Course Introduction & Background
CMSC 420 - Course Overview

- Fundamental data structures and the algorithms for building and maintaining them
- Mathematical methods for analyzing their efficiencies
- Applications and implementation
- See the course syllabus: 
Elements of Data Structures

- How to store, access, and manipulate data
- Fundamental to Computer Science

**Examples:**

- **Information Retrieval:** Web search, forensic search, image search
- **Geographic Information Systems:** Proximity searching (How many people live within 25 miles of the Mississippi river?)
- **Text/String Search:** Symbol tables in a compiler, document search
- **Networking:** Shortest paths/distances in a graph
- **Computer Graphics/VR:** Visibility culling - What can be seen from a given vantage point?
Elements of Data Structures

- Basic aspects of a data structure:
  - **Modeling**: How real-world objects presented as abstract mathematical entities
  - **Operations**: The allowed functions to store, access, and manipulate these entities and their formal meanings
  - **Representation**: Concrete representations in memory (e.g., array vs. linked list)
  - **Algorithms**: Computational approaches to execute these operations

- What we will teach:
  - **Tools**: Common techniques in the design of data structures
  - **Design principles**: Methods that can be widely applied in data structure design
  - **Overall**: How to design data structures, how to implement these designs, and how to evaluate them
Two Angles Towards a Common Goal

- **Theoretical:**
  - Mathematical description of the resources needed by a data structure
  - Criteria:
    - Query time
    - Construction time
    - Update time
    - Memory requirements (and possible tradeoffs with query time)

- **Practical/Empirical:**
  - Ease of implementation
  - Efficiency on actual data sets
Review: Asymptotic Analysis

- **Principles of Asymptotic Analysis:**
  - Running time as function of *basic parameters* (input size, number of vertices/edge in a graph, basic properties of the input set, dimension of the space)
  - Worst-case or average-case?
    - **Worst-case:** Maximum running time for a given input size
    - **Average-case:** Expected running time over a given probability distribution of inputs
    - **Randomized:** When algorithms use randomization, it is common to express running time in the worst-case over inputs, but in the expected-case over randomized choices
    - **Amortized:** Running time averaged over a long sequence of operations

- **Asymptotic notation** ("big-O") to focus on growth rate:

\[ T(n) = 22n^2 + 7n \log n + 5n^3 = O(n^3) \]
Review: Asymptotic Analysis

- **Examples:**
  - $O(1)$ (**Constant time**): Can’t beat this!
    - Examples: Stack push/pop, hashing (expected case)
  - $O(\alpha(n))$ (**Inverse of Ackermann’s function**):
    - Insanely slow-growing, but not constant. $\alpha(n) \leq 5$ if $n \leq$ number of atoms in universe
    - Example: Disjoint-set union/find (used in Kruskal’s MST algorithm)
  - $O(\log \log n)$
    - Example: Van Emde Boas Trees - For storing small integers
  - $O(\log n)$ (**Logarithmic time**): “Gold standard” for comparison-based structures
    - Examples: AVL tree, Red-Black tree, 2-3 tree, ...
More Examples:

- $O((\log n)^c)$ (Polylogarithmic time):
  - Example: Orthogonal range searching in 3-dimensions and higher
- $O(n^p)$, where $0 < p < 1$ (Sublinear polynomial time):
  - Examples: Nearest neighbor searching, spherical range searching
- $O(n)$ (Linear time)
- $O(n \log n)$
  - Standard for any algorithm based on comparison-based sorting
- $O(n^p)$, where $p$ is any constant (Polynomial time)
- $O(p^n)$, where $p > 1$ is any constant (Exponential time)
Summary

- Course Overview
- Basic elements of Data Structure Design
- Review of Asymptotic Analysis