Advanced Data Structures

An introduction
Topics

- Data structures
  - (review) stack, list, array, BST
  - (new) Trees, heaps, union-find, hash tables, specialized (for spatial & strings)
- Algorithm design & analysis
  - Sorting, graph algorithms
- Applications
So, what is a data structure?

A container for data that allows organized access and manipulation
Why study data structures?

- Example problem
  - Given: a set of N numbers
  - Goal: search for number k

- Solution
  - Store numbers in an array of size N
  - Linearly scan array until k is found or array is exhausted

- Number of checks
  - Best case: 1
  - Worst case: N
  - Average case: N/2
Why study data structures?

Solution #2

- Store numbers in a binary search tree
- Search tree until find k
- Number of checks
  - Best case: 1
  - Worst case: $\log_2 N$
  - Average case: $(\log_2 N)/2$
Analysis

- Does it matter?
- $N$ vs. $\log_2 N$
Analysis

- Assume
  - \( N = 1,000,000,000 \)
    - 1 billion (Walmart transactions in 100 days)
    - 1 GHz processor = \( 10^9 \) cycles per second
  - 1 cycle per transaction

- \( O(N) \) algorithm
  - 1 billion transactions = > 1 billion clock cycles

- \( O(lg \, N) \) algorithm
  - 1 billion transactions => 30 clock cycles
Example 2

- Scheduling job in a printer

- Write a code to manage the printer queue
- Functions to support
  - Insert, delete
- Special accommodations needed for:
  - Priority
  - Dynamic update
  - Scheduling challenges
Example 3

- Exploring the Facebook connection network

  - Write a code to tell who is connected to who (directly or indirectly) through your Facebook profile

  - Degrees of separation
Example 4

- Pattern matching

- Write a code to do Google search on your web database
Summary

- Keep the data organized
- Choice of data structures matters
- Appropriate data structures ease design & improve performance

Challenge
- Design appropriate data structure & associated algorithms for a problem
- Analyze to show improved performance