Sorting Algorithms

Anh Huynh

1. Use Big O Notation to describe the time complexity of an algorithm that takes 4N+16steps.- **1 pt**

   $O(N)$ Since according to the rules of Big O notations, the notation ignores constants and numbers that are not an exponent.

2. Use Big O Notation to describe the time complexity of an algorithm that takes 2N2. **1 pt**

   $O(N^2)$ ignore constants and numbers that are not an exponent.

3. Use Big O Notation to describe the time complexity of the following function, which returns the sum of all numbers of an array after the numbers have been doubled: - **2 pt**

```
def double_then_sum(array)
  doubled_array = []

  array.each do |number|
    doubled_array << number *= 2
  end

  sum = 0

  doubled_array.each do |number|
    sum += number
  end
  return sum
end
```

The first .each goes through all the numbers once (N times).
The second .each goes through the numbers again (N times).
They're sequential and not nested so that means it's O(2N) but then we ignore the constant so the function is $O(N)$

4. Use Big O Notation to describe the time complexity of the following function, which accepts an array of strings and prints each string in multiple cases: - **2 pts**

```
def multiple_cases(array)
  array.each do |string|
    puts string.upcase
    puts string.downcase
    puts string.capitalize
  end
end
```

The method loops over the array once so N iterations for all elements. Inside the .each loop, the method executes 3 operations (upcase, downcase, capitalize), which is $3 * N$ but after dropping constants, it becomes $O(N)$

5. The next function iterates over an array of numbers, and for each number whose index is even, it prints the sum of that number plus every number in the array. What is this function's efficiency in terms of Big O Notation? - **4 pts**

```
def every_other(array)
  array.each_with_index do |number, index|
    if index.even?
      array.each do |other_number|
              puts number + other_number
      end
    end
  end
end
```

The outer loop runs N times (once per element) and inside the loop, there is an if loop. This means that if index.even is true, which is half of the elements since half the elements have even indices, it runs another .each loop on the even indices ($O(N)$).
$(N/2) * N = N^2/2$ then we drop the constant so $O(N^2)$