CSCI 104
Overview

Mark Redekopp

Aaron Cote

Administrivia 1

- In-Person
  - We are post-pandemic. In-person attendance is expected. No remote attendance
  - Zoom recordings will not be automatically posted. You may request 2 lectures.
- CS 103 / 170 Preparation
  - Basic if, while, for constructs and functions
  - Arrays, linked-lists
  - Structs, classes (constructors, destructors, operator overloading, copy semantics, inheritance)
  - **Dynamic memory allocation and pointers**
  - Basics of Recursion
  - Asymptotic Notation: Big-O/Theta/Omega notations
- All other content is on our website (https://bytes.usc.edu/cs104/)

Administrivia 2

- Syllabus
  - https://bytes.usc.edu/cs104/syllabus/
  - Exams: 1 midterm and 1 final
  - Six assignments.
    - Each assignment has a written component and a programming component
    - Key: **Start early, work consistently,** and meet the "checkpoint" schedule.
- Expectations
  - Class should be interactive. Speak up directly (I don't mind being interrupted) or raise your hand.
  - I'll give you my best, you give me yours…
    - Attendance, participation, asking questions, academic integrity, take an interest
  - Treat CS104 right!
  - Let's make this fun

Organizing Your Data

- Intentionally vague question: *"Should you always **sort** your data?"*
  - No. What are the tradeoffs?
  - An **Insert** operation becomes more expensive, but a **Lookup** operation becomes less expensive
  - In a backup system, you are constantly inserting information, and you rarely (hopefully never) performing lookups on that information.
- How should you organize your data? What is the best data structure?
  - The answer is, invariably, “it depends.”
  - Otherwise, this class would be called “Data Structure” (singular), I’d teach it to you today, and everyone would go home and get an A.
  - Demo…Need 2 volunteers

Data Structure Consideration

- **Some questions to consider:**
  - Will you search the data often?
  - Will data be added in small, frequent chunks?
  - Will data be added in large, infrequent chunks?
- Besides Insert and Lookup, what other operations are common?
  - **Remove** and **Update**
- Which of these operations you need, and how frequently you need each one, will dictate which data structure you select!
  - There is a data structure called a "Heap" which is really good at all of these operations… except Lookup!
  - Others, such as AVL Trees, are able to do all 4 operations fairly well (but they are worse than Heaps on every operation except Lookup!)
  - Yet others, such as Hash Tables, are usually lightning fast, but are probabilistic and occasionally produce very bad runtimes.

Why Data Structures Matter?

- Modern applications process vast amount of data
- Adding, removing, searching, and accessing are common operations
- Various data structures allow these operations to be completed with different time and storage requirements

Data Structure	Insert	Lookup	Get-Min
Unsorted List	Θ(1)	Θ(n)	Θ(n)
AVL Tree	Θ(log n)	Θ(log n)	Θ(log n)
Heap	Θ(log n)	Θ(n)	Θ(1)

Recall Θ (n) indicates that the actual run-time is bounded by some expression a*n for some $n > n_0$ (where a and n_0 are constants)

Importance of Complexity

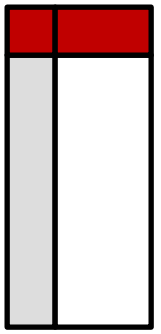
Problem Size	Bit operations used					
n =	log n	n	n log n	n^2	2^n	n!
10	3×10^{-11} s	10^{-10} s	3×10^{-10} s	10^{-9} s	10^{-8} s	3×10^{-7} s
10^2	7×10^{-11} s	10^{-9} s	7×10^{-9} s	10^{-7} s	4×10^{11} yrs	*
10^3	10^{-10} s	10^{-8} s	10^{-7} s	10^{-5} s	*	*
10^4	1.3×10^{-10} s	10^{-7} s	10^{-6} s	10^{-3} s	*	*
10^5	1.7×10^{-10} s	10^{-6} s	2×10^{-5} s	0.1 s	*	*
10^6	2×10^{-10} s	10^{-5} s	2×10^{-4} s	10.2 s	*	*

Abstract Data Types

- Programming students tend to focus on the code and less on the data and its organization
- More seasoned programmers focus first on
  - What data they have
  - How it will be accessed
  - How it should be organized
- An **abstract data type** describes what data is stored and what operations are to be performed
- A **data structure** is a specific way of storing the data implementing the operations
- Example **ADT**: List
  - Data: items of the same type in a particular order
  - Operations: insert, remove, get item at location, set item at location, find
- Example **data structures** implementing a List: Linked List, array, etc.

Another ADT

- `add(key, value)`
  - The key is a unique identified that we can use to find the value in the future.
  - `add("Tetris", 3)`
- `lookup(key)`
  - `Lookup("Tetris")`, to find `"Tetris"` sales rank
- `remove(key)`
  - `remove("Tetris")`, to remove "Tetris".
- This ADT is known as a **map**. We could implement the above map using a sorted list. So, is a sorted list an ADT?
  - No! The sorted list is the data structure. The map is the ADT.



Course Goals

01

Learn basic and advanced techniques for implementing data structures and analyzing their efficiency

- Will require mathematical analysis from CS 170

02

Learn how to identify the best data structure for your needs.

03

Learn object-oriented design principles that make your code readable, modular, and extensible