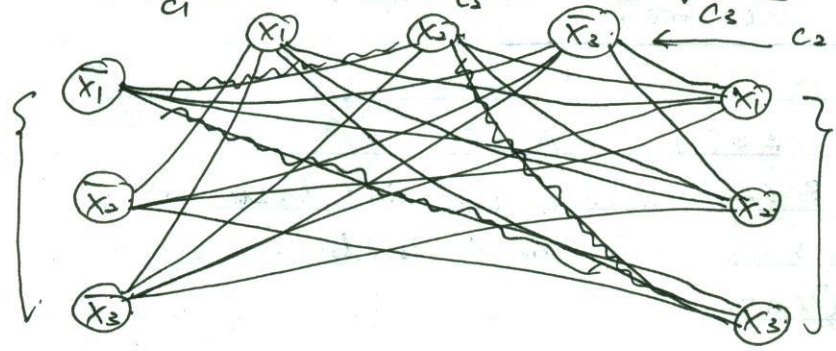
Then C is consistent. So any ~~two~~ two of them are consistent, and there is an edge ~~between~~ between them. Hence C forms a k-clique.

10/13

3 SAT $\leq_p$ CLIQUE

$\varphi = (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (x_1 \vee x_2 \vee \bar{x}_3) \wedge (x_1 \vee x_2 \vee x_3)$

C_1 C_2 C_3



$\Delta \{\bar{x}_1, x_2, x_3\}$ $A = \{x_1 = 0, x_2 = 1, x_3 = 1\}$

Proof of Equivalence condition. φ is satisfiable iff G has a k-clique.

($\Rightarrow$) Proved in last class

($\Leftarrow$) Suppose G has a k-clique, k = the # of clauses = the # of triples

Since there exists no edge between the same triple, the k-clique must have exactly one node from each of the k triples. Let these k nodes be $\{l_1, l_2, \cdots, l_k\}$ where the l_i are literals, l_i from the triple for clause C_i. By definition of the reduction, any two of these literals are consistent.

So $\{l_1, l_2, \cdots l_k\}$ is consistent, hence there is an assignment satisfying $\{l_1, l_2, \cdots l_k\}$ making each l_i true.

So clause l_i is made true by this assignment. So this assignment satisfies φ.

It remains to show reduction can be computed in polynomial time.

The size of φ can be measured by n = the # of literals in φ = $3k$

The reduction generates $3k$ nodes and at most $\frac{3 \times \cancel{k} \times 3 \times (k-1)}{2}$ (k is # of clauses)

$\because$ undirected graph ~~both~~ each edge counted twice.

$3 \cdot \frac{n}{3} = n$

$= \frac{9}{2} k(k-1)$ edges

$= \frac{9}{2} \frac{n}{3} (\frac{n}{3} - 1)$

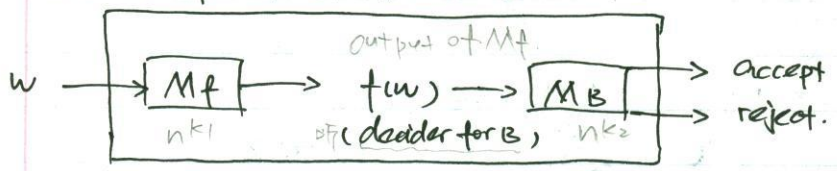
G has n nodes $O(n)$

$\frac{9}{2} \cdot \frac{n}{3} (\frac{n}{3} - 1)$ edges. $O(n^2)$

So, G can be generated in $O(n^2)$ time

Theorem 1 If $A \leq_m B$ and B is decidable, A is decidable.

Decider for A = M_f ; M_B



Theorem 2 If $A \leq_p B$ and $B \in P$, then $A \in P$

proof Let M_f : $A \leq_p B$ and M_B be a poly-time decider for B

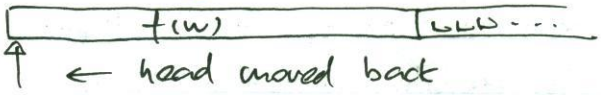
The machine M_f ; M_B in Theorem 1 decides A. It remains to show $W_{M_f ; M_B}(n) = O(n^k)$

Let $W_{M_f}(n) = O(n^{k_1})$ and $W_{M_B} = O(n^{k_2})$

Then by def. of M_f ; M_B,

$W_A(n) = O(W_{M_f}(n) + |f(w)| + W_{M_B}(|f(w)|))$

$n = |w|$ M_f writes $f(w)$ on tape, and the head must be moved back to the leftmost cell.



← head moved back

Since M_f can write at most $W_{M_f}(n)$ symbols on tape,

$|f(w)| \leq W_{M_f}(n) = O(n^{k_1})$. So

$W_A(n) = O(O(n^{k_1}) + O(n^{k_1}) + O((O(n^{k_1}))^{k_2}))$

$= O(O(n^{k_1}) + O(n^{k_1 \cdot k_2}))$

∵ k_2 can be zero $= O(n^{\max(k_1, k_1 \cdot k_2)})$ is in poly-time bound.

∴ use max **End of proof**

Theorem 3 If $A \leq_p B$ and $A \notin P$, the $B \notin P$

proof Suppose $B \in P$ to the contrary. By theorem 2, $A \in P$, contradicting $A \notin P$.

Theorem 4 Let $A, B, C \subseteq \Sigma^*$

(1) $A \leq_p A$ (reflexivity)

(2) $A \leq_p B$ and $B \leq_p C \Rightarrow A \leq_p C$ (transitivity)