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?

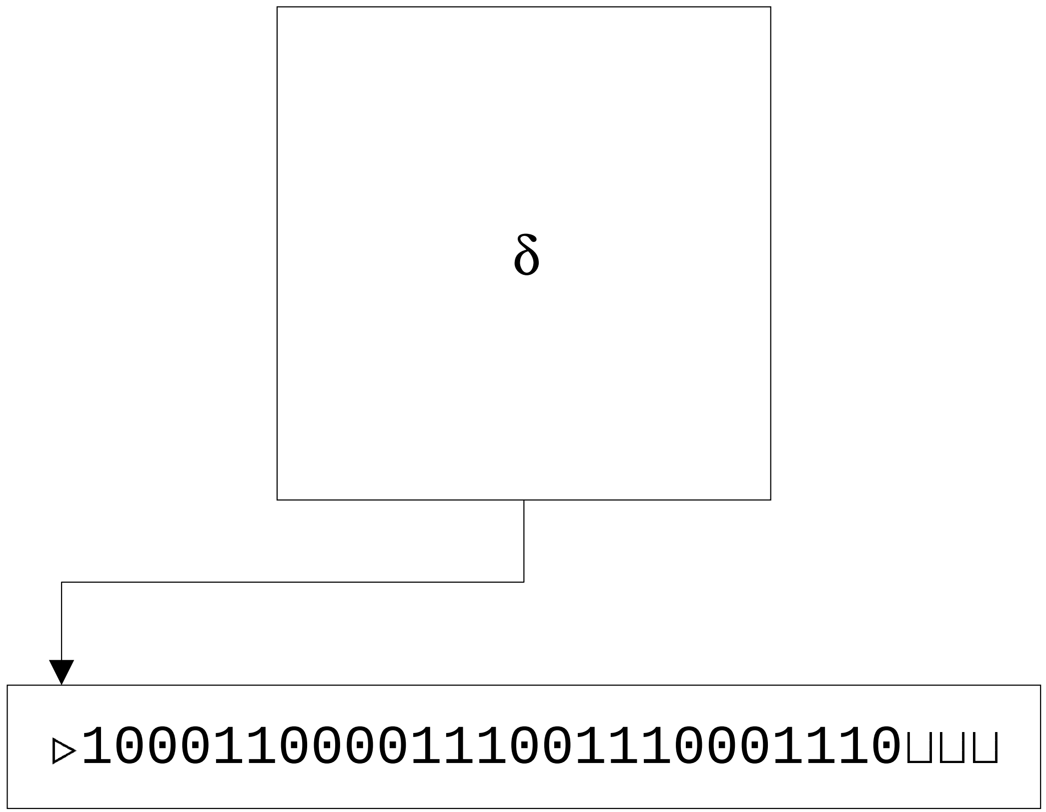
- That can be coded in an **algorithm**.
- An algorithm is a detailed step-by-step method for solving a problem.
  - The Euclidean algorithm for the greatest common divisor is an algorithm.
  - "Let s be the least upper bound of compact set A" is not an algorithm.
  - "Let s be a smallest element of 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.
- K: instruction numbers.
- s: “`main()`” in C.
- Σ: **alphabet** much like the ASCII code.

"Physical" Interpretations (concluded)

- δ: the program with 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 $\delta(q, \rhd) = (p, \rhd, \rightarrow)$ for convenience.

The Operations of TMs

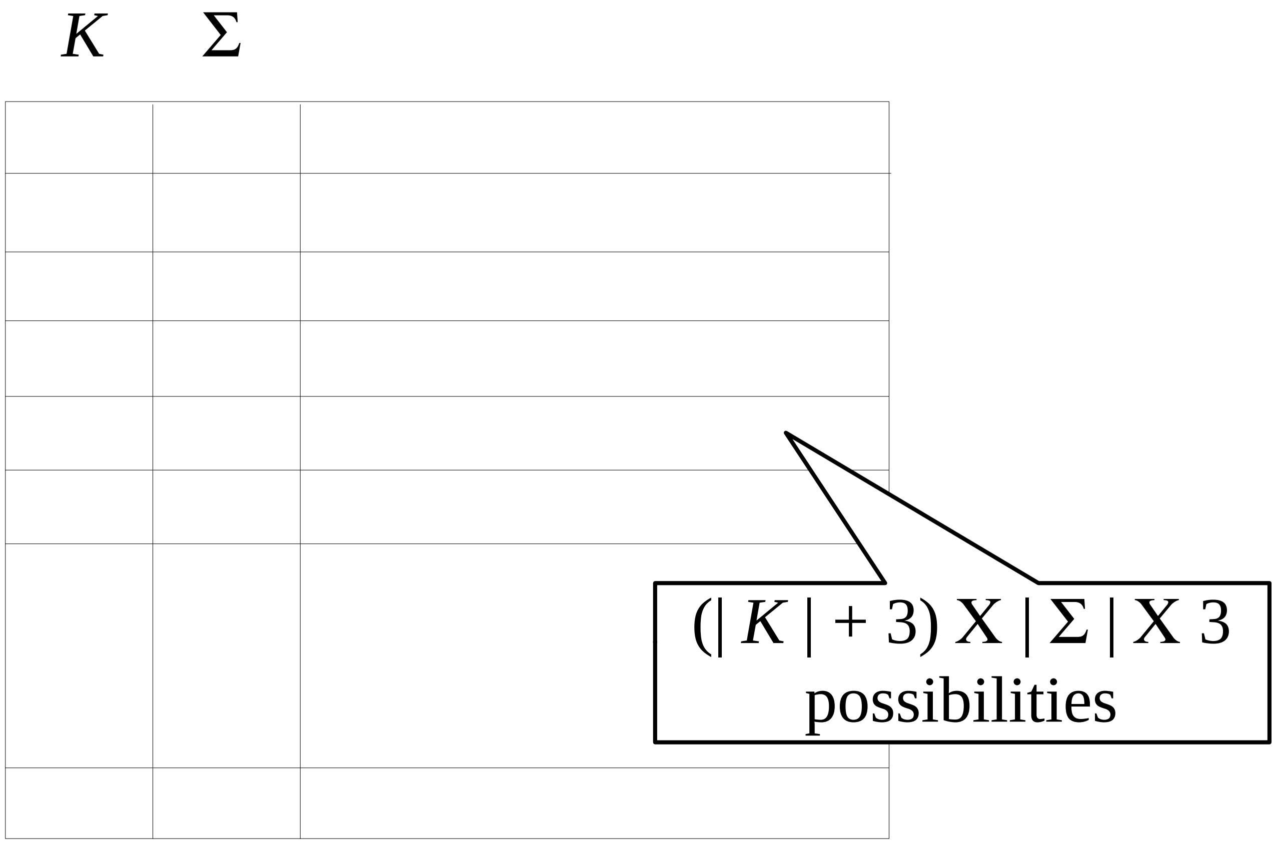
- Initially the state is s.
- The string on the tape is initialized to a $\triangleright$, followed by a *finitely* long string $x \in (\Sigma - \{\sqcup\})^*$.
- x is the **input** of the TM.
  - The input must not contain $\sqcup$s!
- The cursor is pointing to the first symbol, always a $\triangleright$.
- The TM takes each step according to δ.
- The cursor never falls off the left end of the string.
- The cursor may overwrite $\sqcup$ to make the string longer during the computation.

Program 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, a constant—albeit large.
  - A program must have a finite size.
- Different δ's may define the same behavior.


K
Σ
$(|K| + 3) \times |\Sigma| \times 3$
possibilities

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 called the **output** of the computation.
  - y may be empty denoted by ϵ.

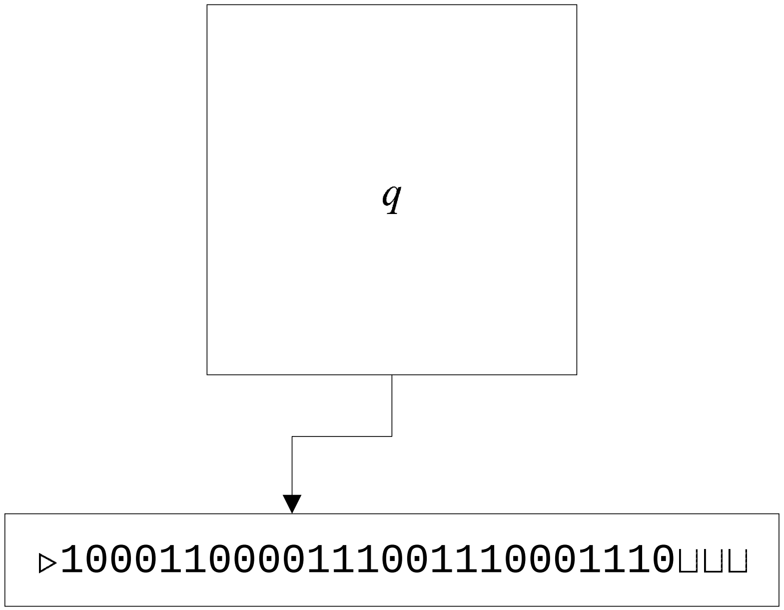
- If M never halts on x, then write $M(x) = \nearrow$.

Programming TMs

- We describe a TM in pseudocode.
- Because of the simplicity of the TM, the model has the advantage when it comes to complexity issues.
  - Imagine developing a complexity theory based on C++.

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

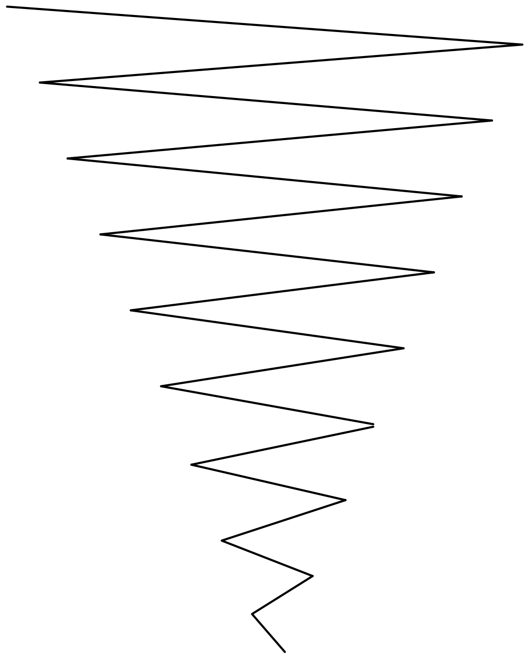
Inserting a Symbol

- 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

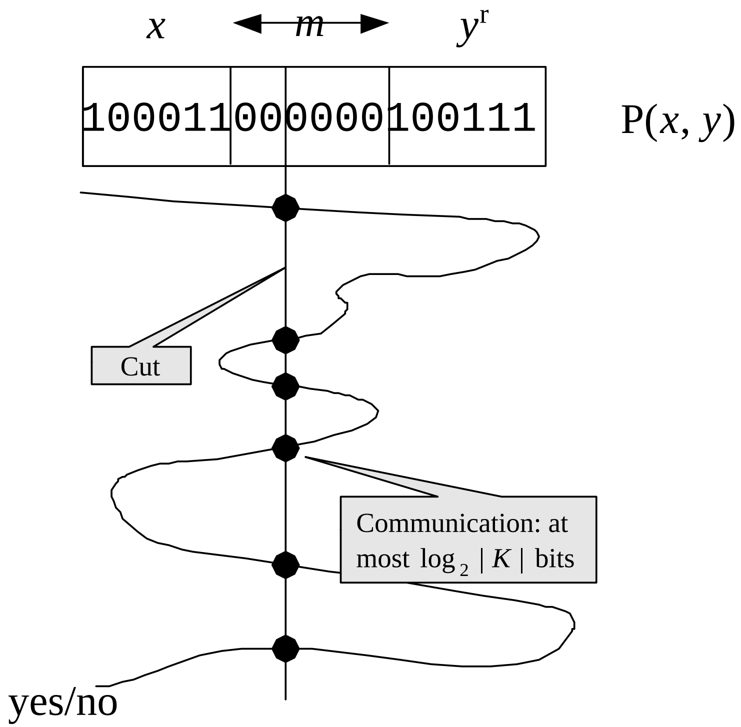
- A string is a **palindrome** if it reads the same forwards and backwards (e.g., `001100`).
- A TM program can be written to recognize palindromes: "yes" for palindromes and "no" for nonpalindromes.
  - It matches the first character with the last character, the second character with the next to last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

10001100000100111



A Matching Lower Bound for Palindrome

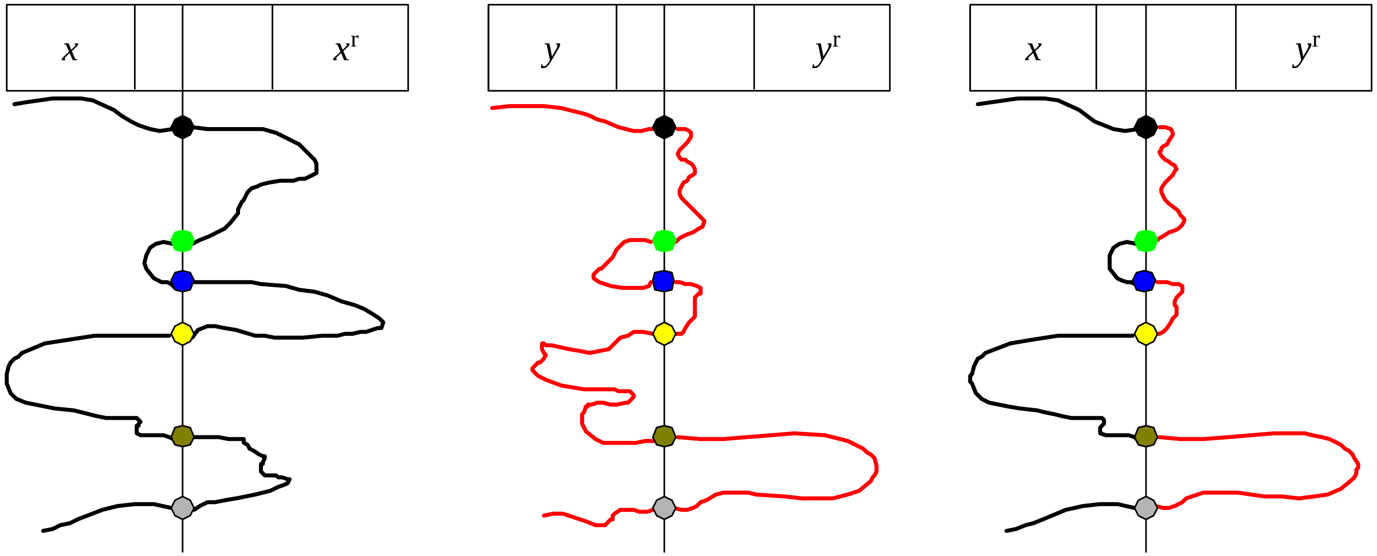
Theorem 1 *Palindrome on single-string TMs takes $\Omega(n^2)$ steps in the worst case.*



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 $\mathrm{P}(x, x)$!
- $\mathrm{C}(x, x)$ is distinct for each x.

The Proof: Cut and Paste



The Proof: Amount of Communications

- Assume $|x| = |y| = m = n/3$.
- 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

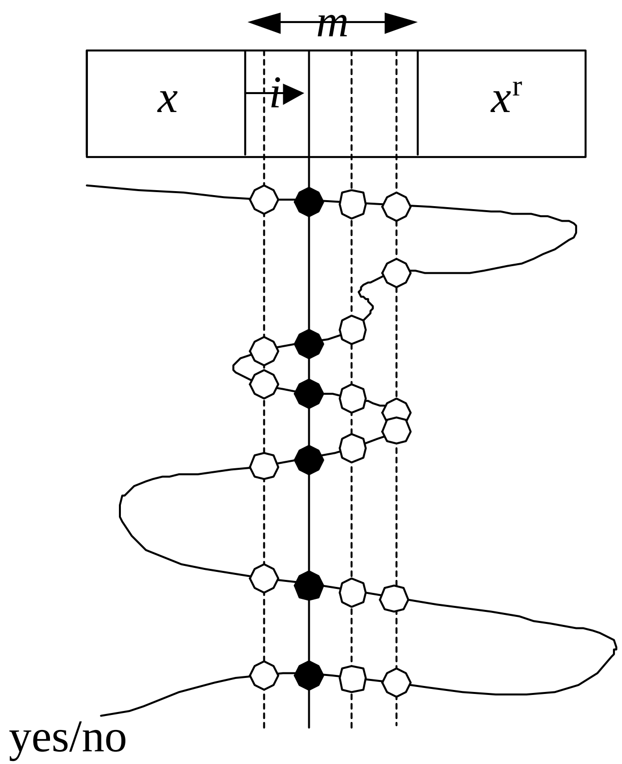
$$\begin{aligned} \sum_{x \in \{0,1\}^m} |\,\mathrm{C}(x,x)\,| &\geq \sum_{x \in \{0,1\}^m, |\,\mathrm{C}(x,x)\,| > \kappa} |\,\mathrm{C}(x,x)\,| \\ &> \kappa \left(2^m - \frac{2^{m+1}}{m} \right) \\ &= \kappa 2^m \frac{m-2}{m}. \end{aligned}$$

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

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

The Proof (continued)

We now lower-bound the 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 for x of length n.
- $T(n) \geq \sum_{i=1}^{m} |\, \mathrm{C}_i(x, x) \,|$.
- As $T(n)$ is the *worst-case* time bound,

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

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.
  - Often 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 is rare and 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 called **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 called a **recursively enumerable language**.

Recursive and Recursively Enumerable Languages

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

- Let TM M decides 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' is constructed by modifying M's program.
- 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** if such an M exists.

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 to be Turing-uncomputable (yet).

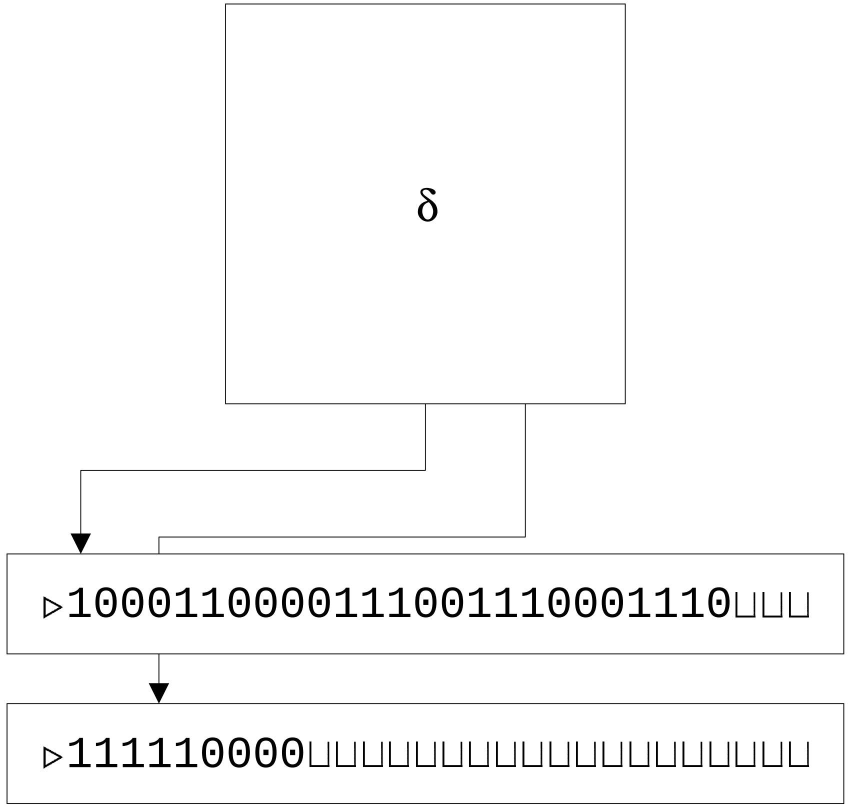
Extended Church's Thesis

- 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 will be binary from now on.

Turing Machines with Multiple Strings

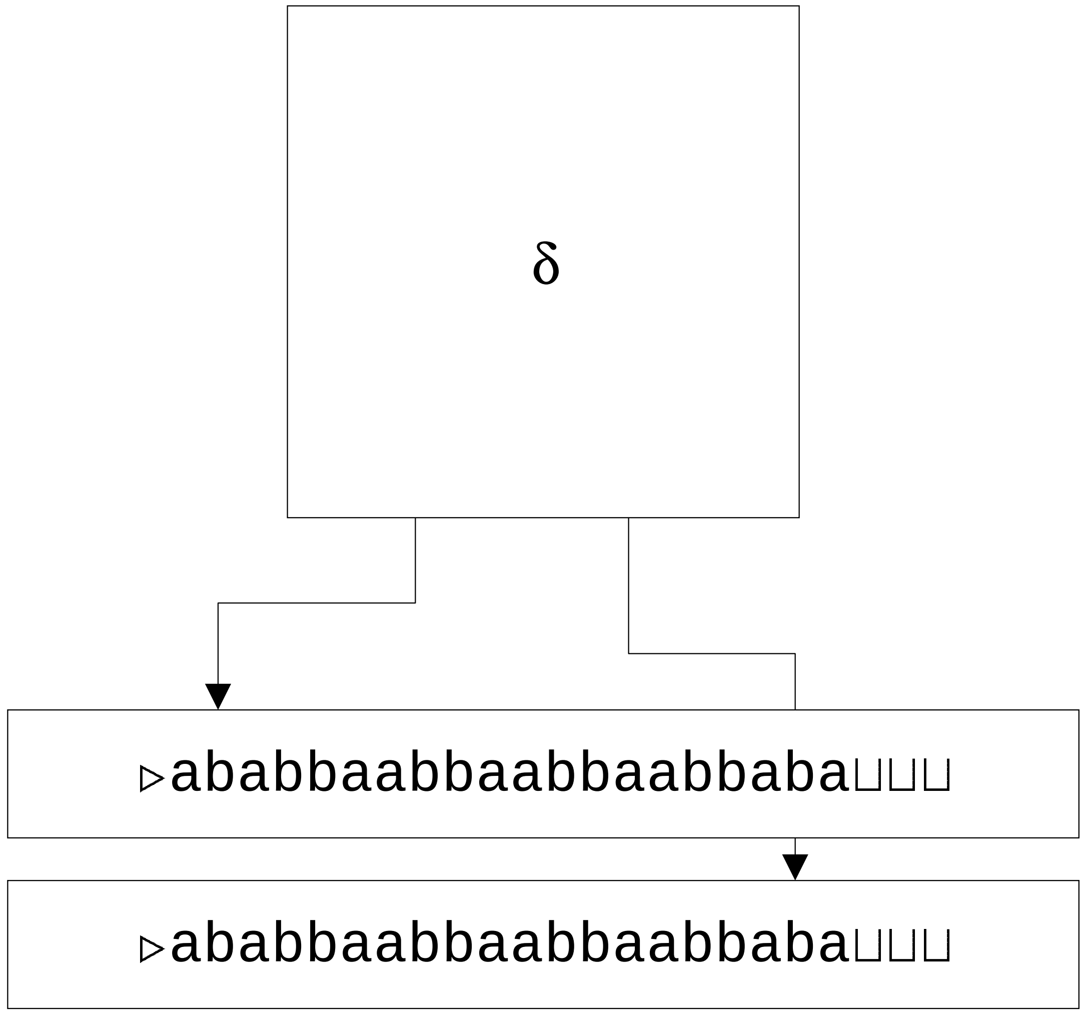
- A k-string Turing machine (TM) is a quadruple $M = (K, \Sigma, \delta, s)$.
- K, Σ, s are as before.
- $\delta : K \times \Sigma^k \to (K \cup \{h, \text{"yes"}, \text{"no"}\}) \times (\Sigma \times \{\leftarrow, \rightarrow, -\})^k$.
- All strings start with a $\triangleright$.
- The first string 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



Palindromes Revisited

- A 2-string TM can decide palindromes 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.


δ
▷ababbaabbaabbaabbaba⊔⊔⊔
▷ababbaabbaabbaabbaba⊔⊔⊔

Configurations and Yielding

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

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

where $w_i u_i$ is the ith string and the ith cursor is reading the last symbol of w_i.

  – Note 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 called a **complexity class**.
  - Palindrome is in $\text{TIME}(f(n))$, where $f(n) = O(n)$.

[a] Hartmanis, Stearns, 1965, Hartmanis, Lewis, Stearns, 1965.