What This Course Is All About

Computability: What can be computed?

- There exist *well-defined* problems that cannot be computed.
- In fact, "most" problems cannot be computed.

Complexity: What is a computable problem's inherent complexity?

- Some computable problems require at least exponential time and/or space; they are **intractable**.
- Some practical problems require superpolynomial resources unless certain conjectures are disproved.
- Other resource limits besides time and space?

Tractability and intractability

- Polynomial in terms of the input size n defines tractability.
  - n, $n \log n$, n^2, n^{90}.
  - Time, space, circuit size, random bits, etc.
- It results in a fruitful and practical theory of complexity.
- Few practical, tractable problems require a large degree.
- Exponential-time or superpolynomial-time algorithms are usually impractical unless correctness is sacrificed.
  - $n^{\log n}$, $2^{\sqrt{n}}$, 2^n, $n! \sim \sqrt{2\pi n}\,(n/e)^n$.

Growth of Factorials

n	$n!$	n	$n!$
1	1	9	362880
2	2	10	3628800
3	6	11	39916800
4	24	12	479001600
5	120	13	6227020800
6	720	14	87178291200
7	5040	15	1307674368000
8	40320	16	20,922,789,888,000

Most Important Results: a Sampler

- An operational definition of computability.
- Decision problems in logic are undecidable.
- Decisions problems on program behavior are usually undecidable.
- Complexity classes and the existence of intractable problems.
- Complete problems for a complexity class.
- Randomization and cryptographic applications.
- Approximability.

What Is Computation?

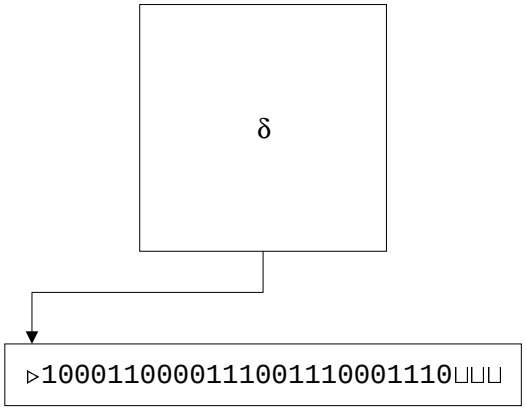
- That can be coded in an **algorithm**.
- An algorithm is a detailed step-by-step method for solving a problem.
  - The Euclidean algorithm for the greatest common divisor is an algorithm.
  - "Let s be the least upper bound of compact set A" is not an algorithm.
  - "Let s be a smallest element of a finite-sized array" can be solved by an algorithm.

Turing Machines[a]

- A Turing machine (TM) is a quadruple $M = (K, \Sigma, \delta, s)$.
- K is a finite set of **states**.
- $s \in K$ is the **initial state**.
- Σ is a finite set of **symbols** (disjoint from K).
  - Σ includes $\sqcup$ (blank) and $\triangleright$ (first symbol).
- $\delta : K \times \Sigma \to (K \cup \{h, \text{"yes"}, \text{"no"}\}) \times \Sigma \times \{\leftarrow, \rightarrow, -\}$ is a **transition function**.
  - $\leftarrow$ (left), $\rightarrow$ (right), and $-$ (stay) signify cursor movements.

[a]Turing (1936).

A TM Schema



"Physical" Interpretations

- The tape: computer memory and registers.
- δ: program.
- K: instruction numbers.
- s: "`main()`" in C.
- Σ: **alphabet** much like the ASCII code.

More about δ

- The program δ has the **halting state** (h), the **accepting state** ("yes"), and the **rejecting state** ("no").
- Given the current state $q \in K$ and the current symbol $\sigma \in \Sigma$,

  $$\delta(q, \sigma) = (p, \rho, D)$$

  specifies the next state p, the symbol ρ to be written over σ, and the direction D the cursor will move afterwards.
- We require $\delta(q, \triangleright) = (p, \triangleright, \rightarrow)$ so that the cursor never falls off the left end of the string.

The Operations of TMs

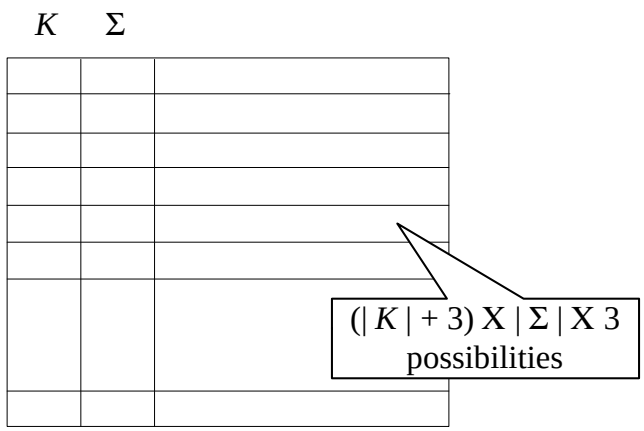
- Initially the state is s.
- The string on the tape is initialized to a $\triangleright$, followed by a *finitely* long string $x \in (\Sigma - \{\sqcup\})^*$.
- x is the **input** of the TM.
  - The input must not contain $\sqcup$s (why?)!
- The cursor is pointing to the first symbol, always a $\triangleright$.
- The TM takes each step according to δ.
- The cursor may overwrite $\sqcup$ to make the string longer during the computation.

Program Size

- A program has a *finite* size.
- The program δ is a function from $K \times \Sigma$ to $(K \cup \{h, \text{"yes"}, \text{"no"}\}) \times \Sigma \times \{\leftarrow, \rightarrow, -\}$.
- $|K| \times |\Sigma|$ lines suffice to specify such a function.
- Given K and Σ, there are

  $$((|K| + 3) \times |\Sigma| \times 3)^{|K| \times |\Sigma|}$$

  possible δ's.
  - This is a constant—albeit large.
- Different δ's may define the same behavior.



The Halting of a TM

- A TM M may **halt** in three cases.

  "yes": The machine **accepts** its input x, and $M(x) =$ "yes".

  "no": The machine **rejects** its input x, and $M(x) =$ "no".

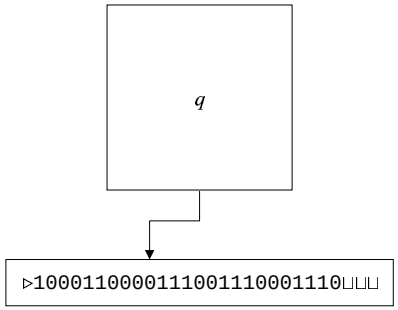
  ***h*:** $M(x) = y$, where the string consists of a $\triangleright$, followed by a finite string y, whose last symbol is not $\sqcup$, followed by a string of $\sqcup$s.
  - y is the **output** of the computation.
  - y may be empty denoted by ϵ.
- If M never halts on x, then write $M(x) = \nearrow$.

Why TMs?

- Because of the simplicity of the TM, the model has the advantage when it comes to complexity issues.
- One can develop a complexity theory based on C++ or Java, say.
- But the added complexity does not yield additional fundamental insights.
- We will describe TMs in pseudocode.

Configurations

- A **configuration** is a complete description of the current state of the computation.
- The specification of a configuration is sufficient for the computation to continue as if it had not been stopped.
  - What does your PC save before it sleeps?
  - Enough for it to resume work later.
- A configuration is a triple (q, w, u):
  - $q \in K$.
  - $w \in \Sigma^*$ is the string to the left of the cursor (inclusive).
  - $u \in \Sigma^*$ is the string to the right of the cursor.



- $w = \triangleright$1000110000.
- $u =$ 111001110001110.

Yielding

- Fix a TM M.
- Configuration (q, w, u) **yields** configuration (q', w', u') in one step, denoted

$$(q, w, u) \xrightarrow{M} (q', w', u'),$$

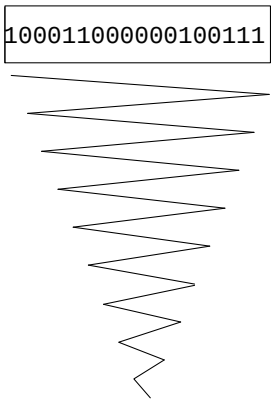
  if a step of M from configuration (q, w, u) results in configuration (q', w', u').
- That configuration (q, w, u) yields configuration (q', w', u') in $k \in \mathbb{N}$ steps is denoted by $(q, w, u) \xrightarrow{M^k} (q', w', u')$.
- That configuration (q, w, u) yields configuration (q', w', u') is denoted by $(q, w, u) \xrightarrow{M^*} (q', w', u')$.

Example: How to Insert a Symbol

- We want to compute $f(x) = ax$.
  - The TM moves the last symbol of x to the right by one position.
  - It then moves the next to last symbol to the right, and so on.
  - The TM finally writes a in the first position.
- The total number of steps is $O(n)$, where n is the length of x.

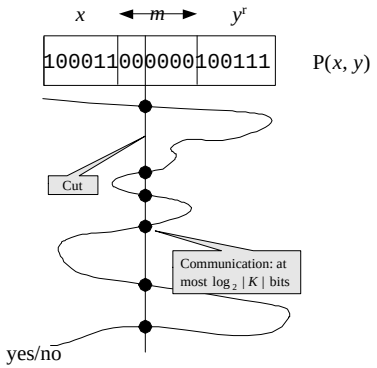
Palindromes

- A string is a **palindrome** if it reads the same forwards and backwards (e.g., `001100`).
- A TM program can be written to recognize palindromes: "yes" for palindromes and "no" for nonpalindromes.
  - It matches the first character with the last character.
  - It matches the second character with the next to last character, etc.
- This program takes $O(n^2)$ steps.



A Matching Lower Bound for PALINDROME

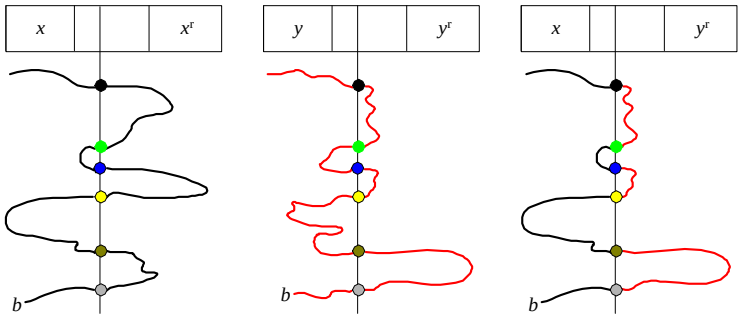
Theorem 1 PALINDROME *on single-string TMs takes* $\Omega(n^2)$ *steps in the worst case.*



The Proof: Communications

- Our input is more restricted; hence any lower bound holds for the original problem.
- Each communication between the two halves across the cut is a state from K, hence of size $O(1)$.
- $\mathrm{C}(x,x)$: the sequence of communications for palindrome problem $\mathrm{P}(x,x)$ across the cut.
- $\mathrm{C}(x,x) \neq \mathrm{C}(y,y)$ when $x \neq y$.
  - Otherwise, $C(x,x) = C(y,y) = C(x,y)$, and $\mathrm{P}(x,y)$ has the same answer as $\mathrm{P}(x,x)$!
- So $\mathrm{C}(x,x)$ is distinct for each x.

Cut and Paste



The Proof: Amount of Communications

- Assume $|\,x\,| = |\,y\,| = m = n/3$.
- $|\,\mathrm{C}(x,x)\,|$ is the number of times the cut is crossed.
- We first seek a lower bound on the total number of communications:

$$\sum_{x \in \{0,1\}^m} |\,\mathrm{C}(x,x)\,|.$$

- Define

$$\kappa \equiv (m+1)\log_{|\,K\,|} 2 - \log_{|\,K\,|} m - 1 + \log_{|\,K\,|}(|\,K\,| - 1).$$

The Proof: Amount of Communications (continued)

- There are $\leq |K|^i$ distinct $\mathrm{C}(x,x)$s with $|\mathrm{C}(x,x)| = i$.
- Hence there are at most

$$\sum_{i=0}^{\kappa} |K|^i = \frac{|K|^{\kappa+1}-1}{|K|-1} \leq \frac{|K|^{\kappa+1}}{|K|-1} = \frac{2^{m+1}}{m}$$

  distinct $\mathrm{C}(x,x)$s with $|\mathrm{C}(x,x)| \leq \kappa$.
- The rest must have $|\mathrm{C}(x,x)| > \kappa$.
- Because $\mathrm{C}(x,x)$ is distinct for each x (p. 35), there are at least $2^m - \frac{2^{m+1}}{m}$ of them with $|\mathrm{C}(x,x)| > \kappa$.

The Proof: Amount of Communications (concluded)

- Thus

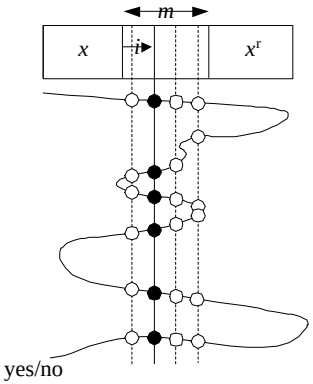
$$\begin{aligned}
\sum_{x\in\{0,1\}^m} |\mathrm{C}(x,x)| &\geq \sum_{x\in\{0,1\}^m, |\mathrm{C}(x,x)|>\kappa} |\mathrm{C}(x,x)| \\
&> \left(2^m - \frac{2^{m+1}}{m}\right)\kappa \\
&= \kappa 2^m \frac{m-2}{m}.
\end{aligned}$$

- As $\kappa = \Theta(m)$, the total number of communications is

$$\sum_{x\in\{0,1\}^m} |\mathrm{C}(x,x)| = \Omega(m2^m). \tag{1}$$

The Proof (continued)

We now lower-bound the worst-case number of communication points.



The Proof (continued)

- $\mathrm{C}_i(x,x)$ denotes the sequence of communications for $\mathrm{P}(x,x)$ given the cut.
- $T(n)$: the worst-case running time, when dealing with $P(y,y)$.
- $T(n) \geq \sum_{i=1}^{m} |\mathrm{C}_i(y,y)|$ as $T(n)$ is the *worst-case* time bound.
- Now,

$$\begin{aligned}
2^m T(n) = \sum_{x\in\{0,1\}^m} T(n) &\geq \sum_{x\in\{0,1\}^m} \sum_{i=1}^{m} |\mathrm{C}_i(x,x)| \\
&= \sum_{i=1}^{m} \sum_{x\in\{0,1\}^m} |\mathrm{C}_i(x,x)|.
\end{aligned}$$

The Proof (concluded)

- By the pigeonhole principle,[a] there exists an $0 \leq i^* \leq m$,

$$\sum_{x \in \{0,1\}^m} |\, \mathrm{C}_{i^*}(x,x) \,| \leq \frac{2^m T(n)}{m}.$$

- Eq. (1) on p. 39 says that

$$\sum_{x \in \{0,1\}^m} |\, \mathrm{C}_{i^*}(x,x) \,| = \Omega(m 2^m).$$

- Hence

$$T(n) = \Omega(m^2) = \Omega(n^2).$$

[a] Dirichlet (1805–1859).

Comments on Lower-Bound Proofs

- They are usually difficult.
  - Worthy of a Ph.D. degree.
- A lower bound that matches a known upper bound (given by an efficient algorithm) shows that the algorithm is optimal.
  - The simple $O(n^2)$ algorithm for PALINDROME is optimal.
- This happens rarely and is model dependent.
  - Searching, sorting, PALINDROME, matrix-vector multiplication, etc.

Decidability and Recursive Languages

- Let $L \subseteq (\Sigma - \{\sqcup\})^*$ be a **language**, i.e., a set of strings of symbols with a finite length.
  - For example, $\{0, 01, 10, 210, 1010, \ldots\}$.
- Let M be a TM such that for any string x:
  - If $x \in L$, then $M(x) =$ "yes."
  - If $x \notin L$, then $M(x) =$ "no."
- We say M **decides** L.
- If L is decided by some TM, then L is **recursive**.
  - Palindromes over $\{0,1\}^*$ are recursive.

Acceptability and Recursively Enumerable Languages

- Let $L \subseteq (\Sigma - \{\sqcup\})^*$ be a language.
- Let M be a TM such that for any string x:
  - If $x \in L$, then $M(x) =$ "yes."
  - If $x \notin L$, then $M(x) = \nearrow$.
- We say M **accepts** L.
- If L is accepted by some TM, then L is a **recursively enumerable language**.
  - Technically, a recursively enumerable language can be generated by a TM, thus the name.

Recursive and Recursively Enumerable Languages

Proposition 2 *If L is recursive, then it is recursively enumerable.*

- Let TM M decide L.
- We next modify M's program to obtain M' that accepts L.
- M' is identical to M except that when M is about to halt with a "no" state, M' goes into an infinite loop.
- M' accepts L.

Turing-Computable Functions

- Let $f : (\Sigma - \{\sqcup\})^* \to \Sigma^*$.
  - Optimization problems, root finding problems, etc.
- Let M be a TM with alphabet Σ.
- M **computes** f if for any string $x \in (\Sigma - \{\sqcup\})^*$, $M(x) = f(x)$.
- We call f a **recursive function**[a] if such an M exists.

[a] Gödel (1931).

Church's Thesis or the Church-Turing Thesis

- What is computable is Turing-computable; TMs are algorithms (Kleene 1953).
- Many other computation models have been proposed.
  - Recursive function (Gödel), λ calculus (Church), formal language (Post), assembly language-like RAM (Shepherdson & Sturgis), boolean circuits (Shannon), extensions of the Turing machine (more strings, two-dimensional strings, and so on), etc.
- All have been proved to be equivalent.
- No "intuitively computable" problems have been shown not to be Turing-computable (yet).

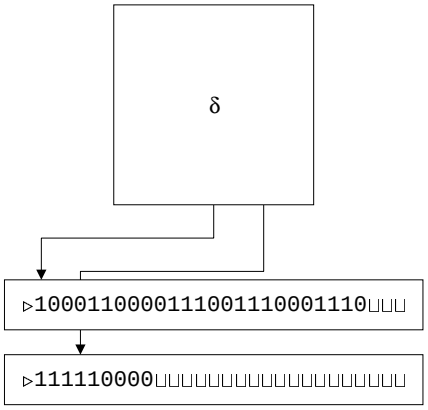
Extended Church's Thesis

- All "reasonably succinct encodings" of problems are *polynomially related*.
  - Representations of a graph as an adjacency matrix and as a linked list are both succinct.
  - The *unary* representation of numbers is not succinct.
  - The *binary* representation of numbers is succinct.
    - 1001 vs. 111111111.
- All numbers for TMs will be binary from now on.

Turing Machines with Multiple Strings

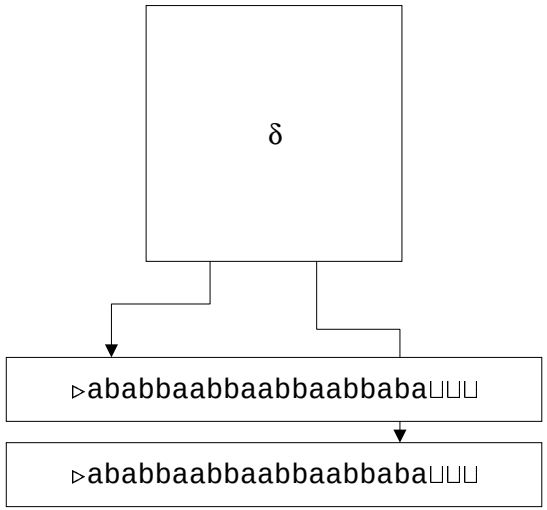
- A k-string Turing machine (TM) is a quadruple $M = (K, \Sigma, \delta, s)$.
- K, Σ, s are as before.
- $\delta : K \times \Sigma^k \to (K \cup \{h, \text{“yes”}, \text{“no”}\}) \times (\Sigma \times \{\leftarrow, \rightarrow, -\})^k$.
- All strings start with a $\triangleright$.
- The first string contains the input.
- Decidability and acceptability are the same as before.
- When TMs compute functions, the output is on the last (kth) string.

A 2-String TM



PALINDROME Revisited

- A 2-string TM can decide PALINDROME in $O(n)$ steps.
  - It copies the input to the second string.
  - The cursor of the first string is positioned at the first symbol of the input.
  - The cursor of the second string is positioned at the last symbol of the input.
  - The two cursors are then moved in opposite directions until the ends are reached.
  - The machine accepts if and only if the symbols under the two cursors are identical at all steps.



Configurations and Yielding

- The concept of configuration and yielding is the same as before except that a configuration is a $(2k+1)$-triple

$$(q, w_1, u_1, w_2, u_2, \ldots, w_k, u_k).$$

  - $w_i u_i$ is the ith string.
  - The ith cursor is reading the last symbol of w_i.
  - Recall that $\rhd$ is each w_i's first symbol.
- The k-string TM's initial configuration is

$$(s, \overbrace{\rhd, x, \rhd, \epsilon, \rhd, \epsilon, \ldots, \rhd, \epsilon}^{2k}).$$

Time Complexity

- The multistring TM is the basis of our notion of the time expended by TM computations.
- If for a k-string TM M and input x, the TM halts after t steps, then the **time required by M on input x** is t.
- If $M(x) = \nearrow$, then the time required by M on x is ∞.
- Machine M **operates within time** $f(n)$ for $f : \mathbb{N} \to \mathbb{N}$ if for any input string x, the time required by M on x is at most $f(|\,x\,|)$.
  - $|\,x\,|$ is the length of string x.
  - Function $f(n)$ is a **time bound** for M.

Time Complexity Classes[a]

- Suppose language $L \subseteq (\Sigma - \{\sqcup\})^*$ is decided by a multistring TM operating in time $f(n)$.
- We say $L \in \text{TIME}(f(n))$.
- $\text{TIME}(f(n))$ is the set of languages decided by TMs with multiple strings operating within time bound $f(n)$.
- $\text{TIME}(f(n))$ is a **complexity class**.
  - PALINDROME is in $\text{TIME}(f(n))$, where $f(n) = O(n)$.

[a]Hartmanis and Stearns (1965), Hartmanis, Lewis, and Stearns (1965).

The Simulation Technique

Theorem 3 *Given any k-string M operating within time $f(n)$, there exists a (single-string) M' operating within time $O(f(n)^2)$ such that $M(x) = M'(x)$ for any input x.*

- The single string of M' implements the k strings of M.
- Represent configuration $(w_1, u_1, w_2, u_2, \ldots, w_k, u_k)$ of M by configuration

$$(q, \rhd w_1' u_1 \lhd w_2' u_2 \lhd \cdots \lhd w_k' u_k \lhd \lhd)$$

  of M'.
  - $\lhd$ is a special delimiter.
  - w_i' is w_i with the first and last symbols "primed."

The Proof (continued)

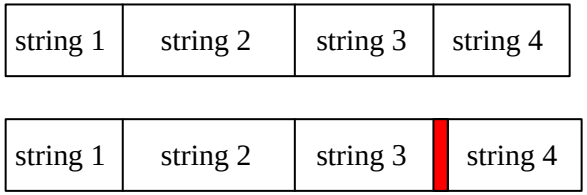
- The initial configuration of M' is

$$(s, \rhd \rhd' x \lhd \overbrace{\rhd' \lhd \cdots \rhd' \lhd}^{k-1 \text{ pairs}} \lhd).$$

- To simulate each move of M:
  - M' scans the string to pick up the k symbols under the cursors.
    - The states of M' must include $K \times \Sigma^k$ to remember them.
    - The transition functions of M' must also reflect it.
  - M' then changes the string to reflect the overwriting of symbols and cursor movements of M.

The Proof (continued)

- It is possible that some strings of M need to be lengthened.
  - The linear-time algorithm on p. 31 can be used for each such string.
- The simulation continues until M halts.
- M' erases all strings of M except the last one.
- Since M halts within time $f(|\,x\,|)$, none of its strings ever becomes longer than $f(|\,x\,|)$.
- The length of the string of M' at any time is $O(kf(|\,x\,|))$.



The Proof (concluded)

- Simulating each step of M takes, *per string of* M, $O(kf(|\,x\,|))$ steps.
  - $O(f(|\,x\,|))$ steps to collect information.
  - $O(kf(|\,x\,|))$ steps to write and, if needed, to lengthen the string.
- M' takes $O(k^2 f(|\,x\,|))$ steps to simulate each step of M.
- As there are $f(|\,x\,|)$ steps of M to simulate, M' operates within time $O(k^2 f(|\,x\,|)^2)$.

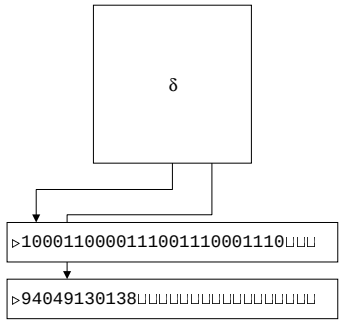
Linear Speedup

Theorem 4 *Let $L \in TIME(f(n))$. Then for any $\epsilon > 0$, $L \in TIME(f'(n))$, where $f'(n) = \epsilon f(n) + n + 2$.*

- Let L be decided by a k-string TM $M = (K, \Sigma, \delta, s)$ operating within time $f(n)$.
- Our goal is to construct a k'-string $M' = (K', \Sigma', \delta', s')$ operating within the time bound $f'(n)$ and which *simulates* M.
- Set $k' = \max(k, 2)$.
- We encode $m = \lceil 6/\epsilon \rceil$ symbols of M in *one* symbol of M' so that M' can simulate m steps of M within 6 steps.

The Proof (continued)

- $\Sigma' = \Sigma \cup \Sigma^m$.
- Phase one of M':
  - M' has states corresponding to $K \times \Sigma^m$.
  - Map each block of m symbols of the input $\sigma_1\sigma_2\cdots\sigma_m$ to the *single* symbol $(\sigma_1\sigma_2\cdots\sigma_m) \in \Sigma'$ of M' to the second string.
  - Doable because M' has the states for remembering.
- This phase takes $m\lceil |x|/m \rceil + 2$ steps.
  - The extra 2 comes from the enclosing symbols $\triangleright$ and $\bigsqcup$.

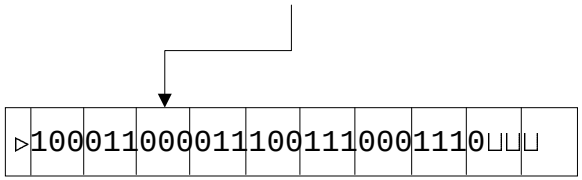


- $m = 3$.
- 3-ary representation, with $\bigsqcup \to 2$.

The Proof (continued)

- Treat the second string as the one containing the input.
  - If $k > 1$, use the first string as an ordinary work string.
- M' simulates m steps of M by six or fewer steps, called a **stage**.
- A stage begins with M' in state $(q, j_1, j_2, \ldots, j_k)$.
  - $q \in K$ and $j_i \le m$ is the position of the ith cursor within the m-tuple scanned.
  - If the ith cursor of M is at the ℓth symbol after $\triangleright$, then the $(i+1)$st cursor of M' will point to the $\lceil \ell/m \rceil$th symbol after $\triangleright$ and $j_i = ((\ell - 1) \bmod m) + 1$.

The Proof (continued)



- $m = 3$.
- $\ell = 8$.
- $\lceil \ell/m \rceil = \lceil 8/3 \rceil = 3$.
- $j_i = ((8-1) \bmod 3) + 1 = 2$.

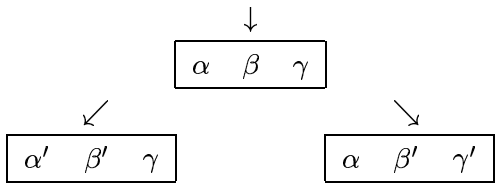
The Proof (continued)

- Then M' moves all cursors to the left by one position, then to the right twice, and then to the left once.
  - This takes 4 steps.
  - No cursor of M can in m moves get out of the m-tuples scanned by M' above.
- M' now "remembers" all symbols (of Σ') at or next to all cursors.
  - M' contains states in $K \times \{1, 2, \ldots, m\}^k \times \Sigma^{3mk}$.
  - That is an $m^k \cdot |\Sigma|^{3mk}$-fold increase.

The Proof (concluded)

- M' has all the information needed for the next m moves of M!
- Hard-code that into its program δ'.
- δ' now implements the changes in string contents and state brought about by the next m moves of M.
  - This takes 2 steps: One for the current m-tuple and one for one of its two neighbors.
- The total number of M' steps is at most 6 per stage.
- The total number of M' steps is at most

$$|x| + 2 + 6 \times \left\lceil \frac{f(|x|)}{m} \right\rceil \leq |x| + 2 + \epsilon f(|x|).$$



Implications of the Speedup Theorem

- State size can be traded for speed.
  - $m^k \cdot |\Sigma|^{3mk}$-fold increase to gain a speedup of $O(m)$.
- If $f(n) = cn$ with $c > 1$, then c can be made arbitrarily close to 1.
- If $f(n)$ is superlinear, say $f(n) = 14n^2 + 31n$, then the constant in the leading term (14 in this example) can be made arbitrarily small.
  - *Arbitrary* linear speedup can be achieved.
  - This justifies the asymptotic big-O notation.
- 1-bit, 4-bit, 8-bit, 16-bit, 32-bit, 64-bit, 128-bit CPUs, and so on.

P

- By the linear speedup theorem, any polynomial time bound can be represented by its leading term n^k for some $k \geq 1$.
- If L is a polynomially decidable language, it is in $\text{TIME}(n^k)$ for some $k \in \mathbb{N}$.
- The union of all polynomially decidable languages is denoted by P:

$$\text{P} = \bigcup_{k>0} \text{TIME}(n^k).$$

- Problems in P can be efficiently solved.

Charging for Space

- We do not want to charge the space used only for input and output.
- Let $k > 2$ be an integer.
- A **k-string Turing machine with input and output** is a k-string TM that satisfies the following conditions.
  - The input string is *read-only*.
  - The last string, the output string, is *write-only*.
  - So its cursor never moves to the left.
  - The cursor of the input string does not wander off into the $\sqcup$s.

Space Complexity

- Consider a k-string TM M with input x.
- We may assume $\sqcup$ is never written over a non-$\sqcup$ symbol.
- If M halts in configuration $(H, w_1, u_1, w_2, u_2, \ldots, w_k, u_k)$, then the **space required by M on input x** is $\sum_{i=1}^{k} |w_i u_i|$.
- If M is a TM with input and output, then the space required by M on input x is $\sum_{i=2}^{k-1} |w_i u_i|$.
- Machine M **operates within space bound $f(n)$** for $f : \mathbb{N} \to \mathbb{N}$ if for any input x, the space required by M on x is at most $f(|\, x \,|)$.

Space Complexity Classes

- Let L be a language.
- Then

$$L \in \text{SPACE}(f(n))$$

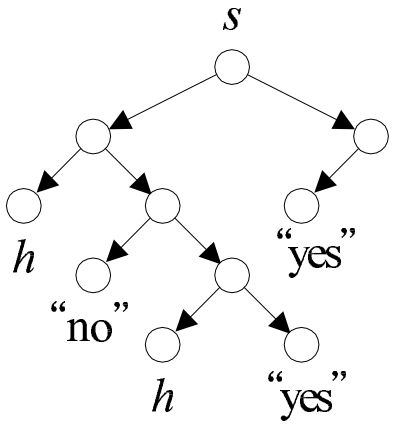
  if there is a TM with input and output that decides L and operates within space bound $f(n)$.
- $\text{SPACE}(f(n))$ is a set of languages.
  - PALINDROME $\in \text{SPACE}(\log n)$: Keep 3 pointers.
- As in the linear speedup theorem (Theorem 4), constant coefficients do not matter.

Nondeterminism[a]

- A **nondeterministic Turing machine** (**NTM**) is a quadruple $N = (K, \Sigma, \Delta, s)$.
- K, Σ, s are as before.
- $\Delta \subseteq K \times \Sigma \to (K \cup \{h, \text{“yes”}, \text{“no”}\}) \times \Sigma \times \{\leftarrow, \rightarrow, -\}$ is a relation, not a function.
  - For each state-symbol combination, there may be more than one next steps—or none at all.
- A configuration yields another configuration in one step if there *exists* a rule in Δ that makes this happen.

[a]Rabin and Scott (1959).

Computation Tree and Computation Path



Decidability under Nondeterminism

- Let L be a language and N be an NTM.
- N **decides** L if for any $x \in \Sigma^*$, $x \in L$ if and only if there is a sequence of valid configurations that ends in “yes.”
  - It is not required that the NTM halts in all computation paths.
- So if $x \notin L$, then no nondeterministic choices should lead to a “yes” state.
- Determinism is a special case of nondeterminism.

An Example

- Let L be the set of logical conclusions of a set of axioms.
- Consider the nondeterministic algorithm:
  1: $b :=$ `false`;
  2: **while** the input predicate $\phi \neq b$ **do**
  3: Generate a logical conclusion of b by applying some of the axioms; {Nondeterministic choice.}
  4: **end while**
  5: "yes";
- This algorithm decides L.

Complementing a TM's Halting States

- Let M decide L, and M' be M after "yes" $\leftrightarrow$ "no".
- If M is a (deterministic) TM, then M' decides $\bar{L}$.
- But if M is an NTM, then M' may not decide $\bar{L}$.
  - It is possible that both M and M' accept x.