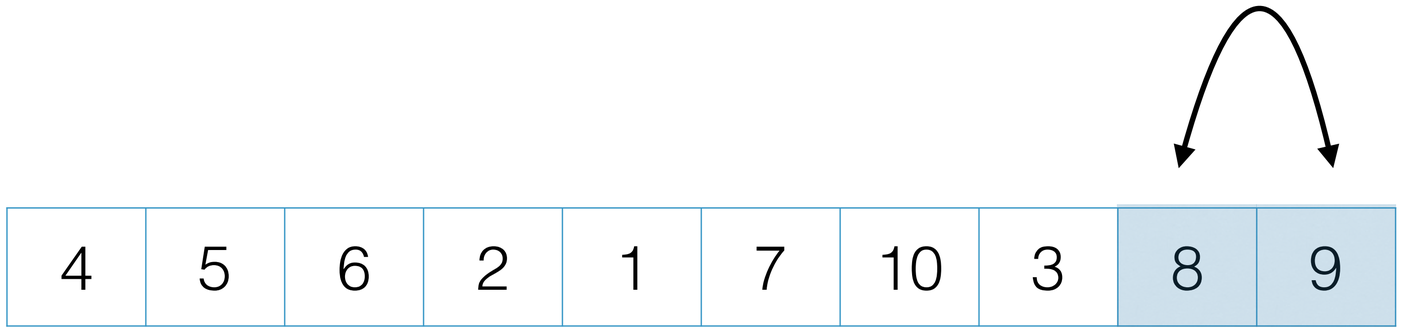
BUBBLE SORT

FOR EACH ITERATION, EVERY ELEMENT IS COMPARED WITH ITS NEIGHBOR AND SWAPPED IF THEY ARE NOT IN ORDER.

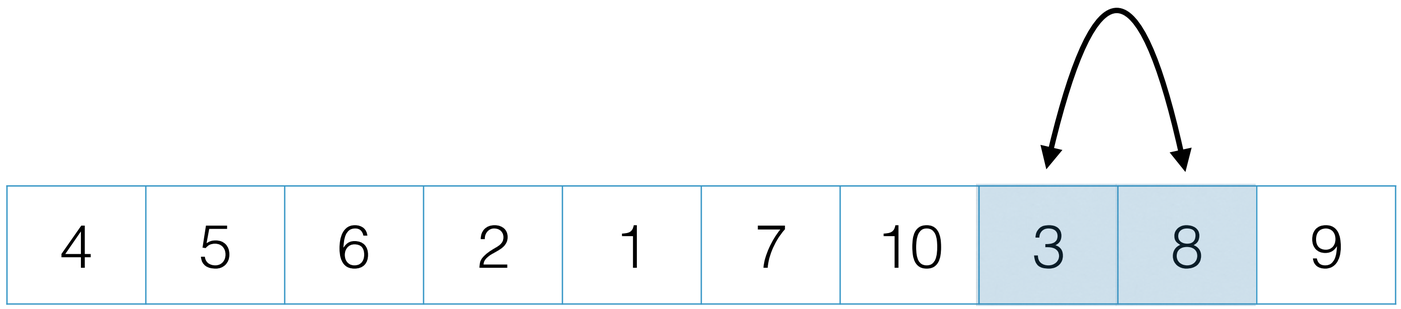
THIS RESULTS IN SMALLER ELEMENTS BUBBLING TO THE BEGINNING OF THE LIST

AT THE END OF THE FIRST ITERATION, THE SMALLEST ELEMENT IS IN THE RIGHT POSITION, AT THE END OF THE SECOND ITERATION THE SECOND SMALLEST IS IN THE RIGHT POSITION AND SO ON

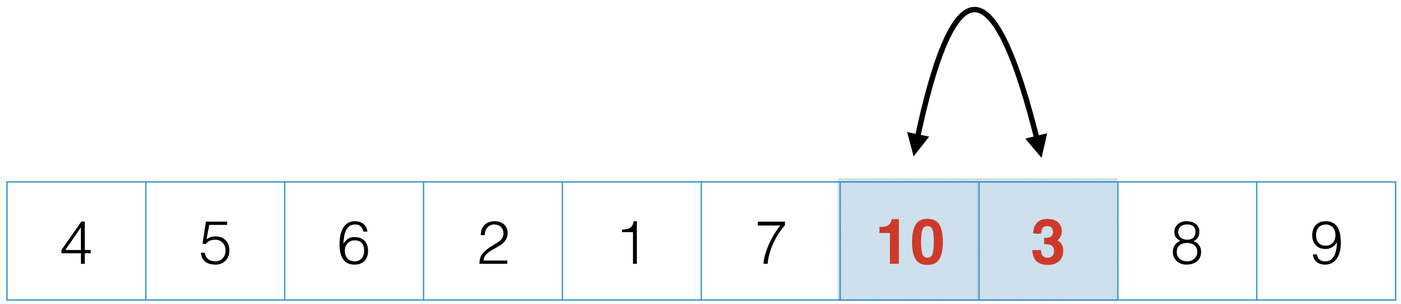
BUBBLE SORT



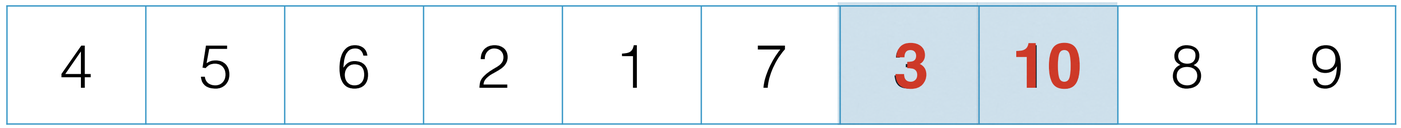
BUBBLE SORT



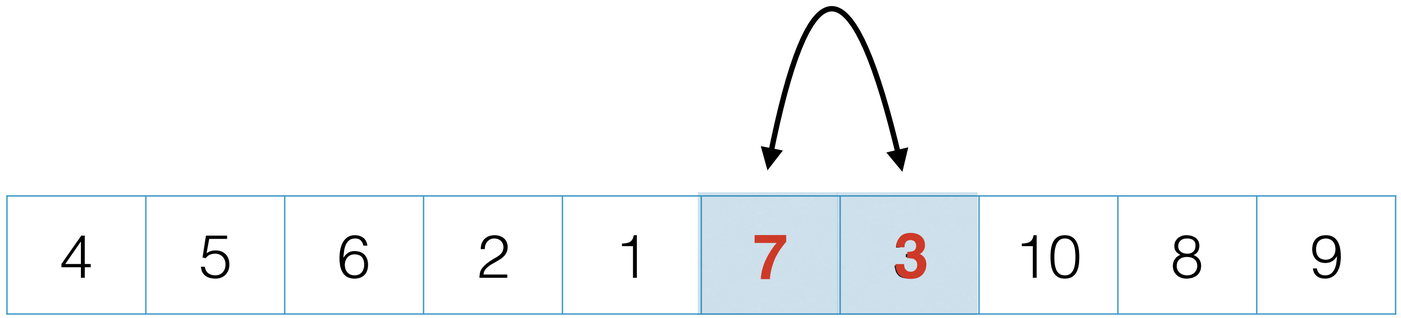
BUBBLE SORT



BUBBLE SORT

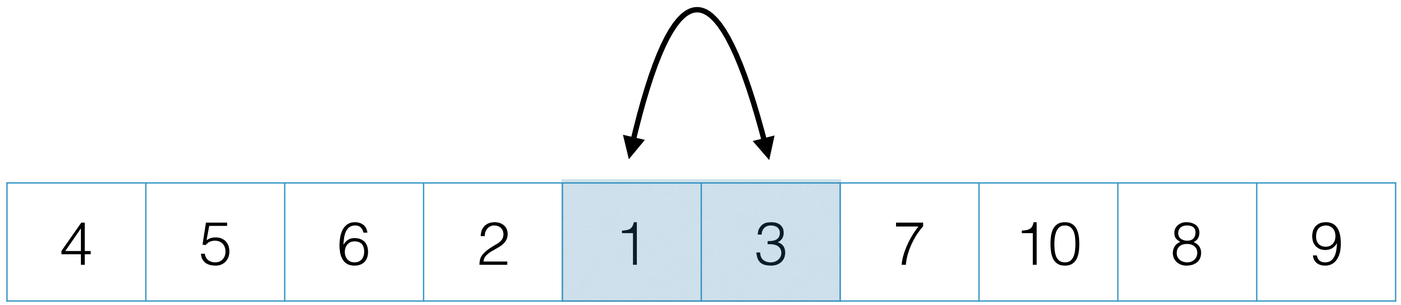


BUBBLE SORT

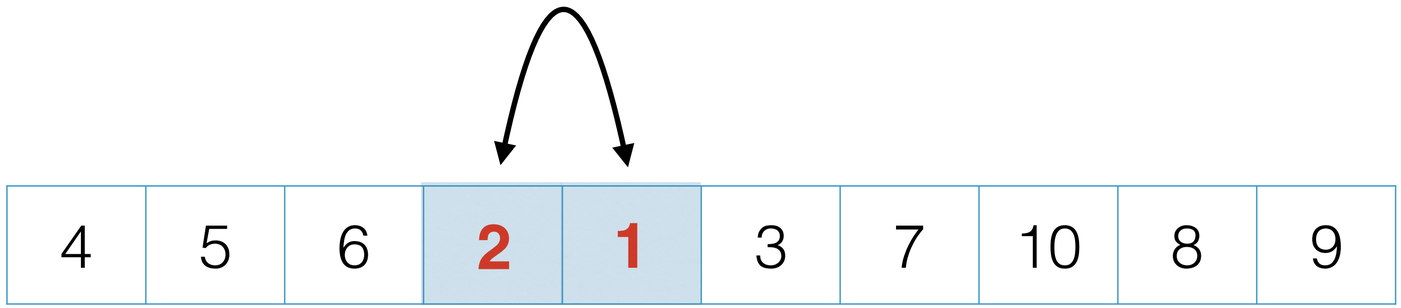


BUBBLE SORT

BUBBLE SORT

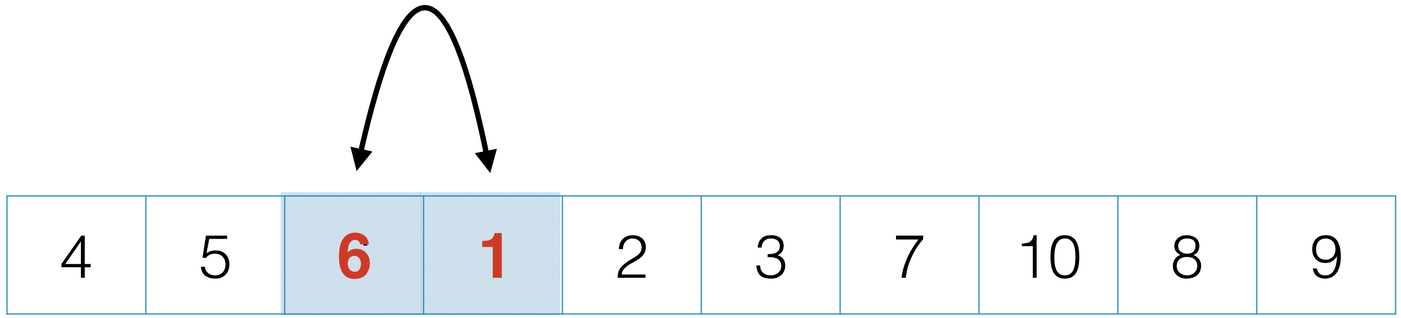


BUBBLE SORT

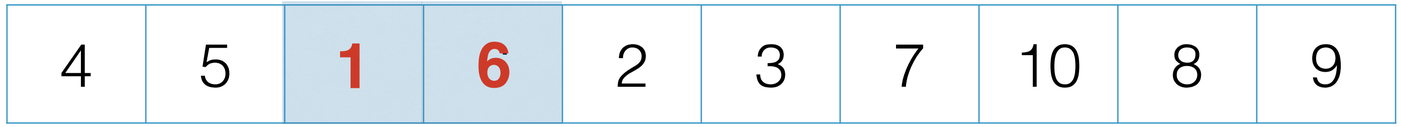


BUBBLE SORT

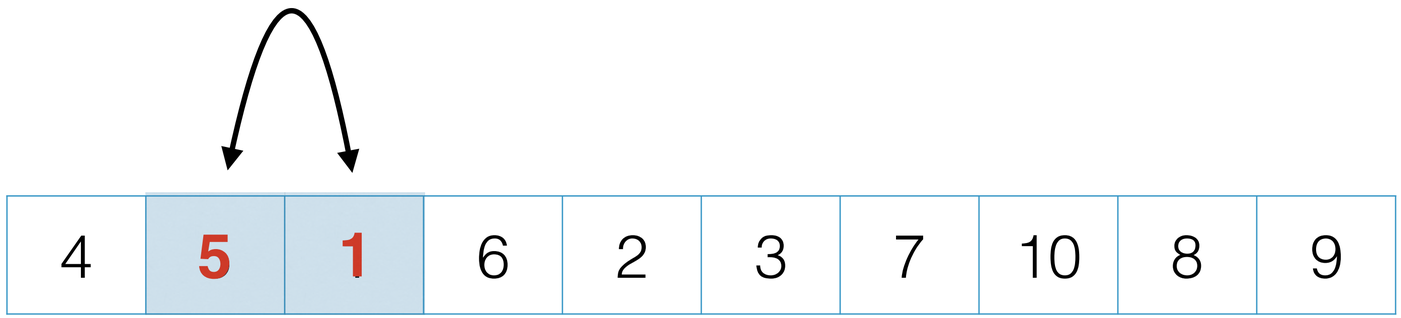
BUBBLE SORT



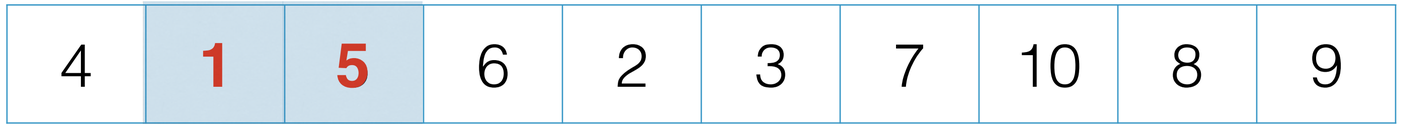
BUBBLE SORT



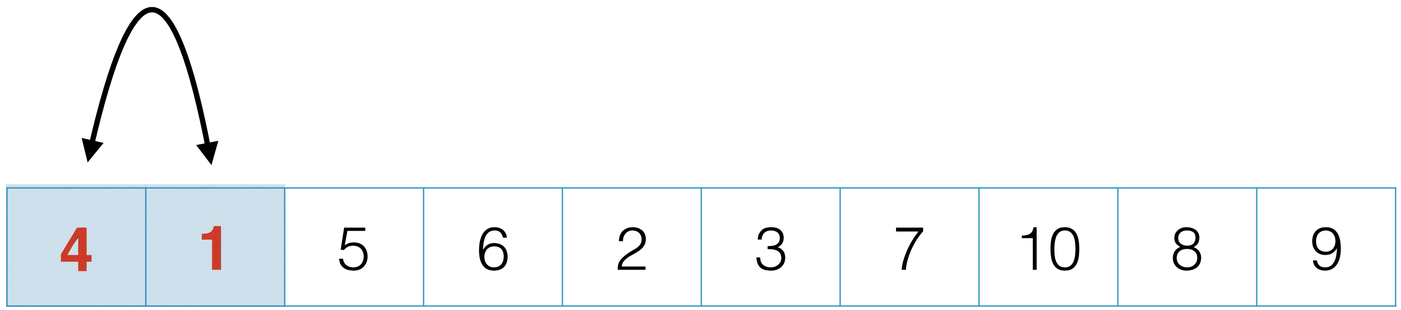
BUBBLE SORT



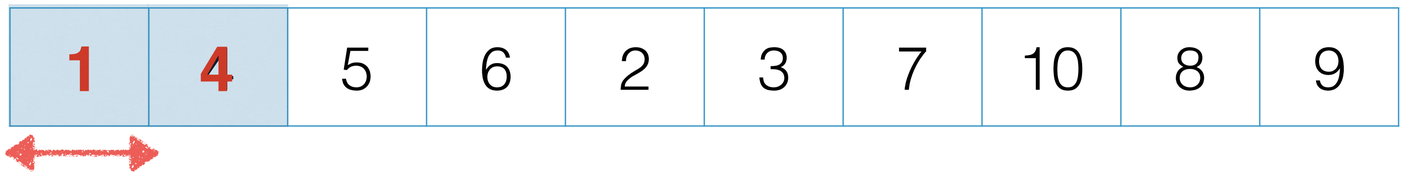
BUBBLE SORT



BUBBLE SORT



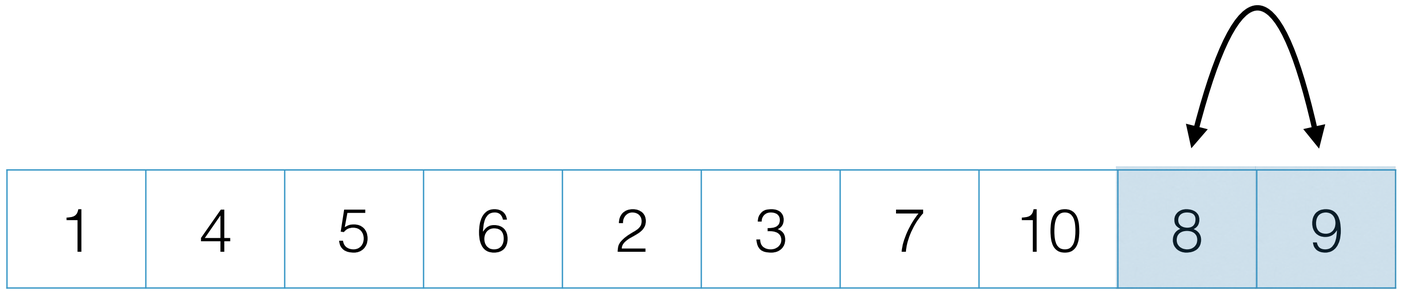
BUBBLE SORT



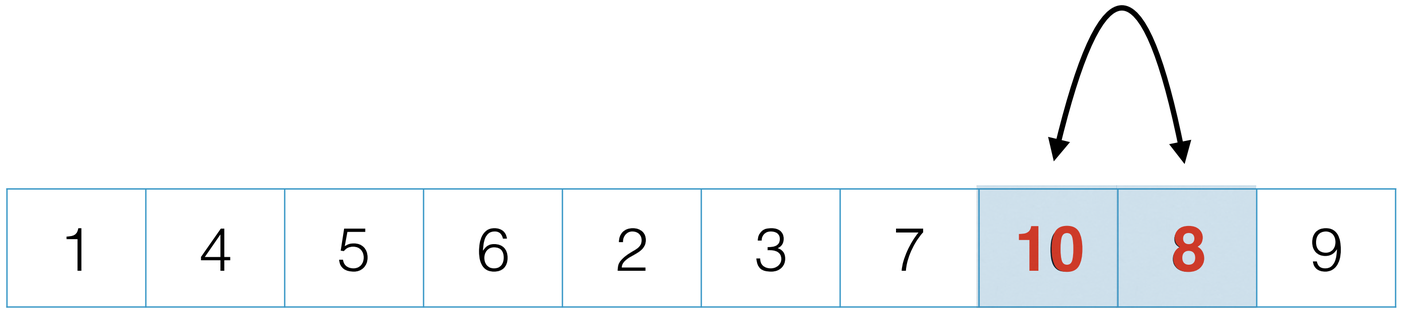
SORTED

1 IS NOW IN THE
CORRECT POSITION

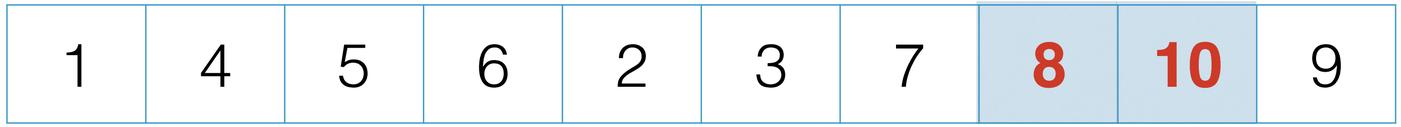
BUBBLE SORT



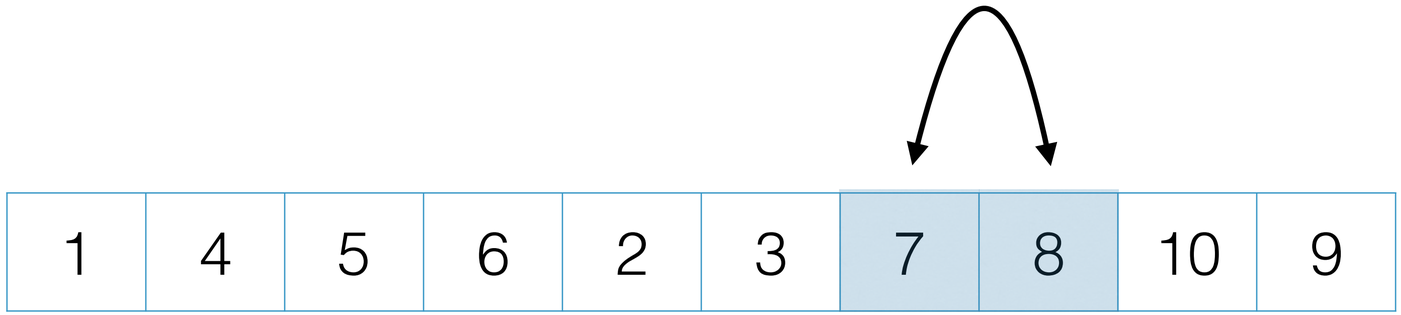
BUBBLE SORT



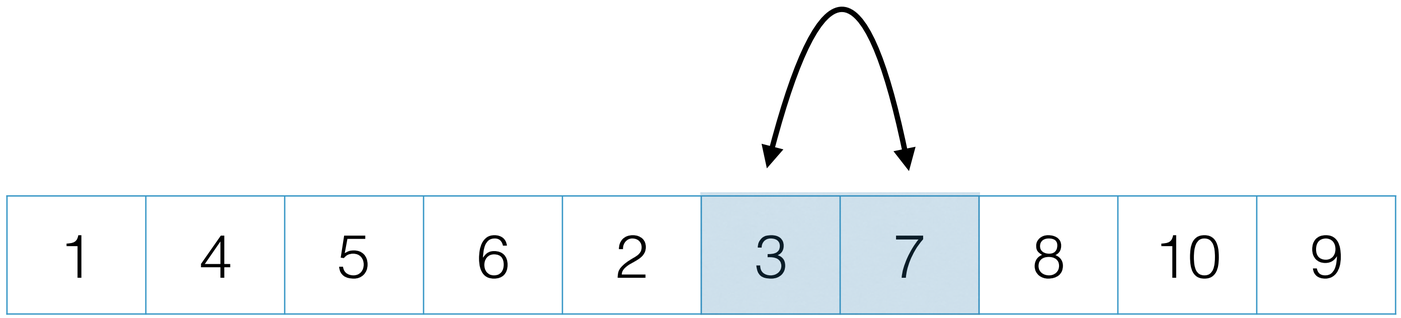
BUBBLE SORT



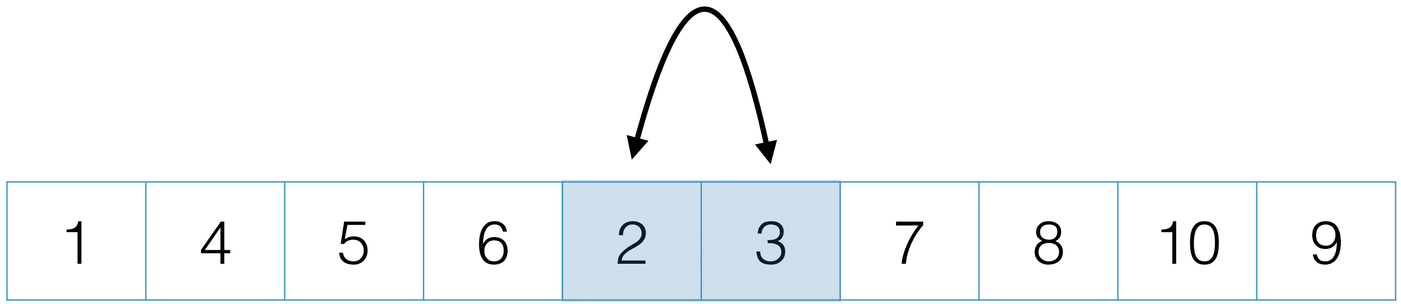
BUBBLE SORT



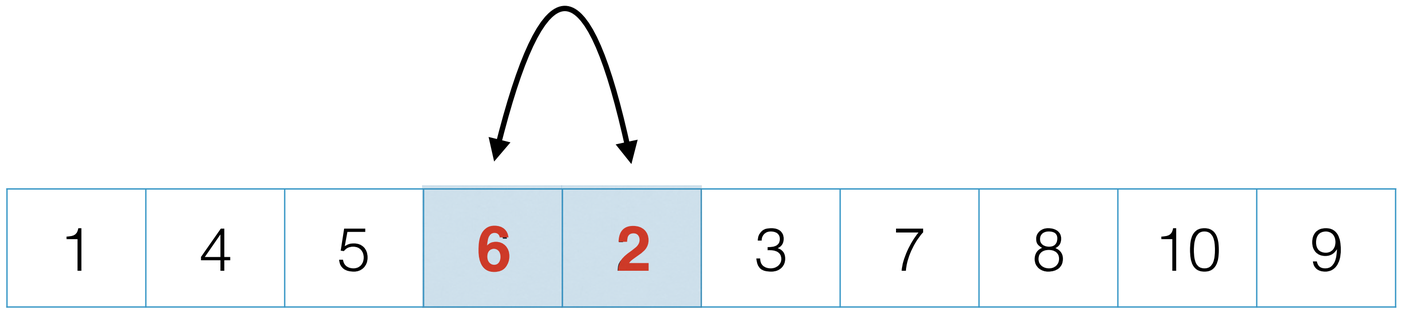
BUBBLE SORT



BUBBLE SORT

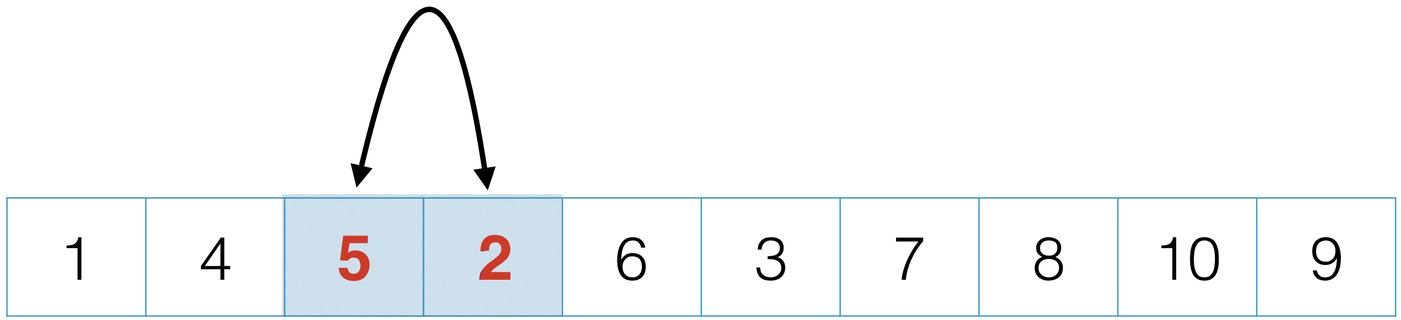


BUBBLE SORT

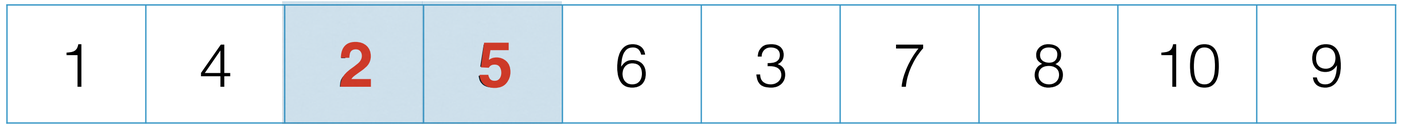


BUBBLE SORT

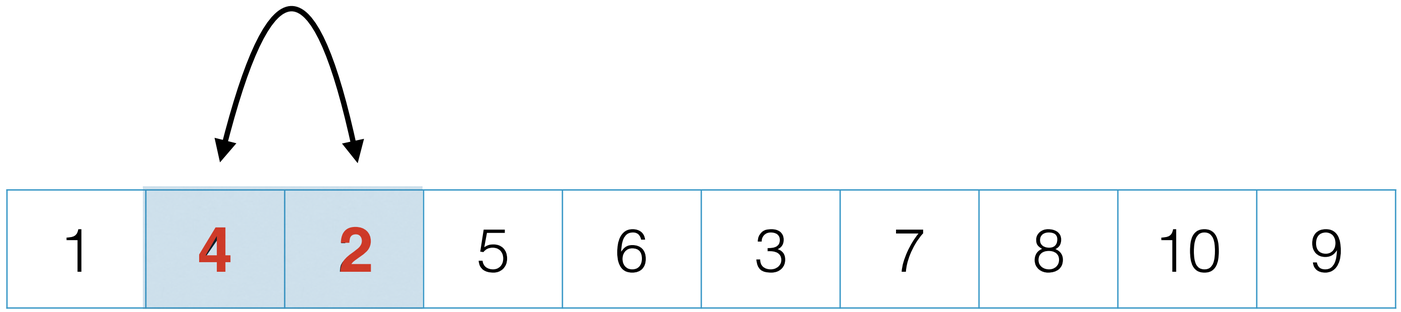
BUBBLE SORT



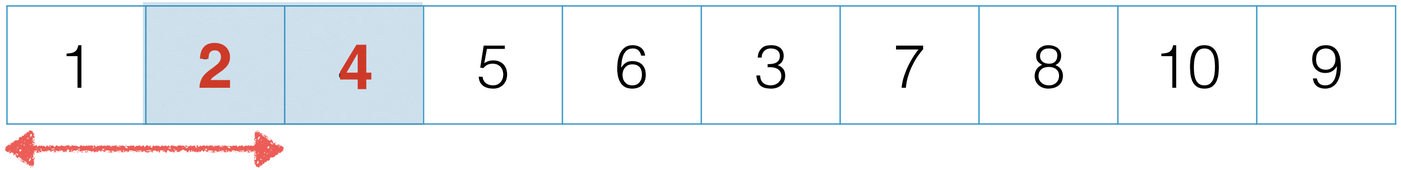
BUBBLE SORT



BUBBLE SORT

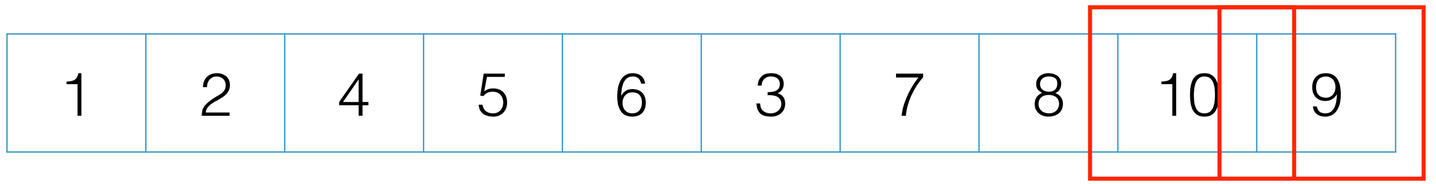


BUBBLE SORT

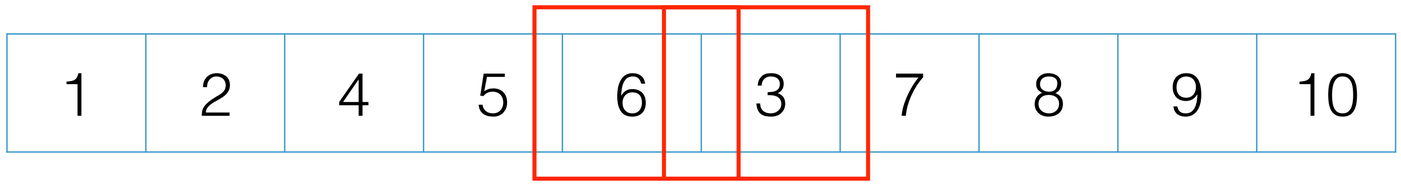


1 AND 2 ARE NOW IN
THE CORRECT POSITION

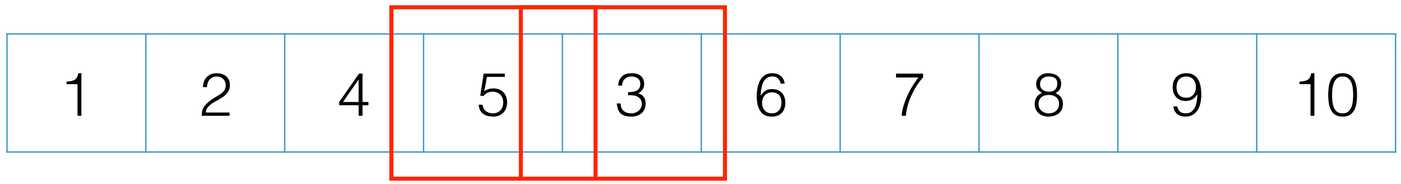
BUBBLE SORT



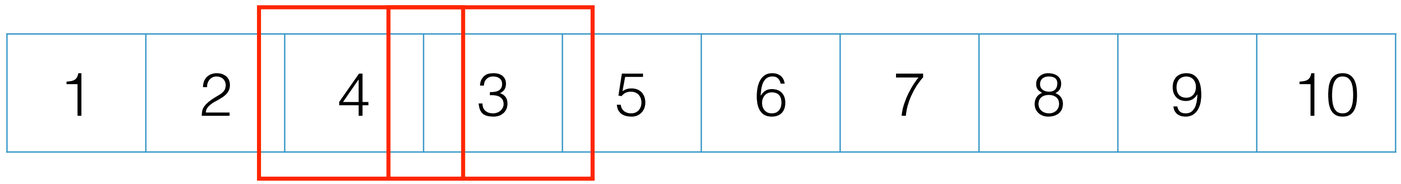
BUBBLE SORT



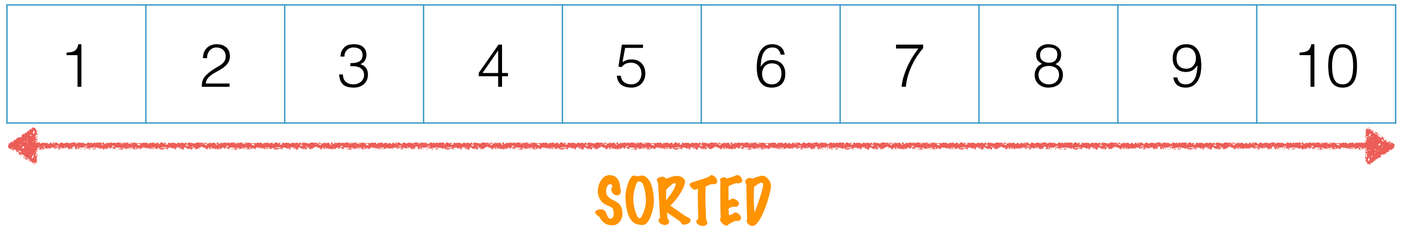
BUBBLE SORT



BUBBLE SORT



BUBBLE SORT



THE INTERESTING THING HERE IS THAT NOT ONLY ARE 1,2, 3 IN THE RIGHT POSITION - THE ENTIRE LIST IS SORTED

IN THE NEXT ITERATION WE PERFORM NO SWAPS. THIS IS AN INDICATION THAT THE LIST IS COMPLETELY SORTED

WE CAN BREAK OUT OF THE LOOP EARLY!

BUBBLE SORT

```
public static void bubbleSort(int[] listToSort) {
    for (int i = 0; i < listToSort.length; i++) {
        boolean swapped = false;
        for (int j = listToSort.length - 1; j > i; j--) {
            if (listToSort[j] < listToSort[j - 1]) {
                swap(listToSort, j, j - 1);
                swapped = true;
            }
        }
        print(listToSort);
        if (!swapped) {
            break;
        }
    }
}
```

THE VARIABLE "SWAPPED" KEEPS TRACK OF WHETHER ANY SWAP WAS PERFORMED IN THE SECOND LOOP, IF NOT, THEN THE LIST IS SORTED AND WE CAN BREAK OUT EARLY

KEEP TRACK OF WHETHER A SINGLE SWAP WAS PERFORMED

NOTE THAT `j` STARTS AT THE LAST ELEMENT IN THE LIST AND COMPARES IT TO THE ELEMENT JUST BEFORE IT

IF NO SWAP WAS PERFORMED THE LIST IS COMPLETELY SORTED, BREAK OUT EARLY

THE SMALLEST ELEMENT BUBBLES TO THE CORRECT POSITION BY COMPARING ADJACENT ELEMENTS

SO IN THE WORST CASE (IF THE LIST IS ORIGINALLY SORTED IN DESCENDING ORDER) "N" ELEMENTS ARE CHECKED AND SWAPPED FOR EVERY SELECTED ELEMENT TO GET TO THE RIGHT POSITION

CHECKING "N" ELEMENTS FOR EACH OF "N" SELECTED ELEMENTS

THE COMPLEXITY OF BUBBLE SORT IS $O(N^2)$

IT IS A STABLE SORT

IT TAKES O(1) EXTRA SPACE, IT SORTS IN PLACE

IT MAKES $O(N^2)$ COMPARISONS AND $O(N^2)$ SWAPS