Increasing Array

Elizabeth Goodwin

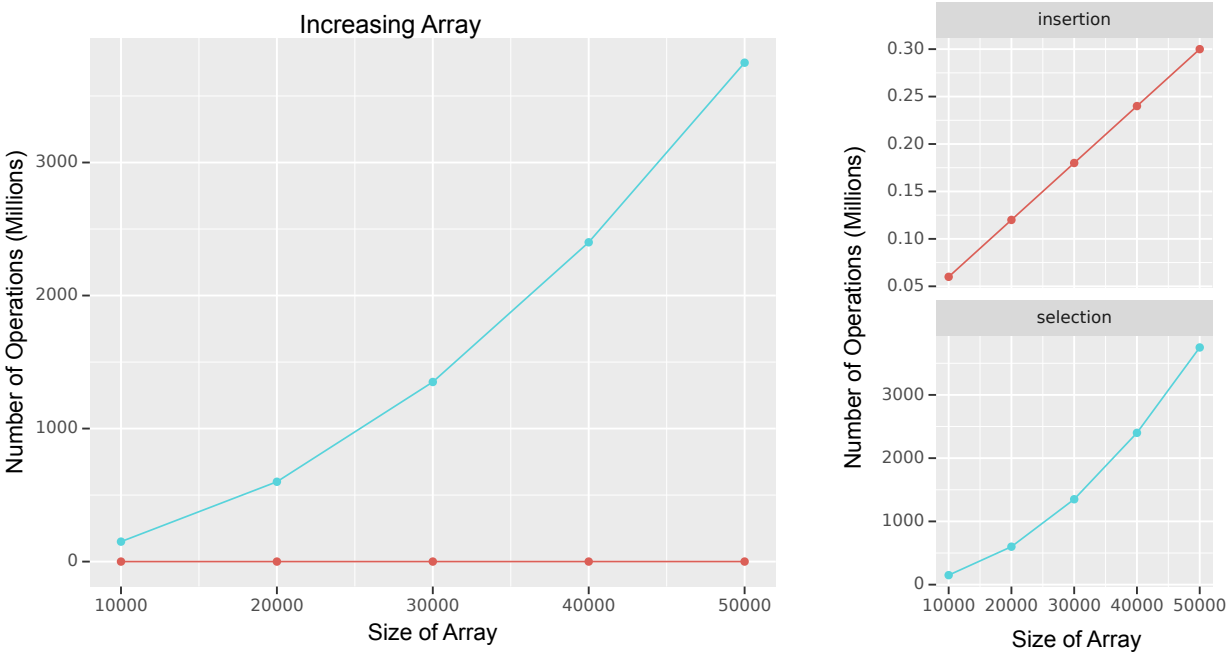

Figure 1: Increasing Array

The difference between the two sorting algorithms is most clearly demonstrated in the Increasing case. Insertion sort dramatically outperforms selection sort to the degree that you can barely see the slope on the insertion sort graph. This is because of one key reason: The runtime of selection sort does not depend on what it finds.

For each value of the sorted array, selection sort searches unsorted (to the right) elements in the array no matter what. It's goal is to find the smallest not already sorted in the array, and it isn't able to know that until it has scanned all of the elements. When presented with an increasing array, it has no way of knowing it is already in order or stopping the search, and has to process each row anyways.

Insertion sort does not have this problem. For each sorted element, it only compares them to elements *it has already seen* (to the left), which it knows is already in order. So when processing a new element, it can just scan left until it finds one smaller than it, and then insert it there. This works because it has already sorted the existing elements, and thus knows that any elements further left must be even smaller than the last element. For an increasing array, this has a profound effect. Every new element it encounters must be greater than those before it. So, it only has to check the element directly to it's left before knowing where to insert it. The closer an array is to already being sorted, the fewer times the while loop needs to run to insert each additional element, and the better it will perform.