CS61b Midterm 2 Study Guide, by Ivan ”Vania” Smirnov

Trees:
Properties:

- Used for hierarchical structure where subsets logical.
- Traversal is O(N), N = number of nodes.
- Find is O(log_k N), assuming bushy tree with k children at nodes.
- Insertion - add to some random child, then bubble up.
- Deletion - remov node, promote child, make sure tree is balanced.
- (Bonus) Quadtrees - use coordinate notation, divide space into 4.

Traversals:

- Inorder - Traverse child.left, node, child.right (BST only) (Infix Notation)
- Post-order - traverse children, visit node (polish (?))
- Pre-order - visit node, traverse children (Prefix notation)
- BFS - use queue, add each node to end, traverse first node.(see HKn slides)

Data Structures:

- Queue - FIFO. Pop from front, add to back.
- Heap (Priority Queue)
  - Functions: add, findLargest, removeLargest
  - Node is always greater than anything in children (or less than.).
  - Insertion - lgN. Process - insert and reheapify.
  - Removal - move (bottom, right) to top node, then reheapify (swap node with larger child).
  - Array storage - store in array starting at 1. Node at position k has children at $2k$ and $2k + 1$, and parent at floor(k/2)
- Stack - FILO. Add and pop to ends only.
- Tree - Nodes with children.
- List - can add at any place, get size, find items, etc. Standard stuff.

Sorting:

- Insertion Sort - $\Theta(Nk)$ comparisons and moves, where k is max distance of displacement.
- Quicksort - Find pivot. Split into 3 lists - less, equal, grater pivot. Recursivley process low and high end. For ”small enough” list, do insertion sort. $\Theta(NlgN)$ on average, $O(N^2)$
- Mergesort - divide into 2, recursively sort each half, merge. $\Theta(NlgN)$
- Shell’s sort - sort distant elements then closer elements. Every 15, then every 7, every 3, etc. Serves to cut down inversions over large distances.
- Heapsort - Use heap property to get smallest (or largest) and add it to the result list. Keep doing this till heap empty (since heap reheapifies itself after each get())
- Radix - LSD (right to left) or Most Significant Digit (left to right). Sort on each bit, making sure sort is stable. By the end this is sorted!
- Quickselection - use quicksort, to narrow down search space untill you find it. Use index to determine which half to go into.
- Selection sort - keep selecting X item and moving into result.
- Counting sort - Given an array L of integersm iterate through L to find the counts of each integer in L. Scan the counts array to produce an array A of running sums LESS THAN the current value. Reconstruct the sorted array by iterating through A and adding keys to S in sorted order. Running Time: Linear

Game Searching:

- Backtracking - always turn ”left”, if get stuck, go back to last non-checked point and turn ”right”.
- Minimax - I choose Max value, opponent chooses min value
- Use tree of possibilities.
- alpha-beta pruning - if opponent already has min move for me, he will not check moves greater than that min.
- Likewise, if I have great move, I won’t check moves worse than that.
- Since game trees often very large, need to have some sort of static eval - assign weights to different aspects and sum them for an evaluation of a position.

Java stuff:

- Parameterization - java does unboxing in the background.
- Ranges (views) - provide subset view of original data

Hashing:

- used for sets - finding stuff in large data sets.
- Hash function - converts item into bucket number. Should have uniform distribtion.
- Num of buckets = items / load factor.
- Chaining - when there is a collision, have bucket be a linked list, so you can add it to the end of the list in a bucket. (same hashcode does not imply equality, but converse is true.)
- add, find, delete all O(1) time.

Sorting and Searching:

- Sorting - arranges items in some sort of order - for retrieval, comparison, etc.
- Internal sort - keeps all data in primary memory
- External Sort - break down into batches, use external media for intermediate storage
- Comparison sort - only sorts on intrinsic order of keys.
- Selection sort - pick next largest (or smallest) and append it to current result.

1. Relax.
2. You will do GREAT!
3. The ”A” is yours.