comes to complexity issues.
Because of the simplicity of the TM (but not its

They are equivalent anyway.

describe a TM with pseudocode.
It is not without loss of generality, in most cases, to

We will skip the details.

Programming TMs
A configuration is a triple \((n, m, b)\) where \(n \in \mathbb{N}^*\), \(m \in \mathbb{N}^*\), and \(b \in \mathbb{B}\). The specification of a configuration is sufficient for the current state of the computation, and \(n \geq m\) is the string to the right of the cursor, and \(m \geq n\) is the string to the left of the cursor (inclusive).

What does your PC save before it enters the sleep mode?

The computation to continue as if it had not been stopped.

A configuration is a complete description of the configurations.
\[ \begin{align*}
\text{10001100001110011100011100} &= n \quad \star \\
\text{01001100001110011100011100} &= m \quad \star \\
\end{align*} \]
denoted by

That configuration yields

\[
(\langle n, m, b \rangle) \xrightarrow{W} (n, m, b)
\]

in (\langle n, m, b \rangle)

That configuration

\[
(\langle n, m, b \rangle) \xrightarrow{W} (n, m, b)
\]

If a step of \( W \) from configuration (\langle n, m, b \rangle) results in (\langle n, m, b \rangle)

\[
(\langle n, m, b \rangle) \xrightarrow{W} (n, m, b)
\]

step denoted

Configuration \( (n, m, b) \) yields (\langle n, m, b \rangle)

Fix a T.M. \( W \).

Yielding
of $x$.

The total number of steps is $O(n)$, where $n$ is the length.

- \textbf{The TM initially writes a in the first position.}
- \textbf{The TM moves the last symbol of $x$ to the right by one position, from right to left.}
- \textbf{The TM moves the next symbol to the right by one position.}
- \textbf{We want to compute $f(x) = ax$.}
- \textbf{Inserting a Symbol}

...
There is a matching lower bound of \( \Omega(n^2) \).

- This program takes \( O(n^2) \) steps.
- etc.

- The second character with the next to last character,
- It matches the first character with the last character,
- "yes" for palindromes and "no" for nonpalindromes.

A TM program can be written to recognize palindromes:

- and backwards (e.g., 001100).
- A string is a palindrome if it reads the same forwards and backwards.
Language

- Palindromes over \{0,1\} constitute a recursive recursive language.

If \( L \) is decided by some TM, then \( L \) is called a

We say \( M \) decides \( L \).

\[ "\text{"no"} = (x)_M \text{ if } x \not\in L \]

\[ "\text{"yes"} = (x)_M \text{ if } x \in L \]

Let \( M \) be a TM such that for every string \( x : \)

Let \( M \) be a TM with a finite length.

Decidability and Recursive Languages
recursively enumerable language.

If $T$ is accepted by some TM, then $T$ is called a

$\mathcal{W}$ accepts $T$.

$\checkmark = (x) \mathcal{W}$, then $T \not\in \mathcal{W}$

$\psi$ = $(x) \mathcal{W}$, then $T \in \mathcal{W}$

Let be a TM such that for any string $x$:

of symbols with a finite length.

Let $T$ be a language, i.e., a set of strings

Acceptability and Recursively Enumerable Languages
\[ L \text{ is clearly accepted by } M'. \]

Program

\[ M \text{ can be constructed by slightly modifying } M. \]

forever and never hails.

\[ M \text{ halts with a "no" state, } M \text{ moves its cursor to the right} \]

\[ M \text{ is identical to } M \text{ except that when } M \text{ is about to} \]

\[ L \text{ TM } M \text{ decides } L. \]

\[ \text{enumerable.} \]

Proposition 1. If \( L \) is recursive, then \( L \) is recursively enumerable.

Recursive and Recursively Enumerable Languages
We call a recursive function $f$ if such an exists.

\((x)f = (x)\mathcal{W}\)

Let $W$ be a TM with alphabet $\Xi$.

Let $\Xi$ be a TM with alphabet $\Psi$.

Turing-Computable Functions

Optimization problems, root finding problems, etc.
to be Turing-unecomputable (yet).

- No "infinitive computable" problems have been shown
  All have been proved to be equivalent.

- Various extensions of the Turing machine (more
  string, two-dimensional strings, and so on), etc.

- Shepherson & Sturgis, boolean circuits (Shannon)
  formal language (Post), assembly language-like RAM
  – Recursively function (Gödel), lambda calculus (Church)

- Many other computation models have been proposed.

- What is computable is Turing-computable, TMs are
  Church's theses of the Church-Turing theses
AII numbers will be binary from now on.

* 1001 vs. 11111111.

– The binary representation of numbers is succinct.  
– The unary representation of numbers is not succinct.

and as a linked list are both succinct.

– Representations of a graph as an adjacency matrix

  polynomially related.

All “reasonably succinct encodings” of problems are

Extended Church’s Theorems
When TM's compute functions, the output is on the last string. 

- Decidability and acceptability are the same as before.
- The first string contains the input.
- All strings start with a specific character.
- \( \forall \left( \{\leftarrow, \rightarrow\} \times \mathcal{X} \right) \times \left( \{\text{"no","yes"},\eta\} \cap \mathcal{Y} \right) \leftarrow \mathcal{X} \times \mathcal{Y} : \emptyset \)
  \( \forall \mathcal{X}, \mathcal{Y} \), \( \emptyset \) is as before.

\( (s, \mathcal{X}, \emptyset, \emptyset) = \mathcal{W} \)

A \( k \)-string Turing machine \( \mathcal{T} \mathcal{M} \) (\( k \)) is a quadruple Turing Machines with Multiple Strings.
A 2-String TM
steps.

symbols under the two cursors are identical at all
positions. The machine accepts the input if and only if the
machine has advanced to the end of the input.

- The two cursors are then moved in opposite
directions until the ends are reached.

- The last symbol of the input.

- The cursor of the second string is positioned at the
  symbol of the input.

- The cursor of the first string is positioned at the first
  symbol of the input.

- It copies the input to the second string.

A 2-string TM can decide palindromes in \(O(n^2)\) steps.

Palindromes Revisited
The k-string TMD's initial configuration is

\[ (s', e, e', c', e', \ldots, e, e') \]

\[ \text{where } \cdot \text{ is the } j\text{th string and the } j\text{th cursor is reading.} \]

\[ \left( w_1, w_2, \ldots, w_n \right) \]

The concept of configuration and yielding is the same as

Configurations and Yielding
Function $(u)f$ is a time bound for $M$. *(u)f \leq f(\|x\|)$ is the length of string $x$. 

If for any input string $x$, the time required by $M$ on $x$ is at most $f(\|x\|)$, then the time required by $M$ on $x$ is \( f(\|x\|) \). 

If for any input string $x$, the time required by $M$ on $x$ is \( t \) steps, then the time required by $M$ and input $x$, the $TM$ halts after time \( t \) steps. 

If for a $k$-string $TM$ and input $x$, the $TM$ halts after time \( t \) steps, then the $TM$ is the basis of our notion of the $TM$ halts after time \( t \) steps.

Time Complexity
- Palindromic is in \( \text{TIME} \), \( (u)f \) is a complexity class.

- \( (u)f \) is the set of languages decided by TMs within multiple strings operating with time bound \( (u)f \).

- \( (u)f \in \text{TIME} \) \( \exists \) \( T \) with \( (u)f \) multiple \( T \) operating in time is decided by a language complexity class

- Suppose language is decided by a

\( \text{TIME} \) Complexity Classes
The Simulation Technique

Given any k-string \( M \) operating within time \( f(n) \), there exists a (single-string) \( M \) operating within time \( f(n) \) such that \( (x)M = (x)M \) for any input \( x \).

By computation

Represent computation \((\#_{n_{1}}, n_{2}, \ldots, n_{m}, \#)\) of \( M \).

The single string of \( M \) implements the k strings of \( M \).

\( \text{Theorem 2} \)
of symbols and cursor movements of $M$. Then, changes the string to reflect the overwriting.

* The transition functions of $M$ must also reflect it.
* Remember them.

The states of $M$ must include $y$ to $\exists x \in X$ (the cursors).

$M$ scans the string to pick up the $k$ symbols under $w$.

To simulate each move of $M$:

\[
\begin{array}{c}
\{ \text{\textbf{\textgreater,\textless,\ldots,\textbf{\textgreater,\textless\textgreater}} \text{\textless,\textbf{\textgreater,\textless}} x,\text{\textless,\textbf{\textgreater,\textless}} \text{\textless}} \}\n\end{array}
\]

$k - 1$ pairs

The initial configuration of $M$ is

The proof (continued)
The proof (continued)

- $\mathcal{M}$ erases all strings of $\mathcal{M}$ except the last one.
- The simulation continues until $\mathcal{M}$ halts.
  - each such string.
  - The linear-time algorithm on p. 26 can be used for
  - then.
  - It is possible that some strings of $\mathcal{M}$ need to be
within time \((|x|) f \gamma y) O\) operates on \(W\) in \(|x|\) steps.

• As there are \(|x|\) steps of \(W\) to simulate, \(W\) in \(|x|\) steps is hence \((|x|) f \gamma y) O\) steps to write and, if needed, to lengthen the string.

• The total number of \(W\) steps is hence \((|x|) f \gamma y) O\) steps to collect information and

• Simultaneously each step, \(W\) takes, per string of \(W\)

• \((|x|) f \gamma y) O\) steps to write and, if needed, to lengthen the string.

• The total length of the string of \(W\) at any moment is \((|x|) f \gamma y) O\) steps to collect information and

• The total length of the string of \(W\) at any moment is \((|x|) f \gamma y) O\) steps to write and, if needed, to lengthen the string.

Since \(W\) halts within \(|x|\) steps, \(\gamma\) none of its strings ever becomes longer than \(|x|\).
can simulate m steps of \( M \) within six steps.

We encode m symbols of \( M \) in one symbol of \( M' \) so that

\[
\text{Set } \ell' = \max(\ell', 2).
\]

and which simulates \( M \).

Our goal is to construct a \( k \)-stirng \( \mathcal{A} \).

\[
(u)f \text{ operates within time }
\]

\[
(\mathcal{A}, \mathcal{Z}_{\ell'}, 0, s) = \mathcal{A}
\]

\[
L \text{ be decided by a } k \text{-stirng } \mathcal{A}
\]

\[
\ell + u + (u)f \in \mathcal{A}
\]

\[
\text{Let } L \in \text{TIME} \text{ then for any } \epsilon > 0,
\]

\[
\text{Theorem 3 } \text{Let } L \in \text{TIME} \text{ then for any } \epsilon > 0,
\]

Linear Speedup
This takes \(|m/|x|| + 2\) steps.

- Double because \(M\)'s has the start for remembering.

- \(M\)'s to the second string.

- \(\mathcal{Z}\) of the single symbol \((\omega_1 \cdots \omega_m) \in \mathcal{Z}\) to the block of \(m\) symbols of the input.

- Map each block of \(m\) symbols of the input.

- \(\mathcal{M}\)'s states corresponding to \(\mathcal{K} \times \mathcal{Z}\).

\[ m \in \mathcal{M} \cup \mathcal{Z} \]

The Proof (continued)
Compressing symbols: Enlarging the Word Length

3-ary representation, with $\prod = 2$.

$m = 3$.
The proof (continued)
$\forall \alpha = 1 + (r \mod \alpha) = \exists\beta$, $\exists \gamma = \lfloor \gamma / \beta \rfloor$, $\exists \delta = \gamma$, $\exists \epsilon = \gamma$. 

The Proof (continued)
predict the next \( m \) moves of \( W \).

Because no cursor of \( W \) can get out of the \( m \)-tuples, it needs states in \( \mathcal{M} \) — cursors.

This takes \( 4 \) steps.

Then moves all cursors to the left by one position, then to the right twice, and then to the left once.

The proof (continued)
The Proof (continued)
Choose \( m = \left\lfloor \frac{6}{\epsilon} \right\rfloor \) to complete the proof.

\[
\left| \frac{m}{(|x|)^{\theta}} \right| \times 6 + \frac{7}{2} + |x|
\]

The total number of \( M \) steps is at most

The total number of \( M \) steps is at most 6 per stage.

one for one of its two neighbors.

This takes 2 steps. One for the current \( m \)-tuple and

\( M \) moves of \( M \).

string contents and state brought about by the next \( m \)

\( M \)'s uses \( \theta \) function to implement the changes in

The Proof (continued)
This justification the asymptotic big-O notation.

Arbitrary linear speedup can be achieved.

made arbitrarily small.

constant in the leading term (1.4 in this example) can be

If \((u) = \frac{1}{2}n + 3n^2\), then the

\((u)\) is superlinear.

\((u)\) close to 1.

If \(cn < c\), then \(c\) can be made arbitrarily

We can trade state size for speed.

Implications of the Speedup Theorem
Think of $P$ as efficiently solvable problems.  

$$0 \leq \sum_{\gamma}^\cup \gamma \in \mathit{TIME}(n^\gamma).$$

denoted by $P$, that is,

The union of all polynomially decidable languages is

$$\sum_{\gamma}^\cup \gamma \in \mathit{TIME}(n^\gamma) \text{ for some } \gamma \in \mathbb{N}.$$

If $L$ is a polynomially decidable language, it is in

some $L \in \mathbb{L}$.  

$\gamma$ can be represented by its leading term $n_{\gamma}$ for

By the linear speedup theorem, any polynomial time

$P$
into the |s.

- The cursor of the input string does not wander off
  - The cursor never moves to the left.
  * The last string, the output string, is write-only.
- The input string is read-only.

- A k-string Turing machine with input and output
  - A k-string Turing machine that satisfies the following conditions.

Let \( k \geq 2 \) be an integer.

and output.

We do not want to change the space used only for input.

Clearing for Space
on $x$ is at most $(|x|)_f$.

If $M = \langle N \rangle$ for any input $x$, the space required by $M$ is $\mathbb{N} \leftarrow \mathbb{N} : f$.

For $(u) \in \mathbb{N} \leftarrow \mathbb{N}$, $M$ operates with space bound $(u)f$.

- If $M$ is a TM with input and output, then the space required by $M$ on input $x$ is $\mathbb{N}$.
- \[ |n^i_m| \leq \frac{1}{2} \sum_{j=1}^{2^m} \] then the space required is $\mathbb{N}$.
- If $M$ halts in configuration $n_1n_2\cdots n_jH$.
- Consider a $k$-string TM with input $x$.

Space Complexity
coefficients do not matter.

As in the linear speedup theorem (Theorem 3), constant

- palindrome is in SPACE($\log n$).

- SPACE is a set of languages.

and operates within space bound $L$.

If there is a TM with input and output that decides $L$

\[
((u)f) \in \text{SPACE}
\]

Then

Let $L$ be a language.

Space Complexity Classes
Determinism is a special case of nondeterminism.

If there exists a rule in $\exists$ that makes this happen, a configuration yields another configuration in one step more than one next steps—or none at all.

For each state-symbol combination, there may be a relation, not a function.

\[
\{\text{←, →}\} \times S \times (\{\text{"no", "yes"}, \text{"y"}\} \cap \mathcal{Y}) \leftarrow S \times \mathcal{Y} \supset \exists \nabla
\]

$S$, $\mathcal{Y}$'s are as before.

\[
(\nabla, S, \mathcal{Y}, s) = N \text{ an nonteterministic Turing machine (NTM) is a}
\]

Nondeterminism
Computation Tree and Computation Path
to a „yes“ state.

So if \( x \notin T \), then no nondeterministic choices should lead computation paths.

It is not required that the NTM halts in all

is a sequence of valid configurations that ends in „yes.“

decides if for any \( x \in X \) there

Let \( T \) be a language and \( N \) be an NTM.

Decidability under Nondeterminism
Possible that both $M$ and $M'$ accept $x$.

But if $M$ is an NTM, then $M'$ may not decide $I$.

• If $M$ is a TM, then $M'$ decides $I$.

• Let $M$ decide $I$, and $M'$ be $M$ after "yes" or "no".

Complementing a TM’s Halting States
A Nondeterministic Algorithm for Satisfiability

\[ \text{end if} \]
\[ \text{if} \ '\text{no}' \ then \]
\[ \text{else} \]
\[ \text{if} \ '\text{yes}' \ then \]
\[ \text{then} \]
\[ (u x, \ldots, x_1, x_1) \phi \]
\[ \text{end if} \]
\[ \text{end for} \]
\[ \text{Guess } x \in \{0, 1\} \]
\[ \text{for } i = 1, 2, \ldots, \text{ do} \]
\[ u \text{ is a boolean formula with } n \text{ variables.} \]
General Paradigm: Guess a “proof” and verify it.

- State
  - Truth assignment that results in the “Yes” path
  - If there is a computation $\phi$ – if is satisfiable
  - If the computation path corresponds to a particular truth assignment out of $2^n$.
- Every computation tree is a complete binary tree of depth $n$.
- The algorithm decides language $\{ \phi : \phi \text{ is satisfiable}\}$.

Analysis
Both problems are extremely hard.

- A total distance at most \( B \) where \( B \) is an input.
- The decision version TSP (D) asks if there is a tour with total distance of the shortest tour of the cities.
- The Traveling Salesman Problem (TSP) asks for the

We are given \( n \) cities \( 1, 2, \ldots, n \) and integer distances

The Traveling Salesman Problem
The degree of nondeterminism is $n$.

```
if end if
'no',
else
'yes',

if $B \subseteq 1^{+?}x^{+?}x \in \Sigma^*$ are distinct and $u, x, \ldots \in \Sigma^*$
{for convenience},
\{x \in \Sigma^* \}
\{for i \in \{1, 2, \ldots \} \}
\{u \in \Sigma^* \}
\{\text{the $i$th city} \}

do
```

A Nondeterministic Algorithm for TSP (D)
all the computation paths of the NTM.

Turning an NTM into a TM seems to require exploring

We change only the "depth" of the computation tree.

\[ (|x|) f \text{ longer than} \]

\[ \text{ for any } x \in \mathbb{N} \text{, if } \mathbb{N} \text{ does not have a computation path} \]

\[ \text{ and } \mathbb{N} \text{ decides } \mathcal{T} \]

\[ \text{ where } \mathbb{N} \leftarrow \mathbb{N} : f \text{ decides } \mathcal{T} \]

Non-deterministic machine \( \mathbb{N} \) decides \( \mathcal{T} \) in time \( u \).

Time Complexity under Non-determinism
\( \text{NTIME} = \text{NTIME} \)

- (u) \text{TIME}(u) \text{TIME}
- (u) \text{TIME}(u) \text{TIME}

\text{Time Complexity Class under Nondeterminism}
Computer science is if \( P = NP \).

- The most important open problem in theoretical computer science is \( P \neq NP \).
  - TSP (T).
  - Graph coloring.
  - Hamiltonian path.
  - Boolean satisfiability (SAT).

Think of NP as efficiently verifiable problems.

Clearly \( P \subseteq NP \).

\[ \forall \gamma > 0, \quad NP = \bigcap \mathcal{N}^{\gamma}(u) \cap \mathcal{N}^{\gamma}(u) \]

Define \( NP \).