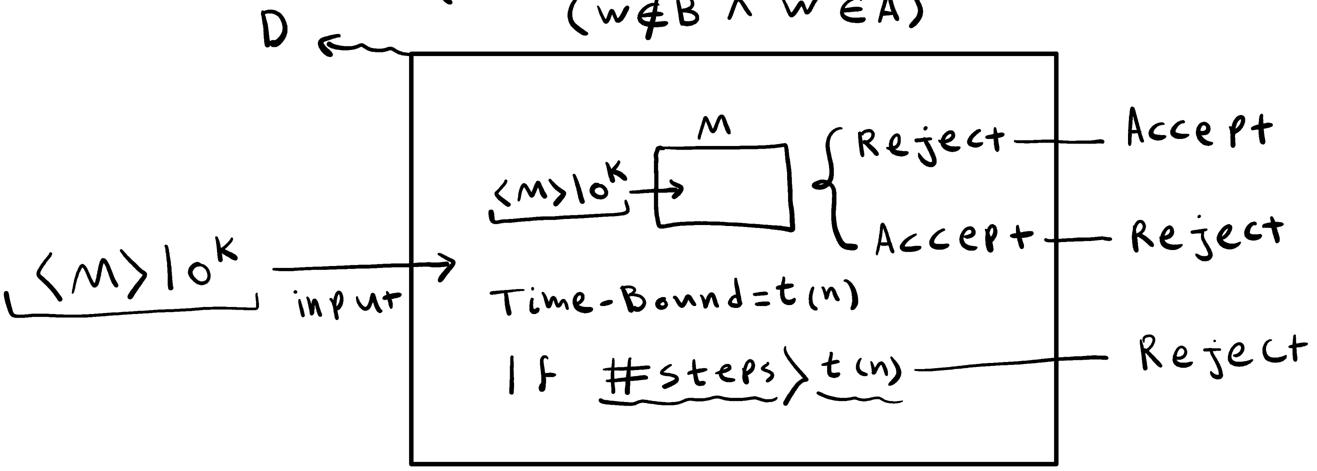
Time Hierarchy Thm

Sipser 9.10 : [تنها تفاوت ضرب در $Lg(t(n))$ است]

اگر تابع $t(n)$، time Constructible باشد، وجود دارد زبان decidable با نام A که $A \in TIME(O(t(n).Lg(t(n))))$ و در زمان $o(t(n))$، decidable نباشد.

Proof: یک TM برای زبان $A \in TIME(t(n).Lg(t(n)))$ با نام D می‌سازیم ونشان می‌دهیم برای هر زبان B که در زمان $o(t(n))$ قابل decide است $A \neq B$ خواهد بود. پس A در زمان $o(t(n))$ قابل decide نیست.

$$(\exists w)(w \in B \wedge w \notin A) \vee (w \notin B \wedge w \in A) \leftarrow A \neq B$$



① زمان $(B) \geq t(n)$, s.t. $(\forall B)$

وجود دارد ← $w \in B$

که ← $w \notin A$

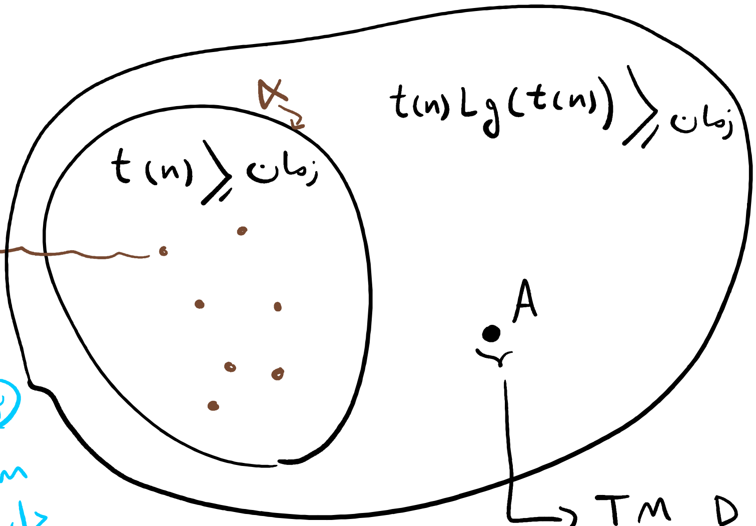
یا

وجود دارد ← $w \notin B$

که ← $w \in A$

② نشان می‌دهیم برای یک A

TM در زمان $O(t(n)Lg(t(n)))$ داریم.



* ② در زمان $O(t(n)Lg(t(n)))$ اجرا شود.

* اگر وجود داشته باشد TM با نام D' به شکلی که زمان $(D') = o(t(n))$ و $L(D') = A$ به تناقض می‌رسیم.

$\alpha = \{ B_i \mid$ وجود دارد TM با نام M_i به شکلی که $L(M_i) = B_i$ و زمان اجرای M_i از $t(n)$ کمتر مساوی باشد $\}$

دلیل نام قطری سازی

$\alpha \rightarrow$

	$\langle M_1\rangle 10^{K_1}$	$\langle M_2\rangle 10^{K_2}$	$\langle M_3\rangle 10^{K_3}$	$\langle M_4\rangle 10^{K_4}$
B_1	1			
B_2		0		
B_3			1	
B_4				0

رفتار ماشین D روی قطر

$\text{If } \langle M_i\rangle 10^{K_i} \in B_i \Longrightarrow \langle M_i\rangle 10^{K_i} \notin D$

$\text{If } \langle M_i\rangle 10^{K_i} \notin B_i \Longrightarrow \langle M_i\rangle 10^{K_i} \in D$

$D(\langle M \rangle 10^K) =$ ① Let $n \leftarrow |\langle M \rangle 10^K|$

② Compute $t(n)$ using time constructibility and store the $t(n)$ in a binary counter. Decrement this counter before each step used to carry out stage 3 and 4. If the counter ever hit 0, reject

③ Simulate M on $\langle M \rangle 10^K$.

④ If M accepts, then reject. If M rejects, Then accept

- زمانی که می‌گوییم M در زمان $O(t(n))$ اجرا می‌شود، یعنی برای هر $c>0$ وجود دارد K به شکلی که برای هر $n \geq K \longleftarrow$ $c\,f(n) \geq$ زمان اجرای M روی ورودی به اندازه‌ی n. پس ممکن است برای برخی از n های کوچکتر از K برقرار نباشد. پس اگر به جای ورودی $\langle M \rangle 10^K$ به آن ورودی $\langle M \rangle$ را بدهیم و $K > |\langle M \rangle|$ آن‌گاه ممکن است برنامه کامل اجرا نشده تمام شود و ما نتوانیم مطمئن شویم که به ازای ورودی $\langle M \rangle$ دو ماشین M و D متفاوت عمل می‌کنند. برای همین D را به گونه‌ای طراحی می‌کنیم که روی ورودی $\langle M \rangle 10^K$ برای هر K ثابت M را روی ورودی $\langle M \rangle 10^K$ تا زمان $t(n)$ شبیه‌سازی کند. از آن‌جا که وجود دارد K برای هر کدام از زبان‌های قابل اجرا در $O(t(n))$ پس اگر ماشین در زمان $O(t(n))$ اجرا شود، $\langle M \rangle 10^K$ حتما تا انتها روی M شبیه‌سازی می‌شود.

پس D ورودیِ $\langle M\rangle 10^k$ را گرفته و اجرای $M(\langle M\rangle 10^k)$ را شبیه‌سازی می‌کند. همچنین تعداد قدم های اجرای M روی $\langle M\rangle 10^k$ را می‌شماریم. شبیه سازی تا حداکثر $t(n)$ قدم اجرا می‌شود.

در این فاصله اگر M روی ورودی Halt کرد خروجی D برعکس خروجی M خواهد بود.

در اثبات، مهمترین بخش درنظرگرفتن هزینه‌ی شبیه سازی با شمارش قدم های شبیه سازی می باشد.

ما می‌خواهیم D درزمان $O(t(n)\lg(t(n)))$ اجرا شود و همچنین زبان آن با هر زبان که درزمان $o(t(n))$ اجرا می شود متفاوت باشد.

جلد ی سیپسر، ص۳۶۹ :

روشِ فعلی شبیه سازی TM تکنواره روی TM تکنواره با شمارش قدم سربار لگاریتمی دارد. اگر روشی بهتری برای شبیه‌سازیِ تکنواره روی تکنواره برای تعداد مشخص قدم داشته باشیم به شکلی سربار ثابت باشد، می شد فاکتور لگاریتمی را از قضیه پاک کرد. چنین شبیه سازی کارا شناخته نشده است.

به طور شهودی، THT که tight تر باشد درست است ولی درحال حاضر ما نمی‌دانیم چگونه آن را اثبات کنیم ..... برای مدل های محاسباتی دیگر این ممکن است.

$D(\langle M\rangle 10^k) =$

No cost ⟵ ① Let n be the $|\langle M\rangle 10^k|$

$O(t(n))$ ⟵ ② Compute $t(n)$ using time Constructibility and store the $t(n)$ in a binary counter. Decrement this counter before each step used to carry out stage 3 and 4. If the counter ever hit 0, reject

③ Simulate M on $\langle M\rangle 10^k$.

④ If M accepts, then reject. If M rejects, Then accept

خط ① ⟵ محاسبه‌ای در خط ① وجود ندارد. یک نام گذاری ساده است.
[اگر می‌بایست محاسبه شود هزینه‌ی محاسبه $O(n \lg n)$ بود ولی نیازی نیست. می‌توانیم
به w که $|w|=n$ به چشم 1^n برای محاسبه‌ی $t(n)$ نگاه کنیم.]

خط ② ⟵ $t(n)$ در زمان $O(t(n))$ قابل محاسبه است، چون
$t(n)$ یک تابع time Constructible است. برای محاسبه‌ی $t(n)$ اول باید
input با طول n خوانده شود. چون $t(n) \geq n$ پس کل زمان $O(t(n)+n) =$
$O(t(n))$ می‌باشد.

خط ③ برای شبیه سازی هر قدم می‌بایست $\delta_M(q_n, a)$ که در آن q_n
حالت فعلی و a کاراکتر زیر هد مجازی M است محاسبه شود. جواب (q_y, b, D)
اطلاعات لازم برای به روز رسانی را به ما می‌دهد

پس برای به روز رسانی نوار به ۳ شی زیر نیاز داریم:

1. δ_M

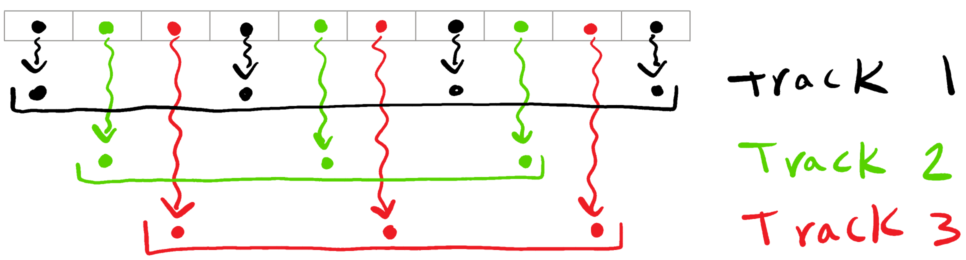
2. حالت فعلی [current state]

3. کارکتر روی هر مجازی M

هر ۳ شی جایی روی تک‌نوار نوشته شدند.
اگر فاصله‌ی این سه شی روی نوار زیاد باشد، هر قدم شبیه سازی طولانی خواهد بود.

Track بندی نوار:

نوار را به ۳، Track تقسیم می‌کنیم:



Note: توجه شود، اگر تعداد Trackهای c ثابت باشد، سرباران هم ثابت است.

Track1 ← محتوای نوار M که آن را شبیه‌سازی می‌کنیم.

Track2 ← شامل δ_Mهای M باشد. توجه شود که δ_M به شکل مجموعه‌ای از ۵تایی ها می باشد: $\delta_M(q_x,a)=(q_y,b,D) \rightsquigarrow (q_x,a,q_y,b,D)$

و شامل حالت فعلی هم می باشد [currentState]

Look-up Table

$\delta_M :=$

q_x	a	q_x	□	R
q_y	b	q_z	b	R
q_z	b	q_z	b	L
q_x	a	q_y	a	L
q_x	a	q_z	b	R

اگر محتوای نوار babba باشد و هد روی دومین حرف باشد، بدون در نظر گرفتن Track سوم داریم:

b	x	x	$\hat{a}$	q_z	?	b	q_x	?	b	a	?	a	q_x	?

....

Current State

Note: اگر همواره δ_M و currentState درکنار هر مجازی M باشند، هزینه‌ی Lookup ثابت است. چون $|\delta_M|+|Lg|Q||=O(|\langle M\rangle|)$

هر حالت در $Lg|Q|$ بیت

برای همین محتوای Track 2 را نزدیک به هد مجازی نگه می داریم.
برای این کار با هر حرکت هد M ، محتوای Track2 را در راستای حرکت
هد، shift می دهیم. چون محتوای Active ، Track 2 ، از
مرتبه ی $O(|\langle M \rangle|)$ یعنی ثابت است. پس shift این محتوا
هزینه ی ثابت به ازای هر قدم ایجاد می کند.

Note : اندازه ی محتوای Active ، Track 2 تنها به M وابسته است و
نه به اندازه ی ورودی M ؛ پس آن را ثابت می گیریم.

* پس اگر Timer نداشته باشیم سربار شبیه سازی ثابت است.

در Track3 ، Timer را ذخیره می کنیم. آن ابتدا با $t(n)$ مقداردهی
اولیه شده است و با هر قدم اجرا یک واحد از آن کم می شود.
به روشنی اندازه ی آن $Lg(t(n))$ می باشد. برای آن که محتوای Active
این Track را نزدیک به هد مجازی M نگه داریم با هر قدم آن را در راستای
حرکت هد shift می دهیم. shift و decreament در هر قدم $O(Lg(t(n)))$
سربار زمانی دارد.

در نتیجه A در زمان $O(Lg(t(n))\,t(n))$ اجرا می شود.

حال می‌خواهیم نشان دهیم A در زمان $o(t(n))$، تصمیم‌پذیر نیست.

برهان خلف: اگر A در زمان $o(t(n))$، تصمیم‌پذیر باشد، پس برای آن یک decider با نام D' وجود دارد که آن را در زمان $dg(n)$ برای یک ثابت d تصمیم می‌گیرد. هم‌چنین $g(n) \in o(t(n))$ می‌باشد. [d وابسته به M است]. پس وجود دارد K به شکلی که برای هر $n \geq K$، $dg(n) < t(n)$، پس روی $\langle D' \rangle 10^K$، D تا انتها اجرا خواهد شد. [زمان کافی برای رسیدن به انتهای شبیه‌سازی دارد.] پس اگر $D'(\langle D' \rangle 10^K) = 1$ باشد $D(\langle D' \rangle 10^K) = 0$ است و برعکس. به عبارتی اگر $\langle D' \rangle 10^K \in L(D')$ آن‌گاه $\langle D' \rangle 10^K \notin L(D)$ و اگر $\langle D' \rangle 10^K \notin L(D')$ آن‌گاه $\langle D' \rangle 10^K \in L(D)$ خواهد بود. پس $L(D') \neq L(D)$ از آن جا که $L(D) = A$ پس این تناقض است. ✗

پس A در زمان $o(t(n))$ تصمیم‌پذیر نیست. ■

نتیجه:

وجود دارد $A \in TIME(t(n) Lg(t(n)))$ که A در $o(t(n))$ تصمیم پذیر نیست

پس برای هر $t_1(n)$ و $t_2(n)$ که $\underbrace{t_1(n) Lg(t_1(n)) = o(t_2(n))}$

داریم $\underbrace{TIME(t_1(n)) \subsetneq TIME(t_2(n))}$

چون می‌توانیم با زمان $t_1(n) Lg(t_1(n))$ ماشین D را بسازیم که $L(D) \notin TIME(t_1(n))$

آ روراً: اگر $f(n) Lg(f(n)) = o(g(n)) \Rightarrow TIME(f(n)) \subsetneq TIME(g(n))$

HS66

Hennie & stearn

https://doi.org/10.1145/321356.321362

two-tape Simmulation of Multitape TM

↳ نتیجه: $2Tape\text{-}UTM$ که $KTape\text{-}TM$ را که زمان T دارد در $\underbrace{TLgT}$ ، Simmulate می کند.

[Note: Timer می‌تواند به شکل یک نوار اضافی به TM اضافه شود.]

① ایده‌ی Track بندی

② ایده‌ی هِد ثابت:

یک زمان دلخواه اجرای $M(x)$ به شکل زیر است:

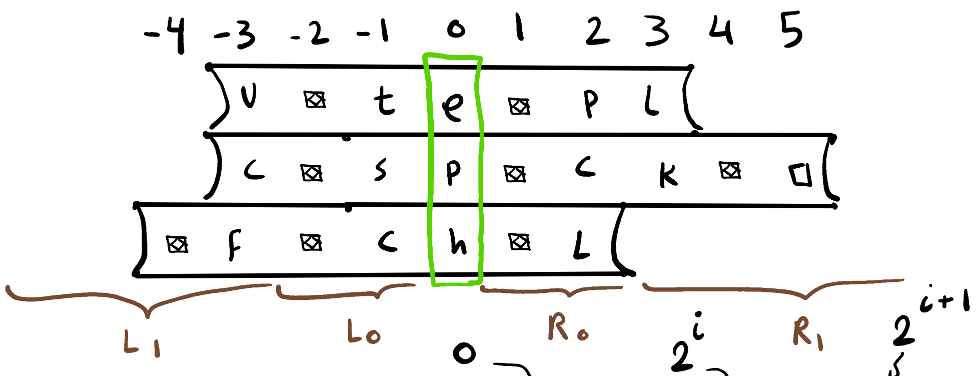
T_1:		O	m	I	D		wt_1
T_2:	A	V	B	D			wt_2
T_3		F	S	S	Z	P	wt_3

$T_1 \rightsquigarrow LSH$, $T_3 \rightsquigarrow RSH$

T_1:	O	m	I	D			wt_1
T_2:		A	V	B	D		wt_2
T_3:		F	S	S	Z	P	wt_3

$\rightsquigarrow (I, V, S) \in \Gamma^3 \rightsquigarrow k=3$

حرکت هِد با شیفت simulate می‌شود.



$|R_i| = 2^{i+1}$

[bidirectional]

• هر zone می‌تواند ۳ حالت empty، half-full و full را در طول محاسبه داشته باشد.

همان الگوریتم را داریم. TM دو نواره است و ورودی متغیر نواره. همه‌ی استدلال‌ها به همان شکل باقی می‌مانند. می‌توان counter را روی یکی از نوارهای TM ورودی فرض کرد. از آنجا که سربار simmulation، $\underline{T \lg T}$ است، خط ۳ از $T \lg T$ بهتر نمی‌شود.

پس باند tight تری نداریم.