30.3 Quicksort Performance Caveats

"Quicksort" was chosen by Tony Hoare as the name for this particular partition sort algorithm. Coincidentally, quicksort is empirically the fastest sort for most common situations.

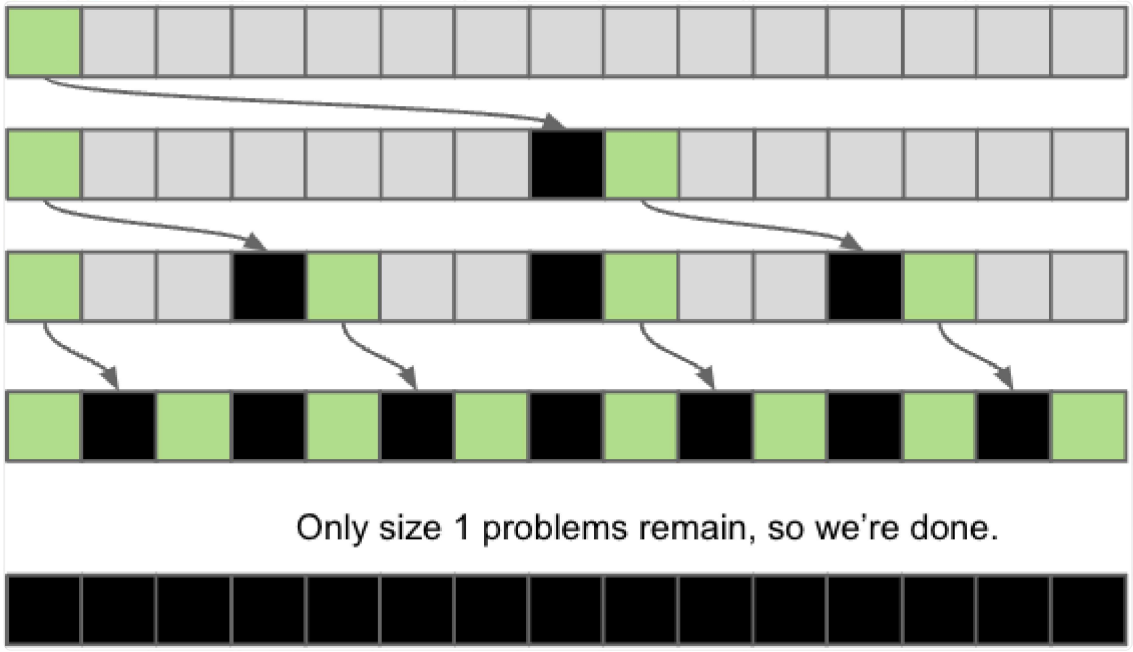
How fast is Quicksort exactly? To answer this question, we need to count the number and calculate the difficulty of the partition operations employed in Quicksort.

In the theoretical runtime analysis, partitioning costs $\Theta(K)$ time, where $\Theta(K)$ is the number of elements being partitioned.

However, what complicates runtime analysis for Quicksort is that the the overall runtime will depend on where the pivot ends up.

Best Case

For instance, the best case runtime is when the pivot chosen always lands in the middle (in other words, the partition always picks the median element of the array).



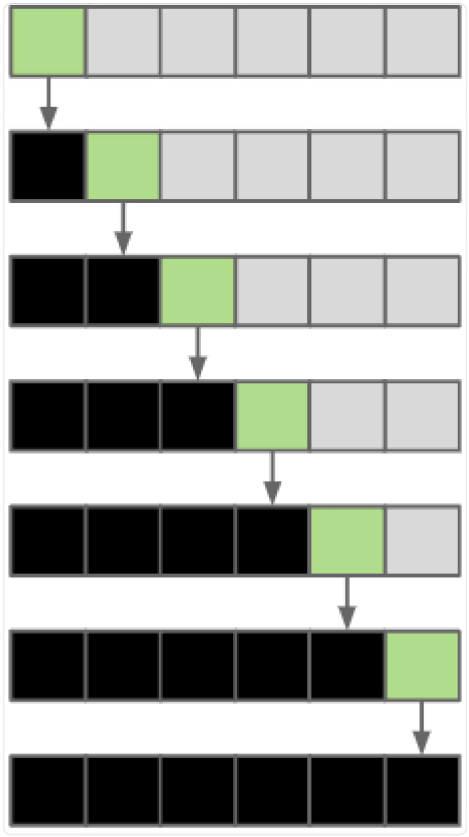
Best Case

In the best case, the total work done at each level is approximately O(N). To see this, the first partition pivots on one element and requires checking all N elements to determine whether they go on the right or the left of the pivot element. The second layer repeats this, but for N/2 elements on two separate pivots (since we are recursively partitioning on the two halves). Thus, each level is O(N) work.

The overall runtime becomes $\Theta(NH)$, where H = # of layers = $\Theta(logN)$. Thus, the total runtime for Quicksort in the best case is $\Theta(NlogN)$.

Worst Case

The worst case for runtime occurs when the pivot chosen at each partition lands at the beginning of the array. In this scenario, each layer still has to do O(N) work, but now there are H = # of layers = N layers, since every single element has to be pivoted on. Thus, the worst case runtime is $\Theta(N^2)$.



Worst Case: $\Theta(N^2)$

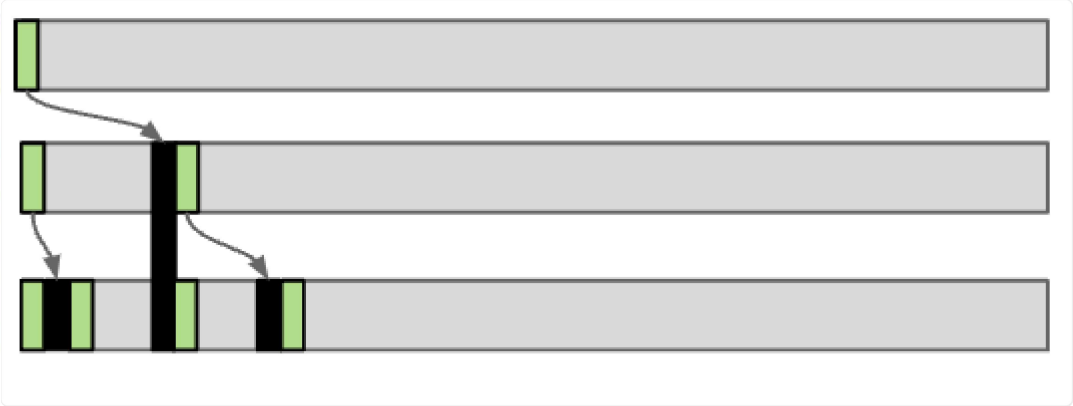
Quicksort vs Mergesort Performance

Theoretical Analysis	Quicksort	Mergesort
Best Case	$\Theta(NlogN)$	$\Theta(NlogN)$
Worst Case	$\Theta(N^2)$	$\Theta(NlogN)$

In comparison, Mergesort seems to be a lot better, since Quicksort suffers from a theoretical worst case runtime of $\Theta(N^2)$. So how can Quicksort be the fastest sort empirically? Quicksort's advantage empirically comes from the fact that on average, Quicksort is $\Theta(NlogN)$.

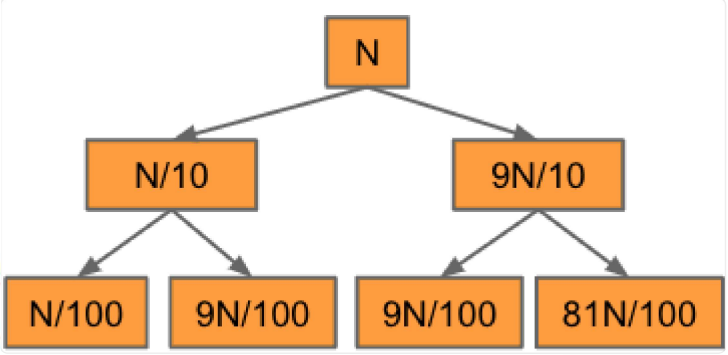
Quicksort's Performance Argument #1: 10% Case

One argument for why Quicksort is the fastest sort empirically supposes that the pivot always ends up at least 10% from either edge.



Pivot is ~10% from the left edge.

If this is the case, then the runtime is still O(NH), where O(N) is the work at each level and H is the # of levels. Since the pivot always lands at least 10% from either edge, H is approximately $\log_{10/9} N$ = O(log N). Thus, overall runtime is O(N log N).

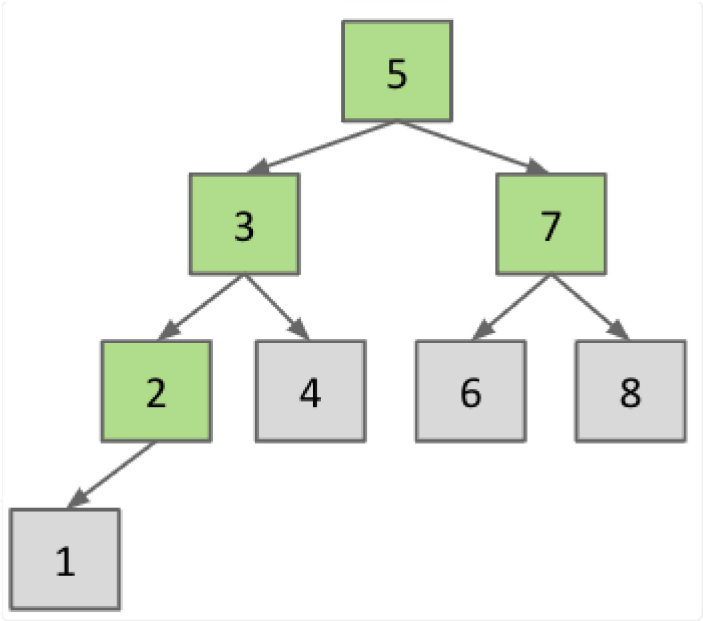


10% case partition work at each level.

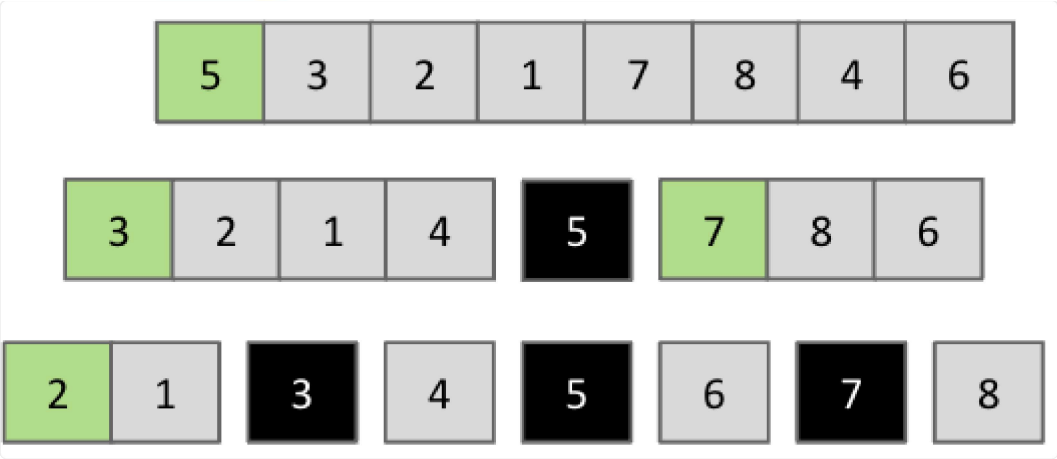
In other words, even if you are not lucky enough to have a pivot that is near the middle, if you can at least have a pivot that is always 10% from the edge, the runtime will be O(N log N).

Quicksort's Performance Argument #2: Quicksort is BST Sort

From another lens, Quicksort can be seen as a form of BST sort. This is because the compareTo calls that Quicksort employs between each element and the pivot element in each partition is the same as the compareTo calls performed in BST insert.



BST insert



Quicksort partitions on the "same" BST elements as above.

Since random insertion into a BST takes O(N log N) time, Quicksort can be argued to have similar performance.

Empirical Quicksort Runtime

Empirically, for N items, Quicksort uses an average of ~2N ln N compares to complete (across 10,000 trials with N = 1000). For more information, check out this link.

Avoiding the Worst Case of Quicksort

As you can see from above, the performance of Quicksort (both in terms of order of growth and in constant factors) depends critically upon **how you select the pivot, how you partition around the pivot, and other optimizations that you can add to speed things up.**

Given certain conditions, Quicksort can result in $\Theta(N^2)$, which is much worse than $\Theta(NlogN)$. Some of these conditions include **bad ordering,** where the array is already in sorted order (or almost sorted order), and **bad elements,** where the array has all duplicates.

Previous
30.2 Quicksort Algorithm

Next
30.4 Summary

Last updated 1 year ago