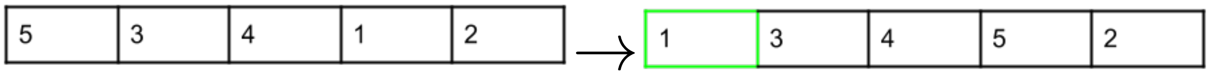
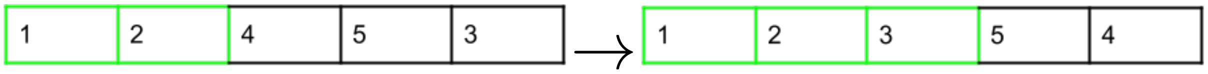
Selection Sort

1 Selection Sort

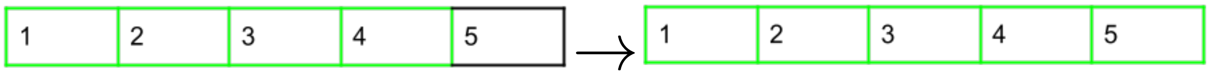
The first sort that we will go over is ***Selection Sort***. Selection sort, n the most basic terms, is finding the minimum element in the list and swaps it to the front to be in its proper places.



We start off with our unsorted array, then we go through all of the unsorted array and we look for the minimum element, which is 1. Once we find 1, we swap it with the 1st element in the array which is 5. We have now created 2 subarrays, one sorted and one unsorted.



We have now go and search for the tiniest element in the subarray which is 2. We then swap it with the 1st element in the subarray. We do a similar process for the element 3. We now have a sorted subarray of size 3 and an unsorted array of size 2.



We finish off by finding the minimum element between 4 and 5. We pick 4, put it in the ordered array which leaves us with just 5. After this, we are left with only 1 element, 5. We just add that element to our ordered array and we are finished.

Let's analyze the runtime of this sort. For any given list, we will have 2 subarrays, one that is sorted and one that is unsorted. Each iteration of the sort, we find the smallest element in the unsorted array and then move it to the end of the sorted array. To do this, the first index of the unsorted array is swapped with smallest element. We always need to go through the full unordered array to find the minimum element, this leads us to do N work for the first iteration $N-1$ the second and so forth. We are left with the sum of $N+N-1+N-2....+3+2+1$ which can be rewritten as $1+2+3+...+N-1+N$ which is just N^2. This means that the runtime in all cases would be $\Theta(N^2$.

The space complexity of selection sort is just $\Theta(N)$ because all we need to do is manipulate pointers (keep track of the end of the sorted array/the start of the unsorted). We can do all our work in the same array so no extra space would be needed.

Selection sort is a stable sort because you look for the minimum each time. If the minimum happens to occur two times in a list simply pick the item that came first to come earlier in your sorted sub array.