ECE-374-B: Lecture 0 - Logistics and Strings/Languages

Lecturer: Nickvash Kani
August 22, 2023
University of Illinois at Urbana-Champaign
Course Administration
Instructional Staff

• Instructor:
  • Nickvash Kani
  • Abhishek Umrawal

• Teaching Assistants:
  • Sung Woo Jeon
  • Sindhu Vydana
  • Sandhya Perumenki
  • Sumedh Vemuganti
  • Weiyang Wang
  • Jack Chen
  • Yueyi Shen
  • Haoyuan You

• Office hours: TBD, See course webpage

• Contacting us: Use private notes on Piazza to reach course staff. Direct email only for sensitive or confidential information.
Section A vs B

This semester, the two sections will be run completely independently.

- Different lectures.
- Different homeworks, quizzes, exams.
- Different grading policies.
This semester, the two sections will be run completely independently.

- Different lectures.
- Different homeworks, quizzes, exams.
- Different grading policies.

Section B will be in-person only. Recordings will be attempted but not guaranteed.
Online resources

- **Webpage**: General information, announcements, homeworks, quizzes, course policies
  [https://ecealogo.com](https://ecealogo.com)
- **Submission (Gradescope)**: Written homework submission and grading, regrade requests. Exams will be uploaded there as well.
- **Communication (Piazza)**: Announcements, online questions and discussion, contacting course staff (via private notes)
- **Gradebook (Canvas)**: Announcements, online questions and discussion, contacting course staff (via private notes)

See course webpage for links

**Important**: check Piazza/course web page at least once each day
Discussion Sessions/Labs

- 50min problem solving session led by TAs
- Two times a week
- Go to your assigned discussion section
- Bring pen and paper!
Discussion Sessions/Labs

- 50min problem solving session led by TAs
- Two times a week
- Go to your assigned discussion section
- Bring pen and paper!

Discussion sections will have questions that appear on the homework. If you skip, you’re just making more work for yourself later.
Again all policy information should be on course website: https://ecealgo.com

Any questions?
Over-arching course questions
This course introduces three distinct fields of computer science research:

- **Computational complexity.**
  - Given infinite time and a certain machine, is it possible to solve a given problem.

- **Algorithms**
  - Given a deterministic Turing machine, how fast can we solve certain problems.

- **Limits of computation.**
  - Are there tasks that our computers cannot do and how do we identify these problems?
Why not just focus on Algorithms?

When someone asks you, ”How fast can you compute problem X”, they are actually asking:

- Is X solvable using the deterministic Turing machines we have at our disposal?
- If it is solvable, can we find the solution efficiently (in poly-time)?
- If it is solvable but we don’t have a poly time solution, what problem(s) is it most similar too?
Course divided into three parts:

- Basic automata theory: finite state machines, regular languages, hint of context free languages/grammars, Turing Machines
- Algorithms and algorithm design techniques
- Undecidability and NP-Completeness, reductions to prove intractability of problems
Goals

• Algorithmic thinking
• Learn/remember some basic tricks, algorithms, problems, ideas
• Understand/appreciate limits of computation (intractability)
• Appreciate the importance of algorithms in computer science and beyond (engineering, mathematics, natural sciences, social sciences, ...)

9
Formal languages and complexity
(The Blue Weeks!)
First 5 weeks devoted to language theory.
Why Languages?

First 5 weeks devoted to language theory.
But why study languages?
Consider the following problem:

**Problem**  Given two \( n \)-digit numbers \( x \) and \( y \), compute their product.

**Grade School Multiplication**
Compute “partial product” by multiplying each digit of \( y \) with \( x \) and adding the partial products.

\[
\begin{array}{c}
\phantom{1}3141 \\
\times \phantom{2}2718 \\
\underline{+} \\
\phantom{1}6282 \\
\hline
8537238
\end{array}
\]
Time analysis of grade school multiplication

- Each partial product: $\Theta(n)$ time
- Number of partial products: $\leq n$
- Adding partial products: $n$ additions each $\Theta(n)$ (Why?)
- Total time: $\Theta(n^2)$
- Is there a faster way?
Fast Multiplication

• $O(n^{1.58})$ time [Karatsuba 1960] disproving Kolmogorov’s belief that $\Omega(n^2)$ is best possible
• $O(n \log n \log \log n)$ [Schonhage-Strassen 1971].

**Conjecture:** $O(n \log n)$ time possible

• $O(n \log n \cdot 2^{O(\log^* n)})$ time [Furer 2008]
• $O(n \log n)$ [Harvey-van der Hoeven 2019]

Can we achieve $O(n)$? No lower bound beyond trivial one!
Equivalent Complexity

Does this mean multiplication is as complex as another problem that has a $O(n \log n)$ algorithm like sorting/QuickSort?
Equivalent Complexity

Does this mean multiplication is as complex as another problem that has a $O(n \log n)$ algorithm like sorting/QuickSort? How do we compare? The two problems have:

- Different inputs (two numbers vs n-element array)
- Different outputs (a number vs n-element array)
- Different entropy characteristics (from a information theory perspective)
An algorithm has a runtime complexity.
A problem has a complexity class!

Recognized by:
- Turing machines
- Linear bounded automata
- Push-down automata
- DFAs, NFAs, RegEx

Problems do not have run-time since a problem $\neq$ the algorithm used to solve it. *Complexity classes are defined differently.*

How do we compare problems? What if we just want to know if a problem is "computable".
Definition

1. An **algorithm** is a step-by-step way to solve a problem.

2. A **problem** is some question that we’d like answered given some input. It should be a decision problem of the form "Does a given input fulfill property X."

3. A **Language** is a set of strings. Given a alphabet, $\Sigma$ a language is a subset of $\Sigma^*$
**Definition**

1. An **algorithm** is a step-by-step way to solve a problem.

2. A **problem** is some question that we’d like answered given some input. It should be a decision problem of the form ”Does a given input fulfill property X.”

3. A **Language** is a set of strings. Given a alphabet, $\Sigma$ a language is a subset of $\Sigma^*$ A language is a formal realization of this problem. For problem $X$, the corresponding language is:

$$L = \{ w \mid w \text{ is the encoding of an input y to problem X and the answer to input y for a problem X is ”YES” } \}$$

A decision problem $X$ is ”YES” is the string is in the language.
How do we define the multiplication problem as a language?
Define \( L \) as language where inputs are separated by comma and output is separated by \( | \).
Machine accepts a \( x*y=z \) if \( "x*y|z" \) is in \( L \). Rejects otherwise.
How do we define the multiplication problem as a language? Define L as language where inputs are separated by comma and output is separated by |.

Machine accepts a x*y=z if ”x*y|z” is in L. Rejects otherwise.

\[
L_{MULT2} = \left\{ 1 \times 1|1, \quad 1 \times 2|2, \quad 1 \times 3|3, \ldots \\
2 \times 1|2, \quad 2 \times 2|4, \quad 2 \times 3|6, \ldots \\
\vdots \quad \vdots \quad \vdots \\
n \times 1|n, \quad n \times 2|2n, \quad n \times 3|3n, \ldots \right\} \quad (1)
\]
Language of sorting

We do the same thing for sorting.

Define $L$ as language where inputs are separated by comma and output is separated by $|$. Machine accepts a $[i_1, i_2, \ldots] = \text{sort} \left( \{i_1, i_2, \ldots\} \right)$ if "x[|]z[|]" is in $L$. Rejects otherwise.
Language of sorting

We do the same thing for sorting.

Define $L$ as language where inputs are separated by comma and output is separated by $|$.  

Machine accepts a $[i_1, i_2, \ldots] = \text{sort}(\{i_1, i_2, \ldots\})$ if "x[[]|z[[]] is in $L$. Rejects otherwise.

$$L_{Sort2} = \left\{ \begin{array}{ccc} 1, 1|1, 1 & 1, 2|1, 2 & 1, 3|1, 3, \ldots \\ 2, 1|1, 2 & 2, 2|2, 2 & 2, 3|2, 3, \ldots \\ \vdots & \vdots & \vdots \\ n, 1|1, n & n, 2|2, n & n, 3|3, n, \ldots \end{array} \right\}$$

(2)
Language of sorting

We do the same thing for sorting.

Define L as language where inputs are separated by comma and output is separated by |.

Machine accepts a $[i_1, i_2, \ldots] = \text{sort}([i_1, i_2, \ldots])$ if "x[]|z[]" is in L. Rejects otherwise.

$$L_{Sort2} = \left\{ 1,1|1,1 \quad 1,2|1,2 \quad 1,3|1,3,\ldots \\
2,1|1,2 \quad 2,2|2,2 \quad 2,3|2,3,\ldots \\
\vdots \quad \vdots \quad \vdots \\
n,1|1,n \quad n,2|2,n \quad n,3|3,n,\ldots \right\}$$

If the same type of machine can recognize both languages, then that gives us an upperbound top their hardness.
How do we formulate languages?
Strings
An alphabet is a **finite** set of symbols.

Examples of alphabets:

- \( \Sigma = \{0, 1\} \),
- \( \Sigma = \{a, b, c, \ldots, z\} \),
- ASCII.
- UTF8.
- \( \Sigma = \{\langle w \rangle \text{forward}, \langle a \rangle \text{strafe left}, \langle s \rangle \text{back}, \langle d \rangle \text{strafe right} \} \)
Definition

1. A string/word over \(\Sigma\) is a finite sequence of symbols over \(\Sigma\). For example, ‘0101001’, ‘string’, ‘\langle\text{moveback}\rangle\langle\text{rotate90}\rangle’

2. \(x \cdot y \equiv xy\) is the concatenation of two strings

3. The length of a string \(w\) (denoted by \(|w|\)) is the number of symbols in \(w\). For example, \(|101| = 3, |\epsilon| = 0\)

4. For integer \(n \geq 0\), \(\Sigma^n\) is set of all strings over \(\Sigma\) of length \(n\). \(\Sigma^*\) is the set of all strings over \(\Sigma\).

5. \(\Sigma^*\) set of all strings of all lengths including empty string.

Question: \{‘0’, ‘1’\}^* =
• $\epsilon$ is a string containing no symbols. It is not a set
• $\{\epsilon\}$ is a set containing one string: the empty string. It is a set, not a string.
• $\emptyset$ is the empty set. It contains no strings.

**Question**: What is $\{\emptyset\}$
• If $x$ and $y$ are strings then $xy$ denotes their concatenation.

• **Concatenation** defined recursively:
  
  - $xy = y$ if $x = \epsilon$
  - $xy = a(wy)$ if $x = aw$

• $xy$ sometimes written as $x \cdot y$.

• Concatenation is **associative**: $(uv)w = u(vw)$ hence write $uvw \equiv (uv)w = u(vw)$

• **not** commutative: $uv$ not necessarily equal to $vu$

• The **identity** element is the empty string $\epsilon$:

  $$\epsilon u = u \epsilon = u.$$
Substrings, prefixes, Suffixes

**Definition**

$v$ is a substring of $w$ $\iff$ there exist strings $x, y$ such that $w = xvy$.

- If $x = \epsilon$ then $v$ is a prefix of $w$.
- If $y = \epsilon$ then $v$ is a suffix of $w$. 
A subsequence of a string $w[1...n]$ is either a subsequence of $w[2...n]$ or $w[1]$ followed by a subsequence of $w[2...n]$.

**Example**

$EE37$ is a subsequence of $ECE374B$
A subsequence of a string $w[1...n]$ is either a subsequence of $w[2...n]$ or $w[1]$ followed by a subsequence of $w[2...n]$.

**Example**

$EE37$ is a subsequence of $ECE374B$

**Question**: How many sub-sequences are there in a string $|w| = 6$?
Definition
If $w$ is a string then $w^n$ is defined inductively as follows:

- $w^n = \epsilon$ if $n = 0$
- $w^n = w w^{n-1}$ if $n > 0$

Question: $(ha)^3 =$.
Answer the following questions taking $\Sigma = \{0, 1\}$.

1. What is $\Sigma^0$?

2. How many elements are there in $\Sigma^n$?

3. If $|u| = 2$ and $|v| = 3$ then what is $|u \cdot v|$?

4. Let $u$ be an arbitrary string in $\Sigma^*$. What is $\epsilon u$? What is $u \epsilon$?
Languages
Definition
A language $L$ is a set of strings over $\Sigma$. In other words $L \subseteq \Sigma^*$.
Definition
A language $L$ is a set of strings over $\Sigma$. In other words $L \subseteq \Sigma^*$.

Standard set operations apply to languages.

- For languages $A, B$ the concatenation of $A, B$ is $AB = \{xy \mid x \in A, y \in B\}$.
- For languages $A, B$, their union is $A \cup B$, intersection is $A \cap B$, and difference is $A \setminus B$ (also written as $A - B$).
- For language $A \subseteq \Sigma^*$ the complement of $A$ is $\overline{A} = \Sigma^* \setminus A$. 
Set Concatenation

**Definition**
Given two sets $X$ and $Y$ of strings (over some common alphabet $\Sigma$) the concatenation of $X$ and $Y$ is

$$XY = \{xy \mid x \in X, y \in Y\}$$

(3)

**Question:** $X = \{ECE, CS\}$, $Y = \{340, 374\} \quad \implies \quad XY = \{\}$
Definition

1. $\Sigma^n$ is the set of all strings of length $n$. Defined inductively:
   - $\Sigma^n = \{\epsilon\}$ if $n = 0$
   - $\Sigma^n = \Sigma \Sigma^{n-1}$ if $n > 0$

2. $\Sigma^* = \bigcup_{n \geq 0} \Sigma^n$ is the set of all finite length strings
3. $\Sigma^+ = \bigcup_{n \geq 1} \Sigma^n$ is the set of non-empty strings.

Definition

A language $L$ is a set of strings over $\Sigma$. In other words $L \subseteq \Sigma^*$.

Question: Does $\Sigma^*$ have strings of infinite length?
Problem
Consider languages over $\Sigma = \{0, 1\}$.

1. What is $\emptyset^0$?
2. If $|L| = 2$, then what is $|L^4|$?
3. What is $\emptyset^*$, $\{\epsilon\}^*$?
4. For what $L$ is $L^*$ finite?
5. What is $\emptyset^+$?
6. What is $\{\epsilon\}^+$?
Let’s review what we learned.

- A **character** \((a, b, c, x)\) is a unit of information represented by a symbol: (letters, digits, whitespace)
- A **alphabet** \((\Sigma)\) is a set of characters
- A **string** \((w)\) is a sequence of characters
- A **language** \((A, B, C, L)\) is a set of strings
Let’s review what we learned.

- A **character**\( (a, b, c, x) \) is a unit of information represented by a symbol: (letters, digits, whitespace)
- A **alphabet**\( (\Sigma) \) is a set of characters
- A **string**\( (w) \) is a sequence of characters
- A **language**\( (A, B, C, L) \) is a set of strings
- A **grammar**\( (G) \) is a set of rules that defines the strings that belong to a language
Languages: easiest, easy, hard, really hard, really hard

- Regular languages.
  - Regular expressions.
  - DFA: Deterministic finite automata.
  - NFA: Non-deterministic finite automata.
  - Languages that are not regular.
- Context free languages (stack).
- Turing machines: Decidable languages.
- TM Undecidable/unrecognizable languages (halting theorem).
• Regular languages.
  • Regular expressions. ← Next lecture
  • DFA: Deterministic finite automata.
  • NFA: Non-deterministic finite automata.
  • Languages that are not regular.
• Context free languages (stack).
• Turing machines: Decidable languages.
• TM Undecidable/unrecognizable languages (halting theorem).
Check the course website (https://ecealgo.com) for lab and hw schedule.