CMPSC 465 Data Structures & Algorithms
Fall 2025 Ke Chen and Yana Safonova

HW 2

Due September 22, 10:00 pm

Instructions: You are encouraged to solve the problem sets on your own, or in groups of three to five people, but you must write your solutions strictly by yourself. You must explicitly acknowledge in your write-up all your collaborators, as well as any books, papers, web pages, etc. you got ideas from.

Formatting: Each part of each problem should begin on a new page. Each page should be clearly labeled with the problem number and the problem part. The pages of your homework submissions must be in order. When submitting in Gradescope, make sure that you assign pages to problems from the rubric. You risk receiving no credit for it if you do not adhere to these guidelines.

Late homework will not be accepted. Please, do not ask for extensions since we will provide solutions shortly after the due date. Remember that we will drop your lowest two scores.

This homework is due Monday, September 22, at 10:00 pm electronically. You need to submit it via Gradescope. Please ask on Canvas about any details concerning Gradescope.

1. (20 pts.) **Lower Bounds.** Consider the decision-tree model of the following problems. Apply the leaf-counting argument to obtain the information theory lower bounds (ITLB) for each of them:

(a). Find the median of an unsorted list of elements.

(b). Given a set of n elements, determine whether they are all distinct.

(c). Given two sorted list of n_1 and n_2 elements respectively, produce a merged list of sorted elements.

2. (20 pts.) **Overlapping Intervals.** You are given a list of n intervals $[x_i, y_i]$, where x_i, y_i are integers with $x_i \leq y_i$. The interval $[x_i, y_i]$ represents the set of integers between x_i and y_i. For instance, the interval [3,6] represents the set $\{3,4,5,6\}$. Define the *overlap* of two intervals I, I' to be $|I \cap I'|$, i.e. the number of integers that are members of both intervals. Design a divide-and-conquer algorithm that, when given n intervals, finds and outputs the pair of intervals with highest overlap (you may resolve ties arbitrarily). A trivial $\Theta(n^2)$ algorithm can be achieved by comparing all pairs of intervals; look for something better.
(*Hint:* Try splitting the list using the left endpoints of the intervals.)

3. (20 pts.) **Peak Elements**. Given an array with n distinct elements, a peak element is one that is strictly larger than both of its neighbors (or larger than its only neighbor, if it is on the edge of the array). For example, in [4,9,3,10], there are two peak elements: the 9 and the 10. Design an algorithm with $\Theta(\log n)$ running time to find the index of a peak element in an array. If an array has multiple peak elements, you may return any of them.

4. (20 pts.) **Median of Medians of Medians** Recall that in the median-of-medians selection algorithm, if we use group size 3, the time complexity satisfies $T(n) \leq T(n/3) + T(2n/3) + O(n)$. We cannot conclude $T(n) = O(n)$ based on this recurrence relation. The following algorithm aims to solve this issue by grouping twice and use the median of medians of medians as the pivot.

MoMoMSelect(A, k):

1. Partition A into $n/3$ groups, each of size 3.
2. Let M be the list of medians of these $n/3$ groups.
3. Partition M into $n/9$ groups, each of size 3.
4. Let M' be the list of medians of these $n/9$ groups.
5. Find the median p of M' by a recursive call MoMoMSelect(M', $|M'|/2$).
6. Use p as the pivot to partition A and recurs on the appropriate subarray as in Select.

What guarantee can you derive about the pivot p? Find the recurrence relation describing the worst-case running time of this algorithm. What is the solution to the recurrence relation?

5. (20 pts.) **Transpose of a Matrix.** The transpose of a matrix A, denoted A^T, is a new matrix whose rows are the columns of A. The standard way to compute the transpose of an $n \times n$ matrix takes $O(n^2)$ time.

(a). Consider an $n \times n$ matrix A where n is a power of 2. Partition it into four $n/2 \times n/2$ sub-matrices:

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

Show how to compute the transpose A^T by recursively computing the transposes of the sub-matrices.

(b). Write a recurrence relation for the time complexity $T(n)$ of this recursive algorithm. Solve the recurrence relation and determine the asymptotic time complexity. Does this recursive approach offer a performance improvement over the standard algorithm?