CS 584/684 Algorithm Design and Analysis

Homework 2

Portland State U, Winter 2021 *01/12/21*

Lecturer: Fang Song *Due: 01/19/21*

Instructions. This problem set contains 5 pages (including this cover page) and 4 questions. A random subset of problems will be graded.

- Your solutions will be graded on *correctness* and *clarity*. You should only submit work that you believe to be correct, and you will get significantly more partial credit if you clearly identify the gap(s) in your solution. It is good practice to start any long solution with an informal (but accurate) summary that describes the main idea. You may opt for the "I take 15%" option.
- You need to submit a PDF file before the deadline. Either a clear scan of you handwriting or a typeset document is accepted. You will get 5 bonus points for typing in LaTeX (Download and use the accompany TeX file).
- You may collaborate with others on this problem set. However, you must **write up your own solutions** and **list your collaborators and any external sources** for each problem. Be ready to explain your solutions orally to a course staff if asked.
- For problems that require you to provide an algorithm, you must give a precise description of the algorithm, together with a proof of correctness and an analysis of its running time. You may use algorithms from class as subroutines. You may also use any facts that we proved in class or from the book.

1. (Recurrence) Solve the following recurrences.
   (a) (5 points) $A(n) = 2A(n/4) + \sqrt{n}$
   (b) (5 points) $B(n) = 2B(n/4) + n$
   (c) (5 points) $C(n) = 3C(n/3) + n^2$
   (d) (5 points (bonus)) $D(n) = \sqrt{n}D(\sqrt{n}) + n$

2. (Quicksort) We were not precise about the running time of Quicksort in class (for a good reason). We will give some case studies in this problem (and appreciate the subtlety).

   (a) (10 points) Given an input array of n elements, suppose we are unlucky and the partitioning routine produces one subproblem with $n - 1$ elements and one with 0 element. Write down the recurrence and solve it. Describe an input array that costs this amount of running time to get sorted by Quicksort.

   (b) (5 points) Now suppose that the partitioning always produces a 9-to-1 proportional split. Write down the recurrence for $T(n)$ and solve it.

   (c) (5 points) What is the running time of Quicksort when all elements of the input array have the same value?

3. (Akinator's trick) Play the game Akinator online (`https://en.akinator.com/`), and answer the questions below.

   (a) (10 points) Given a *sorted* array A with distinct numbers, we want to find out an i such that $A[i] = i$ if exists. Give an $O(\log n)$ algorithm.

   (b) (10 points) Consider a sorted array with distinct numbers. It is then rotated k (k is unknown) positions to the right, and call the resulting array A. (Example: (8,9,2,3,5,7) is the sorted array (2,3,5,7,8,9) rotated to the right by 2 positions) Design as efficient an algorithm as you can to find out if A contains a number x.

   Exercise (do not turn in). Can you think of some real-world problems that the techniques in your algorithms could be useful?

4. (Counting inversions) Given a sequence of n *distinct* numbers $a_1, \ldots, a_n$, we call (a_i, a_j) an *inversion* if $i < j$ but $a_i > a_j$. For instance, the sequence $(2, 4, 1, 3, 5)$ contains three inversions $(2, 1)$, $(4, 1)$ and $(4, 3)$.

   (a) (15 points) Given an algorithm running in time $O(n \log n)$ that counts the number of inversions. (Hint: does Merge-sort help?) Can you also output all inversions?

   (b) (10 points (bonus)) Let's call a pair a *significant inversion* if $i < j$ and $a_i > 2a_j$. Given an $O(n \log n)$ algorithm to count the number of significant inversions.