Decreasing Array

Elizabeth Goodwin

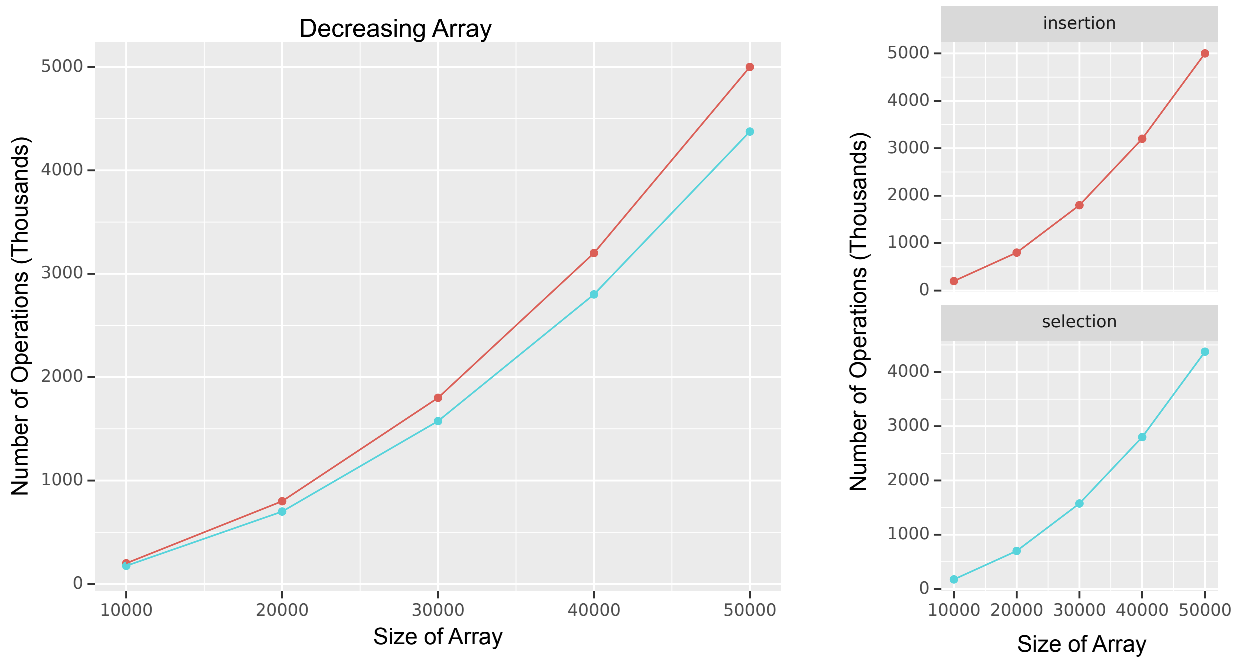

Figure 1: Decreasing Array

Decreasing is the only clearly winning array for selection sort. This is for one key reason: a decreasing array is the worst possible case for insertion sort. In most cases, selection sort is slower than insertion sort. This is because for each element it wants to sort, it must loop through all other elements not already sorted. This will be about the same no matter what, hence similar performance across all the arrays.

Insertion sort is different, however. For each new element, it compares it against the element previously sorted. If it is greater than the element, they swap. It keeps doing this procedure until it finds one less than it, and then stops. As all of the values it is checking have already been sorted, we know that there can't be another greater number further along, and it must be in the correct place.

Most of the time, this is great. In the decreasing case, this provides zero advantages. It will have to check against every single element it has checked already every time, as the correct position will by design always be the furthest away. So, it ends up requiring the same number of checks as selection sort. However, the checking process in insertion sort requires checking if the index has reached zero for each loop, which is one more step than selection sort. This means selection sort beats insertion sort, but only when its very close to it's worse case. As you can see on the plot, it's still quite close.