CSC-421 Applied Algorithms and Structures
Fall 2021-22

Instructor: Iyad Kanj
Office: CDM 832
Phone: (312) 362-5558
Email: ikanj@cs.depaul.edu
Office Hours (Office/Zoom): Mon. 4:00-5:30 & Wed. 10:00-11:30
Course Website: https://d2l.depaul.edu/

Assignment 2

(Due October 11)

Please upload your submission as a single PDF file on D2L. If your submission consists of more than one file, convert all your files into a single PDF file and upload it.

1. (15 points) Solve the following recurrence relations. You do not need to give a $\Theta()$ bound for (a) and (b); it suffices to give the $O()$ bound that results from applying the Master theorem. You may assume that $T(n) = O(1)$ for $n = O(1)$.

   (a) $T(n) = 2T(n/3) + 1$.

   (b) $T(n) = 7T(n/7) + n$.

   (c) $T(n) = T(n-1) + 2$.

2. (15 points) Give a recursive version of the algorithm **Insertion-Sort** based on the following paradigm: to sort $A[1..n]$, we first sort $A[1..n-1]$ recursively and then insert $A[n]$ in its appropriate position. Write a pseudocode for the recursive version of **Insertion-Sort** and analyze its running time by giving a recurrence relation for the running time and then solving it.

3. (15 points) Let A be an array of n numbers. Use the algorithm **Select**(A, k) for finding the k-th smallest element of any array A of numbers to modify **Quick Sort** so that it runs in $O(n \lg n)$ time in

the worst case. Write a pseudocode for the modified **Quick Sort**, and analyze its running time by describing its recurrence relation and solving it. You can use the algorithm/subroutine **Select**() as a black box in the modified **Quick Sort**.

4. (15 points) Give an algorithm that takes as input a positive integer n and a number x, and computes x^n (i.e., x raised to the power n) by performing $O(\lg n)$ multiplications. Your algorithm **CANNOT** use the exponentiation operation, and may use only the basic arithmetic operations (addition, subtraction, multiplication, division, modulo). Moreover, the total number of basic arithmetic operations used should be $O(\lg n)$.

5. (20 points) Although merge sort runs in $\Theta(n \lg n)$ worst-case time and insertion sort runs in $\Theta(n^2)$ worst-case time, the constant factors in insertion sort can make it faster in practice for small problem sizes on many machines. Thus, it makes sense to coarsen the leaves of the recursion by using insertion sort within merge sort when subproblems become sufficiently small. Consider a modification to merge sort in which n/k sublists of length k are sorted using insertion sort and then merged using the standard merging mechanism, where k is a value to be determined.

   (a) (5 points) Show that insertion sort can sort the n/k sublists, each of length k, in $\Theta(nk)$ worst-case time.

   (b) (10 points) Show how to merge the sublists in $\Theta(n \lg (n/k))$ worst-case time.

   (c) (5 points) Given that the modified algorithm runs in $\Theta(nk + n \lg(n/k))$ worst-case time, what is the largest value of k as a function of n for which the modified algorithm has the same running time as standard merge sort, in terms of Θ-notation?

6. (20 points) A group of n Ghostbusters is battling n ghosts. Each Ghostbuster carries a proton pack, which shoots a stream at a ghost, eradicating it. A stream goes in a straight line and terminates when it hits the ghost. The Ghostbusters decide upon the following strategy. They will pair off with the ghosts, forming n Ghostbuster-ghost pairs, and then simultaneously each Ghostbuster will shoot a stream at his chosen ghost. As we all know, it is very dangerous to let streams cross, and so the Ghostbusters must choose pairings for which no streams will

cross. Assume that the position of each Ghostbuster and each ghost is a fixed point in the plane and that no three positions are collinear.

(a) (10 points) Argue that there exists a line passing through one Ghostbuster and one ghost such that the number of Ghostbusters on one side of the line equals the number of ghosts on the same side. Describe how to find such a line in $O(n \lg n)$ time.

(b) (10 points) Give an $O(n^2 \lg n)$-time algorithm to pair Ghostbusters with ghosts in such a way that no streams cross.

7. **Suggested Programming Problems on LeetCode; not to be submitted.** Below are the titles of coding problems that I suggest that you do in LeetCode.com (under the tab "Problems"). You can test your submissions and view the solutions in LeetCode.

(i) Merge k sorted lists.

(ii) Majority element.

(iii) Search a 2D Matrix.