COSC 302: Analysis of Algorithms — Spring 2018
Prof. Darren Strash
Colgate University

Problem Set 4 — Divide and Conquer II, Average-Case Analysis, Heaps, Non-Comparison Sorting

Due by 4:30pm Friday, Feb. 23, 2018 as a single pdf via Moodle (either generated via LaTeX, or concatenated photos of your work). Late assignments are not accepted.

This is an *individual* assignment: collaboration (such as discussing problems and brainstorming ideas for solving them) on this assignment is highly encouraged, but the work you submit must be your own. Give information only as a tutor would: ask questions so that your classmate is able to figure out the answer for themselves. It is unacceptable to share any artifacts, such as code and/or write-ups for this assignment. If you work with someone in close collaboration, you must mention your collaborator on your assignment.

Suggested practice problems (not to be turned in): 4.3-1, 4.3-8, 4.4-2, 4.4-4, 4.4-6, 4-3

1. *Hobbies.* Suppose you are given a group of n people each of which have exactly one hobby (though their specific hobbies are unknown to you). Suppose that you can find out if any two people have the same hobby by asking them to compare hobbies and then they tell you *yes* or *no*—in the process you do not learn what their hobbies are, only that they are the same or different. Now, you are tasked with determining if more than $n/2$ people in the group have the same hobby.

   Describe an algorithm that determines if more than $n/2$ people have the same hobby by asking $O(n \lg n)$ pairs of people to compare hobbies.

   Hints: Consider the following when designing your algorithm.

   (a) The combine step here is critical: the recursion fairy must return more than just "yes" or "no" as the solution to a subproblem.

   (b) Consider all 4 combinations of "yes"/"no" from the recursion fairy.

      i. Is a "yes" answer possible when both subproblems are "no"?

      ii. How many subproblems must return "yes" for the total problem to be "yes"?

      iii. Do you need to compare additional hobbies in case of a "yes" answer? How many comparisons can you do and still have $O(n \lg n)$ 'running time'? (i.e., what should your recurrence be?)

2. Suppose we are given a set S of n items, each with a value and a weight. For any element $x \in S$, we define two subsets

   - $S_{<x}$ is the set of all elements of S whose value is smaller than the value of x.
   - $S_{>x}$ is the set of all elements of S whose value is larger than the value of x.

   For any subset $R \subseteq S$, let $w(R)$ denote the sum of the weights of elements in R. The weighted median of S is any element x such that $w(S_{<x}) \leq w(S)/2$ and $w(S_{>x}) \leq w(S)/2$. Describe and analyze an algorithm to compute the weighted median of a given weighted set in $O(n)$ time. Your input consists of two unsorted arrays $S[1..n]$ and $W[1..n]$, where for each index i, the i-th element has value $S[i]$ and weight $W[i]$. You may assume that all values are distinct and all weights are positive.[1]

3. Suppose we are given an array $A[1..n]$ of distinct integers, and we wish to use linear search to find an element with maximum value. Prove that the expected number of comparisons made by linear search until the maximum element is *encountered*[2] is approximately[3] $n/2$. Use the following template for guidance.

   (a) We will assume that the input consists of distinct integers, and that all permutations of these integers are equally likely. What is the probability that a particular element $A[i]$ is the maximum element in A?

   (b) We need an indicator random variable. Recall that indicator random variables are used to *count* events, being 1 when you encounter an event, and 0 when you don't. What do we want to count here? What does a summation look like for this problem? Give a summation as a template, and temporarily use this summation to guide you in the next steps.

   Hint: You don't **need** a double summation for this problem. However, you **can** solve it with a double summation.

   (c) Devise an indicator random variable. Your indicator random variable should be 1 when a given element meets your criteria, and 0 otherwise.

   (d) Is it sufficient to simply to have a summation that includes your indicator random variable and no other multiplicative factors? Why or why not? If you need some additional multiplicative factor, what is it?

   (e) Now compute the expected value of your summation by combining together your indicator random variable, your probability, and any multiplicative factor you might have.

[1] From Jeff Erickson's *Algorithms and Models of Computation* (`http://www.cs.illinois.edu/~jeffe/teaching/algorithms`); Chapter 1: Recursion.

[2] Note that this is different that finding the maximum element. As we may encounter the maximum element, but not be assured it is truly maximum until we have evaluated all elements in the array.

[3] Note that I am not giving you the exact number, just something close (intuitively). Your number should be exact, and not asymptotic.

4. In this problem, you will execute a sequence of operations on a given array, which is not yet a heap. The operations are cumulative: each operation is executed on the result of the previous operation. For each operation, draw the binary tree representation of the output. The first operation is applied to the following array:

$$A = [10, 9, 6, 5, 3, 8, 7, 2, 1, 0].$$

   (a) First, draw the contents of the array A in the binary tree representation.

   (b) $\textsc{Max-Heapify}(A, 3)$

   (c) $\textsc{Heap-Extract-Max}(A)$

   (d) $\textsc{Heap-Increase-Key}(A, 9, 4)$

5. (Problem 8.2-4 from CLRS) Describe an algorithm that, given n integers in the range 0 to k, preprocesses its input and then answers any query about how many of the n integers fall into a range $[a..b]$ in $O(1)$ time. Your algorithm should use $\Theta(n + k)$ preprocessing time.