Data Structures Study Guide

Professor Andrew Rosen

December 7, 2017

Programming Topics

- Big O notation
- Recursion (especially related to trees)
- Sorting
  - Insertion Sort
  - Quicksort and partition
  - Merge Sort
- Huffman Encoding

Data Structures For the exam, you will not write any data structures, but you need to know how to write code that uses them.

- Lists
  - Types:
    - ArrayList
    - LinkedList
  - Big O for `add, remove, get, set` differs based on type.
- Stacks
- Queues
- Trees
  - Tree vs Binary Tree vs Binary Search Tree
  - Tree algorithms are recursive and are either `find`-based or `traversal`-based.
- Heap
  - Always a complete tree
  - How to turn array into heap in $O(n)$ time.

- Hash Tables
  - They build:
    - Sets
    - Maps
  - Hashing
    - `.hashCode()` to generate a unique key
    - `% table.length` to find a place to put it.
    - Two keys can end up directed to the same index on the table, causing a *collision*.
  - Resolving Collisions
    - Open addressing - probe for an empty spot, grab the first one we find
      - Linear probing - if we have a collision at $index$, look at $index + 1$, then $index + 2$, and so on until we hit an empty spot.
      - Quadratic probing - if we have a collision at $index$, look at $index + 1^2$, then $index + 2^2$, and so on until we hit an empty spot.
    - Chaining - Each slot in the table is actually a Linked List. When we have a collision, we just add to the end that index's linked list.
- Graphs
  - Representations:
    - Adjacency List
    - Adjacency Matrix
  - Traversal and Path algorithms
    - BFS
    - DFS
    - Dijkstra's

1 Example Finals From Other Institutions

Some of these cover material not applicable to our class.

Link

Another

Yet another

One more