Complete Intro to Computer Science (Brian Holt)

- Link to Course (Frontend Masters)
- Link to Website (Notes)

Sort Algorithms:

Bubble Sort

- **Big O** - Worst (reverse sorted) & Average Cases : `O(n^2)` | Best Case (already sorted) : `O(n)`
- **Spatial complexity** - `O(1)`
- **Stable** & **Destructive**

```
function bubbleSort(nums) {
  // 'swapped' is a 'Sentinel Value':
  let swapped = false;
  do {
    swapped = false;
    for (let i = 0; i < nums.length; i++) {
      if (nums[i] > nums[i + 1]) {
        const temp = nums[i];
        nums[i] = nums[i + 1];
        nums[i + 1] = temp;
        swapped = true;
      }
    }
  } while (swapped);
  return nums;
}
```

Insertion Sort

- **Big O** - Worst (reverse sorted) & Average Cases : `O(n^2)` | Best Case (already sorted) : `O(n)`

- **Spatial complexity** - `O(1)`
- **Stable (only if we make it that way)** & **Destructive**
- Notes: it is still better than Bubble sort because it has smaller coefficients (even though same time complexity). It is also better than Merge sort and Quick sort when we are sure the array is already almost sorted.
- Additional resource 1
- Additional resource 2

```
function insertionSort(nums) {
  for (let i = 1; i < nums.length; i++) {
    let numberToInsert = nums[i]; // the numberToInsert number we're looking to insert
    let j; // the inner counter

    // loop from the right to the left
    for (j = i - 1; nums[j] > numberToInsert && j >= 0; j--) {
      // move numbers to the right until we find where to insert
      nums[j + 1] = nums[j];
    }

    // do the insertion
    nums[j + 1] = numberToInsert;
  }
  return nums;
}
```

- Fun fact : a tail call optimized recursion (i.e. tail-recursive) method can result in an infinite loop and not a stack overflow, so it is a slightly better (less disastrous) scenario!