

ECE-374-B: Lecture 1 - Logistics and Strings/Languages

Lecturer: Nickvash Kani

January 17, 2023

University of Illinois at Urbana-Champaign

Course Administration

Instructional Staff

- **Instructor:**
 - Nickvash Kani
- **Teaching Assistants:**
 - Sung Woo Jeon
 - Nicholas Bampton
 - Gautham Varma
 - Sindhu Vydana
 - Sandhya Perumenki
- **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.

Section A vs B

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://ecealgo.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

Grading Policy

Grading Policy: Overview

- Homeworks: 24%
- There will be approximately 9 HWs with ~~3~~⁴⁻⁵ questions each.
- Homeworks need to be submitted on Gradescope. ^{43 problems}
- Only the top ~~21~~³⁴ question grades will be considered for final grade calculation

Grading Policy: Overview

- Homeworks: 24%
- Midterm/Final exams: 75% ($3 \times 25\%$)

Exam dates:

- Midterm 1: Thurs, Feb 16, 12:30pm–1:45pm
- Midterm 2: Tues, Apr 04, 12:30pm–1:45pm
- Midterm 3: Thurs, Apr 27, 12:30pm–1:45pm
- Final: TBD

One exam will be dropped Drop policies should eliminate need for conflict exams.

Grading Policy: Overview

- Homeworks: 24%
- Midterm/Final exams: 75% ($3 \times 25\%$)
- ICES Form: 1%

Piazza is crucial for improving the course experience but only 20-35% participate.

Will ask to upload a screenshot that confirms your completion of the ICES eval before grade estimates.

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.

Advice

- Attend lectures, please ask plenty of questions.
- Attend discussion sessions.
- Don't skip homework and don't copy homework solutions. Each of you should think about all the problems on the home work - do not divide and conquer.
- Start homework early! Your mind needs time to think.
- Study regularly and keep up with the course.
- This is a course on problem solving. Solve as many as you can! Books/notes have plenty.
- This is also a course on providing rigorous proofs of correctness. Refresh your 173 background on proofs.
- Ask for help promptly. Make use of office hours/Piazza.

Common Misconceptions

- Since I got into the B section, I'm guaranteed to pass.
 - Nope, plenty of people have failed in prior semesters.
- I can cram the material the night(s)/week before the exam and do well.
 - I think most students that have gone through this course will disagree.
- Looking at lots of problem solutions is a good way to study the material.
 - There is an enormous difference between understanding and knowing. Understanding is easy, knowledge is difficult.

Miscellaneous

Please contact instructors if you need special accommodations.

Lectures are being taped (hopefully). The issue is that these recording systems are prone to failure. While I make no promises, I will try my best to record the lectures.

See course webpage for additional information.

Over-arching course questions

High-Level 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 Structure

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

Week	Tuesday Lecture	Wed Lab	Thursday Lecture	Fri Lab
Aug 23-26	Adminis trivia and course goals Introduction and history (Sari's Videos, Lec 1 e1) (B: slides, scribbles, video)	String Induction (Sari's Induction notes) Chandra's induction notes e1 (Solutions) (Recording)	Languages and regular expressions (Sari's Videos, Lec 2 e1) (B: slides, scribbles)	Regular expressions (Solutions) (Recording)
Aug 30 - Sep 2	DFA's, intuition, definitions, closure properties Automata Tutor e1, ELAP e1, Mahesh's DFA notes e1, Sari's Videos, Lec 3 e1 (B: slides, scribble)	DFA construction (Solutions) (Recording)	Non-Determinism, NFAs (Sari's Videos, Lec 4 e1) (B: slides, scribble)	Language transformations (Solutions) (Recording)
Sep 6-9	Equivalence of DFAs, NFAs, and regular expressions (Sari's Videos, Lec 5 e1) (B: slides, scribble)	Regex to NFA to DFA (Solutions) (Recording)	Tooling Sets and Proving Non-Regularity (Mahesh's DFA notes e1, Fall 2015 TAC: Tooling Sets Notes e1, Sari's Videos, Lec 6 e1) (B: slides, scribble)	Proving Non-Regularity (Solutions) (Recording)
Sep 13-16	Context-free languages and grammars (Sari's Videos, Lec 7 e1) (B: slides, scribble)	Context-free grammars (Solutions) (Recording)	Turing machines: history, formal definitions, examples, variations (Sari's Videos, Lec 8 e1) (B: slides, scribble)	Turing Machines (Solutions) (Recording)
Sep 20-23	Universal Turing machines (Sari's Videos, Lec 8 e1) (B: slides, scribble)	Midterm 1 Review (effectively office hours)	Midterm 1 - Thursday Sep 22 12:30-13:45 (Skill set, folder, cheat sheet)	No instruction
Sep 27-30	Reductions & Recursion (Sari's Videos, Lec 10 e1, Notes on Solving Recurrences e1) (B: slides, scribbles)	Binary search (Solutions) (Recording)	Divide and conquer: Selection, Karatsuba (Sari's Videos, Lec 11 e1) (B: slides, scribbles)	Divide and Conquer (Solutions) (Recording)
Oct 4-7	Backtracking (Sari's Videos, Lec 12 e1) (B: slides, scribbles)	Backtracking (Solutions) (Recording)	Dynamic programming (Sari's Videos, Lec 13 e1) (B: slides, scribbles)	Dynamic programming (Solutions) (Recording)
Oct 11-14	More Dynamic programming (Sari's Videos, Lec 14 e1) (B: slides, scribbles)	More Dynamic programming (Solutions) (Recording)	Graphs, Basic Search (Chandra's Graph notes e1, Sari's Videos, Lec 15 e1) (B: slides, scribbles)	Even more DP (Solutions) (Recording)
Oct 18-21	Directed Graphs, DFS, DAGs and Topological Sort (Chandra's Graph notes e1, Sari's Videos, Lec 16 e1) (B: slides, scribbles)	Graph Modeling (Solutions) (Recording)	Shortest Paths: BFS and Dijkstra (Chandra's Graph notes e1, Sari's Videos, Lec 17 e1) (B: slides, scribbles)	Shortest Paths (Solutions) (Recording)
Oct 25-28	Bellman-Ford, Dynamic Programming on DAGs (Chandra's Graph notes e1, Sari's Videos, Lec 18 e1) (B: slides, scribbles)	More Shortest Paths (Solutions) (Recording)	MST Algorithms (B: slides, scribbles)	MST (Solutions) (Recording)
Nov 1-4	Midterm 2 (Recursion/DP/Graph Algorithms) - Tuesday, Nov 1 12:30-13:45 (Skill set, folder, cheat sheet)	No instruction	Reductions (Sari's Videos, Lec 21 e1) (B: slides, scribbles)	Reductions (Solutions) (Recording)
Nov 8-11	SAT, NP and NP-Hardness (Sari's Videos, Lec 23-24 e1) (B: slides, scribbles)	NP-hardness reductions (Solutions) (Recording)	More NP-Hardness (Sari's Videos, Lec 23-24 e1) (B: slides, scribbles)	More NP-Hardness (Solutions) (Recording)
Nov 15-18	Undecidability 1 (Sari's Videos, Lec 25 e1) (B: slides, scribbles)	Undecidability reductions (Solutions) (Recording)	Undecidability 2 (Sari's Videos, Lec 26 e1) (B: slides, scribbles)	No instruction
Fall Break (Nov 19-27). Have fun.				
Nov 29 - Dec 2	Optional review for Midterm 3 (B: slides, scribbles)	Optional Review for midterm 3 (effectively office hours)	Midterm 3 (Reductions/P-NP/Decidability) - Thursday, Dec 1 15:30-16:45 (Skill set, folder, cheat sheet)	
Dec 6-9	Wrap-up, closing remarks Optional review for Final Exam (B: slides, scribbles)	Optional Review for final exam	Reading Day ICES Forms Due	
Final Exam - TBD (Skill set, cheat sheet)				

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, ...)

Formal languages and complexity (The Blue Weeks!)

Why Languages?

First 5 weeks devoted to language theory.

Why Languages?

First 5 weeks devoted to language theory.

But why study languages?

Multiplying Numbers

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.

$$\begin{array}{r} 3141 \\ \times 2718 \\ \hline 25128 \\ 3141 \\ 21987 \\ 6282 \\ \hline 8537238 \end{array}$$

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
- $O(n \log n \log \log n)$ [Schönhage-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?

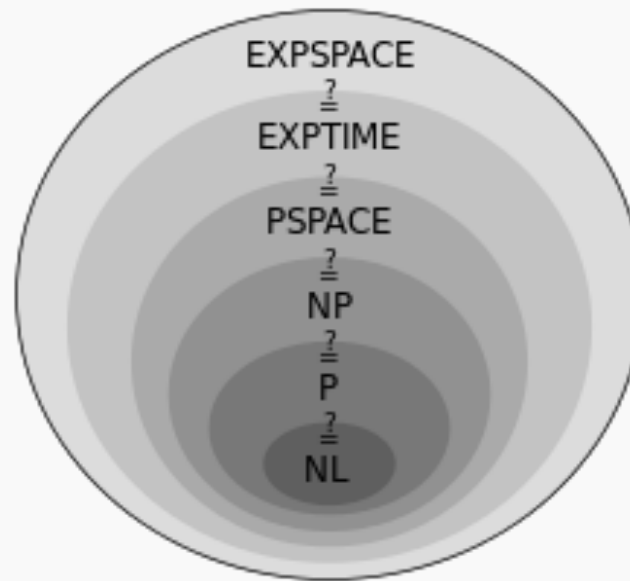
Equivalent Complexity

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)

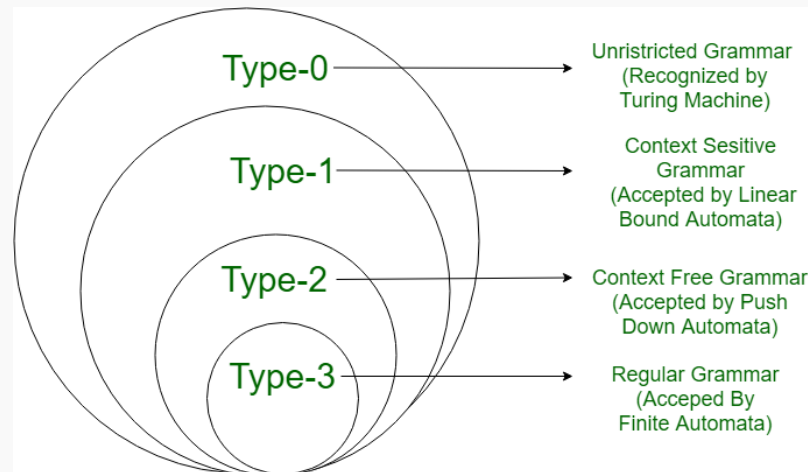
Languages, Problems and Algorithms ... oh my! II

An algorithm has a runtime complexity.



Languages, Problems and Algorithms ... oh my! III

A problem has a complexity class!



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".

Algorithms, Problems and Languages ... oh my! I

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, Σ a language is a subset of Σ^*

Algorithms, Problems and Languages ... oh my! I

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, Σ a language is a subset of Σ^* A language is a formal realization of this problem. For problem X, the corresponding language is:

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

A decision problem X is "YES" if the string is in the language.

Language of multiplication

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.

Language of multiplication

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.

$$L_{MULT2} = \left\{ \begin{array}{lll} 1 \times 1|1, & 1 \times 2|2, & 1 \times 3|3, \dots \\ 2 \times 1|2, & 2 \times 2|4, & 2 \times 3|6, \dots \\ \vdots & \vdots & \vdots \\ n \times 1|n, & n \times 2|2n, & n \times 3|3n, \dots \end{array} \right\} \quad (1)$$

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

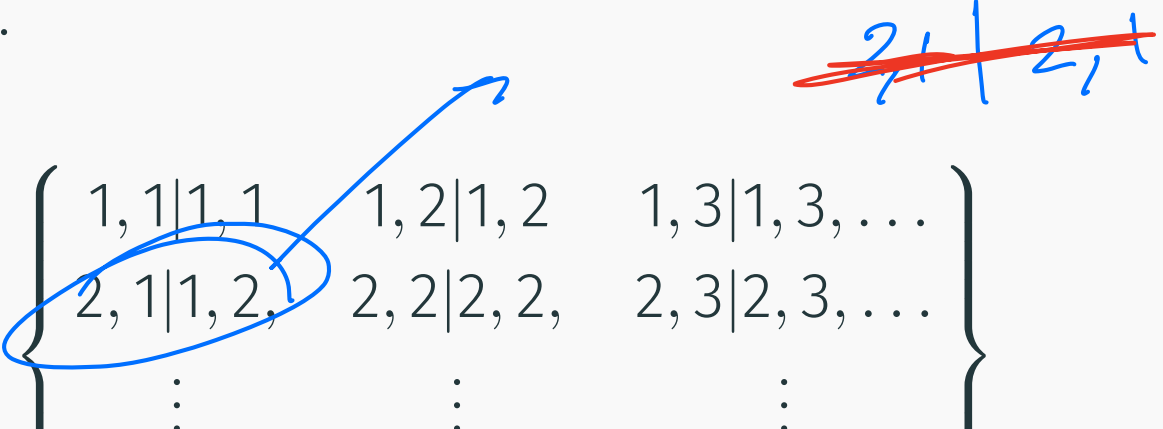
$$L_{\text{Sort2}} = \left\{ \begin{array}{ccc} 1, 1|1, 1 & 1, 2|1, 2 & 1, 3|1, 3, \dots \\ 2, 1|1, 2, & 2, 2|2, 2, & 2, 3|2, 3, \dots \\ \vdots & \vdots & \vdots \\ n, 1|1, n, & n, 2|2, n, & n, 3|3, n, \dots \end{array} \right\} \quad (2)$$

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


$$L_{\text{Sort2}} = \left\{ \begin{array}{lll} 1, 1|1, 1 & 1, 2|1, 2 & 1, 3|1, 3, \dots \\ 2, 1|1, 2, & 2, 2|2, 2, & 2, 3|2, 3, \dots \\ \vdots & \vdots & \vdots \\ n, 1|1, n, & n, 2|2, n, & n, 3|3, n, \dots \end{array} \right\} \quad (2)$$

If the same type of machine can recognize both languages, then that gives us an upperbound to their hardness.

How do we formulate languages?

Strings

Alphabet

An **alphabet** is a **finite** set of symbols.

Examples of alphabets:

- $\Sigma = \{0, 1\},$
- $\Sigma = \{a, b, c, \dots, z\},$
- ASCII.
- UTF8.
- $\Sigma = \{\langle \text{moveforward} \rangle, \langle \text{moveback} \rangle, \langle \text{moveleft} \rangle, \langle \text{moveright} \rangle\}$

String Definition

Definition

1. A **string/word** over Σ is a **finite sequence** of symbols over Σ . 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 $n \geq 0$, Σ^n is set of all strings over Σ of length n . Σ^* is the set of all strings over Σ .
5. Σ^* set of all strings of all lengths including empty string.

Question: $\{ 'a', 'c' \}^* =$

Emptiness

- ϵ 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 $\{\emptyset\}$

Concatenation and properties

- 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 $x \bullet y$.
- 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 ϵ :

$$\epsilon u = u\epsilon = u.$$

Substrings, prefixes, Suffixes

Definition

v is **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

Subsequence

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

kapa is a subsequence of *knapsack*

Subsequence

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

kapa is a subsequence of *knapsack*

Question: How many sub-sequences are there in a string $|w| = 5$?

String exponent

Definition

If w is a string then w^n is defined inductively as follows:

$$w^n = \epsilon \text{ if } n = 0$$

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

Question: $(\textit{blah})^3 =$.

Rapid-fire questions -strings

Answer the following questions taking $\Sigma = \{0, 1\}$.

1. What is Σ^0 ?
2. How many elements are there in Σ^n ?
3. If $|u| = 2$ and $|v| = 3$ then what is $|u \bullet v|$?
4. Let u be an arbitrary string in Σ^* . What is ϵu ? What is $u \epsilon$?

Languages

Languages

Definition

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

Languages

Definition

A **language** L is a set of strings over Σ . 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 $\bar{A} = \Sigma^* \setminus A$.

Set Concatenation

Definition

Given two sets X and Y of strings (over some common alphabet Σ) the **concatenation** of X and Y is

$$XY = \{xy \mid x \in X, y \in Y\} \quad (3)$$

Question: $X = \{fido, rover, spot\}, Y = \{fluffy, tabby\} \implies$
 $XY = .$

Σ^* and languages

Definition

1. Σ^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 Σ . In other words $L \subseteq \Sigma^*$.

Question: Does Σ^* have strings of infinite length?

Rapid-Fire questions - Languages

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^* , $\{\epsilon\}^*$, ϵ^* ?
4. For what L is L^* finite?
5. What is \emptyset^+ ?
6. What is $\{\epsilon\}^+$, ϵ^+ ?

Terminology Review

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**(Σ) is a set of characters
- A **string**(w) is a sequence of characters
- A **language**(A, B, C, L) is a set of strings

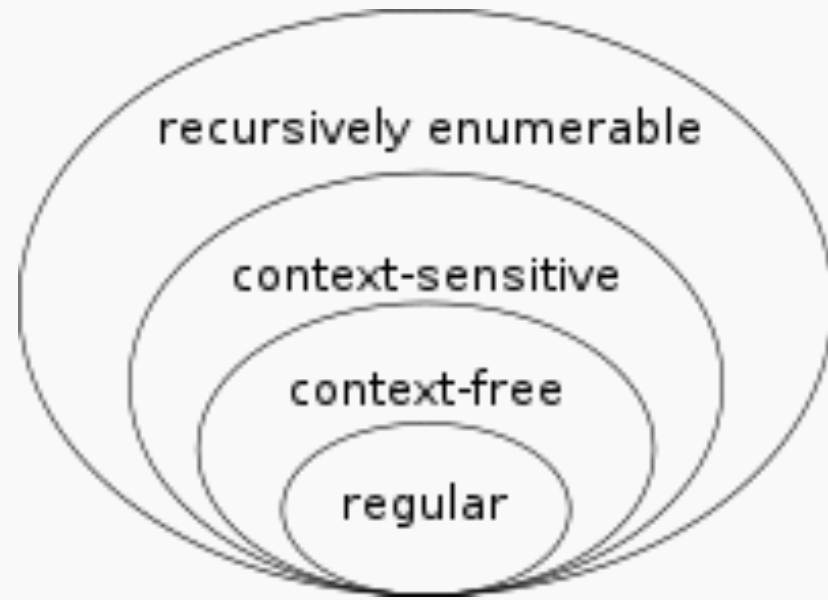
Terminology Review

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**(Σ) 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

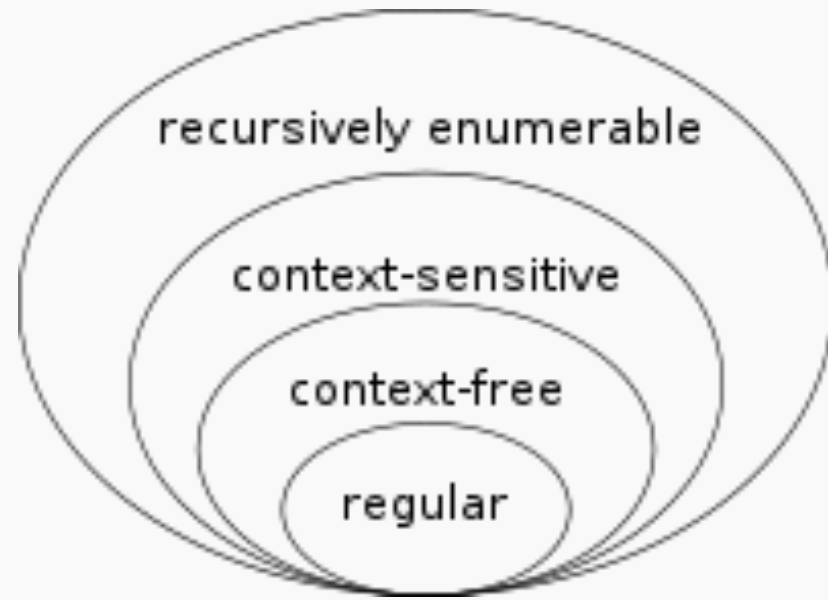
Languages: easiest, easy, hard, really hard, reallyⁿ hard

- 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).



Languages: easiest, easy, hard, really hard, reallyⁿ hard

- 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).



That's it for now

Reminders:

- NO LAB tomorrow (Wednesday 01/18/23)
 - First lab will be on Friday
- First homework will be **assigned** on Monday (Jan 23)
- Website will be updated during this week.
- Use Piazza if you have any questions/