Insertion Sort (Implementation)

(Reading time: under 3 minutes)

We need to create a function that receives the array that we want to sort as an argument.

```
function insertionSort(array) {
}
```

Just like with bubble sort, we need to loop over every item in the array. We declare a temporary variable to store the value of the current item, and declare a new variable `j`, which is equal to the index of the element that we will compare our current element to. The initial value of this index is one index lower than the currently checked item's index.

```
function insertionSort(array) {
   for (let i = 0; i < array.length; i++) {
     const temp = array[i];
     let j = i - 1;
    }
}
```

We need to check whether `j` is not a negative number, as we don't want to compare elements with negative indexes. If the index of `j` is smaller than **0**, we have reached the end of the array. Next, we need to check whether the value of the element on that index in the array, is higher than the current value. If that's the case, the elements are not in the correct position, and need to swap!

```
while (j >= 0 && array[j] > temp) {
  array[j + 1] = array[j];
  j--;
}
```

Let's say that we were working with the array `[4, 5, 1, 2, 3]`, and **1** was our

active value. The index of the currently active element is then **2**, so `temp` is equal to `array[2]` which is **5**, and `j` is equal to **1**. Then, **1** gets replaced with **5**. The array is now `[4, 5, 5, 2, 3]`; `array[2]` is now equal to **5**, and `j` is equal to **0**. We still need to swap **5** with **1**! As we know that the index of the new position of the currently active value is always one index higher than the value of `j`, we set `array[j + 1]` equal to the value that temp holds, which is the currently active value. Now, the array is `[4, 1, 5, 2, 3]`!

```
array[j + 1] = temp;
```

```
function insertionSort(array) {
   for (let i = 0; i < array.length; i++) {
     let temp = array[i];
     let j = i - 1;
     while (j >= 0 && array[j] > temp) {
       array[j + 1] = array[j];
       j--;
     }
     array[j + 1] = temp;
    }
  return array;
}
```

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