SCHOOL OF ENGINEERING AND TECHNOLOGY

TCSS 343 - Assignment 2

October 20, 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 can 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

For this item consider the problem of sorting the odd elements in a list of integers while keeping the even ones unchanged in their original places.

Sort the odd elements in a list (ODDSORT)
Input: $A[1\ldots n]$ a list of integers.
Output: List $A'[1\ldots n]$ such that $A'[i] = A[i]$ if $A'[i]$ is even, and $A'[i] \le A'[j]$ for all $1 \le i \le j \le n$ such that $A'[i]$ and $A'[j]$ are both odd.

Let $OS(A[a\ldots b])$ represent the output of the ODDSORT problem on input $A[a\ldots b]$. Let $\alpha \parallel \beta$ denote the concatenation of lists α and β, and let $minx(A[a\ldots b])$ denote the index (in range $[a\ldots b]$) of the minimum odd element from list $A[a\ldots b]$.

(1 point) 1. Below is a self-reduction for the ODDSORT problem. State a recursive algorithm using pseudocode for solving the ODDSORT problem based on this self-reduction.

$$OS(A[a\ldots b]) = \begin{cases} A & \text{if } a \ge b \\ A[a] \parallel OS(A[a+1\ldots b]) & \text{if } a < b \text{ and } A[a] \text{ is even or } a = k \\ A[k] \parallel OS(A[a+1\ldots k-1] \parallel A[a] \parallel A[k+1\ldots b]) & \text{if } a < b \text{ and } A[a] \text{ is odd and } a < k \\ & \text{with } k = minx(A[a\ldots b]) \end{cases}$$

(2+2 points) 2. Using the same reduction as in item above, now state a recurrence $T(n)$ that expresses the worst case running time of the recursive algorithm assuming that minx() runs in linear time. Solve that recurrence and state the tight (Big Theta) bound on $T(n)$. Hint: look for a similar recurrence in the course notes.

(1 point) 3. Let $t_1 = a + \lfloor (b-a)/3 \rfloor$ and $t_2 = b - \lceil (b-a)/3 \rceil$ denote the indices at one third and two thirds the way from a to b, respectively. Let $pairsort(X, Y)$ be a function that returns $Y \parallel X$ if X and Y are both odd and $X > Y$, otherwise returns $X \parallel Y$. Below is another self-reduction for the ODDSORT problem. State a recursive algorithm using pseudocode for solving the ODDSORT problem based on this self-reduction.

$$SOS(A[a\ldots b]) = \begin{cases} A & \text{if } a \ge b \\ pairsort(A[a], A[b]) & \text{if } b = a+1 \\ A''' & \text{if } b > a+1 \\ \text{with } A' \leftarrow SOS(A[a\ldots t_2]) \parallel A[t_2+1\ldots b] & \\ A'' \leftarrow A'[a\ldots t_1] \parallel SOS(A'[t_1+1\ldots b]) & \\ A''' \leftarrow SOS(A''[a\ldots t_2]) \parallel A''[t_2+1\ldots b] & \end{cases}$$

(2+2 points) 4. Using the same reduction as in item 3 above, state a recurrence $T'(n)$ that expresses the worst case running time of the recursive algorithm. From this recurrence, find a tight (Big-Theta) expression for $T'(n)$. Hint: the Master Theorem may be the easiest solution here.

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

2.2 EXPLORE

For this item consider the problem of finding the concatenation of even-length strings from a list, where the notation $s \parallel t$ means the concatenation of strings s and t in that order and the simple function $\text{len}(s) \geq 0$ returns the length (number of characters) on string s.

`Concatenation of All Even-Lengthed Strings in a List (CONCATEVEN)`
`Input:` $S[a \ldots b]$ `list of strings.`
`Output:` $c = S[i_1] \parallel \ldots \parallel S[i_m]$ `where` $i_1 < \cdots < i_m$ `and` $\text{len}(S[i_k])$ `is even for all` $1 \leq k \leq m$, `or else the empty string (denoted` ε`) if list` S `contains no string of even length.`

Let $C2(S[a \ldots b])$ represent the output of the `CONCATEVEN` problem on input $S[a \ldots b]$.

(4 points) 1. State two different divide-and-conquer self-reductions for the problem above.

(4 points) 2. Give recursive algorithms (in pseudo-code) based on your divide-and-conquer self-reductions to solve the problem above.

(2 points) 3. What are the worst-case runtimes of the solutions you have obtained? (Just state the runtimes. You do not need to show your work.)

2.3 EXPAND

Consider the following recurrence $T(n)$:

$$T(n) = \begin{cases} 1 & \text{if } n < 9 \\ 3T(\lfloor \frac{n}{9} \rfloor) + \sqrt{n} & \text{if } n \geq 9 \end{cases}$$

(4 points) 1. Use the repeated substitution method to come up with a good guess for a bound $g(n)$ on the recurrence $T(n)$.

(3 points) 2. State and prove by induction a theorem showing $T(n) \in O(g(n))$ for the same function $g(n)$ guessed above. Hint: use sentinels to simplify the analysis.

(3 points) 3. State and prove by induction a theorem showing $T(n) \in \Omega(g(n))$ for the same function $g(n)$ above, thereby proving that $T(n) \in \Theta(g(n))$. Hint: use sentinels to simplify the analysis.

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

2.4 CHALLENGE

Consider the following observation on the randomized Quicksort algorithm. Because the pivot is chosen uniformly at random among the n input element, with probability $\frac{1}{2}$ the chosen pivot's position on the sorted list will be between $\frac{n}{4}$ and $\frac{3n}{4}$ (i.e. a good pivot), and with probability $\frac{1}{2}$ the chosen pivot's position on the sorted list will be between 1 and $\frac{n}{4}$ or between $\frac{3n}{4}$ and n (i.e. a bad pivot).

(1 point) 1. State a recurrence that expresses the worst case for bad pivots.

(1 point) 2. State another recurrence that expresses the worst case for good pivots.

(3 points) 3. Prove by induction that a suitable combination of the two recurrences above expressing the expected worst case for any pivot is $\Theta(n \log n)$.

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.