CS 374: Algorithms & Models of Computation

Goal:
- Techniques to design algms
  - How to solve problems on computers (efficiently)
    - Need math defn/model of computation
  - Understand what problems can or can't be solved
    - Prove mathematically

Course Overview

I. Models of Computation
   - Finite automata $\Leftrightarrow$ regular expr
   - Context-free grammars
   - Turing machines

II. Algorithms Design
   - Divide & conquer
   - Dynamic Programming
   - Greedy

III. Undecidability & NP-Completeness

Ex1: Given $n$ numbers, can you find 3 summing exactly to 100?

(3SUM)

$\text{e.g. } 82, 43, 19, 96, 32, 74, 25$

Yes
brute force: $O(n^3)$ time
cleverer alg’m: $O(n^2)$
fastest alg’m: still open!
[current record: about $O\left(\frac{n^2}{\log^3 n}\right)$
by C.’17]

Ex2 Given $n$ polygons & a box, can you pack them in box?

Ex3 Given $n$ polygons, can you tile the entire plane?

no alg’m is possible!

PART I: MODELS OF COMPUTATION

Math Prelims

Strings
Strings

A string is a finite sequence of symbols from a finite set \( \Sigma \) called alphabet.

E.g. strings over \( \Sigma = \{0, 1\} \):
- 0110, 01, 0

Empty string is denoted \( \varepsilon \).
Let \( \Sigma^* = \{ \text{all strings over } \Sigma \} \).

Let \( x, y \) be strings.

a) length \( |x| \)
   - e.g. \( |01101| = 4, \quad |101| = 2, \quad |\varepsilon| = 0 \)

b) concatenation \( xy \)
   - e.g. \( x = 01, \ y = 101 \)
   \( \Rightarrow xy = 01101 \)
   \( (xy)z = x(yz) \)
   \( |xy| = |x| + |y| \)
   \( \varepsilon x = x \varepsilon = x \)

c) \( i^{th} \) power \( x^i = x \ldots x \) \( i \) times
   - e.g. \( (101)^3 = 101101101 \)
   \( x^0 = \varepsilon \)

d) \( x \) is a substring of \( y \) if \( y = wxz \) for some strings \( w, z \)
   (prefix if \( w = \varepsilon \), suffix if \( z = \varepsilon \))

e) other ops:
   \( x^R = \text{reverse of } x \).
\[ x^R = \text{reverse of } x \]

(can be defined recursively:
\[
 x^R = \begin{cases} 
 y^R a & \text{if } x = ay \ a \in \Sigma, y \in \Sigma^* \\
 \epsilon & \text{if } x = \epsilon 
\end{cases}
\]

\[(xy)^R = y^Rx^R\]

**Languages**

A **language** is a set of strings (i.e. subset of \( \Sigma^* \))

\[\{ 0110, 01, 0 \}\] is a lang.
\[\{ x \in \{0,1\}^* \mid |x| \text{ is even} \}\]
\[\{ \text{all words in English dictionary} \} \text{ (over } \Sigma = \{',,.,!', 'z'\})\]

finite, boring
\[\{ \text{all syntactically valid Java programs} \}\]
more interesting
\[\{ \text{all prime numbers in binary} \}\]

(\text{decision problems can be encoded as languages})

Let \( L_1, L_2 \) be languages.

a) \( \text{union } L_1 \cup L_2 \)
\[ \text{intersection } L_1 \cap L_2 \]
\[ \text{complement } \bar{L}_1 = \Sigma^* \setminus L_1 \]
\[ \text{difference } L_1 \setminus L_2 = L_1 \cap \bar{L}_2 \]
b) \[ L_1 L_2 = \{ xy \mid x \in L_1, y \in L_2 \} \]

\[ \text{e.g. } L_1 = \{0, 00\}, \quad L_2 = \{1, 01\} \]
\[ \Rightarrow L_1 L_2 = \{01, 001, 0001\} \]

\[ \text{e.g. } L_1 = \{0, 00, 000, \ldots\} \]
\[ L_2 = \{1, 11, 111, \ldots\} \]
\[ \Rightarrow L_1 L_2 = \{0^i 1^j \mid i, j \geq 1\} \]

c) \[ i^{\text{th}} \text{ power } L^i = \underbrace{L \cdot \cdots \cdot L}_{i \text{ times}} \]

\[ \text{e.g. } \{1, 01\}^2 = \{11, 101, 011, 0101\} \]
\[ L^0 = \{\varepsilon\}. \]

d) \[ \text{Kleene star } L^* = \bigcup_{i=0}^{\infty} L^i \]
\[ = L^0 \cup L^1 \cup L^2 \cup \ldots \]

\[ \text{e.g. } \{01\}^* = \{\varepsilon, 01, 0101, 010101, \ldots\} \]
\[ \{0, 1\}^* \text{ as defined before} \]
\[ \{1, 01\}^* = \{\varepsilon, 1, 01, 11, 101, 011, 0101, 111, 1101, \ldots\} \]
\[ = \{x \in \{0, 1\}^* \mid x \text{ does not contain } 00 \text{ as a substring} \}
\[ \text{and } x \text{ ends in } 1\} \]
There are countably many strings (but countably many Java programs)