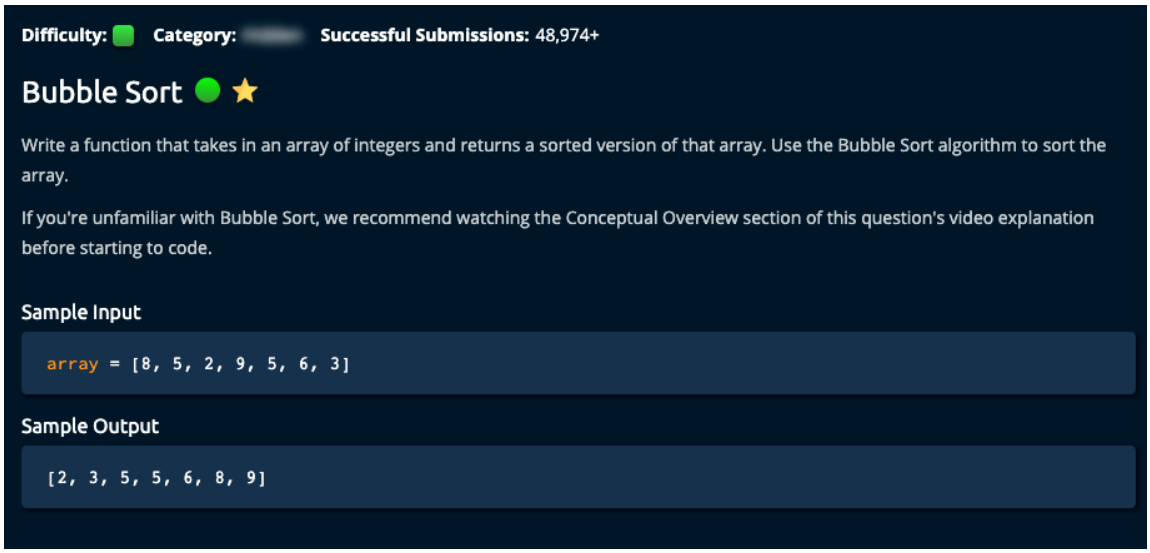
Code Challenge #11 Bubble Sort (Easy)



Difficulty: **Category:** **Successful Submissions:** 48,974+

Bubble Sort

Write a function that takes in an array of integers and returns a sorted version of that array. Use the Bubble Sort algorithm to sort the array.

If you're unfamiliar with Bubble Sort, we recommend watching the Conceptual Overview section of this question's video explanation before starting to code.

Sample Input

```
array = [8, 5, 2, 9, 5, 6, 3]
```

Sample Output

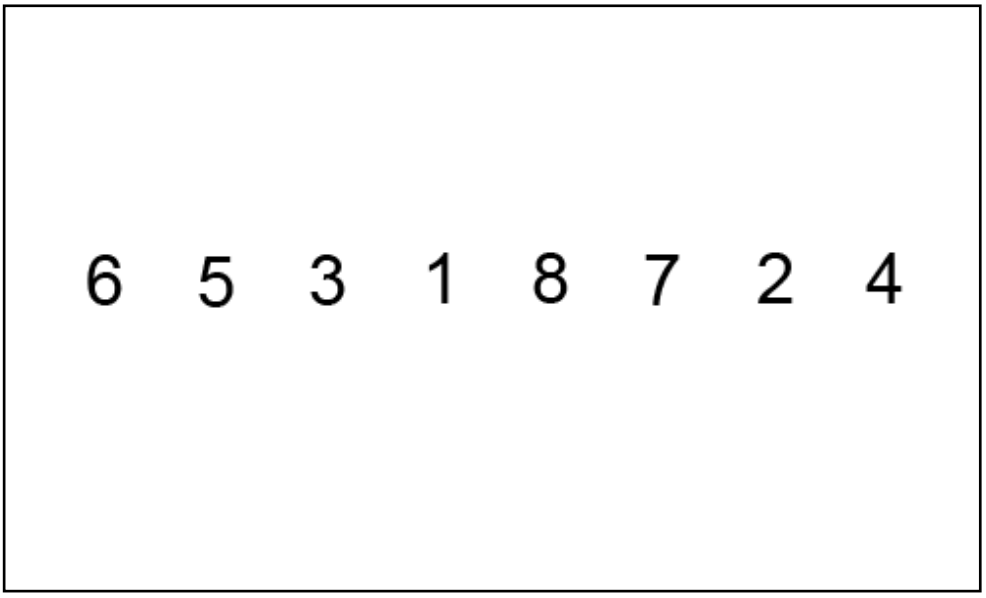
```
[2, 3, 5, 5, 6, 8, 9]
```

Solution #1

```
1. function bubbleSort(array) {
2.   let isSorted = false;
3.   let counter = 0;
4.   while (!isSorted) {
5.     isSorted = true
6.     for (let i = 0; i < array.length - 1 - counter; i++) {
7.       if (array[i] > array[i + 1]) {
8.         swap(i, i + 1, array)
9.       isSorted = false
10.        }
11.      }
12.      counter++;
13.    }
14.    return array;
15. }
16.
17. function swap(i, j, array) {
18.    const temp = array[j];
19.    array[j] = array[i];
20.    array[i] = temp;
21. }
```

Explanation

Bubble sort is a sorting algorithm that swaps adjacent elements depending on their side. It is called bubble sort due to the largest values moving or bubbling towards the end of the list. See animated gif below.



The bubble sort uses a technique with a let variable called isSorted. This isSorted variable is initially set to false. We also have a let counter variable with an initial value of 0. We then use a while loop where we use the bang operator in order to turn the isSorted variable to a Boolean value of true. Within the while loop we assign the isSorted to a true value. We make this assumption inside the while loop that the array is sorted and then check if it is. We check it using an for loop where we declare i is a variable set to 0. As long as i is less than array.length – 1 – counter we increment i. The counter variable is for code efficiency because on each loop we don't need to check the end items due to a bubble sort pushing the largest values towards the right end of the array. We then check if the value on the left array[i] > array[i + 1]. If it is we then we use a helper function called swap to swap the values. We then assign a false value to the isSorted variable in order to continue the loop. The loop will stop if we don't get to the if statement. Outside of the if statement we increment the counter variable by one. The final line of the code for this function is to return the array. The helper function called swap is a simple function that takes in three arguments. The three arguments are i, j and array. Inside the helper function we assign the value of array[j] to a const variable called temp. We then assign the value of array[i] to array[j]. We then assign the value of variable temp to array[i]. The Big O is O(n^2) for worst case scenario and average case. The best case is O(n) if the array is already sorted.