**The Heap and the Quick**

* Why does Quicksort switch to insertion sort?
  + Quicksort is efficient with very large input
    - So when quicksort breaks the list down to smaller and smaller lists, insertion sort becomes more efficient.
* What happens if the partition is 90 – 10?
  + AFAIK the complexity stays at nlogn
    - Update, I believe this changes it to O(nlogn).
      * nlogn is the average time of quicksort normally so this is an improvement?
* Which pivot choice is used by Bentley and McIllroy?
  + The left-most value
* Why would you not use a random pivot?
  + Honestly I’m not sure if I can think of any case to not randomize the pivot
    - Seems like randomizing the pivot reduces the chance of hitting worst case
* What if there are multiple repeated elements? (Better partition?)
  + Yep, three way partition
    - Left, mid, right
      * 0 to left, left to mid, mid to right, right to len(n)
* Heapsort and Quicksort are both nlogn. Tradeoffs?
  + While a well implemented quicksort is usually quicker than heapsort, quicksort can take up to O(n) auxiliary space, where heapsort uses only O(1).
    - Anything else?
* Why would one try an iterative implementation of either?
  + I don’t see a reason to try this in either quick or mergesort.
* Which sort would be easier to implement in parallel?
  + QUICKSORT

**Sorting Theory and Its Limits**

* What does it mean for a sort to be stable?
  + If two equal keys appear in the same order in sorted output as they appear in the input data set.
* If we could find the median of an array in O(n), could we improve quicksort runtime?
  + Yes! We could make it to be O(nlogn), as opposed to O(n2)
* Can we make all sorting algorithms seen previously stable?
  + I think so?
    - CHECK THIS

**Unweighted Graph Algorithms**

* Not sure how to put this in a word doc so just check your pictures if we work on these in person

**Weighted Graph Algorithms**

* How can we detect negative cycles?
  + Use Bellman-Ford
* How could we stop early?
  + What does this mean lol