Week 14: Sorting

Friday, December 11, 2020 11:59 AM

Review

Selection Sort

→ Sorted & unsorted sections

→ repeatedly find smallest element in unsorted section & swap with first element in unsorted

→ $O(n^2)$

Insertion Sort

→ Sorted & unsorted sections

→ go backwards through sorted section & find where current unsorted element belongs, then insert it.

→ worst case: $O(n^2)$

→ best case: $O(n)$

* efficient for small (≤ 10) lists

Divide and Conquer Algorithms

→ break up a problem into similar smaller problems, then solve small problems

Merge Sort

→ recursively split list in half

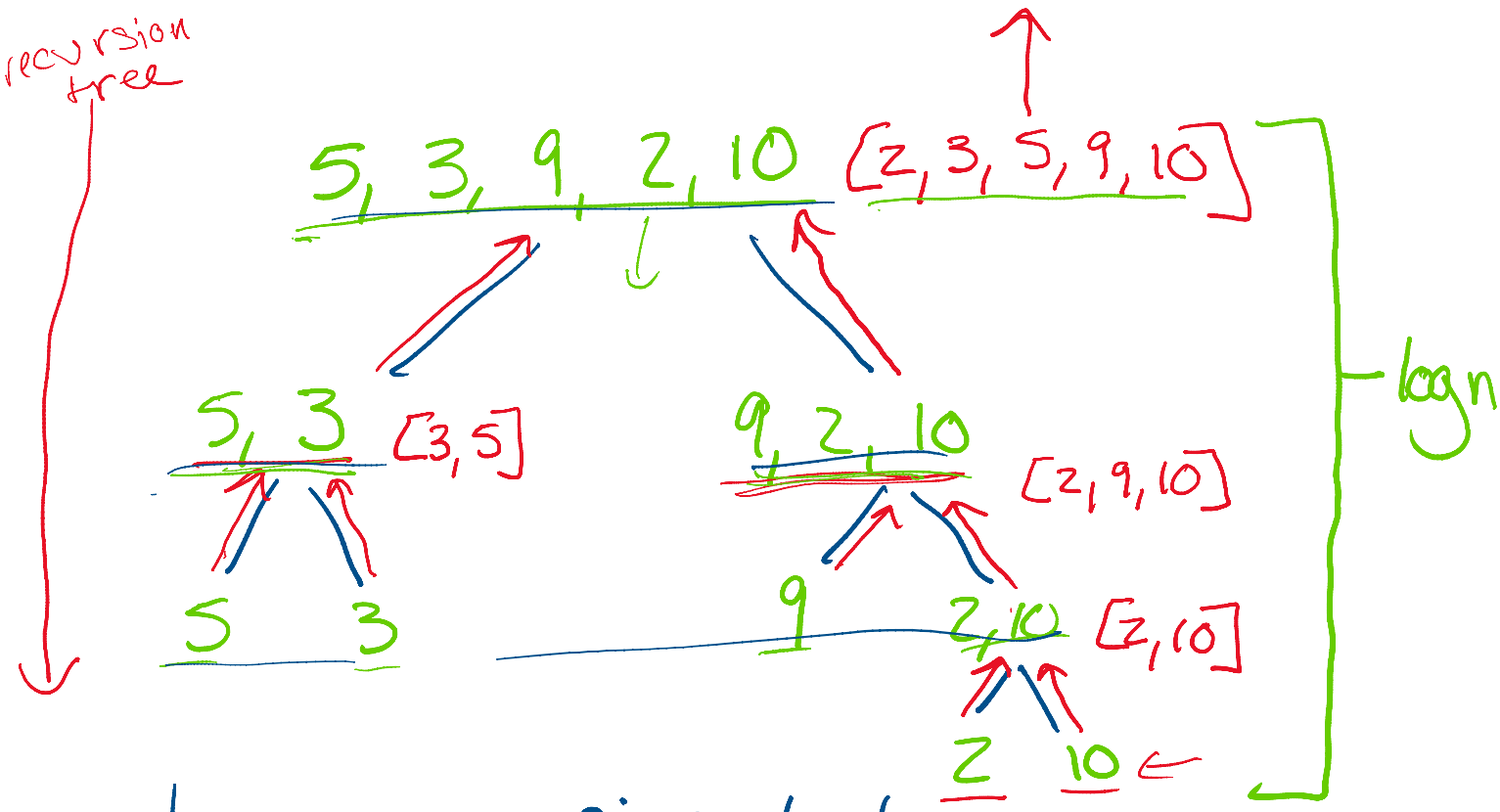
→ merge sorted lists

→ go through left & right, putting smaller element next in returned array

L: 2, 5, 8, 10

R: 1, 5, 7, 9, 11

Our: 1, 2, 5, 5, 7, 8, 9, 10, 11



→ log n recursive levels

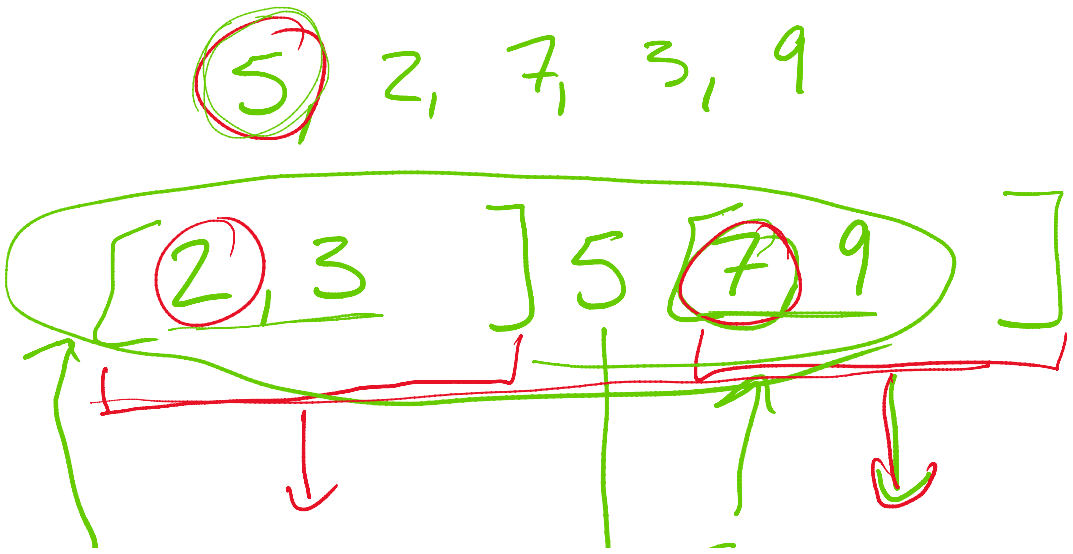
→ O(n) work at each level

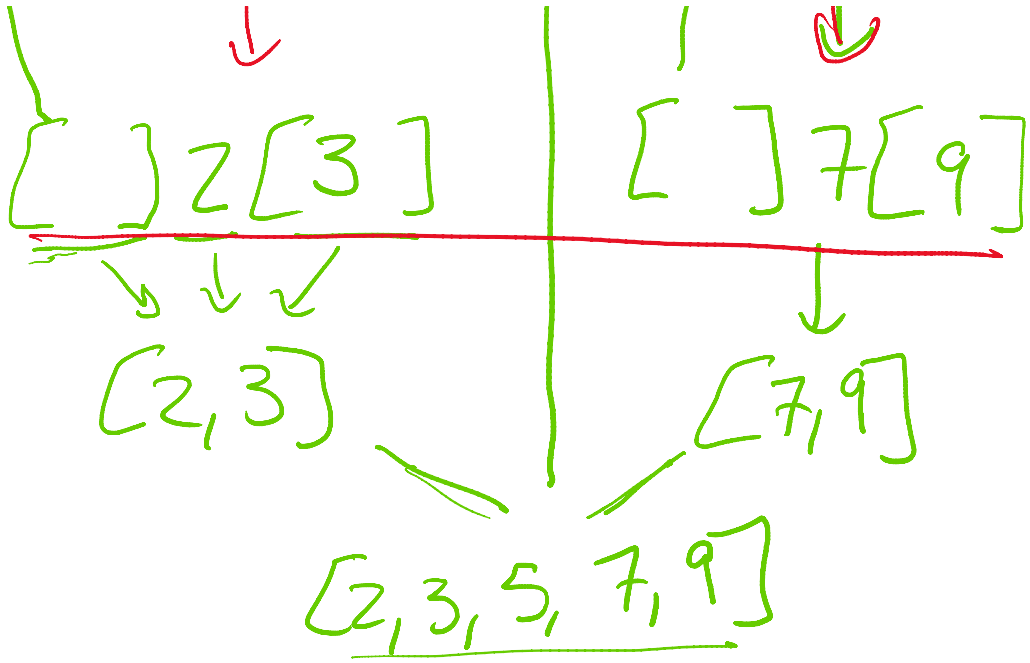
*runtime: n log n
  *guaranteed

*downside: merge needs to create new arrays
  *wastes quite a lot of space

Quick Sort

- choose a pivot
  - choose first/last
  - choose random element
    - * very high probability of being a good pivot
- partition elements
  - left: < pivot
  - right: > pivot
  - center: = pivot
- recurse on left & right





2, 3, 5, 7, 9

3, 5, 7, 9

5, 7, 9

7, 9

9

→ runtime: depends on pivot
  → good pivot:
    → half in left half in right
    → log n recursive levels

→ log n recursive levels
→ n work in partition
* O(n log n) *average

→ bad pivot:
→ only remove 1 element instead of splitting
→ O(n) recursive levels
→ O(n) in partition
* O(n²)

* upside: don't need extra space