COMP 3711 – Design and Analysis of Algorithms
2019 Spring Semester – Written Assignment # 2
Distributed: March 6, 2019 – Due: March 15, 2019
Corrected version of March 8, 2019, 12:00

Your solutions should contain (i) your name, (ii) your student ID #, and (iii) your email address
Notes:

- Please write clearly and briefly.
- Please follow the guidelines on doing your own work and avoiding plagiarism given on the class home page.
  In particular ***don't forget to acknowledge individuals who assisted you, or sources where you found solutions.*** Failure to do so will be considered plagiarism.
- Also follow the submission guidelines on the class web page. Use white, unwatermarked paper, e.g., no student society stationary.
  - If handwritten, solutions should be single sided, start a new page for every problem, be single column and leave space between consecutive lines and more space between paragraphs.
  - If typed, try to use an equation editor, e.g., latex , the equation editor in MS-WORD or whatever the equivalent is in whatever typesetting system you are using
- This assignment is due by 13:00 (1PM) on Friday March 15, 2019 in BOTH hard AND soft copy formats. A hard copy should be deposited in one of the two COMP3711 assignment collection boxes outside of room 4210. A soft copy for our records in PDF format should also be submitted via the online CASS system. See the Assignment 1 page in Canvas for information on how to submit online.
- The default base for logarithms will be 2, i.e., $\log n$ will mean $\log_2 n$. If another base is intended, it will be explicitly stated, e.g., $\log_3 n$.

Update The original released version of this assignment had incorrect formulas in the definitions of the median and quartile problems in problem 1. This version has corrected those errors.

Update The original version asked for a more complicated function (with four input parameters) as a solution to Question 4. This version simplifies the requirements on the problem and only has two parameters. We will also accept a solution to the original version of the problem.

Problem 1 [25 pts] Recall the *Selection* problem. *Given an array A of n unsorted values and an integer $k \leq n$ return the k-smallest item in A.*

Algorithmically, suppose you are given an array$A[1 \ldots n]$, $p < r$ and k. A procedure $Select(A, p, r, k)$ should find and report the k-smallest item in $A[p..r]$

In class you learned a simple $O(n)$ *randomized* algorithm for solving Selection. There also exists a (much more complicated) $O(n)$ worst-case time Selection algorithm.

A special case of the Selection problem is the *Median-Problem* which finds the middle ordered item in the subarray. Formally, $MEDIAN(A, p, r)$ should return $Select\left(A, p, r, \lceil \frac{r-p+1}{2} \rceil\right)$.

A special case of the Selection problem is the *Quartile-Problem* which finds the $\frac{1}{4}$ ordered item in the subarray. Formally, $Quart(A, p, r)$ should return $Select\left(A, p, r, \lceil \frac{r-p+1}{4} \rceil\right)$.

In this problem assume you are given a linear time procedure $Quart(A, p, r)$, i.e., it solves the Quartile problem in $O(r - p + 1)$ time.

Also assume that you are also given the linear time code to implement the Partition procedure taught in class and can call it as a subroutine in your algorithm.

1. **Show how, using $Quart()$ as a subroutine, you can solve the median problem in linear, i.e., $O(r - p + 1)$, time.**

   Justify the running time of your algorithm.

   Further restriction: Your solution must be performed in-place (no new array can be created). In particular, one alternative solution would be to extend the array to size m so that all the entries in $A[n+1, \ldots, m] = \infty$ and $m > 2n$. After this, you would be able to solve the median problem in the original array with just one call to the quartile problem in the extended array. With the further restriction, this is not possible.

2. Now suppose that you are given a linear time algorithm for solving the median problem.

   Show how, using this as a subroutine, Quicksort can be modified to run in $O(n \log n)$ *worst-case* time.

   Explain why your modified version runs in $O(n \log n)$ time.

   You MAY assume that all items in the array have distinct values.

3. (For this part you may assume that n and k are powers of 2) .
   Again assume that you are given a linear time algorithm for solving the median problem.

Given an array of size n, and $k \leq n$ find an $O(n \log k)$ algorithm that reorders the items in A so they are partitioned into k parts with items in each part less than or equal to the items in the next part.

More formally, let $\Delta = n/k$. Then, for $i = 0, 1, \ldots k-2$, all items in

$$A[i\Delta, i\Delta + 1, \ldots, (i+1)\Delta - 1]$$

are less than or equal to all of the items in

$$A[(i+1)\Delta, (i+1)\Delta + 1, \ldots, (i+2)\Delta - 1].$$

As an example if $k = 4$ and the original array was

$$[1, 3, 5, 7, 9, 11, 13, 15, 2, 4, 6, 8, 10, 12, 16, 14]$$

one legal output would be

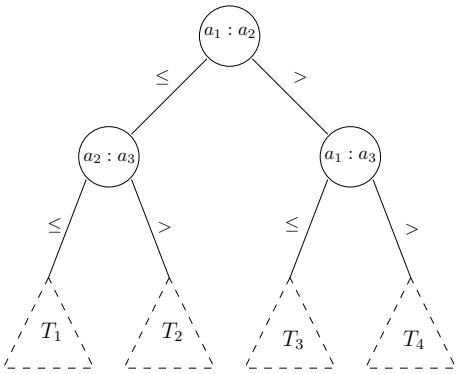
$$[1, 3, 2, 4, 7, 6, 5, 8, 12, 11, 10, 9, 13, 14, 16, 15]$$

Problem 2 [25 pts]

The figure below shows part of the decision tree for Insertion Sort (from page 12 of our introductory set of slides) operating on a list of 4 numbers, a_1, a_2, a_3, a_4.

Please expand subtree T_3, i.e., show all the internal (comparison) nodes and leaves in subtree T_3.

Note that the left branch of a comparison $(a : b)$ is the case $a \leq b$ and the right branch is $a > b$.



Problem 3: [25 pts]

Consider a long river, along which n houses are located. You can think of this river as an x-axis; the houses locations are given by their coordinates on this axis in a sorted order.

Your company wants to place cell phone base stations along the river, so that every house is within 8 kilometers of one of the base stations. Give an $O(n)$-time algorithm that minimizes the number of base stations used.

As well as giving the algorithm and showing that it runs in $O(n)$ time, you must prove that it yields an optimal solution.

Problem 4 [25 pts] **Updated March 8, 2019, 12:00.**

Suppose that X and Y are two sequences of equal size, sorted in increasing order. The first sequence is stored in sorted order in the array $X[1..n]$; the second sequence is stored in sorted order in the array $Y[1..n]$;

The goal of this problem is to find an efficient algorithm to find the median element in the combined set of $2n$ elements.

As an example, if

$$X = [1, 3, 7, 8, 10, 11],\ Y = [2, 4, 9, 13, 17, 19],$$

then the median item is $X[4] = 8$.

Note that this could easily be "solved" in $O(n)$ time if you merge the two sorted arrays into one sorted array and return its n'th element. You should find something that runs in time $O(\log n)$. (Hint: Use a variation of binary search).

More specifically, you should design a procedure that is called as $MED(X, Y)$ that returns the correct answer.

In writing your solution you may assume that n is known as a global constant (so it does not need to be passed to the function).

Your solution to this problem should be separated into three parts.

(a) First provide clearly documented pseudocode for the procedure.

(b) Next, explain what your algorithm does and why it is correct. Be explicit in proving the correctness for all cases of the algorithm.

(c) Finally, derive and state a tight upper bound on the running time of your algorithm.