CS/ECE 374 A: Intro to Algorithms & Models of Computation

http://courses.engr.illinois.edu/cs374/sp2022/A

Instructors: Timothy Chan, Ruta Mehta

Lectures & Labs:
- First week: completely online
- Afterwards: lectures in-person (ECEB 1002) & streamed live on Zoom & recorded
- Labs: mix of in-person & zoom sessions (TBA)

Office Hrs.: Online (Zoom/Discord) (more details later)

Piazza
(note: please be courteous & respectful to others!)

HWs:
- weight = 1 HW problem
- 11 Guided Problem Sets (GPS) on PrairieLearn (autograded)
- 11 Written Homeworks
  - each = 2 HW problems
  - may work in groups ≤ 3
  - total = 33 HW problems
  - no late HWs!
  (but may drop 6 problems)
  (if illness/examining circumstances, ask instructors ...)

Exams:
- Midterm 1: Feb 21 Mon 7p-9:30p (Conflict: TBA)
- Midterm 2: Apr 11 Mon 7p-9:30p
- Final: TBA
  (proctor via Zoom)

Grades:
- HWs: 28%
- Midterm 1: 21%
- Midterm 2: 21%
- Final: 30%

Option 1: fixed cut-offs
Option 2: curved
Take better of two
(see web pages)
Overview

Introduction to CS Theory

Goal 1: how to solve problems (efficiently)
algorithm design & analysis

Goal 2: how to show that a problem can't be solved (efficiently)
mathematically prove

Outline

Part I. Models of Computation
finite automata ⇔ regular exprs
context-free grammars
Turing machines

Part II. Algorithm Design Techniques
divide & conquer
dynamic programming*
greedy
graph algorithms

Part III. NP-Completeness* & Undecidability

Example: Given n numbers,
can there exist 3 numbers summing to 100?
e.g. 81, 95, 43, 20, 32, 74, 25
brute-force algm: \(O(n^3)\) time
smarter algm: \(O(n^2)\) time...
fastest? OPEN
\[ \sim O\left(\frac{n^2}{\log^2 n}\right) \quad [C'2018] \]
Ex 2. Given $n$ polygons & 1 rectangle, can they be packed in rectangle?

[Diagram showing 5 shapes numbered 1 to 4.]

No efficient algo' believed to be possible
\[ \text{NP-complete} \]

Ex 3. Given $n$ polygons.
Can they tile the entire plane?
(assuming infinite copies)

[Diagram showing shapes being combined to form a larger shape.]

No algo' possible (undecidable)

---

Part I. Models of Computation

Math Preliminaries

Strings

A string is a finite sequence of symbols from a finite set $\Sigma$.

E.g., strings over $\Sigma = \{0, 1\}$:

$$1001 \quad 01 \quad 101 \quad 0$$

Let $\varepsilon$ denote the empty string.

Let $\Sigma^*$ denote \{all strings over $\Sigma$\}. 

---

Inputs to program, etc.
Let \( \Sigma \) denote \{all strings over \( \Sigma \}\}.

Let \( x, y \) be strings.

a) length \( |x| \)
   
   e.g. \( |1001| = 4, \ |\varepsilon| = 0 \)

b) concatenation \( xy \)
   
   e.g. \( x = \varepsilon, y = 1001 \Rightarrow xy = 011001 \)
   
   \((xy)_\varepsilon = x(y\varepsilon) \quad xy \neq yx \)
   
   \( |xy| = |x| + |y| \)

   \( 3x = x, \quad x\varepsilon = x \)

   \( i \)th power \( x^i = \underbrace{x \cdots x}_{i \text{ times}} \)

   e.g. \( (101)^3 = 101101101 \)
   
   \[
   \begin{align*}
   x^0 &= \varepsilon \\
   x^i &= x \cdot x^{i-1} \\
   |x^i| &= i |x|
   \end{align*}
   \]

d) \( x \) is a substring of \( y \) if

   \( y = w \cdot x \cdot z \) for some strings \( w, z \)

   (prefix if \( w = \varepsilon \), suffix if \( z = \varepsilon \))

e) other ops:

   \( x^R \) = reverse of \( x \)

   \[
   x^R = \begin{cases}
   \varepsilon & \text{if } x = \varepsilon \\
   y^Ra & \text{if } x = ay \text{ for some } a \in \Sigma, \ y \in \Sigma^* \\
   \end{cases}
   \]

   (convention: Symbols \( a, b, c, \ldots \)
   
   Strings \( x, y, z, \ldots \)

   \((xy)^R = y^Rx^R \quad (\text{lab la})\)
**Languages**

A language is a set of strings (over $\Sigma$)

(i.e. $L \subseteq \Sigma^*$)

- e.g. $\{1001, 01, 101, 0\}$
- finite, boring

- infinite, more interesting

- $\{x \in \{0, 1\}^* : |x| \text{ is odd}\}$
- $\{\text{all prime numbers written in binary}\}$
- $\{\text{all syntactically valid Python programs}\}$

(languages can encode all decision problems)

Let $L_1, L_2$ be languages.

a) Union $L_1 \cup L_2$

Intersection $L_1 \cap L_2$

Complement $L_1^c = \Sigma^* \setminus L_1$

Difference $L_1 \setminus L_2 = L_1 \cap \overline{L_2}$

b) Concatenation $L_1 L_2 = \{xy : x \in L_1, y \in L_2\}$

- e.g. $L_1 = \{0, 00\}$, $L_2 = \{1, 01\}$
  $L_1 L_2 = \{01, 0001, 001\}$

- e.g. $L_1 = \{0, 00, 000, \ldots\} = \{0^i : i \geq 1\}$
  $L_2 = \{1, 11, 111, \ldots\} = \{1^j : j \geq 1\}$
  $L_1 L_2 = \{0^i 1^j : i \geq 1, j \geq 1\}$

Wrong

- $L_1 L_2 = \{0^i 1^j : i \geq 1, j \geq 1\}$

c) $i^{th}$ power $L^i = L \cup \cdots \cup L$
c) $i$th power: $L^i = \underbrace{L \cdots L}_{i \text{ times}}$

  e.g. $\{1,01\}^2 = \{11, 0101, 101, 01\}$
  $L^0 = \{\varepsilon\}$
  $L^i = L \cdot L^{i-1}$

d) Kleene star

  $L^* = \bigcup_{i=0}^{\infty} L^i = L^0 \cup L \cup L^2 \cup \ldots$

  e.g. $\{01\}^* = \{\varepsilon, 01, 0101, 010101, \ldots\}$

  $\{1,01\}^* = \{\varepsilon, 1, 01, 11, 0101, 101, 011, 111, 010101, 10101, 0111, \ldots, \ldots\}$

  $= \{ x \in \{0,1\}^* : \ x \text{ does not contain } 00 \text{ as a substring } \land \text{ does not end } 0 \}$

  (proof?)

  $\{0,1\}^* = \text{all strings over } \{0,1\}$