Week 1 Assignment

WEEK 1 ASSIGNMENT

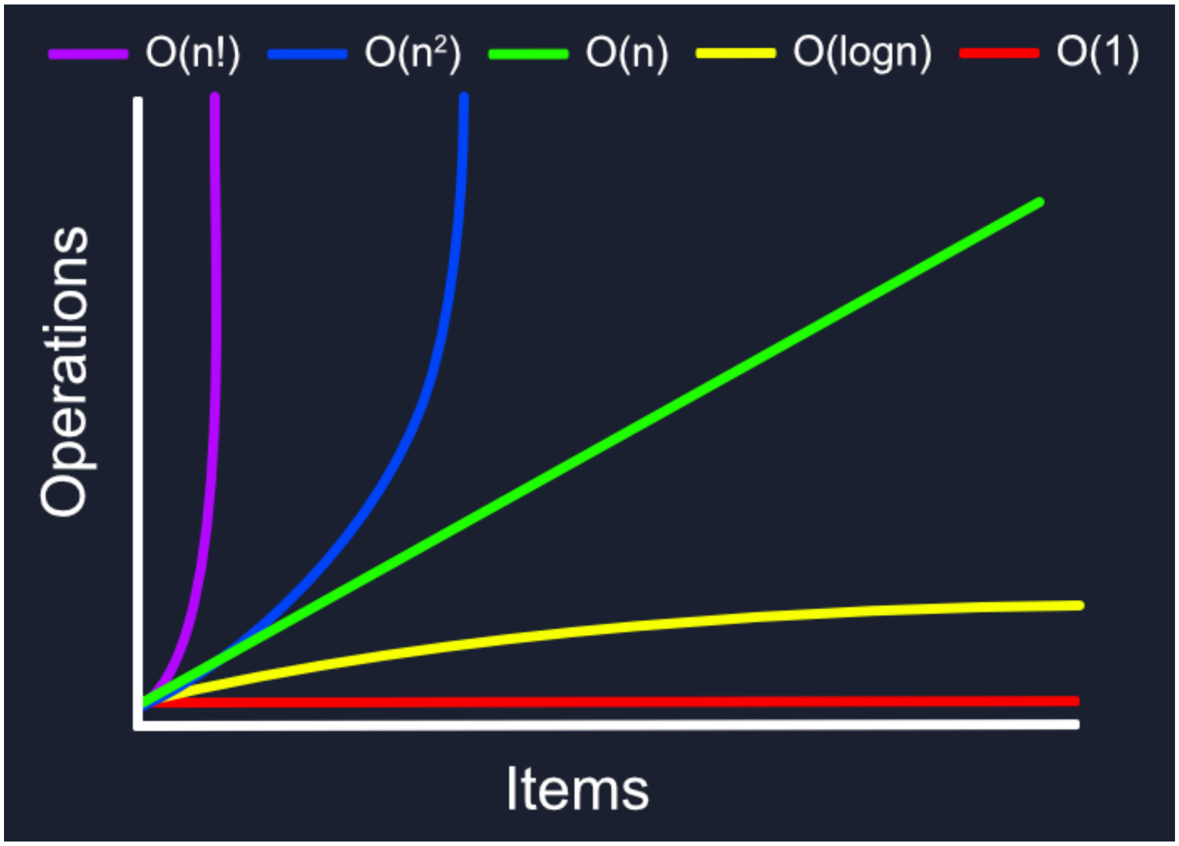
Directions: Please answer the following questions on a VS-Code Editor page. Push up the code to a branch on GitHub called: week-1

Due: Saturday by 9 AM CST

Problem 1: In 3-4 sentences, please explain what Big O Notation is.

Problem 2: For each of the time complexities shown below:

- Name the complexity
- Rank from 1-5 (1 being the best, 5 being the worst)
- Describe in 1-2 sentences of the complexity
- Provide a code example of how the complexity works



Problem 3: Name 3 reasons why we care about Big O and we care about code performance.

Problem 4: What is the problem of using a time method such as performance.now() to measure how "fast" a code runs on our machines.

Problem 5: Given the following piece of code:

- Explain what the TOTAL time complexity is
  - For example, if a function had one linear and a nested for…loop, it would be: n + n^2
- Explain what the CONSOLIDATED time complexity is
  - For example, if a function had one linear and a nested for…loop, it would condense to: n^2

```
const someFunction = (arr1) => {
    arr1.push(1).pop()

    for (let i = 0; i < arr1.length; i++) {
        console.log('do something 2')
    }

    for (let i = 0; i < arr1.length; i++) {
        console.log('do something 3')
    }

    for (let i = 0; i < arr1.length; i++) {
        for (let i = 0; i < arr1.length; i++) {
            console.log('do something 3')
        }
    }
}
```

Problem 6: Given the following piece of code:

- Explain what the TOTAL time complexity is
  - For example, if a function had one linear and a nested for…loop, it would be: n + n^2
- Explain what the CONSOLIDATED time complexity is
  - For example, if a function had one linear and a nested for…loop, it would condense to: n^2

```
const someFunction1 = (arr1) => {
    let sum = arr1[1] + arr[2]

    while (condition) {
        sum = arr[5] + arr[7]
    }

    for (let i = 0; i < arr1.length; i++) {
        for (let i = 0; i < arr1.length; i++) {
            for (let i = 0; i < arr1.length; i++) {
                console.log('do something 3')
            }
        }
    }
}
```

Problem 7: Please explain in 3-5 sentences why we can ignore constants and consolidate our time complexities.

Problem 8: In 2-3 sentences, please explain what space complexity is and why we care.

Problem 9: Given the following data TYPES, label what the **space** complexity is for each one:

- Boolean
- Undefined
- Null
- Numbers
- String
- Array
- Object

Problem 10: Give two reasons when you should use a array and when you should use a object.

Problem 11: Given the following object methods, label what the **TIME** complexity is for each one:

```
const obj = {
    name: 'tony'
}
//inserting
obj.age = 44;

//removing
delete obj.age;

//searching 1
obj.hasOwnProperty['name']

//searching 2
for (const prop in obj) {
    console.log(obj[prop])
}

//accessing
obj.age //44

//retrieving keys
Object.keys(obj)

//retrieving values
Object.values(obj)
```

Problem 12: Given the following array methods, label what the <u>**TIME**</u> complexity is for each one:

```
const arr2 = [1, 2, 3, 4, 5, 6, 7];

//inserting 1
arr2.push(8)

//inserting 2
arr2.unshift(0)

//removing 1
arr2.pop()

//removing 2
arr2.shift()

//searching 1
const findNumber = arr2.find(num => num === 2)

//searching 2
for (let i = 0; i < arr2.length; i++) {
    if (arr2[i] === 2) {
        return arr2[i]
    }
}

//retrieving
const getNumber = arr2[3]

//method 1
const double = arr2.map(num => num * 2)

//method 2
const removeAndAddNewNumber = arr2.splice(1, 1, 5)

//method 3
const getSum = arr2.reduce((total, num) => total + num, 0)

//method 4
for (const num of nums) {
    console.log(num * 2)
}

//method 5
const convertToString = arr2.join(' ')

//method 6
const reversed = arr2.reverse();
```

Problem 13: For each one of these code blocks, please identify the time & space complexity and explanation of why it is.

Problem 1:

```
function findFirstIndexOfNumber(number, array) {
    for (let i = 0; i < array.length; i++) {
      if (array[i] === number) {
        return i;
      }
    }
    return -1
  }
```

Problem 2:

```
function findEachIndexOfNumber(number,array) {
    let arrayOfIndexes = [];
    array.forEach(function(element, index) {
      if (element === number) {
        arrayOfIndexes.push(index);
      }
    });
    return arrayOfIndexes;
  }
```

Problem 3:

```
const array = [36, 14, 1, 7, 21];

function higherOrLower(array) {
  if (array[array.length -1 ] > array[0]) {
    return "Higher";
  else if (array[array.length -1 ] < array[0]) {
    return "Lower";
  } else {
    return "Neither";
  }
}
```

Problem 4:

```
const array = [1,2,3,4,5,6,7,8];

function determineSumOfSequentialArray(array) {
   let sum = 0;
   for (let i = 0; i < array.length; i++) {
       sum += array[i];
   }
   return sum;
}
```

Problem 5:

```
const array = [1,2,3,4,5,6,7,8];

function determineSumOfSequentialArray(array) {
  return array.length * (array.length + 1)/2;
}
```

Problem 6:

```
function searchSortedArray(number, array, beginIndex = 0, endIndex = array.length - 1) {
    let middleIndex = Math.floor((beginIndex + endIndex)/2);
    if (array[middleIndex] === number) {
      return middleIndex;
    } else if (beginIndex >= endIndex) {
      return -1;
    } else if (array[middleIndex] < number) {
      beginIndex = middleIndex + 1;
      return recursiveBinarySearch(number, array, beginIndex, endIndex);
    } else if (array[middleIndex] > number) {
      endIndex = middleIndex - 1;
      return recursiveBinarySearch(number, array, beginIndex, endIndex);
    }
  }
```

Problem 7:

```
const array1 = [3, 7, 9, 12, 15, 18, 32];
const array2 = [3, 3, 7, 41, 76];
function compareArrays(array1, array2) {
    let arrayOfPairs = [];
    array1.forEach(function(e, i) {
      array2.forEach(function(e2, i2) {
        if (e === e2) {
          arrayOfPairs.push([i, i2]);
        }
      });
    });
    return arrayOfPairs;
  }
```

Problem 8:

```
function sortByValue(array){
    function swap(array, index1, index2){
      let temporaryValue = array[index1];
      array[index1] = array[index2];
      array[index2] = temporaryValue;
    }
    let count = 1;
    while (count < array.length) {
      let swapCount = 0;
      for (let i=0; i<array.length-count; i++) {
        if (array[i] > array[i+1]) {
          swap(array, i, i+1);
          swapCount++;
        }
      }
     count++;
   }
     return array;
  }
```

Problem 9:

```
function returnDupes(array, array2) {
    let dupeArray = [];
    array.forEach(function(element) {
      if (array2.includes(element)) {
        dupeArray.push(element);
      }
    });
    return dupeArray;
  }
```

Problem 10:

```
function sumFilteredData(array) {
    return array.filter(function(element) {
      return ((element > 5) && (element < 20))
    }).reduce(function(valueToAdd, currentValue) {
      return valueToAdd + currentValue;
    }, 0);
  }
```