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

Worksheet 2

Monday, Sep 15, 2025

1. **k-th Smallest.** Given two *sorted* arrays A and B of size m and n respectively, and an integer k, $1 \leq k \leq m+n$, design an algorithm to find the k-th smallest number in A and B. Describe your algorithm and analyze the running time of your algorithm. Your algorithm should run in $O(\log(m+n))$ time.

2. **Matrices.** A is an $n \times n$ matrix containing integers. For simplicity, you may assume n is a power of 2. Design a divide-and-conquer algorithm to find the maximum element of A. Write the recurrence relation for the time complexity and report its runtime in Θ notation. What do you observe?

3. **Selection.** Consider the selection algorithm in which the input elements are divided into groups of 7. Write the recurrence to describe the worst-case running time of the algorithm and find the solution of the recurrence.

4. **Selection.** Suppose you have a black box algorithm $A1$ that finds the $\lfloor n/10 \rfloor$-th smallest of n elements (for any given n). Show that you can find the median of n elements by making $O(1)$ calls to $A1$ and using no other pairwise element-comparisons.

5. **Merge.** *A k-way merge operation.* Suppose you have k sorted arrays, each with n elements, and you want to combine them into a single sorted array of kn elements.

   a) Here's one strategy: Using the `merge` procedure, merge the first two arrays, then merge in the third, then merge in the fourth, and so on. What is the time complexity of this algorithm, in terms of k and n?

   b) Give a more efficient solution to this problem, using divide-and-conquer.

6. **Array Rotations.** Consider a rotation operation that takes an array and moves its last element to the beginning. After n rotations, an array $[a_0, a_1, a_2 \ldots a_{m-1}]$ of size m where $0 < n \leq m$, will become:

$$[a_{m-n}, a_{m-n+1}, \ldots, a_{m-2}, a_{m-1}, a_0, a_1, \ldots a_{m-n-1}]$$

   Notice how a_{m-1} is adjacent to a_0 in the middle of the new array. For example, two rotations on the array $[1,2,3,4,5]$ will yield $[4,5,1,2,3]$.

   You are given a list of unique integers `nums`, which was previously sorted in ascending order, but has now been rotated an unknown number of times. Find the number of rotations in $O(\log n)$ time. *(Hint: consider Binary Search.)*