HW 3

Joshua Ortiga

Xin Li

Jonathan Le

December 9, 2022

Problem 5.1. Find a function $f(n)$ that will provide the n^{th} Fibonacci number.

$$f(n) = f(n) - 1 + f(n) - 2$$

Problem 5.2. Take the following functions and order from smallest to largest with respect to their Θ function.
$\log(x)$, $\sqrt[3]{x}$, $\sqrt[500]{x}$, 1, $x^2\log(x)$, $x^{10^{10^{20}}}$, x^x, $\pi(x)$, x^2, 1.1^x, x, $x\log(x)$, 1.9^x, 2.1^x, $x!$

Problem 5.3. Assume that the merge operation requires $\Theta(n)$ work, what is the $\mathcal{O}$ of merge-sort? (Prove it using the method we showed in class)

Problem 5.4. LeetCode problem: Given two input linked lists that represent numbers in reverse order (where the first element of the list is the last digit, etc), write a function that adds the two numbers together and returns a new linked list containing their sum.

```
class Solution:
        def addTwoNumbers(self, l1: ListNode, l2: ListNode) -> ListNode:
                x1 = 0
                x2 = 0
                count = 0

                while l1!=None and l2!=None:
                        x1 += l1.val * 10**count
                        x2 += l2.val * 10**count
                        l1 = l1.next
```

```
        l2 = l2.next
        count += 1

while l1!=None:
        x1 += l1.val * 10**count
        l1 = l1.next
        count += 1

while l2!=None:
        x2 += l2.val * 10**count
        l2 = l2.next
        count += 1

ans = x1 + x2
head = ListNode(ans % 10)
ans //= 10
n1 = head

while ans:
        n1.next = ListNode(ans % 10)
        ans //= 10
        n1 = n1.next

return head
```

Give a function (or two) that represents the number of operations (where operations are defined as how we listed them in class) as a function of the lengths m,n of our two input lists L1,L2 respectively. You might need to use two functions to represent upper/lower bounds. Then provide the most simplified class of Ω, $\mathcal{O}$, and Θ that this code belongs to.

Problem 5.5. *Extra credit*: This problem is a bit challenging so it will be worth 10 points of extra credit. For large values of n the exact number of operations $f(n)$ approaches the following function

$$\lim_{n\to+\infty} f(n) = k \cdot n!$$

where $n!$ is the factorial of n. Find the value of k. Because it's so much extra credit, there will be no partial credit for this.

```
def crappyFactorial(n):
        if n == 0:
                return 1
        elif n == 1:
                return 1
        total = 0
```

```
for (i=0; i < n; i = i + 1):
        total = total + crappyFactorial(n-1)
return total
```

Left as exercise for grader.