Sorting - useful for a zillion purposes

In previous experiences:

"slow sorts" - selection sort
insertion sort
bubble sort

$\rightarrow O(n^2)$

$1 + 2 + 3 + \cdots + \underbrace{n-1}_{\text{ish}} \rightarrow O(n^2)$

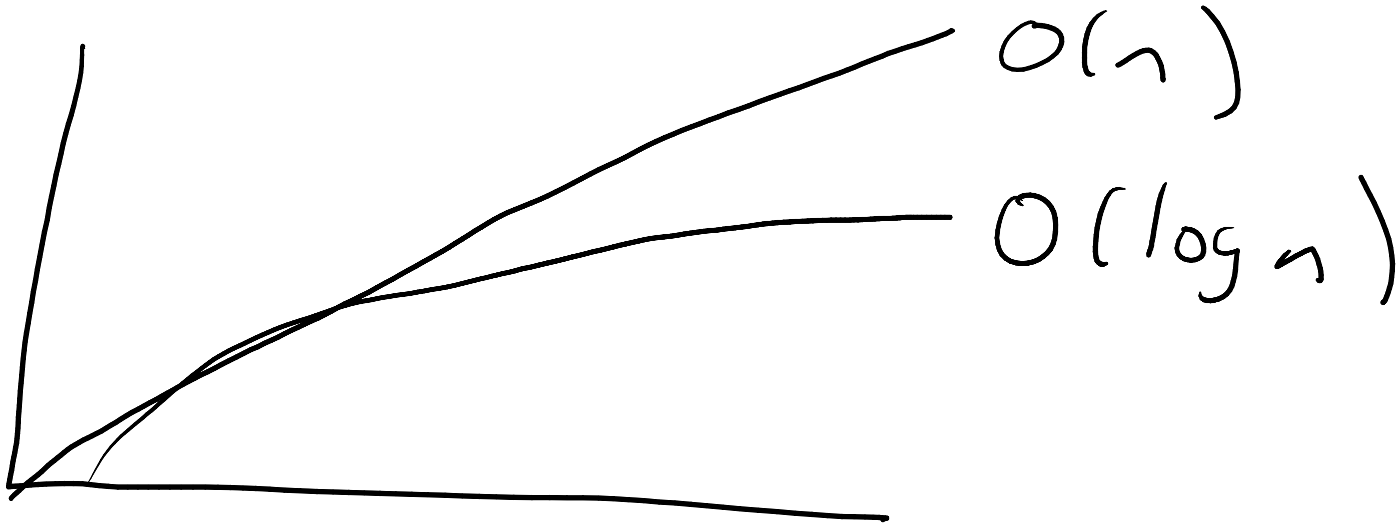
In this class

mergesort
quicksort
heapsort

all are $O(n \log n)$ [with details, caveats, etc]

$O(n^2)$ $\quad$ $O(n \log n)$

$O(n * n)$ $\quad$ $O(n * \log n)$



Mergesort → recursive!

- mergesort left half
- mergesort right half
- merge two halves together

6, 2, 4, 3, 9, 8, 1, 7

6 2 4 3   9 8 1 7

6 2   4 3   9 8   1 7

6   2   4   3   9   8   1   7

2 6   3 4   8 9   1 7

2 3 4 6   1 7 8 9

1 2 3 4 6 7 8 9

How much work?

Splitting takes essaticlly no work at all, because all we really do is accounting

- "mergesort list indices 0 → half"
- "mergesort list incides half+1 → end"

Work is in the merging.

6 2 4 3 9 8 1 7

worst case: n

2 6 3 4 8 9 1 7

worst case: n

2 3 4 6 1 7 8 9

worst case: n

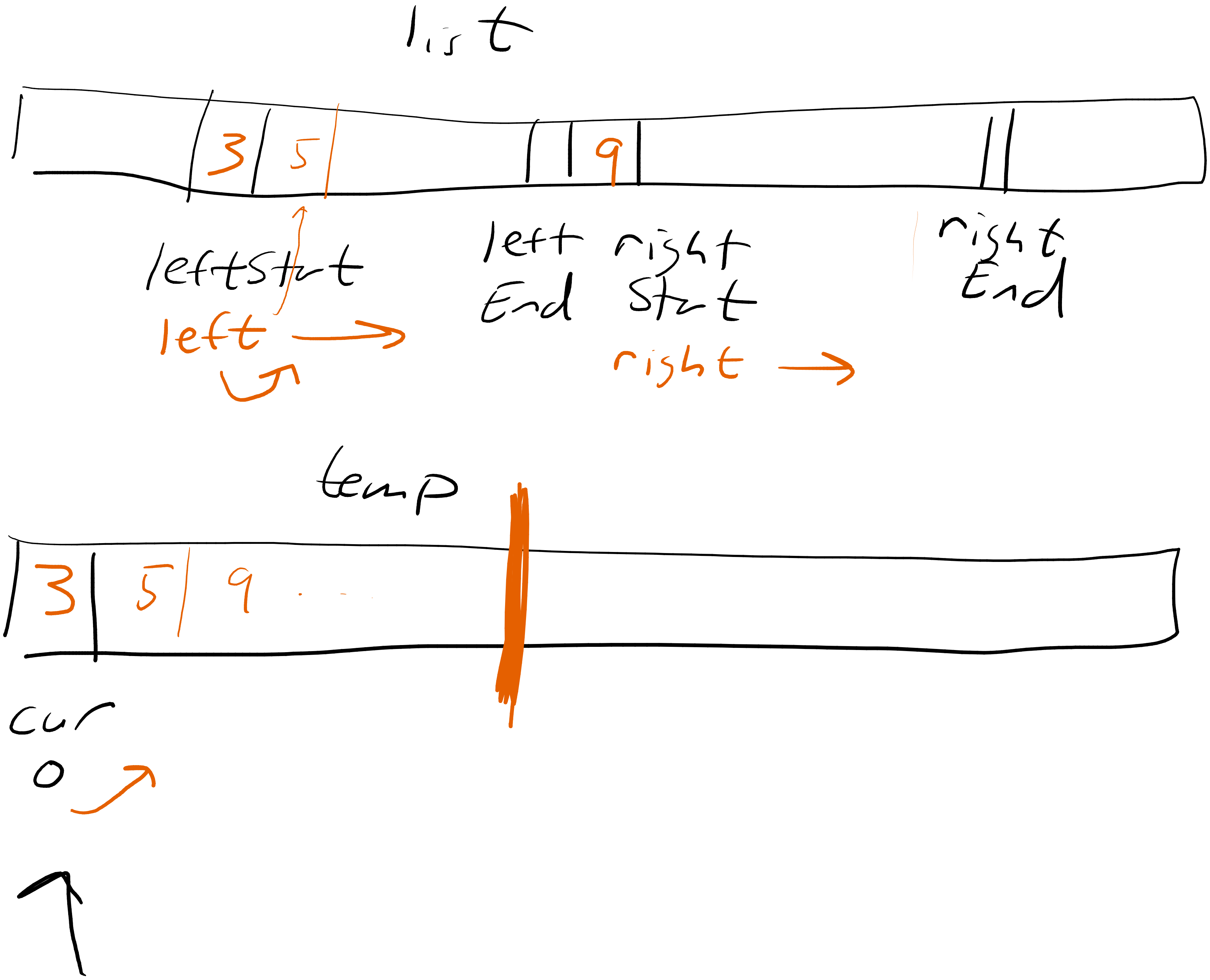
1 2 3 4 6 7 8 9

How many numbers were rewritten?

Performance: n * # of rows above

$\log n$

$n * \log n$



started at 0 in temp list, whereas leftStart in original list was elsewhere