College of Computer & Information Science
Northeastern University
Fall 2018

CS5800
MS Algorithms
4 October 2018

Problem Set 3 (due Thursday, October 11, 9:59 PM)

Instructions:

- The assignment is due at the time and date specified. *No late homework will be accepted.*
- We encourage you to attempt and work out all of the problems on your own. You are permitted to study with friends and discuss the problems; however, *you must write up you own solutions, in your own words.*
- Please refrain from searching online or asking your peers or other students for solutions. The best way to learn the material is to attempt the problem yourself, and if you are stuck, identify where and why you are stuck and seek help to overcome the associated hurdles.
- If you do collaborate with any of the other students on any problem, please *list all your collaborators in your submission for each problem.*
- We require that all homework submissions be neat, organized, and *typeset*. You may use plain text or a word processor like Microsoft Word or LaTeX for your submissions. If you need to draw any diagrams, however, you may draw them with your hand.

1. (4 + 4 = 8 points) Multi-Merge

Suppose you would like to merge k sorted arrays, each of length n, into one sorted array of length nk.

(a) One strategy is to use the merge operation we studied in class to merge the first two arrays to obtain a new sorted array of size $2n$, then merge in the third to obtain a new sorted array of size $3n$, then merge in the fourth, and so on until you obtain the desired sorted array of length nk.

Formalize the above algorithm in pseudocode and analyze its time complexity, in terms of n and k.

(b) Give a more efficient algorithm to this problem, using divide and conquer. Analyze its time complexity, in terms of n and k.

Problem 2. (8 points) Finding widgets of majority type using pairwise testing

You have n widgets, each of which is of a certain *type*. You want to determine whether there is a *majority type*; i.e., if there is a type t such that the number of widgets of type t is *greater than* $n/2$. For instance, the set of 7 widgets with types A, A, B, C, A, C, A, respectively has a majority type (A) since there are 4 widgets of that type. On the other hand, the set of 6 widgets with types A, A, B, C, D, A has no majority type.

Unfortunately, you are unable to determine the type of any given widget. Instead, the only operation available to you is to call a subroutine EqualityTest that takes two widgets as input and returns *True* if both are of the same type and *False* otherwise.

Give a divide-and-conquer algorithm for determining if there is a majority type in a given set of n widgets. The running time of your algorithm is the number of calls made by your algorithm to EqualityTest. Analyze the running time of your algorithm.

Ideally, the running time of your algorithm should be $O(n \log n)$. You will receive extra credit if the running time of your algorithm is $O(n)$.

Problem 3. (4 points) Finding the end of an infinite array?

Suppose you are given a very long array A, whose first n elements are integers (in arbitrary order) and the remaining elements are the special symbol ∞. You can access any position i in A by referring to $A[i]$. However, you do not know n.

Give an algorithm for determining n. Analyze the number of accesses to A made by your algorithm, in terms of n. For full credit, your algorithm must make $O(\log n)$ accesses to A.