Sorting

Saturday, February 17, 2018 1:09 AM

- Sort Key
- Best/ worst/ average case bounds
- In-place sort: O(1) extra space during sorting
- Stable sort: relative order of elements with same key value preserved
- Algorithms can be combined for the best result

Applications

- Uniqueness testing, deleting duplicates, frequency count, set notation, searching (efficiently), dictionary/ directory

Comparison-based

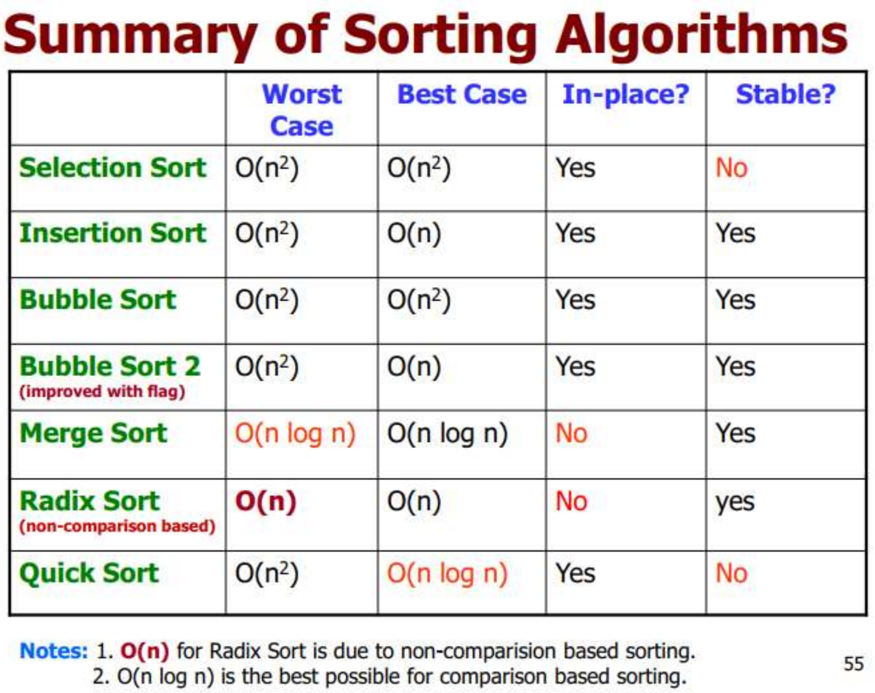
Iterative	Recursive Divide-and-conquer: recursively solve
Selection Sort • Find largest -> put at back • O(n^2)	Merge Sort • Most of the work done in merge() step • merge() called log n times, each step = • Additional temporary arrays needed, inputs have to be copied to original array
Bubble Sort • Need to swap ith and i+1th items? • O(n^2), even for sorted input ○ Improvement: flag to check if input is sorted ○ Mark out which portions are already sorted • Best case = O(n), for outer iteration	Quick Sort • Pivot p, split data into 2 parts recursively • Pivot is randomly selected • Most of the work done in divide step • In-place sorting (only swapping operations • Complexity = O(n) per partition • Best case = depth log n, O(n log n) • Worst case = already sorted; O(n^2) ○ What if order is reversed?
Insertion Sort • Scan backwards and insert • Best case = O(n), worst case = O(n^2)	

Non-comparison based

- Radix Sort and Heap Sort

Radix Sort

- "radix" refers to position of decimal points
- Each data is a character string
- O(d*n); d = number of possible characters
  - Worst case, d is log n, hence O(n log n)
  - d can be fixed or bounded to give O(n)** (usually the case)
- Last digit > second last > .... > first digit



Summary of Sorting Algorithms

	Worst Case	Best Case	In-place?	Stable?
Selection Sort	$O(n^2)$	$O(n^2)$	Yes	No
Insertion Sort	$O(n^2)$	$O(n)$	Yes	Yes
Bubble Sort	$O(n^2)$	$O(n^2)$	Yes	Yes
Bubble Sort 2 **(improved with flag)**	$O(n^2)$	$O(n)$	Yes	Yes
Merge Sort	$O(n \log n)$	$O(n \log n)$	No	Yes
Radix Sort **(non-comparison based)**	**$O(n)$**	$O(n)$	No	yes
Quick Sort	$O(n^2)$	$O(n \log n)$	Yes	No

Notes: 1. **O(n)** for Radix Sort is due to non-comparision based sorting.
2. O(n log n) is the best possible for comparison based sorting.

Java API

- `Array.sort()`
- `Collections.sort(list)  // for list`
- Non-primitive data: apply a comparator