Week 14: Sorting

Friday, December 11, 2020 9:58 AM

Review

Selection Sort

- sorted & unsorted sections
- repeatedly find min of unsorted section & put at end of sorted
- $O(n^2)$

Insertion Sort

- Sorted & unsorted sections
- go backwards through sorted section & find place to insert current unsorted element
- best case: $O(n)$
- worst case: $O(n^2)$

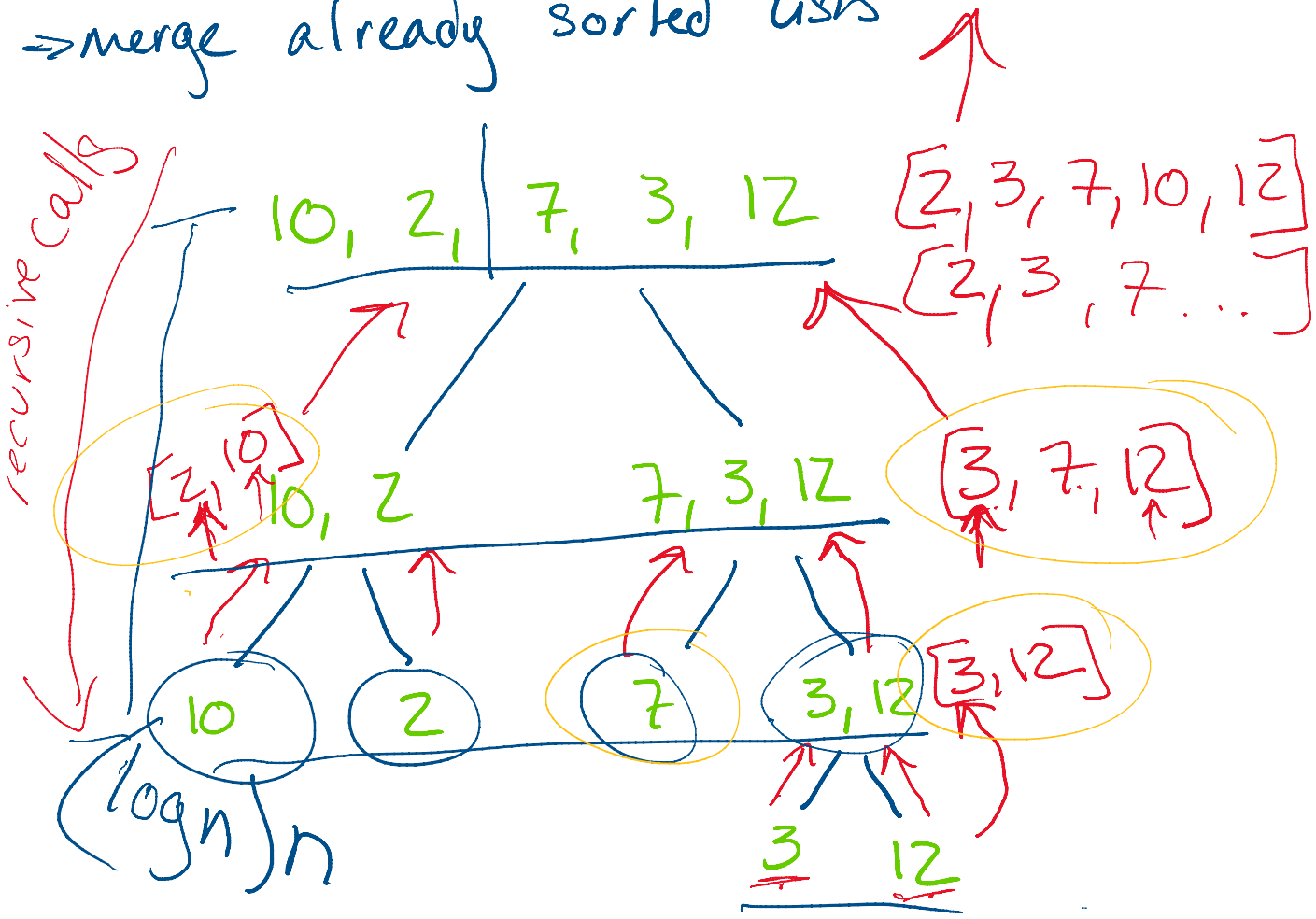
Divide & Conquer Algorithms

- breaks up a problem into multiple subproblems & recursively solve subproblems
  - * usually on logn scale

Merge Sort

Merge Sort

- recursively split list in 2
  - until list 1 or 0 elements
- merge already sorted lists



* create new output array at each merge call
  * space inefficient for arrays

runtime: $O(n \log n)$

- $\log n$: depth of recursion tree

→ logn: depth of recursion (splitting in half)

→ n: at every level, working with all elements

Quick Sort

→ choose a pivot
  → need to choose a good pivot
  → basis of partition
  → choose first/last element
  → choose at random is best & most efficient

→ partition list
  → Left: < pivot
  → Right: > pivot
  → Center: = pivot
  → left & right

→ Recurse on left & right

Runtime:

- → good pivot:
  - → split in half every time
  - → depth of tree: log n
  - → n work at each level to partition list

average* $O(n \log n)$

*no extra space

- → bad pivot
  - → everything on 1 side
  - → essentially just removing 1 element each time
  - → depth of tree: $O(n)$
  - → n work at each level

→ n work at each level

$O(n^2)$