Spring 2022

CSE 317: Design and Analysis of Algorithms

Midterm Exam [All sections]

Thursday, March 10, 2022.
Total marks: 20 points, Duration: 120 minutes.

Name: ______________________________, Student ID: ______________________________

— SOLUTION —

Instructions

- Use of mobile devices is strictly prohibited for the entirety of the exam period.
- Please submit your devices in your bag at the front of the examination room.
- You may keep writing material with you on your desk.
- Please do not use a pencil or a red pen for answers that you want to be graded.
- Please write your answers in the provided answer book.
- Acquisition of answers through unfair means will automatically cancel your exam.
- Keep track of the time.
- Please write your name and student ID on this question paper as well as on answer book clearly.
- Submit the question paper along with the answer book.

— Good Luck —

1. (05 points) Assume f, g, h are three asymptotically nonnegative functions i.e. $f(n) \geq 0$, $g(n) \geq 0$, $h(n) \geq 0$ for all values of $n \in \mathbb{N} \cup \{0\}$. Furthermore, $f(n) = O(g(n))$ and $g(n) = O(h(n))$. Answer following questions as True or False. Justify your answer.

   (a) $f(n) = n \ln n - 5$ and $g(n) = n \log n + 2$.

   Solution: True. $f(n) = O(n \log n), g(n) = O(n \log n)$.

   (b) $g(n) = \sqrt{n^3}$ and $h(n) = 2n^{\frac{3}{2}+\epsilon}$ for some $\epsilon \in \mathbb{R} > 0$.

   Solution: True. $g(n) = O(n^{3/2}), h(n) = O(n^{2/3+\epsilon})$.

   (c) $g(n) = 1/(n^2 + 1)$ and $f(n) = 1/(n + 1)$.

   Solution: False. $g(n) = Of(n)$.

   (d) $f(n) = g(n) = h(n)$.

   Solution: True. Transitivity property.

   (e) $f(n) = 3^n - n^2$ and $h(n) = 2^n$.

   Solution: False. $h(n) = O(f(n))$.

2. (04 points) Solve following recurrences.

   (a) $f(n + 1) = 2f(n) - f(n - 1)$ with $f(0) = 0, f(1) = 1$.

   Solution: The recurrence $f(n+1) = 2f(n)-f(n-1)$ with $f(0) = 0, f(1) = 1$ can be rewritten as following:

   $$f(n) - 2f(n - 1) + f(n - 2) = 0, \quad \text{with } f(0) = 0, f(1) = 1,$$

   it will give us following characteristic equation $r^2 - 2r + 1 = 0$. This characteristic equation has two equal roots as $r^2 - 2r + 1 = (r - 1)^2$, as $r_1 = 1$, $r_2 = 1$. This will give us general solution as:

   $$f(n) = c_1 \cdot r_1^n + c_2 \cdot n \cdot r_2^n.$$

   Solving it for specific solutions will give us $c_1 = 0$ and $c_2 = 1$. Since both r_1 and r_2 are 1 therefore $f(n) = n = \Theta(n)$.

   (b) $g(n) = 7g\left(\frac{n}{3}\right) + \sqrt[3]{2n} - 5$ with $g(1) = 1$.

   Solution: Using Master Theorem from notes, we see that $a = 7$, $c = 3$, $\gamma = 1/3$. We see that $\log_c a = \log_3 7 \approx 1.7712$. So $\gamma < \log_c a$, therefore $g(n) = O(n^{1.7712})$.

3. (04 points) Let $G = (V, E)$ be a directed graph with n vertices. A *dead vertex* is a vertex $d \in V$ such that for all $v \in V$, $(d, v) \notin E$. Devise an algorithm that, given the adjacency matrix of G, determines whether or not G has a dead vertex in time $O(n)$ where $n = |V|$. Prove that the time complexity of your algorithm is $O(n)$.

Solution: Algorithm FIND-DEAD-VERTEX
Input: An directed graph $G = (V, E)$
Output: A dead vertex d from G

1. $d =$ CANDIDATE-DEAD-VERTEX(G)
2. **if** out_degree(d) $= 0$ **then return** d
3. **else return** `''No dead vertex''`

Algorithm: CANDIDATE-DEAD-VERTEX
Input: An directed graph $G = (V, E)$
Output: A candidate dead-vertex v from G

1. **if** $|V| = 1$ **then return** v
2. $A = \emptyset$
3. Pair vertices into at most $n/2$ pairs
4. Add left-over vertex to A
5. **foreach** pair v, w
6.   **if** $(v, w) \in E$ **then** add w to A
7.   **else** add v to A
8. **return** CANDIDATE-DEAD-VERTEX(A)

Time complexity: Let $T(n)$ be the time taken by CANDIDATE-DEAD-VERTEX then $T(n) \leq T(n/2) + O(n)$. Therefore, $T(n) = O(n)$ where $n = |V|$. The time complexity of FIND-DEAD-VERTEX is linear as well, as it checks if the in degree of the candidate dead vertex is zero.

4. Given a list A of size $n \geq 1$ that contains $\lceil \log_2 n \rceil$ different integers. For example, let $A = \langle 9, 2, 9, 2, 1, 2, 1, 1 \rangle$, in this case $n = 8$ and there are $\log_2 8 = 3$ distinct elements. We want the output to be $A' = \langle 1, 1, 1, 2, 2, 2, 9, 9 \rangle$.

   (a) (01 points) Design a brute-force algorithm in time $O(n \log n)$. Justify.

   Solution: Sort the input sequence A using any comparison-based algorithm for example MERGESORT: Time complexity: $O(n \log n)$.

   (b) (03 points) Design an efficient algorithm in time $O(n \log \log n)$. Justify.

Solution: **Algorithm:** EFFICIENT-SORT
Input: A list A of size n
Output: A sorted list A'

1. Create an empty binary-search tree T
2. **foreach** $a \in A$
3.   **if** $a \notin T$ **then**
4.   insert a in T
5.   create a counter c_a for a
6. $c_a = 0$
7.   **else** $c_a = c_a + 1$

Time complexity: The loop on Line 2 will run for n times and for each time it tries to insert an element from the input list A into a binary-search tree T. Since there are only $\lceil \log_2 n \rceil$ different elements in A therefore the size of the tree T will never be more than $O(\log n)$ so inserting an element in T will cost at most $O(\log \log n)$. Resulting in total running time as $O(n \log \log n)$.

(c) (03 points) Design an even better algorithm in time $O(n)$. Justify.

Solution: **Algorithm:** MORE-EFFICIENT-SORT
Input: A list A of size n
Output: A sorted list A'

1. Let D be a dictionary
2. **foreach** $a \in A$
3.   **if** $a \in D$ **then** $D[a] = D[a] + 1$
4.   **else** $D[a] = 1$
5. Sort the elements of the dictionary D.

Time complexity: The loop on Line 2 will run for n times and for each time it increases the counter associated with the dictionary element by 1 if the element exists in the dictionary otherwise it creates the dictionary entry. This step takes $O(1)$ on average. The last step (sorting) will take $O(\log n \cdot \log \log n)$ using any standard sorting algorithm as there are $O(\log n)$ elements in the dictionary. So total time complexity is $O(n) + O(\log n \cdot \log \log n) = O(n)$.