

CS231 Fall 2013 Final Review

Stephanie R Taylor

December 2013

1 Overall

- generic classes (how to use them, how to write them, why we want them)
- inheritance in Java (how to use it, why and when we want it)
- be able to visually depict the data structures (i.e. be able to read code and draw the data structure that results from the code – I will tell you ahead of time which data structure it is)
- how to write good test code
- For each data structure, know the time complexity for insertion, look-up, and removal. Understand the difference between the average case and the worse case.

2 Time and Space complexity

- Big-Oh notation (how to compute it)
- Big-Theta notation (how to compute it)
- Compute Big-Oh and Big-Theta time-complexities for given algorithms (including recursive algorithms)

3 Linked Lists

- singly linked list
- circularly linked list
- doubly linked list

4 Searching Sorting

Be able to recognize and/or describe the following algorithms.

- binary search
- quick sort
- selection sort
- merge sort
- heap sort

5 Binary Trees

- understand the basic implementation strategies for trees
- understand/write code to implement 4 traversals
 - in-order
 - pre-order
 - post-order
 - level order

5.1 Binary Search Trees

- what makes a tree a BST?
- how do we add a node to a BST?
- what can go wrong if we don't balance the tree?

6 Stacks and Queues

- stacks (what operations? how to implement? time complexity?)
- FIFO queues (what operations? how to implement? time complexity?)
- priority queues (what operations? how to implement? time complexity?)

7 Graphs

- different strategies for indicating connections (store some sort of list/map of edges in each vertex or store an adjacency matrix)
- traversals

8 Skip Lists

- Can you follow code that adds something to a skip list? and then draw a picture representing the skip list?

9 Hash Tables/Maps

- how to store key/value pairs
- chaining vs. linear probing
- hash functions