Theory of Computation Lecture Notes

Prof. Yuh-Dauh Lyuu
Dept. Computer Science & Information Engineering
and
Department of Finance
National Taiwan University

Class Information

- Papadimitriou. *Computational Complexity.* 2nd printing. Addison-Wesley. 1995.
  - The best book on the market for graduate students.
  - We more or less follow the topics of the book.
  - More “advanced” materials may be added.
- You may want to review discrete mathematics.

Class Information (concluded)

- More information and future lecture notes (in PDF format) can be found at

  `www.csie.ntu.edu.tw/~lyuu/complexity.html`

- Please ask many questions in class.
  - The best way for me to remember you in a large class.[a]
- Teaching assistants will be announced later.

[a] "[A] science concentrator [...] said that in his eighth semester of [Harvard] college, there was not a single science professor who could identify him by name." (*New York Times*, September 3, 2003.)

Grading

- No roll calls.
- No homeworks.
  - Try some of the exercises at the end of each chapter.
- Two to three examinations.
- You must show up for the examinations, in person.
- If you cannot make it to an examination, please email me beforehand (unless there is a legitimate reason).
- Missing the final examination will earn a “fail” grade.

Problems and Algorithms

I have never done anything “useful.”
— Godfrey Harold Hardy (1877–1947),
A Mathematician’s Apology (1940)

What This Course Is All About

Computability: What can be computed?

- What is computation anyway?
- There are *well-defined* problems that cannot be computed.
- In fact, "most" problems cannot be computed.

What This Course Is All About (continued)

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

- Some computable problems require at least exponential time and/or space; they are **intractable**.
  - Can't you let Moore's law take care of it?[a]
    - A variant of Moore's law says the computing power doubles every 1.5 years.[b]
    - The genome sequence data at the Sanger Centre at Cambridge is doubling each year.[c]

[a] Contributed by Ms. Amy Liu (J94922016) on May 15, 2006.

[b] Moore (1965).

[c] Microsoft (2006).

What This Course Is All About (concluded)

- Some practical problems require superpolynomial resources unless certain conjectures are disproved.
- Other resource limits besides time and space?
  - Program size, circuit size (growth), number of random bits, etc.

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, number of 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.
  - $n^{\log n}$, $2^{\sqrt{n}}$,[a] 2^n, $n! \sim \sqrt{2\pi n}\,(n/e)^n$.

[a] Size of depth-3 circuits to compute the majority function (Wolfovitz (2006)).

Growth of Factorials

n	$n!$	n	$n!$
1	1	9	362,880
2	2	10	3,628,800
3	6	11	39,916,800
4	24	12	479,001,600
5	120	13	6,227,020,800
6	720	14	87,178,291,200
7	5040	15	1,307,674,368,000
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.

Turing Machines

What Is Computation?

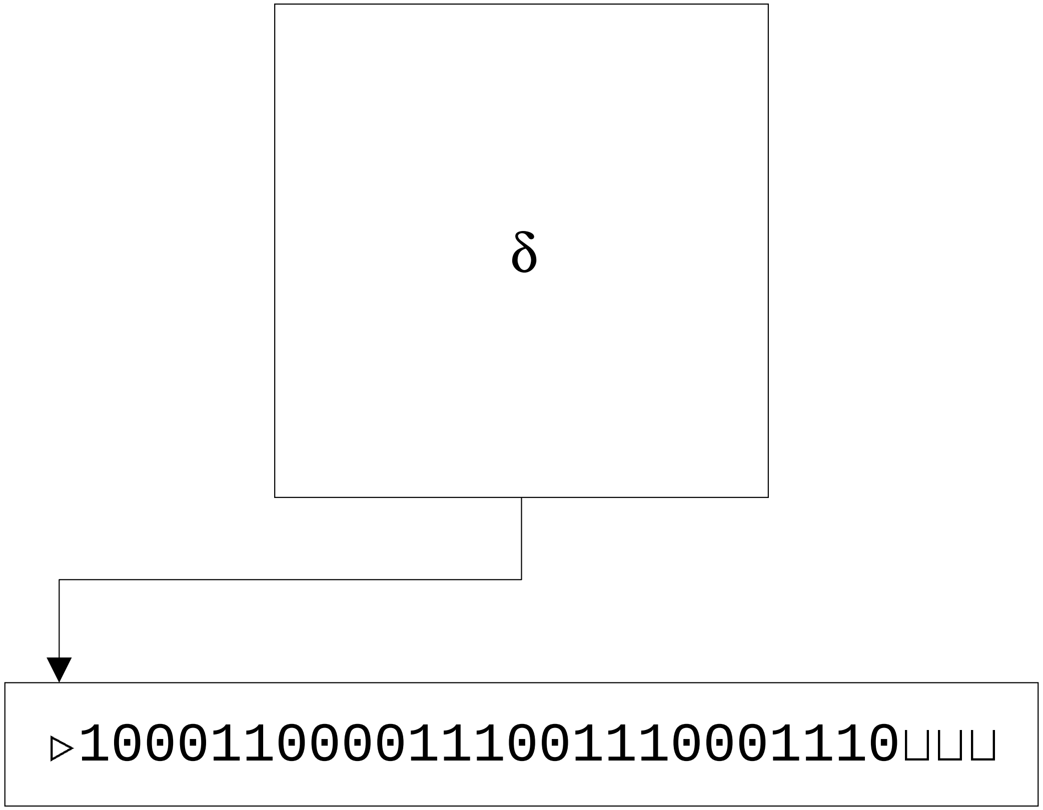
- That can be coded in an **algorithm**.
- An algorithm is a detailed step-by-step method for solving a problem.
  - The Euclidean algorithm for the greatest common divisor is an algorithm.
  - "Let s be the least upper bound of compact set A" is not an algorithm.
  - "Let s be a smallest element of 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).
- $\delta : K \times \Sigma \rightarrow (K \cup \{h, \text{"yes"}, \text{"no"}\}) \times \Sigma \times \{\leftarrow, \rightarrow, -\}$ is a **transition function**.
  - $\leftarrow$ (left), $\rightarrow$ (right), and $-$ (stay) signify cursor movements.

[a] Turing (1936).

A TM Schema



δ

▷1000110000111001110001110⊔⊔⊔

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

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

  - It specifies the next state p, the symbol ρ to be written over σ, and the direction D the cursor will move *afterwards*.

- We require $\delta(q, \rhd) = (p, \rhd, \rightarrow)$ so that the cursor never falls off the left end of the string.

The Operations of TMs

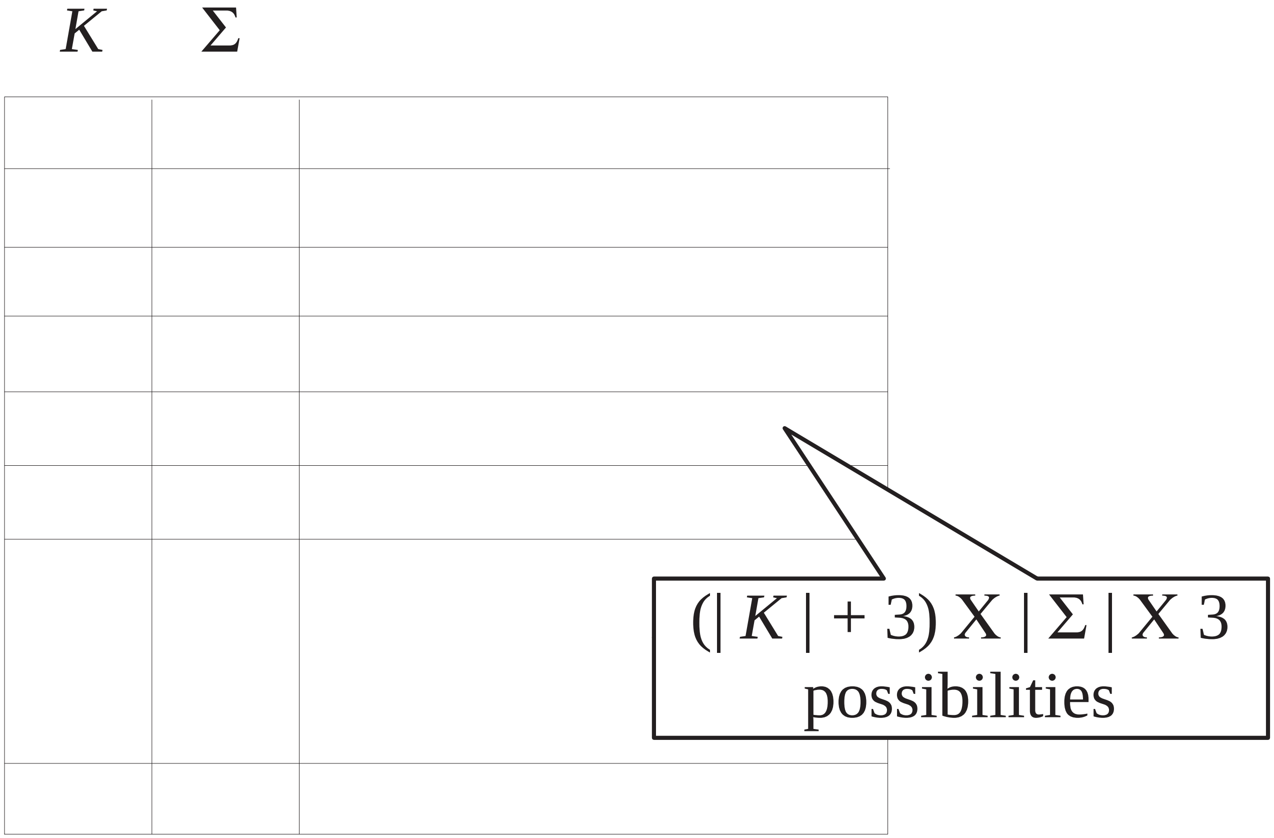
- Initially the state is s.
- The string on the tape is initialized to a $\rhd$, followed by a *finite-length* string $x \in (\Sigma - \{\sqcup\})^*$.
- x is the **input** of the TM.
  - The input must not contain $\sqcup$s (why?)!
- The cursor is pointing to the first symbol, always a $\rhd$.
- The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

Program Count

- A program has a *finite* size.
- Recall that
  $\delta : K \times \Sigma \rightarrow (K \cup \{h, \text{“yes”}, \text{“no”}\}) \times \Sigma \times \{\leftarrow, \rightarrow, -\}$.
- So $|K| \times |\Sigma|$ “lines” suffice to specify a program, one line per pair from $K \times \Sigma$ ($|\,x\,|$ denotes the length of x).
- Given K and Σ, there are

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

  possible δ's (see next page).
  - This is a constant—albeit large.
- Different δ's may define the same behavior.



$(|K| + 3)\ \mathrm{X}\ |\Sigma|\ \mathrm{X}\ 3$ possibilities

The Halting of a TM

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

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

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

  h**:** $M(x) = y$, where the string (tape) consists of a $\rhd$, 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.

Remarks

- A problem is computable if there is a TM that halts with the correct answer.
  - If a TM (i.e., program) does not always halt, it does not solve the problem, assuming the problem is computable.[a]
  - OS does not halt as it does not solve a well-defined problem (but parts of it do).[b]

[a] Contributed by Ms. Amy Liu (J94922016) on May 15, 2006. Control-C is not a legitimate way to halt a program.

[b] Contributed by Mr. Shuai-Peng Huang (J94922019) on May 15, 2006.

Remarks (concluded)

- Any computation model must be physically realizable.
  - A model that requires nearly infinite precision to build is not physically realizable.
  - For example, if the TM required a voltage of exactly 100 to work, it would not be considered a successful model for computation.
- Although a TM requires a tape of infinite length, which is not realizable, it is not a major conceptual problem.[a]
- A tape of infinite length cannot be used to realize infinite precision within a finite time span.[b]

[a] Thanks to a lively discussion on September 20, 2006.

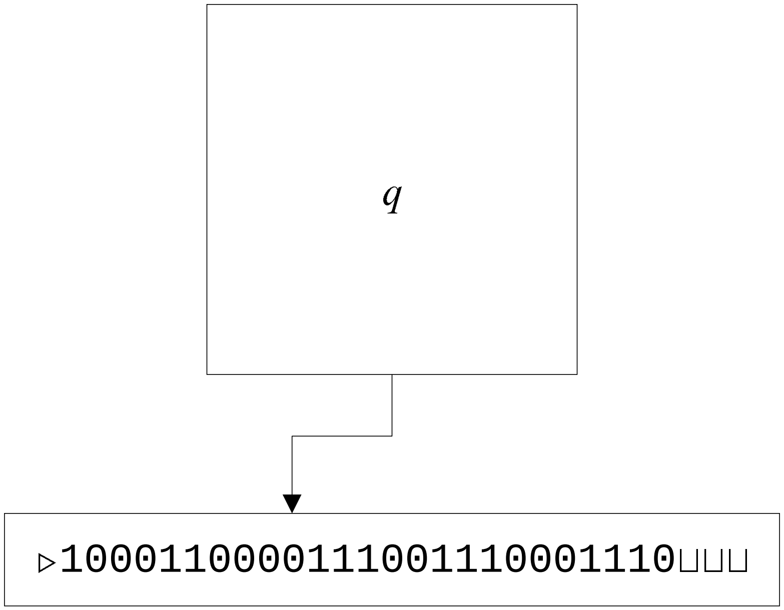
[b] Thanks to a lively discussion on September 20, 2006.

The Concept of Configuration

- 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.
- Similar to the concept of **Markov process** in stochastic processes or dynamic systems.

Configurations (concluded)

- 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.
- Note that (w, u) describes both the string and the cursor position.



q

$\triangleright 1000110000111001110001110 \sqcup\sqcup\sqcup$

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

Yielding

- Fix a TM M.
- Configuration (q, w, u) **yields** configuration (q', w', u') in one step,
$$(q, w, u) \xrightarrow{M} (q', w', u'),$$
if a step of M from configuration (q, w, u) results in configuration (q', w', u').
- $(q, w, u) \xrightarrow{M^k} (q', w', u')$: Configuration (q, w, u) yields configuration (q', w', u') in $k \in \mathbb{N}$ steps.
- $(q, w, u) \xrightarrow{M^*} (q', w', u')$: Configuration (q, w, u) yields configuration (q', w', u').

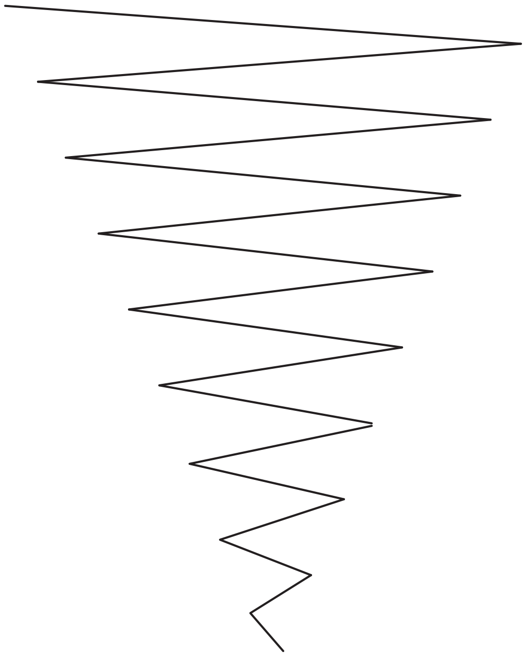
Example: How to Insert a Symbol

- 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:
  - It matches the first character with the last character.
  - It matches the second character with the next to last character, etc. (see next page).
  - "yes" for palindromes and "no" for nonpalindromes.
- This program takes $O(n^2)$ steps.
- Can we do better?

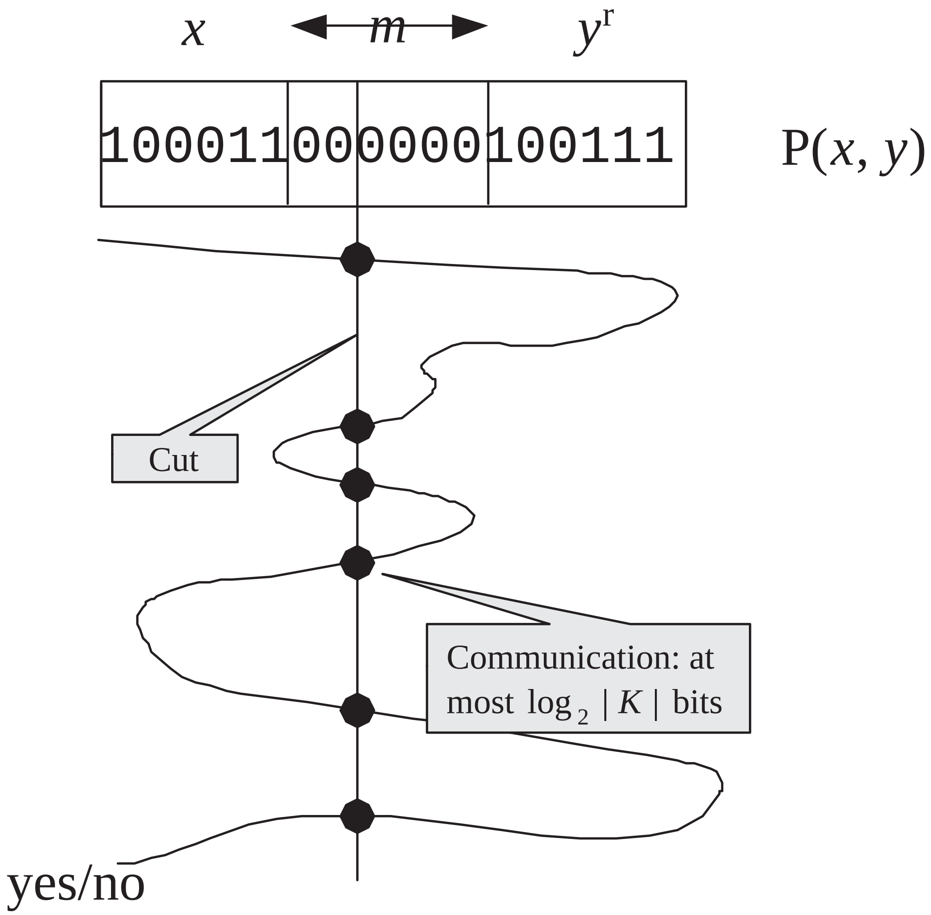
100011000000100111



A Matching Lower Bound for PALINDROME

Theorem 1 (Hennie (1965)) PALINDROME *on single-string TMs takes* $\Omega(n^2)$ *steps in the worst case.*

The Proof: Setup

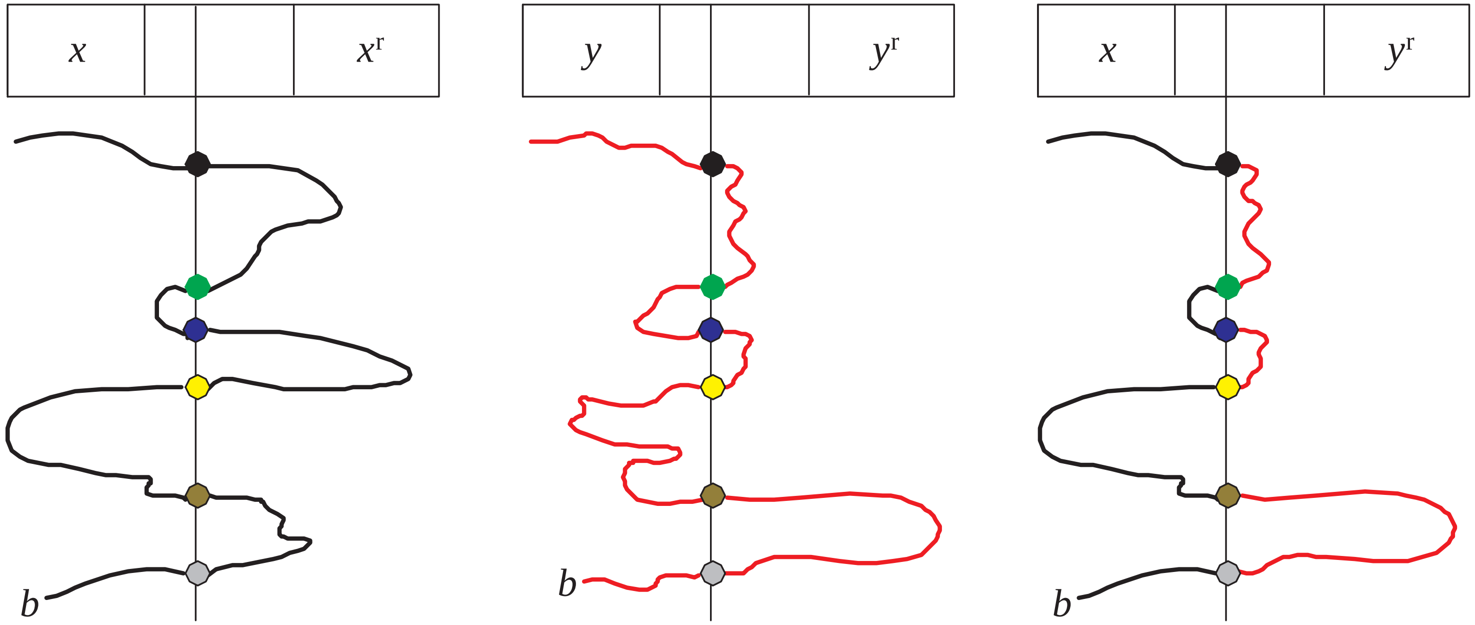


The Proof: Communications

- 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 $O(1)$.
- $\mathrm{C}(x, y)$: the sequence of communications for palindrome problem $\mathrm{P}(x, y)$ *across* the cut.
  - This crossing sequence is a sequence of states from K.

The Proof: Communications (concluded)

- $\mathrm{C}(x, x) \neq \mathrm{C}(y, y)$ when $x \neq y$.
  - Suppose otherwise, $C(x, x) = C(y, y)$.
  - Then $C(y, y) = C(x, y)$ by the cut-and-paste argument (see next page).
  - Hence $\mathrm{P}(x, y)$ has the same answer as $\mathrm{P}(y, y)$!
- So $\mathrm{C}(x, x)$ is distinct for each x.


x
x^r
b
y
y^r
b
x
y^r
b

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. 36), 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

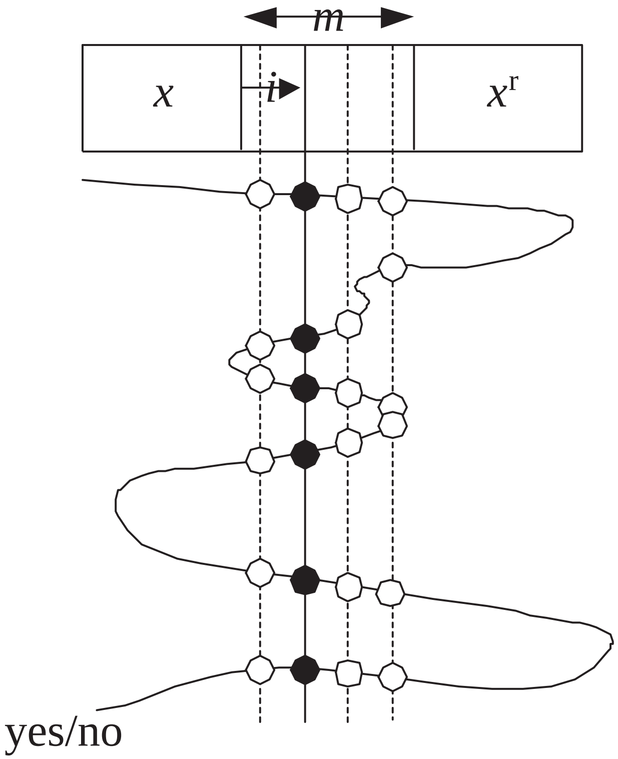
$$\begin{aligned}\sum_{x\in\{0,1\}^m} |\,\mathrm{C}(x,x)\,| &\geq \sum_{x\in\{0,1\}^m, |\,\mathrm{C}(x,x)\,|>\kappa} |\,\mathrm{C}(x,x)\,| \\ &> \left(2^m - \frac{2^{m+1}}{m}\right)\kappa \\ &= \kappa 2^m \frac{m-2}{m}.\end{aligned}$$

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

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

The Proof (continued)

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



The Proof (continued)

- $\mathrm{C}_i(x,x)$ denotes the sequence of communications for $\mathrm{P}(x,x)$ given the cut at position i.
- Then $\sum_{i=1}^{m} |\,\mathrm{C}_i(x,x)\,|$ is the number of steps spent in the middle section for $P(x,x)$.
- Let $T(n) = \max_{x \in \{0,1\}^m} \sum_{i=1}^{m} |\,\mathrm{C}_i(x,x)\,|$.
  - $T(n)$ is the worst-case running time spent in the middle section when dealing with any $P(x,x)$ with $|\,x\,| = m$.
- Note that $T(n) \geq \sum_{i=1}^{m} |\,\mathrm{C}_i(x,x)\,|$ for any $x \in \{0,1\}^m$.

The Proof (continued)

- Now,

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

The Proof (concluded)

- By the pigeonhole principle,[a] there exists an $1 \leq i^* \leq m$,

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

- Eq. (1) on p. 40 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.