32.4 Summary

Quicksort

What can we do to avoid the worst case runtime of $heta(N^2)$ for Quicksort?

- 1. Randomness: pick a random pivot point, shuffle items before sorting
- 2. Smarter Pivot Selection: constant time pivot pick (ex: pick a few and then choose the best out of them), linear time pivot pick like (ex: median)
- 3. Introspection: switch to a different sort if current sort hits recursion depth threshold

Quicksort vs. Mergesort

- Mergesort is faster than Quicksort L3S (leftmost pivot, 3-scan partition) and QuickSort PickTH (median pivot, Tony Hoare partition)
- Quicksort LTHS (leftmost pivot, Tony Hoare partition) is faster than Mergesort!
- Tony Hoare's partitioning:
 - two pointers, one at each end of the items array, walking towards each other
 - left pointer hates larger or equal items, right pointer hates smaller or equal items
 - swapping anything they don't like; stop when the two pointers cross each other
 - new pivot = Item at right pointer.

Quick Select

Quick select helps us quickly find the median with partitioning. Median of an length n array will be around index n /2.

- 1. Initialize array with the leftmost item as the pivot.
- 2. Partition around pivot.
- 3. Partition the subproblem. Repeat the process.

4. Stop when the pivot is at the median index.

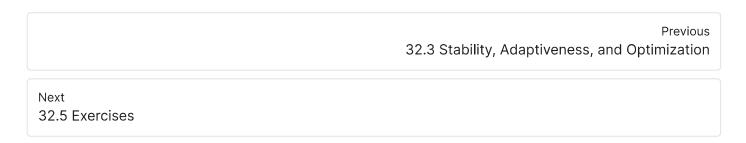
Expected runtime: $\theta(N)$. Worst case runtime (when array is in sorted order): $\theta(N^2)$.

Stability, Adaptiveness, and Optimization

Stability: A sort is stable if the order of equivalent elements is preserved.

Optimizations:

- Adaptiveness sort that exploits the existing order of the array.
- Switch to Insertion Sort when a subproblem reaches size 15 or lower
- Exploit restrictions on set of keys
- Switch from QuickSort if the recursion goes too deep



Last updated 1 year ago

