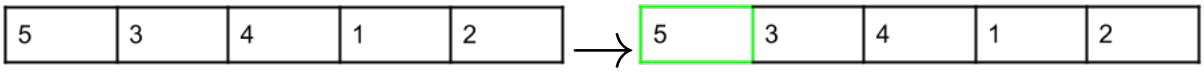
Insertion Sort

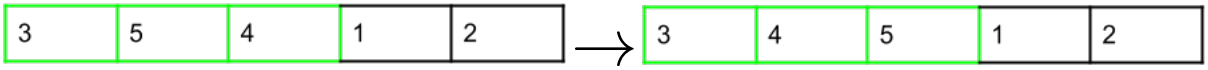
1 Insertion Sort

Insertion Sort is another relatively basic sort. It works as follows, you have 2 subarrays, similar to selection sort. Instead of looking for the minimum each time, you create the ordered array as you move through your unordered one. The process will be illustrated rather extensively in the following example:

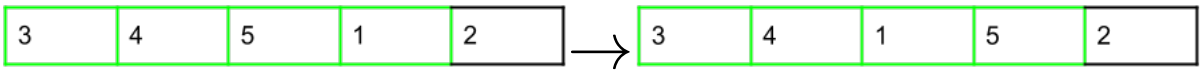


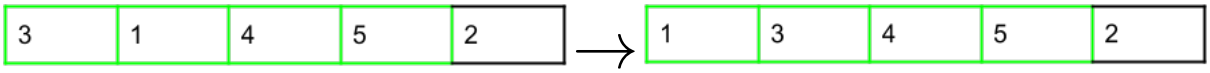
We start off with the unsorted array. We then create 2 subarrays, one sorted and one unsorted. The sorted array would be composed of the first element, 5.

We have now expanded our sorted array to contain 3 and 5. The element 3 is less than 5, so we swap it with 5 in order to get proper ordering.

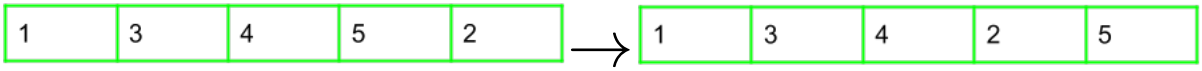


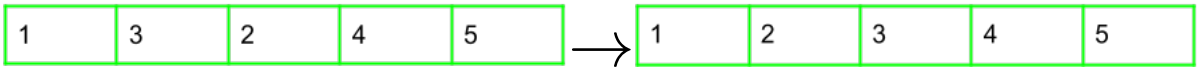
We will now expand the sorted array to include 4. We then swap 4 with 5 since it is less than 5. After this, we compare it to 3. 3 is less than 4, so we do not swap 4 with it.





We add 1 to the sorted array now. We have to swap it with 5 since 1 is less than 5. We swap it with 4 and then 3 because 1 is less than both of them. We now have 1 at the start of our array.





We now will put our final element, 2, into our sorted array. We need to swap it with 5,4, and 3 in order to get it to its rightful spot.

The amount of time that insertion sort takes directly relies on the amount of "inversions" that must be done. to get an idea of what exactly an inversion is, let's take a look at 1. 1 is 3 spots away from its rightful spot, its means that it adds 3 inversions to our total amount in our sort. This means that the runtime of insertion sort is $\Theta(N + K)$ where K is the amount of inversions. In the worst case, there are $\frac{N * (N-1)}{2}$ inversions.
No additional space must be added for insertion sort because everything can be done with pointers, similar

to selection sort. Thus, the total amount of space that insertion sort uses is $\Theta(N)$
Insertion sort is a stable sort because anytime you find 2 elements that are equal, just do not swap them.

That way the one that came earlier originally stays where it was.