CSE 373: Wrap-up

Brad Chamberlain
University of Washington
Autumn 1999

Algorithm Requirements

Space and Time:
- asymptotic analysis for primary effects
- evaluation of secondary effects
  - by inspection
  - by experimentation

Q: How fast/space-efficient is “good enough?”
(e.g., $O(n)$ was bad for `Delete()`, but $O(n\log n)$ was great for `Sort()`…)

A:
Criteria for Good Running Time

Your resources
- how much time/memory can you afford?

Nature of the problem
- some problems are just harder than others
  (e.g., sorting is harder than deletion)

Characteristics of your application
- what problem sizes/input sets will you typically be running on? (be sure to plan for the future)

Maintainability/Elegance
- this tends to dominate software development costs

Evaluating Running Time/Space

O(1)    – ideal
O(log n) – generally as good as ideal
O(n)    – could be better, could be worse
O(n log n) – could be better, could be worse
O(n^2)  – could be better, could be worse
O(2^n)  – unusable
Games Theoreticians Play

Prove that an algorithm is \( \Omega(f(n)) \) by nature
- e.g., sorting using only comparison \(<, >, ==\) cannot be done in less than \( n \log n \) time (Chapter 7)

What’s wrong with this claim:
“\( \text{I wrote a } \text{FindMin()} \text{ operation that runs in } O(\log n) \text{ time on an unsorted list of integers} \)”

More Games Theoreticians Play

Classify problems based on their running times

\[ P - \text{ the set of problems that can be solved in } \text{polynomial time} \]
Observation

- Data is an attribute common to all programs
  - programs process, manipulate, store, display, gather
  - data may be information, numbers, images, sound
- Each program must decide how to store data
- Choice influences program at every level:
  - execution speed
  - memory requirements
  - maintenance (debugging, extending, etc.)

ADT Tensions

Ideal: a fast, elegant ADT that uses little memory

Generates tensions:
- time vs. space
- performance vs. elegance
- generality vs. simplicity
- one operation’s performance vs. another’s
The Myth of ADTs

Not a perfect black box:
- knowing how an ADT will be used can lead to a good choice of implementation
- also, knowledge of an ADT’s implementation may change how a client uses it

*But...* ADTs are still a useful concept

*Use motivates design*

Course Goals

- To introduce several standard data structures
- To teach how data structures are evaluated
- To determine when each data structure is useful

- To give you the ability to design, build, and evaluate your own data structures