CSE332: Data Abstractions

About the Final

Tyler Robison

Summer 2010

Final Logistics

- Final on Friday
  - Usual time: 10:50
  - Usual room: Here (EEB 026)
- One hour
- No notes, no books; calculators ok (but not really needed)
- Info on website under ‘Exams’

Topics (short list)

- Sorting
- Graphs
- Parallelization
- Concurrency
- Amortized Analysis

- NP NOT covered
- Material in Midterm NOT covered

Section Tomorrow

- Review problems
  - Get some more practice with material
- Questions
- Last (?) opportunity for re-grading on hw/project

Preparing for the Exam

- Homework a good indication of what could be on exam
- Check out previous quarters' exams
  - Length differs
  - 326 ones differ quite a bit
  - Final info site has links
- Make sure you:
  - Understand the key concepts
  - Can perform the key algorithms

Sorting Topics

- Know
  - Simple sorts
  - Heap Sort
  - Merge Sort
  - Quick Sort
  - Bucket Sort & Radix Sort
- Know run-times
- Know how to carry out the sort
- Lower Bound for Comparison Sort
  - Won’t be ask to give full proof
  - But may be asked to use similar techniques
  - Be familiar with the ideas

Graph Topics

- Graph Basics
  - Definition; weights; directedness; degree
  - Paths; cycles
  - Connectedness (directed vs undirected)
  - ‘Tree’ in a graph sense
  - DAGs
- Graph Representations
  - Adjacency List
  - Adjacency Matrix
  - What each is; how to use it
- Graph Traversals
  - Breadth-First
  - Depth-First
  - What data structures are associated with each?

Graph Topics

- Topological Sort
- Dijkstra's Algorithm
  - Doesn't play nice with negative weights
- Minimum Spanning Trees
  - Prim's Algorithm
  - Kruskal's Algorithm
- Know algorithms
- Know run-times

Parallelism

- Fork-join parallelism
  - Know the concept; diff. from making lots of threads
  - Be able to write pseudo-code
  - Reduce: parallel sum, multiply, min, find, etc.
  - Map: bit vector, string length, etc.
- Work & span definitions
- Speed-up & parallelism definitions
- Justification for run-time, given tree
- Justification for ‘halving’ each step
- Amdahl’s Law
- Parallel Prefix
  - Technique
  - Span
  - Uses: Parallel prefix sum, filter, etc.
- Parallel Sorting

Concurrency

- Race conditions
- Data races
- Synchronizing your code
  - Locks, Reentrant locks
  - Java's 'synchronize' statement
  - Readers/writer locks
  - Deadlock
  - Issues of critical section size
  - Issues of lock scheme granularity – coarse vs fine
- Knowledge of bad interleavings
- Condition variables
- Be able to write pseudo-code for Java threads, locks & condition variables

Amortized Analysis

- To have an Amortized Bound of O(f(n)):
  - *There does not exist a series of M operations with run-time worse than O(M*f(n))*
- Amortized vs average case
- To prove: prove that no series of operations can do worse than O(M*f(n))
- To disprove: find a series of operations that's worse