Choosing a Pivot

In the very early versions of quicksort, the leftmost element of the partition would often be chosen as the pivot element. Unfortunately, this causes worst-case behavior on already sorted arrays, which is a rather common use-case. The problem was easily solved by choosing either a random index for the pivot, choosing the middle index of the partition or (especially for longer partitions) choosing the median of the first, middle and last element of the partition for the pivot (as recommended by Sedgewick).[7] This "median-of-three" rule counters the case of sorted (or reverse-sorted) input, and gives a better estimate of the optimal pivot (the true median) than selecting any single element, when no information about the ordering of the input is known.

Specifically, the expected number of comparisons (see §Analysis of randomized quicksort) with random pivot selection is $1.386\ n \log n$. Median-of-three pivoting brings this down to $C_{n,2} \approx 1.188\ n \log n$, at the expense of a three percent increase in the expected number of swaps.[5] An even stronger pivoting rule, for larger arrays, is to pick the ninther, a recursive median-of-three, defined as[5]

> ninther(a) = median(median-of-three(first ⅓ of a), median-of-three(middle ⅓ of a), median-of-three(final ⅓ of a))

> Selecting a pivot element is also complicated by the existence of integer overflow. If the boundary indices of the subarray being sorted are sufficiently large, the naïve expression for the middle index, $(lo + hi)/2$, will cause overflow and provide an invalid pivot index. This can be overcome by using, for example, $lo + (hi{-}lo)/2$ to index the middle element, at the cost of more complex arithmetic. Similar issues arise in some other methods of selecting the pivot element.

What is the difference between using $(lo + hi)/2$ and $lo + (hi{-}lo)/2$ to find the mean of two numbers?

You can simple write `(left + right) / 2`.

```
    left + (right-left)/2
=>  2*left/2 + (right-left)/2    //multiply (left * 2/2)
=>  (2*left + right-left)/2
=>  (left + right)/2
```