CSE 102 - Data Structures and Algorithms
Quiz 1 - Solutions

19 May 2022

Name: **Total Marks: 20 Marks**

Roll No: **Duration: 20 mins**

Questions 1 is mandatory. Answer any 3 out of the other 4 questions. Each question carries 5 marks.

1. Write true or false (no justification required):
   (a) If $f(n) = O(s(n))$ and $g(n) = O(r(n))$, then $f(n)/g(n) = O(s(n)/r(n))$. **False**
   (b) An array is an Abstract Data Type but a record is a Data Structure. **False**
   (c) Let $\mathcal{P}$ be the problem of finding if a given input integer is prime or not. Let N be a fixed integer that is given as input to $\mathcal{P}$. Then, the size of the input is N. **False**
   (d) When the input is sorted, although the average case complexity of binary search is $O(log(n))$, the worst case complexity of binary search is $O(n)$. **False**
   (e) If $f(n) = \Theta(n)$, then it is always true that $f(n) = \Omega(n)$. **True**

   Rubric : +1 each for correct answer.

2. Show that if $f(n) = O(s(n))$ and $g(n) = O(r(n))$, then $f(n) + g(n) = O(s(n) + r(n))$.

   Ans: We know that for any two function f_1 and f_2 if $f_1(n) = O(f_2(n))$, then there exists a constant $c > 0$ and a natural number n_0 such that $f_1(n) \leq c \cdot f_2(n)$ for all $n \geq n_0$. Using this, we have that,

$$\exists\ c_1 > 0 \text{ and } n_{01} \in \mathbb{N} \text{ such that } f(n) \leq c_1 \cdot s(n)\ \forall\ n \geq n_{01}: \tag{1}$$

   and

$$\exists\ c_2 > 0 \text{ and } n_{02} \in \mathbb{N} \text{ such that } g(n) \leq c_2 \cdot r(n)\ \forall\ n \geq n_{02} \tag{2}$$

   Let $c_3 = \max\{c_1, c_2\}$ and $n_0 = \max\{n_{01}, n_{02}\}$
   From the above equations, for all $n \geq \max\{n_{01}, n_{02}\}$ we have that

$$f(n) + g(n) \leq c_1 \cdot s(n) + c_2 \cdot r(n) \leq c_3 \cdot (s(n) + r(n)). \tag{3}$$

   This implies that

$$\exists\ c_3 > 0 \text{ and } n_0 \in \mathbb{N} \text{ such that } f(n) + s(n) \leq c_3 \cdot (s(n) + r(n))\ \forall\ n \geq n_{01}. \tag{4}$$

   Hence, $f(n) + g(n) = O(s(n) + r(n))$.

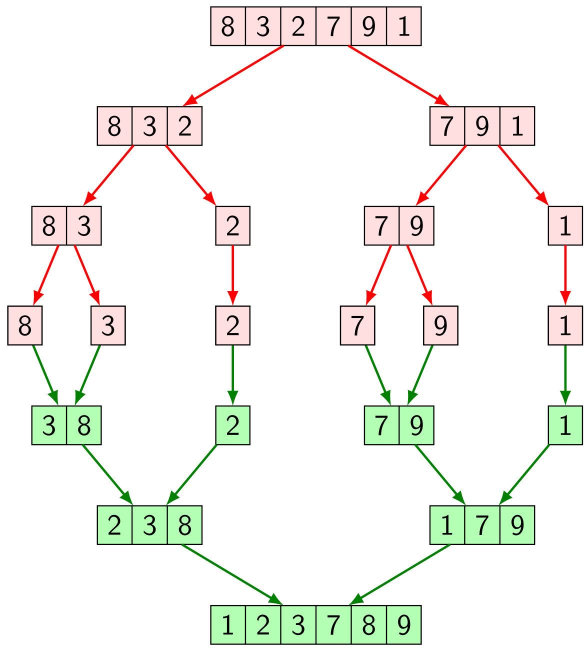
   Rubric : +1 for each equation, labelled above, and +1 for mentioning $c_3 = \max\{c_1, c_2\}$ and $n_0 = \max\{n_{01}, n_{02}\}$

3. Let $\mathcal{A}$ be an algorithm that takes an input x of size n. Then $\mathcal{A}$ makes $2 \cdot n$ iterations in total and the time taken for performing i^{th} iteration is i units of time. Is $\mathcal{A}$ an efficient algorithm? Explain in not more than 2 lines why you call it efficient or not efficient.

Ans: The total number of iterations made by $\mathcal{A}$ is $2 \cdot n$. Since i^{th} iteration takes i units of time, the total time taken by the algorithm is $T = 1 + 2 + 3 + \cdots + (2n) = 2n \cdot (2n+1)/2 = O(n^2)$. So, $\mathcal{A}$ is an efficient algorithm. We call $\mathcal{A}$ an efficient algorithm since it runs in polynomial units of time and not exponential units of time.

Rubric : +2 for mentioning the total time taken $O(n^2)$. +1 for mentioning that the algorithm is efficient. +2 for the correct reasoning. Students could have also written that the algorithm is not efficient with comparisons to O(n) and O(log n), so for such reasoning, out of 3, 1.5 marks should be given.

4. Say you are given a list $L = [8, 3, 2, 7, 9, 1]$. Show step by step how you would perform a merge sort on L.



Rubric : +1 marks for the final solution of the merge sort. +2 marks if they have done the dividing steps correctly. +2 marks if they have done the merging steps correctly.

5. Consider the following modified quick sort algorithm `ModQS` where in each iteration the pivot is chosen by finding the median of the list under consideration (assume that finding median of a list of size k takes $O(k)$ units of time.) Give the recurrence relation of `ModQS` algorithm and compute the time complexity of the algorithm (you can use the Master's theorem given below).

Master's Theorem:

The solution of the recurrence relation $T(n) = aT(n/b) + cnk$, where a and b are integer constants, $a \geq 1, b \geq 2$, and c and k are positive constants, is

$$T(n) = \begin{cases} O(n^{\log_b a}, \text{ if } a > b^k) \\ O(n^k \log(n)), \text{ if } a = b^k \\ O(n^k), \text{ if } a < b^k \end{cases}$$

Ans: In `ModQS` each iteration, we find the median of the list and we break the list into two sub-lists. We also know that the time taken to find the median of a list of size k is $O(k)$. So, we can give the recurrence relation as

$$T(n) = T(n/2) + O(n).$$

Using Master's theorem, we get that the time complexity of `ModQS` is $O(nlog(n))$.

Rubric : +3 marks for the correct recurrence relation (+2 marks for O(n/2) and +1 mark for O(n)). +2 marks for the correct time complexity.