COMP 3711 – Design and Analysis of Algorithms
2021 Fall Semester – Written Assignment # 1
Distributed: Sept 9'th 2021
Due: Sept 20'th 2021, 11:59 PM

Your solution should contain
(i) your name, (ii) your student ID #, and (iii) your email address
at the top of its first page.

Some Notes:

- Please write clearly and briefly. Your solutions should follow the guidelines given at
  https://canvas.ust.hk/courses/38226/pages/assignment-submission-guidelines
  In particular, your solutions should be written or printed on *clean* white paper with no watermarks, i.e., student society paper is not allowed.

- Please also follow the guidelines on doing your own work and avoiding plagiarism as described on the class home page.
  You must acknowledge individuals who assisted you, or sources where you found solutions. Failure to do so will be considered plagiarism.

- The term *Documented Pseudocode* in questions (4) and (5) means that your pseudocode must contain documentation, i.e., comments, inside the pseudocode, briefly explaining what each part does.

- Many questions ask you to explain things, e.g., what an algorithm is doing, why it is correct, etc. To receive full points, the explanation must also be *understandable* as well as correct.

- Please make a *copy* of your assignment before submitting it. If we can't find your submission, we will ask you to resubmit the copy.

- Submit a SOFTCOPY of your assignment to Canvas by the deadline. The softcopy should be one PDF file (no word or jpegs permitted, nor multiple files).
  If your submission is a scan of a handwritten solution, make sure that it is of high enough resolution to be easily read. At least 300dpi and possibly denser.

Problem 1: [25 pts]

In class, we "solved" the Mergesort recurrence

$$\forall n > 1, \quad T(n) \leq T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lceil \frac{n}{2} \right\rceil\right) + n \quad \text{and} \quad T(1) = 1 \qquad (1)$$

by simplifying and assuming that n was a non-negative integral power of two. This permitted replacing Equation (1) with

$$T(n) \leq 2T(n/2) + n \quad \text{and} \quad T(1) = 1.$$

We were then able to use the expansion method (with implicit induction) to derive the *exact* solution

$$T(2^k) \leq 2^k + k2^k. \qquad (2)$$

If $n = 2^k$ this becomes $T(n) \leq n + n \log_2 n$.

We then jumped to say that Equation (2) implies $T(n) = O(n \log n)$.

It doesn't, formally. The expansion method only really solved it for n being a power of 2.

The solution to this homework problem will show, at least for two recurrences, why such simplifications are "legal" and why Equation (1) implies $T(n) = O(n \log n)$ for all $n \geq 1$.

Before attempting this homework, you should first read the "Practice Homework" on the class tutorial page and understand the solution given for that.

That will not only provide some starting intuition for how to solve this problem, it will also illustrate how to formally write solutions.

(a) Let $c > 0$ be constant. Let $T(n)$ be a function satisfying

$$\forall n > 2, \quad T(n) \leq T\left(\left\lfloor \frac{n}{3} \right\rfloor\right) + c \quad \text{and} \quad T(1) = T(2) = 1. \tag{3}$$

(i) Prove using the expansion method that $T(3^k) = O(k)$.

(ii) Let $S(n)$ be some **nondecreasing** function of n.
You are told that $S(n)$ also satisfies $S(3^k) = O(k)$.
Prove that this implies $S(n) = O(\log_3 n)$.

(iii) For all $n \geq 1$, set $R(n) = \max_{1 \leq i \leq n} T(i)$. Prove that

$$\forall n > 2, \quad R(n) \leq R\left(\left\lfloor \frac{n}{3} \right\rfloor\right) + c.$$

(iv) Prove that if $T(n)$ satisfies Equation (3) then $T(n) = O(\log n)$.
You can either do this by using the facts proven in parts (i), (ii) and (iii) (recommended) or directly (much longer).

(b) We learned in class that if $T(n)$ satisfies Equation (1) then it satisfies Equation (2) so

$$T(2^k) = O(k2^k). \tag{4}$$

(v) Let $S(n)$ be some **nondecreasing** function of n that also satisfies Equation (4). Prove that this implies $S(n) = O(n \log_2 n)$.

(vi) For all $n \geq 1$, set $R(n) = \max_{1 \leq i \leq n} T(i)$.
Prove that if $T(n)$ satisfies Equation (1) then

$$\forall n > 1, \quad R(n) \leq R\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + R\left(\left\lceil \frac{n}{2} \right\rceil\right) + n.$$

(vii) Assuming the correctness of Equation (4), prove that $T(n)$ defined by Equation (1) satisfies $T(n) = O(n \log_2 n)$.
You can either do this by using the facts proven in parts (v) and (vi) (recommended) or directly (much longer).

Requirements, Hints and Comments:

- Prove each of parts (i) - (vii) separately in order.
  Make sure that you clearly label each part.

  Leave a few empty lines between each part.
  Solutions not following this format will have points deducted.
- You should assume the correctness of earlier parts when solving later ones. For example, when solving part (iv), you should assume the correctness of parts (i), (ii) and (iii).
- k and n will always denote non-negative integers.
- $F(n)$ being a *non-decreasing function* means $\forall n,\ F(n) \leq F(n+1)$.
- $f(n) = O(g(n))$ means that
  $$\textit{there exists } c > 0 \textit{ and } n_0 \textit{ such that } \forall n > n_0,\ f(n) \leq cg(n).$$
  Your proofs in parts (i), (ii) (iv), (v) and (vii) should explicitly identify the values of c and n_0 used.
- $T(3^k) = O(k)$ and $T(2^k) = O(k2^k)$ mean, respectively, that
  $$\textit{there exists } c > 0 \textit{ and } k_0 \textit{ such that } \forall k > k_0,\ T(3^k) \leq ck.$$
  $$\textit{there exists } c' > 0 \textit{ and } k_0' \textit{ such that } \forall k > k_0',\ T(2^k) \leq c'k2^k.$$
- In part (ii) you only know that $S(n)$ is a non-decreasing function satisfying $S(3^k) = O(k)$. Your proof may not use any other assumptions about $S(n)$.
  Hint: Use the fact that for $x \geq 3$, $\log_3 3x = 1 + \log_3 x \leq 2 \log_3 x$.
- Similarly, in part (v), you may only use the facts that $S(n)$ is a non-decreasing function satisfying $S(2^k) = O(k2^k)$.
- Write full, formal solutions.

Problem 2 [14 pts]

For each pair of expressions (A, B) below, indicate whether A is O, Ω, or Θ of B. Note that zero, one, or more of these relations may hold for a given pair. List the most appropriate relation.

It often happens that some students will get the directions wrong, so please write out the relation in full, i.e., you should write exactly one of the terms or write that *none of the relations is satisfied*.

More explicitly, if any of the relationships are satisfied, you should write the appropriate

$$A = O(B), \quad A = \Omega(B), \quad \text{or} \quad A = \Theta(B)$$

and not just $O(B)$, $\Omega(B)$ or $\Theta(B)$, omitting the A.

If no relationship is satisfied write *none of the relations is satisfied*.

(a) $A = n^3 - n^2 \log n$, $B = 30n^2 - 7n^3 + 2n^4$;

(b) $A = \log_{100}((n+2)!)$, $B = \log_2 n^n$;

(c) $A = (16)^{\frac{1}{\log_n 5}}$, $B = 2^{\sqrt{3 \log_2 n}}$;

(d) $A = \sum_{k=1}^{n} k^4$, $B = 10 \times \binom{n}{5}$;

(e) $A = n^6 2^{n(\log_2 n)^3}$, $B = n^8 + 2021^{2020^{2019}}$;

(f) $A = \sum_{k=1}^{n} \frac{1}{k(k+1)}$, $B = \ln\left(\sum_{k=1}^{n} \frac{1}{k}\right)$;

(g) $A = n(1 + (-1)^n)$, $B = n(1 + (-1)^{n+1})$.

Write one solution per line.

Hints and Comments:

- If only one relationship holds, then the most appropriate relationship is just that relationship. If $A = \Theta(B)$ then you should write $A = \Theta(B)$ (even though both $A = O(B)$ and $A = \Omega(B)$ are both also valid). As examples,
  - If $A_1 = 10n$ and $B_1 = n^2$ then the answer should be "$A_1 = O(B_1)$".
  - If $A_2 = 10n^2$ and $B_2 = n^2 - 20n$ then the answer should be "$A_2 = \Theta(B_2)$".
- It is NOT necessary to provide justifications for your answer. But if you do not provide any justifications and you are wrong, we cannot award you partial credit.

Problem 3 [20 pts]

Give asymptotic upper bounds for $T(n)$ satisfying the recurrences in parts (a) and (b). Make your bounds as tight as possible.

Note that "tight upper bound" means the best possible Big-O answer. So, $T(n) = O(n^5)$ would not be considered a fully correct answer if $T(n) = O(n^4)$ as well.

You must prove your results from scratch using either the expansion or tree method shown in class. Show all of your work.

For (a) you may assume that n is a power of 3; for (b) you may assume that it is a power of 4.

(a) $T(1) = 1; \quad T(n) = 10T(n/3) + n^2$ for $n > 1$.

(b) $T(1) = 1; \quad T(n) = 16T(n/4) + n^2$ for $n > 1$.

Problem 4: [21 pts] Tutorial question DC8 asks you to solve the "stock-profit" problem using divide-and-conquer techniques in $O(n \log n)$ time. The bottom of the first page of the solution slides states:
Note: This problem can be solved using similar ideas that are used for the maximum contiguous subarray problem. Similar to that problem, there is also an $O(n)$ time linear scan solution. That is not what is being requested here, but it is a good exercise to see if you can find it.

You now need to find an $O(n)$ time algorithm for solving the stock-profit problem.

(a) Write your algorithm down in clear documented pseudocode.
Separately, below the algorithm, provide an *understandable* explanation in words as to what the algorithm is doing.

(b) Prove (explain) why your algorithm is correct.

(c) Prove (explain) that your algorithm runs in $O(n)$ time.

Problem 5: [20 pts] **Searching**

The input to the problem is an n element array $A[0, \ldots, n-1]$. Element $A[i]$ is *locally minimal* in array A if $A[i]$ is not larger than all of its neighbors (either the neighbors to both sides, or the neighbor to one side if it's the first or last item). For example, 7, 6 and both 8's are the locally minimal items in the array below

7	**13**	**15**	**12**	**6**	**9**	**11**	**14**	**16**	**23**	**25**	**24**	**8**	**8**

Finding an absolute globally minimal item requires $O(n)$ time to inspect every item. Finding a locally minimal one can be done faster.

You now need to find an $O(\log n)$ time algorithm for finding a locally minimal element in the array. If there are many such items, you only need to return one of them.

(a) Write your algorithm down in clear documented pseudocode.

(b) Prove (explain) why your algorithm is correct.

(c) Prove (explain) that your algorithm runs in $O(\log n)$ time.