SCHOOL OF ENGINEERING AND TECHNOLOGY

TCSS 343 - Assignment 3

November 4, 2020

1 GUIDELINES

Homework should be electronically submitted to the instructor by midnight on the due date. A submission link is provided on the course Canvas Page. The submitted document should be typeset using any common software and submitted as a PDF. We strongly recommend using LaTeX to prepare your solution. You could use any LaTeX tools such as Overleaf, MiKTeX/TeXWorks, TeXShop etc. Scans of handwritten/hand drawn solutions are not acceptable.

Each problem is worth the indicated number of points. Solutions receiving full points must be correct (no errors or omissions), clear (stated in a precise and concise way), and have a well organized presentation. Show your work as partial points will be awarded to rough solutions or solutions that make partial progress toward a correct solution.

Remember: You must write up the solutions completely on your own. You must also list the names of everyone whom you discussed the problem set with. You may not consult written materials other than the course materials in coming up with your solutions.

2 PROBLEMS

2.1 UNDERSTAND

In this problem use the Master Theorem to find and prove tight bounds for these recurrences (2 points each).

1. $$T(n) = \begin{cases} c & \text{if } n < 125 \\ 25T\left(\frac{n}{125}\right) + \sqrt[3]{n^2} & \text{if } n \geq 125 \end{cases}$$

2. $$T(n) = \begin{cases} c & \text{if } n < 16 \\ 8T\left(\frac{n}{16}\right) + \sqrt[3]{n^2}\log n & \text{if } n \geq 16 \end{cases}$$

3. $$T(n) = \begin{cases} c & \text{if } n < 49 \\ 343T\left(\frac{n}{49}\right) + n^2 \log\log n & \text{if } n \geq 49 \end{cases}$$

4. $$T(n) = \begin{cases} c & \text{if } n < 7 \\ 49T\left(\frac{n}{7}\right) + n^2 \log\log n & \text{if } n \geq 7 \end{cases}$$

5. $$T(n) = \begin{cases} c & \text{if } n < 7 \\ 49T\left(\frac{n}{7}\right) + n^2 \log n & \text{if } n \geq 7 \end{cases}$$

Grading You will be docked points for errors in your math, disorganization, lack of clarity, or incomplete proofs.

2.2 EXPLORE

For the following problems stated as pseudo-code, let $A[\ell \ldots r]$ denote the sublist of the integer list A from the ℓ-th to the r-th element inclusive, let `Foo(`$A[1 \ldots n]$`)` denote an algorithm that runs in time $\Theta(n^2)$, and let `Bar(`$A[1 \ldots n]$`)` denote an algorithm that runs in time $\Theta(n \log^2 n)$.

```
Algo(A[1...n])
    If n ≤ 5 Then Return // nothing to do
    Foo(A[1...n])
    Algo(A[1...⌊3n/5⌋])
    Bar(A[1...n])
    Algo(A[⌊n/5⌋...⌊4n/5⌋])
    Foo(A[1...n])
```

```
    Algo(A[⌊2n/5⌋...n])
    Bar(A[1...n])
End Algo.
```

(2 points) 1. State a recurrence that gives the complexity $T(n)$ for algorithm `Algo`.

(3 points) 2. Find the tight complexity of algorithm `Algo`.

```
Flex(A[1...n])
    If n ≤ 1 Then Return // nothing to do
    Foo(A[1...n])
    Flex(A[1...⌊n/√2⌋])
    For i = 1 to n do
        Bar(A[1...⌊2n/3⌋])
    End For
    Flex(A[1...⌊n/√2⌋])
End Flex.
```

(2 points) 3. State a recurrence that gives the complexity $T(n)$ for algorithm `Flex`.

(3 points) 4. Find the tight complexity of algorithm `Flex`.

2.3 EXPAND

Define the median of a collection of N distinct comparable elements to be an element v from that collection that is larger than exactly $\lfloor N/2 \rfloor$ (or, equivalently, smaller than exactly $\lceil N/2 \rceil - 1$) other elements from the same collection.

Consider the problem of finding the median of the elements in two sorted lists of sizes m and n (not necessarily equal), respectively.

(1 point) 1. Express this problem formally (unique name, input conditions, output conditions).

(1+1 points) 2. (a) Describe a simple algorithm to compute that median. (b) Find its tight (Big-Theta) asymptotically complexity as a function of the total number of elements.

(2+1 points) 3. (a) Show how you can use the medians of the two lists to reduce an instance of this problem to smaller sub-instances. (b) State a precise self-reduction for this problem.

(1+2+1 points) 4. (a) State a recursive algorithm that solves the problem based on your reduction. (b) For the special case when $m = n$, obtain from it a recurrence expressing its running time as a function of n. (c) Solve the recurrence and find a tight (Big-Theta) asymptotic bound on the complexity in the worst case.

2.4 CHALLENGE

Mike Rowe is the President, CEO, and only employee of MikeRoweSoft Corporation. He is working on a project to compute Fibonacci numbers, defined by the properties $F_0 = 0$, $F_1 = 1$, and $F_n = F_{n-1} + F_{n-2}$ for $n > 1$.

Mike conjectures that Fibonacci numbers satisfy the intriguing property $F_n = F_m F_{n-m-1} + F_{m+1} F_{n-m}$ for all $n > m \geq 0$. From this conjectured property, he derives the following self-reduction to compute a pair of successive Fibonacci numbers (F_{2n-1}, F_{2n}) given the prior pair of successive Fibonacci numbers (F_{n-1}, F_n), for $n > 1$:

$$\begin{aligned} F_{2n-1} &= F_{n-1}^2 + F_n^2 \\ F_{2n} &= (2F_{n-1} + F_n)F_n \end{aligned}$$

Now he believes this self-reduction provides a better way than the usual $\Theta(n)$ algorithm to compute F_n. However, he has not taken TCSS 343, and hence he has so far been unable to prove his conjecture correct and his algorithm to be better than $\Theta(n)$.

(2 points) 1. Prove that Mike's conjectured property is correct. (Hint: try induction)

(1 point) 1. Write an algorithm to compute (F_{n-1}, F_n) for any $n > 1$ based on the above self-reduction.

(2 points) 1. Find the tight (Big-Theta) asymptotic complexity of this algorithm and show that it indeed beats the usual linear-time method to compute Fibonacci numbers.

Grading Correctness and precision are of utmost importance. Use formal proof structure for the Big-Theta bounds. You will be docked points for errors in your math, disorganization, lack of clarity, or incomplete proofs.