Sorting
Array and String Methods

Complexity: Big O notation

The Big O is basically a way to describe an algorithms “abstract” performance in operation as the operation’s inputs grow in size toward infinity.

Bubble Sort

A. Compare each pair of adjacent elements from the beginning of an array and, if they are in reversed order, swap them.

B. If at least one swap has been done, repeat step A.

Bubble Sort

6 5 3 1 8 7 2 4

Bubble Sort Complexity

Worst-case performance - $O(n^2)$

Best-case performance - $O(n)$

Average performance - $O(n^2)$

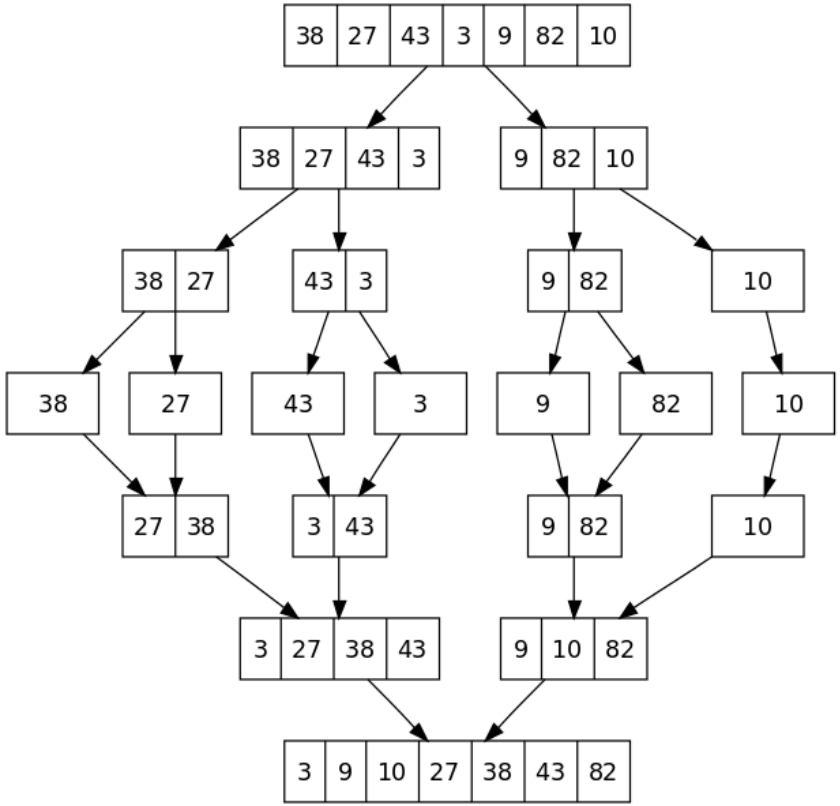
MergeSort

A. If the list is of length 0 or 1, then it is already sorted. Otherwise:
B. Divide the unsorted list into two sublists of about half the size.
C. Sort each sublist recursively by re-applying merge sort.
D. Merge the two sublists back into one sorted list.

MergeSort

6 5 3 1 8 7 2 4

MergeSort - Sorting Tree



MergeSort Complexity

Worst-case performance - O(n log n)

Best-case performance - O(n log n) typical,

O(n) natural variant

Average performance - O(n log n)

https://www.toptal.com/developers/sorting-algorithms

https://www.cs.usfca.edu/~galles/visualization/ComparisonSort.html

http://www.stoimen.com/blog/2010/07/09/friday-algorithms-javascript-bubble-sort/

http://codingmiles.com/sorting-algorithms-bubble-sort-using-javascript/

http://www.stoimen.com/blog/2010/07/02/friday-algorithms-javascript-merge-sort/