HW 5

Joshua Ortiga

Xin Li

Jonathan Le

December 12, 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.

Functions:
$\log(x)$, $\sqrt[3]{x}$, $\sqrt[500]{x}$, 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, $x!$

Sorted:
1, $\sqrt[500]{x}$, $log(\text{x})$, $\sqrt[3]{x}$, $\pi(x)$, x, $x\log(x)$, x^2, $x^2\log(x)$, $x^{10^{10^{20}}}$, 1.1^x, 1.9^x, 2.1^x, $x!$, x^x

1.1^x, 1.9^x, 2.1^x *are relatively equal with each other.*

$\sqrt[500]{x}$, $\sqrt[3]{x}$ *are relatively equal with each other.*

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)

$T(1) = 1$

When splitting the array into a left and right half:
$\Rightarrow T(n) = 2T(\frac{n}{2}) + cn$
$\Rightarrow T(n) = 2T(\frac{n}{2}) + cn$
$\Rightarrow 2T(\frac{n}{4}) + c(\frac{n}{2})$

After performing the merge sort on the left and right half, and then merging

the two sorted halves of the array together:

$\Rightarrow T(n) = 2[2T(\frac{n}{4}) + c(\frac{n}{2})] + cn$
$\Rightarrow 4T(\frac{n}{4}) + 2cn$

General form:

$T(n) = 2^k \cdot T(\frac{n}{2^k}) + kcn$ where $n = 2^k$, $(\frac{n}{2^k}) = 1$, $k = \log_2(n)$

Then substitute k into the general form of T(n) function:

$T(n) = n \cdot T(1) + \log_2(n) \cdot cn$
$T(n) = \mathcal{O}(n \cdot \log(n))$ ■

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.

```
# Definition for singly-linked list.
# class ListNode:
#   def __init__(self, x):
#     self.val = x
#     self.next = None
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 l_1, l_2 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.

Best case scenario:
Let n = len(l_1) = len(l_2).

For all operations outside of any loops, the sum of operations = 14.

Number of operations in the first while loop = 15·n.

Both loops are never executed since all nodes are visited in the first loop when $m = n$. Therefore, number of operations for the second and third loops = 2 (For each loop's comparison operation).

Number of operations in the final while loop = 9·(n-1).

$\therefore$ function for best case is:

$$14+15n+2+9(n-1)$$

Worst case scenario:
Let m = len(l_1), n = len(l_2).

Let x = a number comprised of the values in l_1.
Let y = a number comprised of the values in l_2.

In the worst case, two numbers x and y will be two numbers selected, s.t. the result of the adding $x+y$ yields a result s.t. the number of digits of the result = max(m,n)+1. For example, 1+9=10 or 99+1=100.

For all operations outside of any loops, the sum of operations = 14.

Number of operations in the first while loop = 15·min(m,n).

The while loop iterating over the linked list with less nodes is skipped, so only one operation is executed for the comparison operation.

The while loop for the linked list containing more nodes is not skipped, and it executes 8·abs(m-n) operations.

The number of operations in the final while loop = 9·max(n,m) operations.

∴ the function to find the number of operations in the worst case is:

$$14+(15\cdot min(m,n))+1+(8\cdot abs(m-n))+(9\cdot max(n,m))$$

■

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 an exercise for the grader.