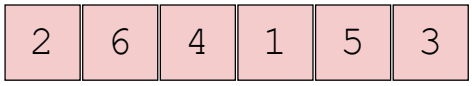
Introduction to Selection Sort

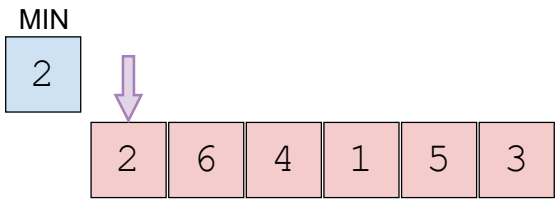
Selection sort works by finding the minimum, and then comparing each element to that minimum to decide its correct position. (Reading time: under 3 minutes)

Selection sort is a simple sorting algorithm, that loops over the array and saves the absolute lowest value. The lowest value is then swapped with the first item in the unsorted array.



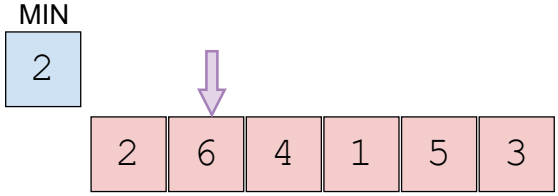
We want to sort the array with values [2, 6, 4, 1, 5, 3] in ascending order.

1 of 32

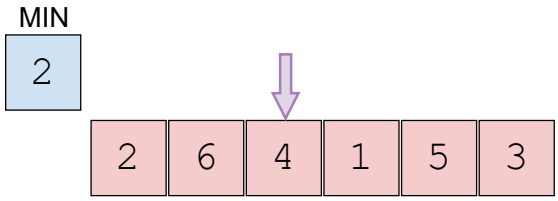


We store the current minimum value, and loop over the array. If we find an item that's lower than the current minimum value, the minimum value will be replaced with that value!

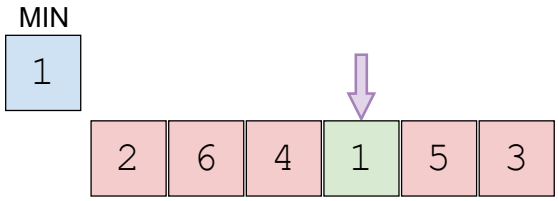
2 of 32



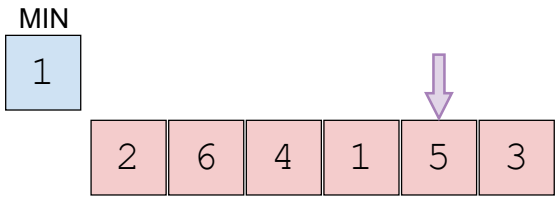
6 > 2, MIN remains the same.



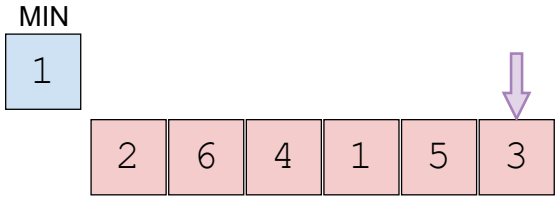
4 > 2, MIN remains the same.



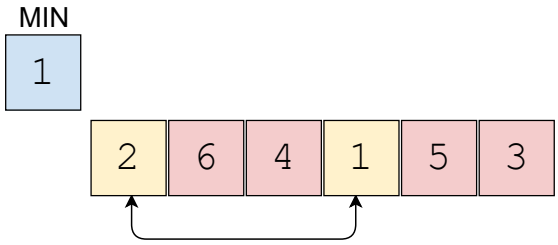
1 < 2, MIN is updated



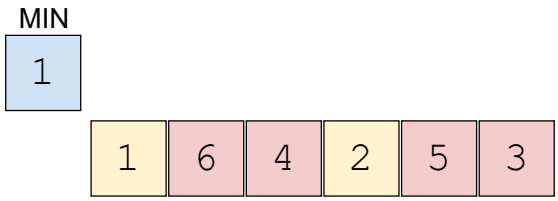
5 > 1, MIN remains the same.



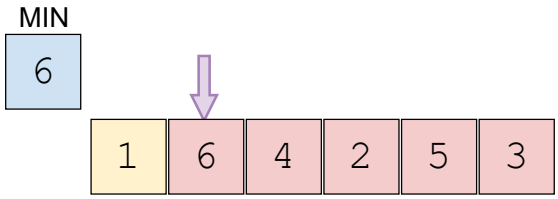
3 > 1, MIN remains the same. We're now at the end of the array.



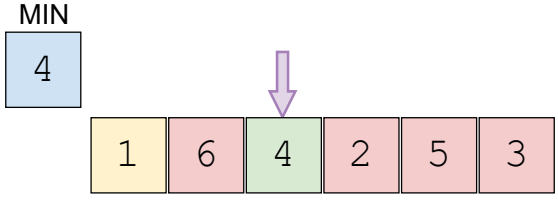
We swap the first item of the unsorted array with the minimum value, and move the sorted array one element.



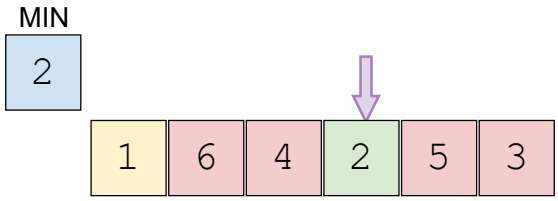
We swap the first item of the unsorted array with the minimum value, and move the sorted array one element.



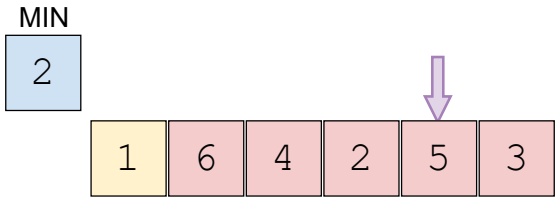
The yellow elements show the elements of the sorted array. Loop starts again. MIN = 6



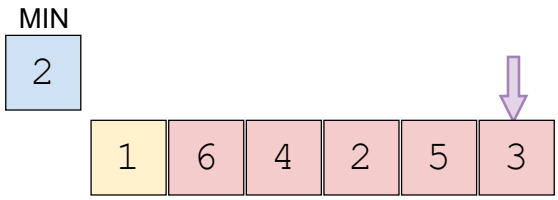
4 < 6, MIN is updated



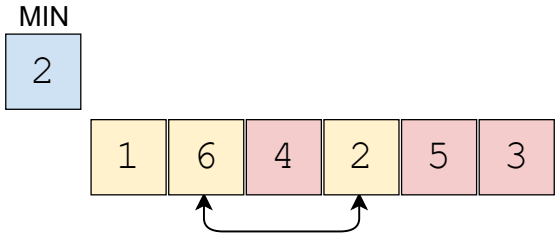
2 < 4, MIN is updated



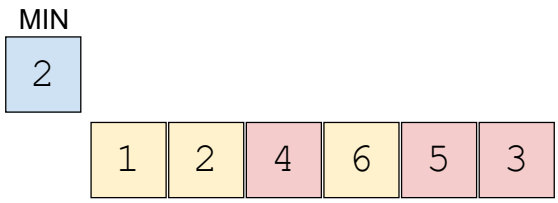
5 > 2, MIN remains the same



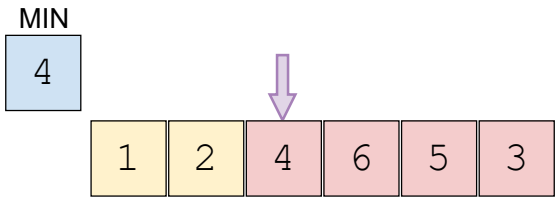
3 > 2, MIN remains the same



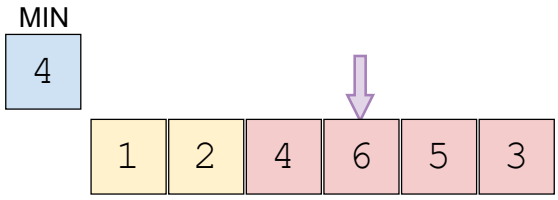
We swap the first item of the unsorted array with the minimum value, and move the sorted array one element.



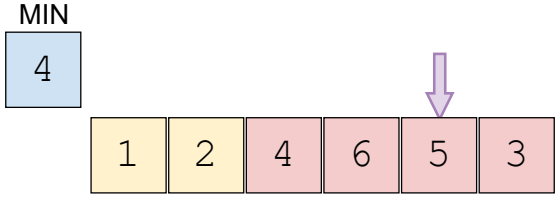
We swap the first item of the unsorted array with the minimum value, and move the sorted array one element.



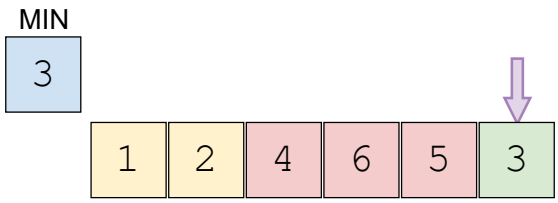
The yellow elements show the elements of the sorted array. We now loop over the unsorted array again. MIN = 4



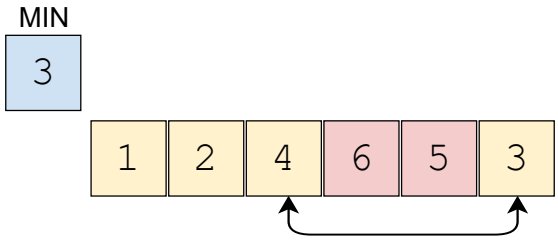
6 > 4, MIN remains the same



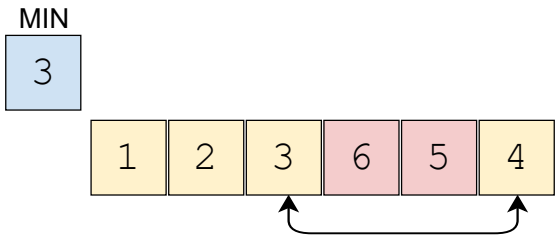
5 > 4, MIN remains the same



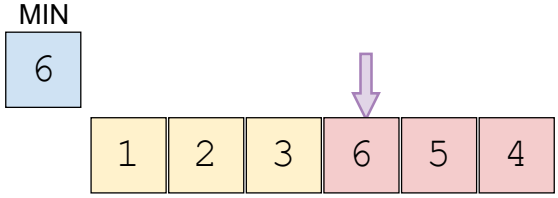
3 < 4, MIN is updated



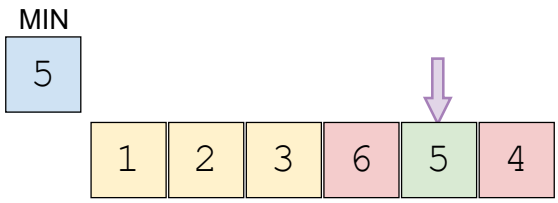
We swap the first item of the unsorted array with the minimum value, and move the sorted array one element.



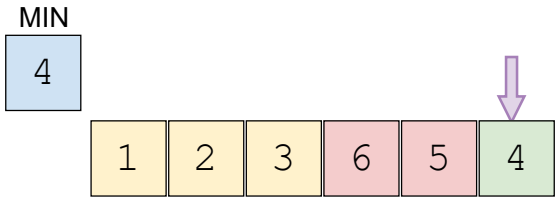
We swap the first item of the unsorted array with the minimum value, and move the sorted array one element.



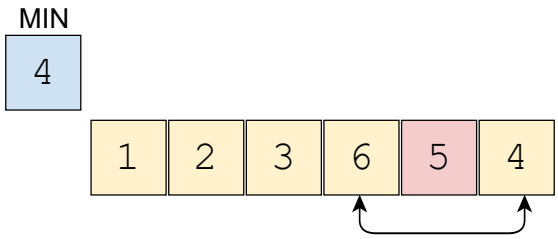
The yellow elements show the elements of the sorted array. We loop over the unsorted array again. MIN = 6



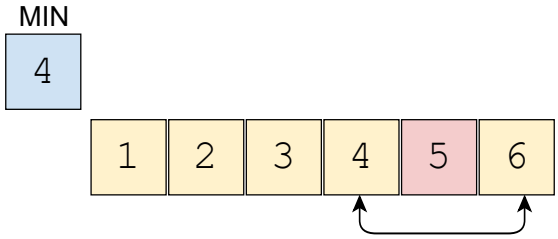
5 < 6, MIN is updated



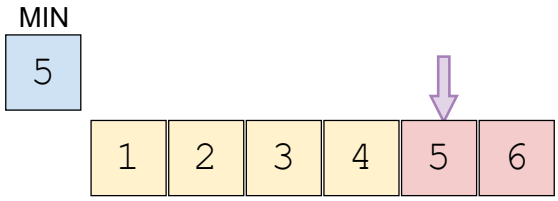
4 < 5, MIN is updated



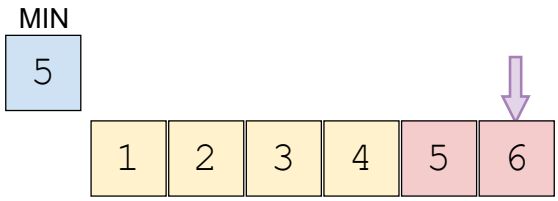
We swap the first item of the unsorted array with the minimum value, and move the sorted array one element.



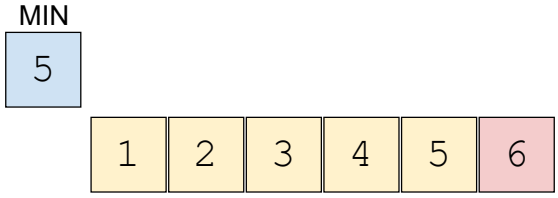
We swap the first item of the unsorted array with the minimum value, and move the sorted array one element.



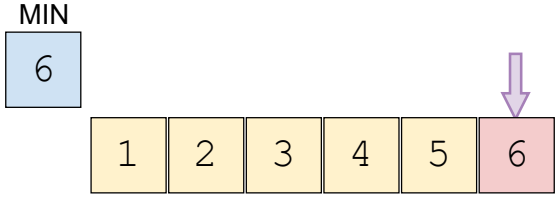
The yellow elements show the elements of the sorted array. We now loop over the unsorted array again. MIN = 5



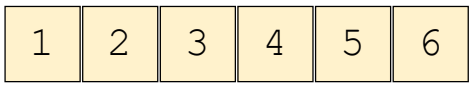
6 > 5, MIN remains the same



5 is at the correct position. The yellow elements show the elements of the sorted array.



We loop over the unsorted array again. MIN = 6



6 is in the correct position. The array has been sorted.

32 of 32

In the next lesson, I will talk about the implementation of this algorithm.