Time Complexity

傅育熙

[Turing] has for the first time succeeded in giving an absolute definition of an interesting epistemological notion, i.e., one not depending on the formalism chosen.

– Kurt Gödel

[Turing machines have] the advantage of making the identification with effectiveness in the ordinary (not explicitly defined) sense evident immediately.

– Alonzo Church

Synopsis

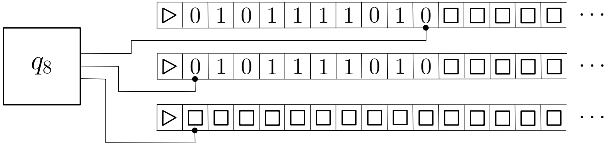
1. Turing Machine
2. Universal Turing Machine
3. Speedup Theorem
4. Time Complexity Class
5. Verification Problem
6. Time Hierarchy Theorem
7. Gap Theorem

Turing Machine

k-Tape Turing Machine

A k-tape Turing Machine $\mathbb{M}$ has k-tapes.

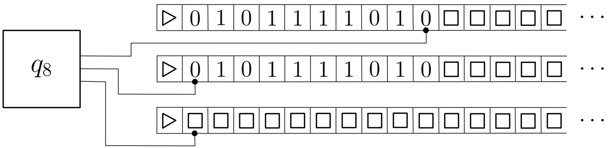
- The first tape is the read-only input tape.
- The other $k-1$ tapes are the read-write work tapes.
  - the k-th tape is also used as the output tape.



k-Tape Turing Machine

A k-tape Turing Machine is described by a tuple (Γ, Q, δ).

1. A finite set Γ of symbol, called alphabet such that $\Gamma \supseteq \{0, 1, \square, \triangleright\}$.
2. A finite set Q of states such that $Q \supseteq \{q_{\texttt{start}}, q_{\texttt{halt}}\}$.
3. A transition function $\delta : Q \times \Gamma^k \to Q \times \Gamma^{k-1} \times \{\texttt{L}, \texttt{S}, \texttt{R}\}^k$.



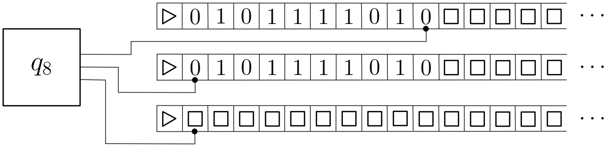
Configuration and Computation

A configuration of a running TM $\mathbb{M}$ consists of the following:

- the state;
- the contents of the work tapes;
- the head positions.

---

initial/start configuration, final configuration; 1-step computation



Problems Solved by Turing Machines

A function $f\colon \{0,1\}^* \to \{0,1\}^*$ is a problem.

- $\mathbb{M}$ computes or solves f if $\mathbb{M}(x) = f(x)$ for every $x \in \{0,1\}^*$.
- "$\mathbb{M}(x) = y$" stands for "$\mathbb{M}$ halts with y written on its output tape if its input tape is preloaded with x".

A function $d\colon \{0,1\}^* \to \{0,1\}$ is a decision problem.

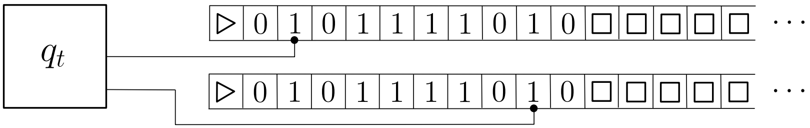
- $\mathbb{M}$ decides d if $\mathbb{M}$ computes d.

A set $L \subseteq \{0,1\}^*$ is a language.

- $\mathbb{M}$ accepts L if $\mathbb{M}$ decides the characteristic function $L(x) = \begin{cases} 1, & \text{if } x \in L, \\ 0, & \text{if } x \notin L. \end{cases}$

A 2-Tape Turing Machine for Palindrome Problem

A detailed transition function is defined in the book. The TM works in linear time.



What if we are only allowed to use TMs with one read-write tape?

Time Function

Suppose $T : \mathbf{N} \rightarrow \mathbf{N}$ and $\mathbb{M}$ computes the problem f.

We say that $\mathbb{M}$ computes f in $T(n)$-time if its computation on every input x requires at most $T(|x|)$ steps.

- $|x|$ denotes the length of x.
- For example $|2^{1024} - 1| = 1024$.

We shall assume that all time functions $\geq n$.

Design Turing Machines for the following functions:

1. $s(x) = x + 1$. [using a TM for $s(x)$ we can implement counter.]
2. $u(x) = 1^x = \underbrace{1 \ldots 1}_{x}$. [we often attach 1^x to disallow a TM to run more than x steps.]
3. $e(x) = 2^x$. [the machine simply outputs 10^x.]
4. $l(x) = \log(x)$. [if $x = 2^k$, the machine outputs $|x| + 1$; otherwise it outputs $|x|$.]

Time Constructible Function

Suppose $T : \mathbf{N} \to \mathbf{N}$ and $T(n) \geq n$.

1. T is time constructible if there is a Turing Machine that computes the function $1^n \mapsto \llcorner T(n) \lrcorner$ in time $O(T(n))$.
2. T is fully time constructible if there is a Turing Machine that upon receiving 1^n stops in exactly $T(n)$-steps.

We shall only care about the time-constructible functions.

Hard-Wiring a Clock to a Turing Machine

Let $\mathbb{M} = (Q_0, \Gamma_0, \rightarrow_0)$ be a k_0-tape TM.

Let $\mathbb{T} = (Q_1, \Gamma_1, \rightarrow_1)$ be a k_1-tape TM that runs in $T(n)$-steps.

We can use $\mathbb{T}$ as a timer to force $\mathbb{M}$ to terminate in no more than $T(n)$-steps.

- The integrated (k_0+k_1)-tape TM consists of two parallel machines. After replicating the input, it operates as the TM specified by
  - $Q = Q_0 \times Q_1$ and $(q, q^1_{\mathtt{halt}}) = q_{\mathtt{halt}}$ for $q \in Q_0$ and $(q^0_{\mathtt{halt}}, q) = q_{\mathtt{halt}}$ for $q \in Q_1$;
  - $\Gamma = \Gamma_0 \times \Gamma_1$;
  - $\rightarrow = \rightarrow_0 \times \rightarrow_1$.

$\mathbb{T}$ is said to be hard-wired to $\mathbb{M}$.

Complexity theory ought to be model independent.

Variants of Turing Machines are equivalent to the k-tape Turing Machines in the sense that they can simulate each other with polynomial overhead.

Oblivious Turing Machine

A TM is oblivious if the locations of its heads at the i-th step of the execution on input x depend only on $|x|$ and i.

Church-Turing Thesis.

Every physically realizable computing device can be simulated by a Turing Machine.

Law of Nature vs Wisdom of Human.

Universal Turing Machine

The confusion of software, hardware and datum lies at the heart of computation theory.

- Finite syntactic objects can be coded up by numbers. [$(101)^*$.]

Turing Machine as String

1. A transition $(p, a, b, c) \to (q, d, e, \mathtt{R}, \mathtt{S}, \mathtt{L})$ can be coded up by say

$$001\dagger1010\dagger1100\dagger0000\dagger\dagger011\dagger1111\dagger0000\dagger01\dagger00\dagger10.$$

2. A transition table can be coded up by a string of the form

$$\ddagger\_\ddagger\_\cdots\_\ddagger\_\ddagger. \tag{1}$$

3. A binary representation of (1) is obtained by using the following mapping:

$$\begin{aligned} 0 &\mapsto 01, \\ 1 &\mapsto 10, \\ \dagger &\mapsto 00, \\ \ddagger &\mapsto 11. \end{aligned}$$

Turing Machine as String

The encodings can be made to enjoy the following property:

1. Every TM is represented by infinitely many strings in $\{0,1\}^*$.
   - If σ encodes a machine, then $0^i\sigma 0^j$ encodes the same machine.
2. Every string in $\{0,1\}^*$ represents some TM.
   - Let all illegal strings code up a specific TM.

- Let $\llcorner \mathbb{M} \lrcorner$ be the binary representation of TM $\mathbb{M}$.
- Let $\mathbb{M}_\alpha$ be the TM represented by the binary string α.

Effective Enumeration of Turing Machine

By fixing an effective bijection from $\{0,1\}^*$ to $\mathbf{N}$, the set of natural numbers, we obtain

$$\mathbb{M}_0, \mathbb{M}_1, \ldots, \mathbb{M}_i, \ldots.$$

The functions defined by these machines are normally denoted by

$$\phi_0, \phi_1, \ldots, \phi_i, \ldots.$$

Universal Turing Machine and Efficient Simulation

Theorem. There is a universal TM $\mathbb{U}$ that renders true the following statements.

1. $\mathbb{U}(x,\alpha) \simeq \mathbb{M}_\alpha(x)$ for all $x,\alpha \in \{0,1\}^*$. [this is Turing's universal machine]
2. If $\mathbb{M}_\alpha(x)$ halts in $T(|x|)$ steps, then $\mathbb{U}(x,\alpha)$ halts in $cT(|x|)\log T(|x|)$ steps, where c is a polynomial of α. [c is independent of $|x|$.]

- The version with $O(T(n))$ time amplification appeared in 1965.
- The present version was published in 1966.

---

1. J. Hartmanis and R. Stearns. On the Computational Complexity of Algorithms. Transactions of AMS, 117:285-306, 1965.
2. F. Hennie and R. Stearns. Two-Tape Simulation of Multitape Turing Machines. Journal of ACM, 13:533-546, 1966.

Proof of Hennie and Stearns

A universal TM has a fixed number of work tapes. How does it deal with a source TM with an unknown number of work tapes?

The solution is to use two work tapes:

- The main tape simulates all the work tapes of the source TM.
- The scratch tape is used to record current state, to indicate zone boundaries, and to speed up shifting.

Proof of Hennie and Stearns

From three bidirectional worktapes to one bidirectional worktape

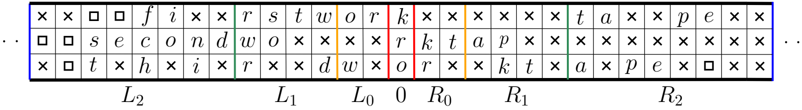
	×	×	□	□	f	i	×	×	r	s	t	w	o	r	k	×	×	×	×	×	×	t	a	×	×	p	e	×	×	
· ·	□	□	s	e	c	o	n	d	w	o	×	×	×	×	r	k	t	a	p	×	×	×	×	×	×	×	×	×	×	· ·
	×	□	t	×	h	×	i	×	r	×	×	d	w	×	o	r	×	×	k	t	×	a	×	p	e	×	□	×	×	
				L_2						L_1			L_0		0	R_0			R_1						R_2					

Imagine that the head is fixed and the tapes are shifting in opposite directions.

One may perceive that a symbol stored in the main tape of $\mathbb{U}$ is a tuple, say (k, r, o).

- The symbols k, r, o are encoded by strings of $\mathbb{U}$.
- The tuple (k, r, o) is also encoded by a string of $\mathbb{U}$.
- $\mathbb{U}$ has to perform a sequence of computation steps to overwrite (k, r, o).
  - the simulation overhead does not depend on $|x|$.

Proof of Hennie and Stearns



The work tape of $\mathbb{U}$ is split into zones.

$$\ldots \mid L_{\log(T)} \mid \ldots \mid L_1 \mid L_0 \mid \_ \mid R_0 \mid R_1 \mid \ldots \mid R_{\log(T)} \mid \ldots$$

where $R_i = [2^{i+1} - 1, 2^{i+2} - 2]$, $L_i = [-2^{i+2} + 2, -2^{i+1} + 1]$ and $|R_i| = |L_i| = 2 \cdot 2^i$.

The universal TM makes use of a special symbol $\times$ for buffer cells.

Proof of Hennie and Stearns

	×	×	□	□	f	i	×	×	r	s	t	w	o	r	k	×	×	×	×	×	×	t	a	×	×	p	e	×	×	
$\cdot\cdot$	□	□	s	e	c	o	n	d	w	o	×	×	×	×	r	k	t	a	p	×	×	×	×	×	×	×	×	×	×	$\cdot\cdot$
	×	□	t	×	h	×	i	×	r	×	×	d	w	×	o	r	×	×	k	t	×	a	×	p	e	×	□	×	×	
					L_2						L_1		L_0		0	R_0			R_1						R_2					

Constraint on the zones: For each $i \in \{0, \ldots, \log(T)\}$,

- L_i is full $\Leftrightarrow$ R_i is empty.
- L_i is half full $\Leftrightarrow$ R_i is half full.
- L_i is empty $\Leftrightarrow$ R_i is full.

The location 0 always contains a non-$\times$ symbol.

$\mathbb{U}$ builds up the zones while the simulation proceeds. The extra overhead is $O(T\log(T))$.

Proof of Hennie and Stearns

Simulating head-moving-to-right by tape-going-to-left:

×	×	□	□	f	i	×	×	r	s	t	w	o	r	k	×	×	×	×	×	×	t	a	×	×	p	e	×	×
□	□	s	e	c	o	n	d	w	o	×	×	×	×	r	k	t	a	p	×	×	×	×	×	×	×	×	×	×
×	□	t	×	h	×	i	×	r	×	×	d	w	×	o	r	×	×	k	t	×	a	×	p	e	×	□	×	×
				L_2					L_1			L_0		0	R_0			R_1						R_2				

$$\Downarrow$$

□	□	f	i	r	s	t	w	×	×	o	r	×	k	t	a	×	p	e	×	×	×	×	×	×	×	×	×	×
□	□	s	e	c	o	n	d	w	o	×	×	×	×	r	k	t	a	p	×	×	×	×	×	×	×	×	×	×
×	□	t	×	h	×	i	×	r	×	×	d	w	×	o	r	×	×	k	t	×	a	×	p	e	×	□	×	×
				L_2					L_1			L_0		0	R_0			R_1						R_2				

The total cost of shifting $= O(2^i)$.

- We need the scratch tape to act as transit storage while the machine is doing shifting.

Proof of Hennie and Stearns

After performing the shift with index i, it takes at least

- $2^i - 1$ right shifts before $L_{i-1}, \ldots, L_0$ become empty; and
- $2^i - 1$ left shifts before $R_{i-1}, \ldots, R_0$ become empty.

In other words, once a shift with index i is performed, the next $2^i - 1$ shifts of that parallel tape only involves indexes less than i.

Consequently there are at most $k\frac{T}{2^i}$ shifts with index i.

$$\sharp(\text{shift}) = O\left(k \sum_{i=1}^{\log(T)} \frac{T}{2^i} 2^i\right) = O\left(T \log(T)\right).$$

Theorem. Suppose L is computed by an $O(T(n))$ time TM for time constructible T. Then there is an oblivious TM that decides L in time $O(T(n)\log T(n))$.

Modify the construction of $\mathbb{U}$ as follows:

1. Mark all the zones with $\times$ and $\Box$. [$T(n)\log T(n)$ is time constructible.]
2. Copy the input to the worktape so that it interleaves with $\times$.
3. Rearrange the contents of zones in an oblivious fashion.

The third task can be accomplished by maintaining a counter.

- Before each simulating step, increase the counter by one.
- If the i-th bit of the counter value has just turned from 0 to 1, the machine sweeps, for a fixed number of time, the zones $L_i, \ldots, L_0, R_0, \ldots, R_i$ to incur a rearrangement.

Corollary (Hennie and Stearns, 1966). If f is computable in time $T(n)$ by a TM using k read-write tapes, then it is computable in time $O(T(n) \log T(n))$ by a TM with two read-write tapes.

Universal Machine and Diagonalization

The existence of universal machines (of all kind) allows one to establish impossibility results using diagonal simulation.

Impossibility via Diagonalization

Using universal TM one can define $\mathtt{UC}$ as follows:

$$\mathtt{UC}(\alpha) = \begin{cases} 0, & \text{if } \mathbb{M}_\alpha(\alpha) = 1, \\ 1, & \text{otherwise.} \end{cases}$$

Alternatively,

$$\mathtt{UC}(\alpha) = \begin{cases} 0, & \text{if } \mathbb{U}(\alpha, \alpha) = 1, \\ 1, & \text{otherwise.} \end{cases}$$

Had some TM $\mathbb{W}$ computed $\mathtt{UC}$, one would have that $\mathbb{W}(\llcorner \mathbb{W} \lrcorner) = 0$ iff $\mathbb{W}(\llcorner \mathbb{W} \lrcorner) = 1$.

Impossibility via Reduction

The halting problem HALT is defined by

$$\mathtt{HALT}(\alpha, x) = \begin{cases} 1, & \text{if } \mathbb{M}_\alpha(x) \text{ terminates,} \\ 0, & \text{otherwise.} \end{cases}$$

Theorem. HALT is not computable by any TM.

Proof.

If HALT were computable by some $\mathbb{M}_{\mathtt{H}}$, then

$$\mathbb{M}_{\mathtt{U}}(\alpha) = \begin{cases} 0, & \text{if } \mathbb{M}_{\mathtt{H}}(\alpha, \alpha) = 1 \wedge \mathbb{M}_\alpha(\alpha) = 1, \\ 1, & \text{otherwise.} \end{cases}$$

would compute UC. □

Speedup Theorem

Given a problem, is there always a best algorithm that solves it?

Blum's Speedup Theorem answers the question in the negative in a most forceful manner one can imagine.

1. Manuel Blum. A Machine-Independent Theory of the Complexity of Recursive Functions. Journal of ACM, 1967.

Blum Complexity Measure

(ϕ_i, Φ_i) is a Blum complexity measure if the following hold:

1. $\Phi_i(x)$ is defined if and only if $\phi_i(x)$ is defined.
2. $\Phi_i(x) \leq n$ is decidable.

Blum Time Function

Given a TM $\mathbb{M}$, the Blum time function $\mathtt{time}_{\mathbb{M}}(x)$ is defined by

$$\mathtt{time}_{\mathbb{M}}(x) = \mu t.(\mathbb{M}(x) \text{ terminates in } t \text{ steps}).$$

We write $\mathtt{time}_i(x)$ for $\mathtt{time}_{\mathbb{M}_i}(x)$.

Fact. $(\phi_i, \mathtt{time}_i)$ is a Blum complexity measure.

Main Lemma. Let r be a total computable function. There is a total computable function f such that given any TM $\mathbb{M}_i$ for f we can construct effectively a TM $\mathbb{M}_j$ with the following properties:

(I) ϕ_j is total and $\phi_j(x) = f(x)$ a.e. (almost everywhere).

(II) $r(\mathtt{time}_j(x)) < \mathtt{time}_i(x)$ a.e..

- for every speedup function r, say $r(z) = e^z$,
- there is a problem f
- such that for any algorithm I that computes f
- one can construct effectively an algorithm J that speeds up I by a ration of r and
- that solves f almost everywhere.

Proof of Main Lemma

By the S-m-n Theorem, there is a total primitive recursive function s such that

$$\phi_{s(e,u)}(x) \simeq \phi_e^{(2)}(u,x). \tag{2}$$

According to the Recursion Theorem there exists some e such that

$$\phi_e^{(2)}(u,x) \simeq g(e,u,x), \tag{3}$$

where $g(e,u,x)$ is obtained by the diagonalisation construction to be described next.

Intuitively $j = s(e, i+1)$.

Proof of Main Lemma

Suppose some finite canceled sets $C_{e,u,0}, \ldots, C_{e,u,x-1}$ are defined.

If $\texttt{time}_{s(e,i+1)}(x)$ is defined for all $i \in \{u, \ldots, x-1\}$, then let

$$C_{e,u,x} = \{i \mid u \le i \le x-1,\ \texttt{time}_i(x) \le r(\texttt{time}_{s(e,i+1)}(x))\} \setminus \bigcup_{y<x} C_{e,u,y};$$

otherwise $C_{e,u,x}$ is undefined.

- Clearly $C_{e,u,x}$ is computable, and if $i \in C_{e,u,x}$ then $\phi_i(x)\downarrow$.

Now $g(e, u, x)$ is defined by diagonalization.

$$g(e,u,x) = \begin{cases} 1 + \max\{\phi_i(x) \mid i \in C_{e,u,x}\}, & \text{if } C_{e,u,x} \text{ is defined,} \\ \uparrow, & \text{otherwise.} \end{cases}$$

Proof of Main Lemma

Fact. $g(e, u, x)$ is a total function.

Proof.

By induction suppose $g(e, u, y)$ is defined for all $y < x$.

- If $u \geq x$, then $C_{e,u,x} = \emptyset$ and consequently $g(e, u, x) = 1$.
- Suppose $u < x$ and $g(e, x, x)$, ..., $g(e, u+1, x)$ are defined.
  - $\phi_{s(e,x)}(x)$, ..., $\phi_{s(e,u+1)}(x)$ are defined by (2) and (3).
  - Hence $\mathtt{time}_{s(e,x)}(x)$, ..., $\mathtt{time}_{s(e,u+1)}(x)$ are defined.
  - It follows that $C_{e,u,x}$ is defined.
  - Consequently $g(e, u, x)$ is also defined.

This completes the downward induction. □

Corollary. For each u the function $\phi_{s(e,u)}(x)$ is total.

Proof of Main Lemma

(I). Some v exists such that $\phi_e^{(2)}(0,x) = \phi_e^{(2)}(u,x)$ for all $x > v$.

Proof.

For each $i < u$ if i appears in $C_{e,u,y}$, then it disappears from $C_{e,u,x}$ for all $x > y$. Let

$$v = \max\{y \mid C_{e,0,y} \text{ contains an index } i < u\}.$$

It is easy to see that $C_{e,0,x} = C_{e,u,x}$ for all $x > v$. □

Now let

$$f(x) = \phi_e^{(2)}(0,x).$$

It follows from (2) and (I) that $\phi_{s(e,u)}(x) = f(x)$ a.e..

Proof of Main Lemma

(II). If $\phi_i(x) = \phi_e^{(2)}(0,x)$ for all x, then $r(\texttt{time}_{s(e,i+1)}(x)) < \texttt{time}_i(x)$ a.e..

If not, i would have been canceled at some stage x, meaning that $i \in C_{e,0,x}$. By the definition of g,

$$\phi_i(x) \neq g(e,0,x) = \phi_e^{(2)}(0,x),$$

contradicting to the assumption.

Blum's Speedup Theorem. Let r be a total computable function. There is a total computable function f such that given any TM $\mathbb{M}_i$ for f there is some TM $\mathbb{M}_j$ for f such that $r(\mathtt{time}_j(x)) < \mathtt{time}_i(x)$ almost everywhere.

W.l.o.g. assume that r is increasing.

By a slight modification of the proof of Main Lemma, we obtain a total computable function f such that given any TM $\mathbb{M}_i$ for f there is a TM $\mathbb{M}_k$ satisfying the following:

1. $\phi_k(x)$ is total and $\phi_k(x) = f(x)$ a.e., and
2. $r(\mathtt{time}_k(x) + x) < \mathtt{time}_i(x)$ a.e..

Some c exists such that $\phi_k(x) = f(x)$ whenever $x > c$. We get a TM $\mathbb{M}_j$ from $\mathbb{M}_k$ by short-cutting computations at inputs $\leq c$.

If x is large enough such that the cost of the short-cutting computations is less than x, then $\mathbb{M}_j$ satisfies $r(\mathtt{time}_j(x)) < \mathtt{time}_i(x)$ a.e..

A less dramatic version of Speedup Theorem, historically appeared earlier, is the so-called Linear Speedup Theorem.

Linear Speedup Theorem

Theorem (Hartmanis and Stearns, 1965). If L is decidable in $T(n)$ time, then for any $\epsilon > 0$ the problem L is decidable in $\epsilon T(n) + n + 2$ time.

Suppose a TM $\mathbb{M} = (Q, \Gamma, \delta)$ accepts L in time $T(n)$.

Design $\widetilde{\mathbb{M}}$ such that a string of m symbols of $\mathbb{M}$ can be encoded by one symbol of $\widetilde{\mathbb{M}}$.

- $\widetilde{\mathbb{M}}$ converts the input in $n + 2$ steps.
- $\widetilde{\mathbb{M}}$ realigns the head in n/m steps.
- $\widetilde{\mathbb{M}}$ uses 5 steps to simulate m steps of $\mathbb{M}$.

The overall time is $\leq n + 2 + \frac{n}{m} + \frac{5}{m} T(n) \leq n + 2 + \frac{6}{m} T(n)$. So we let $m = 6/\epsilon$.

If $T(n) = \omega(n)$, the expression $\epsilon T(n) + n + 2$ can be replaced by $\epsilon T(n)$.

Message from Blum's Speedup Theorem:

- We cannot define time complexity for problems.
- We can of course define time complexity for solutions.

With this remark we proceed to investigate time complexity class.

Time Complexity Class

TIME(_)

Let $T : \mathbf{N} \to \mathbf{N}$ be a time function.

A decision problem $L \subseteq \{0,1\}^*$ is in $\mathbf{TIME}(T(n))$ if there exists a TM that accepts L and runs in time $cT(n)$ for some $c > 0$.

Complexity Class

Intuitively a complexity class is a set of problems having solutions that enjoy certain model independent properties.

In our sense $\mathbf{TIME}(n)$ is not a complexity class.

The Most Important Complexity Class

Strong Church-Turing Thesis. All physically realizable computing devices can be simulated by TM's with polynomial overhead.

$$\mathbf{P} = \bigcup_{c \geq 1} \mathbf{TIME}(n^c).$$

By Strong Church-Turing Thesis, $\mathbf{P}$ is model independent.

Philosophical Questions about $\mathbf{P}$

Identifying $\mathbf{P}$ to the class of feasible problems is a controversial issue.

---

1. Does $\mathbf{P}$ really characterize the class of feasible problems?
2. Is every problem in $\mathbf{P}$ provably in $\mathbf{P}$?

We might never understand a problem whose intrinsic complexity $1024^n/n^{1024}$.
We might understand a problem whose intrinsic complexity is $2^n/n^2$.

Complexity Class $\mathbf{EXP}$

$$\mathbf{EXP} = \bigcup_{c \geq 1} \mathbf{TIME}(2^{n^c}).$$

Verification Problem

For theoretical reasons we introduce a variant of Turing Machine that is not (believed to be) physically realizable.

Turing Machine with Nondeterministic Choice

A NDTM (Nondeterministic Turing Machine) has two transition functions δ_0, δ_1.

We say that $\mathbb{N}$ runs in $T(n)$ time if for every input $x \in \{0,1\}^*$, and every sequence of nondeterministic choices, $\mathbb{N}$ reaches $q_{\mathtt{halt}}$ within $T(|x|)$-steps.

The NDTM's can be effectively enumerated in the same way DTM's are enumerated.

$$\mathbb{N}_0, \mathbb{N}_1, \ldots, \mathbb{N}_i, \ldots.$$

Language Accepted by NDTM

Suppose $\mathbb{N}$ is an NDTM and x is an input.

An NDTM $\mathbb{N}$ accepts x, notation $\mathbb{N}(x) = 1$, if there is some sequence of choices that makes $\mathbb{N}(x)$ reach $q_{\mathtt{halt}}$ and output 1. Otherwise $\mathbb{N}$ refuses x, notation $\mathbb{N}(x) = 0$.

An NDTM $\mathbb{N}$ accepts $L \subseteq \{0,1\}^*$ if $x \in L \Leftrightarrow \mathbb{N}(x) = 1$.

Nondeterministic Turing machines do not have any computational strategy.

The main reason for introducing NDTM is that many problems, such as Vertex Cover, have simple solutions in terms of NDTM.

What nondeterminism provides is the power of guessing.

$\mathbf{NTIME}(\_)$

Suppose $T : \mathbf{N} \to \mathbf{N}$ and $L \subseteq \{0,1\}^*$.

$L \in \mathbf{NTIME}(T(n))$ if L is accepted by an NDTM $\mathbb{N}$ run in $cT(n)$ time for some $c > 0$.

Complexity Class via NDTM

$$\mathbf{NP} = \bigcup_{c \geq 1} \mathbf{NTIME}(n^c),$$

$$\mathbf{NEXP} = \bigcup_{c \geq 1} \mathbf{NTIME}(2^{n^c}).$$

$$\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{EXP} \subseteq \mathbf{NEXP}.$$

How about a universal nondeterministic TM for the nondeterministic TMs?

Snapshot

A snapshot of a k-tape TM $\mathbb{M}$ on some input x at step i is a tuple

$$\langle q, a_1, \ldots, a_k \rangle \in Q \times \underbrace{\Gamma \times \ldots \times \Gamma}_{k},$$

where $a_1, \ldots, a_k$ are the symbols in the cells the readers point to.

Unlike configurations the size of a snapshot depends only on $\mathbb{M}$, not on any input.

A ‘Universal’ Nondeterministic Turing Machine

A ‘universal’ NDTM $\mathbb{V}$ could be designed as follows:

1. It guesses a sequence of snapshots and a sequence of choices by running the input machine on the input value without looking at the worktapes.
2. It then verifies for each worktape of the input machine if the snapshots are legal.
   - To follow the content change of the tape being verified, $\mathbb{V}$ needs an additional tape.

- In the guessing phase, nothing stops $\mathbb{V}$ from guessing forever.
- The time to simulate an accepting run is linear in the run time.
- In application we often apply $\mathbb{V}$ to NDTMs with time constructible time functions.

$\mathbf{TIME}$ vs. $\mathbf{NTIME}$

There are almost no nontrivial results relating nondeterministic complexity class to deterministic complexity class. The following is a rare example.

Theorem (Paul, Pippenger, Szemerédi and Trotter, 1983). $\mathbf{TIME}(n) \subsetneq \mathbf{NTIME}(n)$.

- This is a non-relativizing result.
- Neither $\mathbf{TIME}(n)$ nor $\mathbf{NTIME}(n)$ is a complexity class.
- The proof draws inspiration from Hopcroft, Paul and Valiant's proof of

$$\mathbf{TIME}(S(n)) \subseteq \mathbf{SPACE}(S(n)/\log S(n)).$$

Time Hierarchy Theorem

Time Hierarchy Theorem

Theorem (Hartmanis and Stearns, 1965). If f and g are time constructible such that $f(n)\log f(n) = o(g(n))$, then $\mathbf{TIME}(f(n)) \subsetneq \mathbf{TIME}(g(n))$.

Consider L decided by the following Turing Machine $\mathbb{D}$:

On input x, simulate $\mathbb{M}_x$ on x in $g(|x|)$ steps.
If the simulation is finished in $g(|x|)$ steps, output $\overline{\mathbb{M}_x(x)}$.

By definition $L \in \mathbf{TIME}(g(n))$.

Suppose L were in $\mathbf{TIME}(f(n))$. Let L be decided by $\mathbb{M}_z$ with time bound $2f(n)$ such that z satisfies $f(|z|)\log f(|z|) < g(|z|)$. Then

- $\mathbb{D}(z) = \mathbb{M}_z(z)$ by assumption, and
- $\mathbb{D}(z) = \overline{\mathbb{M}_z(z)}$ since $\mathbb{D}$ can complete the simulation of $\mathbb{M}_z(z)$.

Exponential Hierarchy

$$
\begin{aligned}
\mathbf{EXP} &= \bigcup_{c>1} \mathbf{TIME}(2^{n^c}) \\
\mathbf{2EXP} &= \bigcup_{c>1} \mathbf{TIME}(2^{2^{n^c}}) \\
\mathbf{3EXP} &= \bigcup_{c>1} \mathbf{TIME}(2^{2^{2^{n^c}}}) \\
&\vdots \\
\mathbf{ELEMENTARY} &= \mathbf{EXP} \cup \mathbf{2EXP} \cup \mathbf{3EXP} \cup \ldots
\end{aligned}
$$

Nondeterministic Time Hierarchy

1. Cook showed that $\mathbf{NTIME}(n^{r(n)}) \subsetneq \mathbf{NTIME}(n^{r'(n)})$ if $1 \leq r(x) < r'(x)$.
2. Seiferas, Fischer and Meyer proved that $f(n+1) = o(g(n))$ implies

$$\mathbf{NTIME}(f(n)) \subsetneq \mathbf{NTIME}(g(n)).$$

3. Žák came up with a simpler proof of the separation result.

---

1. Cook. A Hierarchy for Nondetermintstic Time Complexity. Journal of Computer and System Sciences, 1973.
2. Seiferas, Fischer and Meyer. Separating Nondeterministic Time Complexity Classes. Journal of ACM, 1978.
3. Žák. A Turing Machine Time Hierarchy. Theoretical Computer Science, 1983.

Nondeterministic Time Hierarchy Theorem

Theorem. If f and g are time constructible such that $f(n+1) = o(g(n))$, then

$$\mathbf{NTIME}(f(n)) \subsetneq \mathbf{NTIME}(g(n)).$$

Žák's proof is given on the next two slides. An NDTM $\mathbb{Z}$ is defined on the next slide.

1. The head of the input tape and the head of the first worktape keep moving to right in synchrony at full speed.
   - If the input length is 1 or the input contains a 0, $\mathbb{Z}$ rejects.
   - The worktape head writes down 1 in the cell with index 1, it then keeps writing down 0, and occasionally it writes down 1.
   - Let $h_0, h_1, h_2, \ldots$ be the indices of the worktape cells with 1's, defined in Step 2.
2. In the second worktape, $\mathbb{Z}$ enumerates NDTMs hardwired with a $2f(n)$-timer. Let $\mathbb{L}_1, \mathbb{L}_2, \ldots$ be the enumeration.
   - Having generated $\mathbb{L}_i$, $\mathbb{Z}$ computes $\mathbb{L}_{i-1}(1^{h_{i-1}+1})$; $\mathbb{Z}$ then writes down $\mathbb{L}_{i-1}(1^{h_{i-1}+1})$ on the worktape and at the same time it marks 1 at location h_i on the first worktape.
3. Suppose the input is 1^n with $n > 1$. After scanning the input, do the following.
   3.1 If $n = h_i$ then $\mathbb{Z}$ accepts 1^n if and only if $\mathbb{L}_{i-1}(1^{h_{i-1}+1}) = 0$;
   3.2 If $h_{i-1} < n < h_i$ then $\mathbb{Z}$ simulates $\mathbb{L}_{i-1}$ on 1^{n+1} for $g(n)$ steps nondeterministically.

In Step 2, $\mathbb{Z}$ gets $\mathbb{L}_{i-1}$ and $1^{h_{i-1}+1}$ by looking back at the most recent history! $1^{h_{i-1}+1} - 1^{h_{i-2}+1}$

Let L be the language accepted by $\mathbb{Z}$.

1. $L \in \mathbf{NTIME}(g(n))$ since Step 3.1 costs no time and Step 3.2 costs at most $O(g(n))$-time.
2. $L \notin \mathbf{NTIME}(f(n))$.
   - Assume that some NDTM $\mathbb{L}_i$ accepted L in $2f(n)$-time.
   - Let i be so large that the nondeterministic simulation in Step 3.2 can be completed.

   Here is the contradiction.

$$\mathbb{L}_i(1^{h_i+1}) = \mathbb{Z}(1^{h_i+1}) = \mathbb{L}_i(1^{h_i+2}) = \mathbb{Z}(1^{h_i+2}) = \ldots = \mathbb{L}_i(1^{h_{i+1}}) = \mathbb{Z}(1^{h_{i+1}}) \neq \mathbb{L}_i(1^{h_i+1}).$$

- $=$, because both $\mathbb{Z}$ and $\mathbb{L}_i$ accept L,
- $=$, since the simulation in Step 3.2 can be completed, and
- $\neq$, due to the negation in Step 3.1.

The technique used in the proof is called lazy diagonalization.

By Time Hierarchy Theorem we have

$$\mathbf{TIME}(n^c) \subsetneq \mathbf{TIME}(2^{n^c}).$$

Is it true that the inequality

$$\mathbf{TIME}(b(n)) \subsetneq \mathbf{TIME}(2^{b(n)})$$

holds for all total computable function $b(x)$?

Gap Theorem

Theorem. For each total computable function $r(x) \geq x$, a total computable function $b(x)$ exists such that $\mathbf{TIME}(b(x)) = \mathbf{TIME}(r(b(x)))$.

1. Boris Trakhtenbrot. Turing Computations with Logarithmic Delay. Algebra and Logic 3(4):33-48, 1964. (in Russian)
2. Allan Borodin. Computational Complexity and the Existence of Complexity Gaps. Journal of the ACM 19(1):158-174, 1972.


Proof of Gap Theorem

Define a sequence of numbers $k_0 < k_1 < k_2 < \ldots < k_x$ by

$$\begin{aligned} k_0 &= 0, \\ k_{i+1} &= r(k_i) + 1, \quad \text{for } i < x. \end{aligned}$$

The $x+1$ intervals $[k_0, r(k_0)]$, $[k_1, r(k_1)]$, ..., $[k_x, r(k_x)]$ form a partition of $[0, r(k_x)]$.

Let $P(i, k)$ denote the following local (hence decidable) property:

- For every $j \le i$ and every input z to $\mathbb{M}_j$ such that $|z| = i$, either $\mathbb{M}_j(z)$ halts in k steps or it does not halt in $r(k)$ steps.

For each machine $\mathbb{M}_i$ we diagonalize on the input strings whose size is no more than i.

Proof of Gap Theorem

Let $n_i = \sum_{j=0}^{i} |\Gamma_j|^i$, the number of input of size i to $\mathbb{M}_0, \ldots, \mathbb{M}_i$.

- The n_i numbers of computation step cannot fill all of $[k_0, r(k_0)]$, ..., $[k_{n_i}, r(k_{n_i})]$.
- It follows that there is at least one $j \le n_i$ such that $P(i, k_j)$ is true.
- Let $b(i)$ be the least such k_j.

Thus $P(i, b(i))$ is true for all i.

Suppose that $\mathbb{M}_j$ accepts some L in $r(b(n))$ time.

- For every x with $|x| \ge j$ we know by definition that $\mathbb{M}_j(x)$ either halts in $b(|x|)$ steps or does not halt in $r(b(|x|))$ steps.

It follows that for sufficiently large x, $\mathbb{M}_g(x)$ halts in $b(|x|)$ steps.

By Time Hierarchy Theorem $b(x)$ is not time constructible.

Turing Award (1993) in recognition of their seminal paper that lays down the foundations for computational complexity theory.

The terminology computational complexity was introduced by Hartmanis and Stearns.

---

1. J. Hartmanis and R. Stearns. On the Computational Complexity of Algorithms. Transactions of AMS, 117:285-306, 1965.