CS 721: Advanced Algorithms & Analysis

Homework 1, Fall 2018, Total 90 points

Assigned on: Thursday, 08/30/2018
Due on: Tuesday, 09/11/2018

1. A binary search algorithm aims to find a target value within a sorted array by splitting the sorted input array into two subsets of almost equal size and recursively searches in one of these two subsets after making one comparison. Consider the following two modifications of binary search algorithm.

   (a) (10 points) Consider the modified binary search algorithm (called ternary search) so that it splits the sorted array <u>not</u> into two subsets of almost-equal sizes, but into three subsets of sizes approximately one-third. If this ternary search algorithm need to recursively search in one of these three subsets of approximately one-third size, how many comparisons need to be made in each step? Write down the recurrence for this ternary search algorithm and find the asymptotic runtime of this algorithm.

   (b) (10 points) Consider another variation of the binary search algorithm so that it splits a input sorted array into two subsets of sizes approximately one-third and two-thirds respectively. Write down the recurrence for this search algorithm when (i) the search always selects the smaller subset and (ii) the search always selects the larger subset. Find the asymptotic runtime of this algorithm for both these cases.

2. Consider the general k-ary search algorithm (generalization of binary search) which splits a sorted array into k subsets each of approximately size n/k, and by making $(k-1)$ comparisons, recursively searches in one of these k subsets. Clearly, the recurrence relation can be written as,

$$T(n) = T(n/k) + (k-1) \tag{1}$$

   Here k is a variable and can be a function of n as well.

   (a) (10 points) Suppose we set $k = \sqrt{n}$ in equation 1. Find asymptotic running of $T(n)$ in this case.

   (b) (5 points) Now suppose we set $k = \log n$ (where the base of the logarithm is 2) in equation 1. The above recurrence relation becomes,

$$T(n) = T(n/\log n) + (\log n - 1) \tag{2}$$

   Explain why equation 2 can not be solved using Master theorem.

   (c) (10 points) One way to solve equation 2 will be to find explicit upper bound (O) and lower bound (Ω) for $T(n)$. Let us concentrate on the upper bound first. Form the divide and conquer point of view, the recurrence relation $T(n) = T(n/\log n) + (\log n - 1)$ says that a problem of size n is divided into one sub-problem of size $n/\log n$ and "divide plus combine" step takes $(\log n - 1)$ time. To find an upper bound of $T(n)$ consider a separate divide and conquer algorithm which divides a problem of size n into one sub-problem of size $n/2$ and where the "divide plus combine" step takes $\log n$ time. Let the running time of this algorithm be $T_1(n)$. Since subproblem size $n/2$ is bigger than $n/\log n$ for all $n \geq 4$, $T(n) \leq T_1(n)$, and therefore, $T(n) = O(T_1(n))$. Thus solving $T_1(n)$ will give an upper bound of $T(n)$. Write down the recurrence relation of $T_1(n)$ and solve for $T_1(n)$. Assume $T_1(1) = 1$. Show your steps.

(d) (10 points) To find a lower bound of $T(n)$ consider yet another divide and conquer algorithm which divides a problem of size n into one sub-problem of size $n/\sqrt{n} = \sqrt{n}$ and where the "divide plus combine" step takes $\log n - 1$ time. Let the running time of this algorithm be $T_2(n)$. Clearly, $T(n) \geq T_2(n)$ and therefore, $T(n) = \Omega(T_2(n))$. Thus solving $T_2(n)$ will give an lower bound of $T(n)$. Write down the recurrence relation of $T_2(n)$ and solve for $T_2(n)$. Show your steps.

3. Let $f(n)$ and $g(n)$ asymptotically positive functions. State whether the following statements are TRUE or FALSE. In case a statement is TRUE, prove it using definitions of $O, \Omega, \Theta, \omega, o$ etc. In case it is FALSE, provide a counter example.

   (a) (5 points) $f(n) = O((f(n))^2)$.

   (b) (5 points) $f(n) = \Theta(f(n/2))$.

   (c) (5 points) $f(n) = \Omega(\sqrt{f(n)})$.

   (d) (10 points) $= \max(f(n), g(n)) = \Theta(f(n) + g(n))$.

4. 6. (10 points) Using **substitution method** show that the recurrence relation $T(n) = 4T(n/2) + n$ has a solution $T(n) = O(n^2)$.

Submision:

- Preferably all texts and diagrams should be electronically produced. However, for this homework you can scan and upload.
- Your name and page number should appear on each page.
- Entire assignment should be a single PDF file.
- the PDF file should be named in the format HW01_Lastname_Firstname.pdf, for example HW01_Sinha_Kaushik.pdf.
- Submit the pdf file on blackboard.
- Each homework assignment is due at midnight on due date.