CSE 373

Autumn 2012

Looking Forward,
Looking Back

Today's Outline

- **Announcements**
  - Final Exam – next Tues Dec 11th , 2:30-4:20pm
  - Office Hours Next week:
    - Mon 3/10 2:30-3:30pm, Ruth, CSE 360
    - Tues 3/11 10-11am, Tanvir, CSE 216
    - Tues 3/11 12-2pm, Ruth, CSE 360
- **Course Review/Overview**
- **Technical Interviews**
- **Course Evaluations**

Final Exam

- **Final Exam**, Tuesday, Dec 11th , 2012
- 2:30 - 4:20pm in our regular lecture room
- Exam policies
  - Closed book, closed notes. No Calculators allowed.
  - The exam begins promptly at 2:30pm and ends at 4:20pm.
- The Final exam is cumulative, although more weight will be given to topics covered since the second midterm.

Overview and Goals

(From first day handout)

Achieve an understanding of fundamental data structures and algorithms and the tradeoffs between different implementations of these abstractions. Theoretical analysis, implementation, and application. Lists, stacks, queues, heaps, dictionaries, maps, hashing, trees and balanced trees, sets, and graphs. Searching and sorting algorithms.

Topics

- Stacks and Queues, array and list implementations.
- Asymptotic analysis, Big-O. Worst case, upper bound, lower bound, analyzing loops, recurrences, amortized complexity.
- Trees – definitions
- Dictionary ADT
- Binary search trees – Inorder, preorder, postorder traversals, insert, delete, find.
- AVL trees - Single and double rotations, insert, find.

Topics (cont.)

- Binary Heaps - Findmin, Deletemin, Insert. Additional operations of increase, decrease, buildheap.
- D-heaps - Findmin, Deletemin, Insert. Additional operations of increase, decrease, buildheap.
- Disjoint Union/Find. Up-trees. Weighted union (union by size) and path compression.
- Hashing. Properties of good hash functions. Selecting hash table size. Separate chaining and open addressing. Linear Probing, Quadratic Probing, & Double Hashing to resolve collisions. Rehashing.
- The memory hierarchy. Temporal and spatial locality. Data structure choice and the memory hierarchy.

Topics (cont.)

- Graphs. Directed and undirected. Adjacency list and adjacency matrix representations.
  - Topological sorting.
  - Graph searching. Depth-first, breadth-first search.
  - Shortest paths. Dijkstra's algorithm. Greedy Algorithms.
  - Minimum Spanning Tree – Prim's & Kruskal's Algorithms
- Sorting. Insertion sort, Selection sort, Heap sort, Merge sort, Quicksort. Lower bound on comparison sorting. In-place sorting. Stable sorting. Bucket sort, Radix sort.
- B-trees. Motivation, choice of M and L, Insert & delete.

Concepts

- ADT – what it is, why we have them, how to compare implementations
- Comparisons – Running time, Space, Big-O, Data Locality
- Tradeoffs – Pointers vs. Arrays, Space vs. Time
- Algorithm Design – Iteration, Recursion, Greedy Algorithms, Divide and Conquer

Skills

- Use Big-O to help you select the best data structure
- Time your code
- Java programming

Programming Projects

- Implement Stack using Linked list & array
- Implement Heaps (2,3, other heap)
- Solve Problems using data structures:
  - Maze Generation (Disjoint Sets)
  - Dijkstra's for shortest path (Graphs + Dictionary + Heap? + ???), using Java Collections

More Computer Science Courses!!

- CSE 374 Intermediate Programming Concepts & Tools (13wi)
- CSE 417 Algorithms and Complexity (13wi)
- CSE 41? New Course!! (13sp)
- CSE 154 Web Programming (13sp)
- CSE 415 Artificial Intelligence (autumn)
- CSE 410 Computer Systems (Operating Systems & Architecture) (next yr)
- CSE 413 Programming Languages and their Implementation (following year)

Technical Interviews

Good News!

- The best thing you could have done to prepare for a technical interview is to have taken this class!

More Good News!

- Studying for your final exam will also better prepare you for a technical interview!

Even More Good News!

- There are still more resources/things you can do to prepare further!

Resources

Web:

- http://www.cs.washington.edu/students/ugrad/recruiting
- http://www.sellsbrothers.com/fun/msiview/default.aspx?content=question.htm
- http://maxnoy.com/interviews.html
- http://www.glassdoor.com/index.htm
- http://www.careercup.com/

Books:

- *Programming Interviews Exposed*, Mongan, Kindler, and Giguere
- *Cracking the Coding Interview*, Laakmann McDowell
- *Algorithm Design*, Kleinberg, Tardos (textbook for algorithms course)

How to Prepare

- Review this course/textbook/slides/other content on data structures and algorithms
- Study examples of programming interview questions
- Study the company's web site!!
- Talk to people who have interviewed for similar positions recently
- Prepare answers to common questions ahead of time

Common "Soft" Questions

- Why do you want to work **here**?
- What is the biggest project you have worked on (lines of code, number of people, code written by others)?
- What was the most challenging part of that project? How did you solve it?
- Why should we hire you?
- Do you have any (good) questions for me?

Interview Process

- Meet recruiter as a career fair – could get asked some questions on the spot (but don't let that stop you from talking to them!)
- Phone Interview – could include coding in real time via Google Docs
- In person Interview – coding on whiteboard, on paper

During the Interview…

- Keep Talking!!!!
- Try to "think out loud" so interviewer can see your thought process
- Ask Clarifying Questions
  - Sometimes questions are intentionally vague to see if you will ask these questions!
  - E.g. Are negative numbers possible? Will there be duplicate values? Is the value I am looking for guaranteed to be present? Does it matter how much memory I use? How big is N? Can I use this method/class from Java API?

Don't Be Silent!

- Remember, you know a lot about Data Structures and Algorithms! ☺ The interviewer might be using a different word for something you KNOW.
- Say what you DO know that seems relevant:
  - data structures you are considering and why, ones you would reject and why, questions about time and space efficiency
  - Don't be paralyzed trying to come up with most efficient answer on your first try!
- Try to solve a small example ***problem*** (e.g. find topoSort, find MST) to help you see possible algorithms to solve general case
- Think about data structures from this class – does anything seem like it would be useful?

You think you have a solution…

- Come up with a solution that works first! (even if not necessarily the most efficient) then refine it
- Describe your solution in words, to confirm you have understood the problem, before you jump straight to code
- Try your solution with a sample input to confirm it works
- Testing! Check to see if your solution handles edge cases (or at least mention that you know you should do this)
- **Be prepared to describe the running time of your solution in Big-O**
- You may also be asked about Memory use

Through the entire process – TALK – give the interviewer the chance to clarify the question or to offer suggestions

Example Questions

- Implement a stack with a linked list or array
- Determine if a linked list has a cycle
- Do a preorder traversal of a tree WITHOUT using recursion
- Determine if a binary tree is balanced
- Design a hash table
- Find which word occurs most often in a text file
- Given an unsorted array, find the kth element in sorted order
- MANY many more available on the web…

Thanks!