* <http://www.cs.ucf.edu/courses/cop3502/nihan/spr03/sort.pdf>
  + Good examples
  + And the selection problem
* Insertion Sort
  + O(N2)
  + TODO: Algorithm?
* Heap Sort
  + Average case: O(nLogn)
  + Space = O(1)
* Quick sort:
  + Divide and conquer
  + Uses partition
    - Use median of first, last and middle to avoid worst case behavior
  + Worst case N2
  + Best/Average case nLogn
  + O(1) space but stack uses O(n)
  + <https://www.cse.ust.hk/~dekai/271/notes/L01a/quickSort.pdf>
  + Reduce stack to O(nlogn) using:
    - <http://homepages.math.uic.edu/~leon/cs-mcs401-s08/handouts/quicksort.pdf>
  + Worst case of quicksort:
    - Elements are sorted in increasing sequence
    - Elements are already sorted in decreasing sequence
    - Elements are the same.
* 3 way quicksort
  + <http://algs4.cs.princeton.edu/23quicksort/>
* Median Selection:
  + Sort array and return kth element (nlogn)
  + Use partition to return kth element
  + <https://www.cse.ust.hk/~dekai/271/notes/L01a/quickSort.pdf>
* Sorting Stability
  + Stable: Insertion sort, bubble sort, merge sort are stable
  + Unstable: Heap sort or quick sort
* Minimal Tree Stuff
  + Height: Number of edges from a node to deepest level.
  + Number of nodes = 2h+1-1 (maximum/full). This includes the leaves.
* Priority Queue
  + Array Size: N,
    - Last parent: = N/2
    - Parent (i) = i/2
    - Left child(i) = 2\*i
    - Right Child = 2\*i+1
  + Build Heap:
    - N inserts: nLog(n)
    - Start from (N/2), i.e. building all nodes: O(n)
      * Complete Binary tree of height (h) has: 2h+1-1 nodes
      * Sum of height of all nodes is: n-h-1 = n-logn-1
  + Delete Min/Max: O(logn)
  + Insert: O(logn)
  + Heapsort: O(nlogn)
  + Max/Min nodes in a heap:
    - Max Nodes (Full tree)= 2(h+1)-1
    - Min Nodes (last level has one node): 2(h-1+1)-1+1=2h