Quiz 1 Review

High Level

- What is a problem? What is an algorithm? (R01)
- Analyzing running time: **How to count?**
  - **Asymptotics** (R01)
  - **Recurrences** (R03)
  - **Model of computation**: Word-RAM (R01), Comparison (R05)
- How to solve an algorithms problem
  - Reduce to a problem you know how to solve
    - Use a data structure you know (e.g. **search**)
    - Use an algorithm you know (e.g. **sort**)
  - Design a new algorithm (harder, mostly in 6.046)
    - Brute Force
    - Decrease & Conquer
    - Divide & Conquer (like merge sort)
    - Dynamic Programming (later in 6.006!)
    - Greedy/Incremental

Data Structure

Reduce your problem to using a data structure storing a set of items, supporting certain search and dynamic operations efficiently. You should know **how** each of these data structures implement the operations they support, as well as be able to **choose** the right data structure for a given task. (R02-R08)

Sequence Data Structure	Operations $O(\cdot)$				
	Container	Static	Dynamic		
	`build(A)`	`get_at(i)` `set_at(i,x)`	`insert_first(x)` `delete_first()`	`insert_last(x)` `delete_last()`	`insert_at(i, x)` `delete_at(i)`
Array	n	1	n	n	n
Linked List	n	n	1	n	n
Dynamic Array	n	1	n	$1_{(a)}$	n
Sequence AVL	n	$\log n$	$\log n$	$\log n$	$\log n$

Set Data Structure	Operations $O(\cdot)$				
	Container	Static	Dynamic	Order	
	`build(A)`	`find(k)`	`insert(x)` `delete(k)`	`find_min()` `find_max()`	`find_prev(k)` `find_next(k)`
Array	n	n	n	n	n
Sorted Array	$n \log n$	$\log n$	n	1	$\log n$
Direct Access	u	1	1	u	u
Hash Table	$n_{(e)}$	$1_{(e)}$	$1_{(a)(e)}$	n	n
Set AVL	$n \log n$	$\log n$	$\log n$	$\log n$	$\log n$

Priority Queue Data Structure	Operations $O(\cdot)$		
	`build(A)`	`insert(x)`	`delete_max()`
Dynamic Array	n	$1_{(a)}$	n
Sorted Dyn. Array	$n \log n$	n	$1_{(a)}$
Set AVL	$n \log n$	$\log n$	$\log n$
Binary Heap	n	$\log n_{(a)}$	$\log n_{(a)}$

Algorithm

Reduce your problem to a problem you already know how to solve using known algorithms. You should know **how** each of these algorithms can be implemented to solve each problem, as well as be able to **choose** the right algorithm for a given task.

- **Problem:** Sorting n integers (R03-R08)

Algorithm	Time $O(\cdot)$	In-place?	Stable?	Comments
Insertion Sort	n^2	Y	Y	$O(nk)$ for k-proximate
Selection Sort	n^2	Y	N	$O(n)$ swaps
Merge Sort	$n \log n$	N	Y	stable, optimal comparison
Heap Sort	$n \log n$	Y	N	low space, optimal comparison
AVL Sort	$n \log n$	N	Y	good if also need dynamic
Counting Sort	$n + u$	N	Y	$O(n)$ when $u = O(n)$
Radix Sort	$n + n \log_n u$	N	Y	$O(n)$ when $u = O(n^c)$