# MCS 401 – Computer Algorithms I
## Spring 2025
## Problem Set 3

Lev Reyzin

**Due**: 3/5/25 by the beginning of class

**1.** Consider the problem of taking positive integers $x$ and $n$ and computing $x^n$ using as few multiplications as possible. Here, we are not concerned with the cost of performing the multiplications, but only with how many multiplications are done. Naively, $O(n)$ suffice by starting with 1 and multiplying $x$ with the result $n$ times. But observe that

$$x^n = \left\{ \begin{array}{ll} (x^{n/2})(x^{n/2}), & \text{for } n \text{ even} \\ (x^{(n-1)/2})(x^{(n-1)/2})x, & \text{for } n \text{ odd} \end{array} \right\}.$$

Use this observation to come up with a divide and conquer approach to computing $x^n$ that uses asymptotically fewer multiplications than linear in $n$.

**2.** Consider multiplying two $n$-digit integers $x$ and $y$ to get an integer $z = xy$. As we saw in class, the grade-school multiplication algorithm, which takes $O(n^2)$ time, can be vastly improved upon. One such improvement is given by the Toom-Cook algorithm. It divides the digits of $x$ and of $y$ into 3 (approximately) equal parts and uses 5 multiplications instead of the brute-force 9 to get $z$. It also uses a constant number of additions and subtractions. Give the recurrence resulting from this approach and solve the recurrence. How does it compare to the other integer multiplication methods we learned about in class?

**3.** You are given a one dimensional array that may contain both positive and negative integers. Give an $O(n \log n)$ algorithm to find the sum of contiguous (ie. next to one another) subarray of numbers which has the largest sum. For example, if the given array is $[-2, -5, \mathbf{6}, \mathbf{-2}, \mathbf{-3}, \mathbf{1}, \mathbf{5}, -6]$, then the maximum subarray sum is 7 (the subarray is marked in boldface). Argue that your algorithm is correct.

**4.** You are given two arrays, $A$ and $B$, each of which contains $n$ integers. The elements in each array are guaranteed to already be in sorted order in the input, i.e.

$$A[0] \leq A[1] \leq \ldots \leq A[n-1] \text{ and also } B[0] \leq B[1] \leq \ldots \leq B[n-1].$$

Give as fast an algorithm as you can for finding the *median* value of all the $2n$ numbers in both $A$ and $B$. (We define the median of $2n$ numbers to be the average of the $n$th smallest and $n$th largest values.) Argue that your algorithm is correct and give its running time.

**5.** You are given an array $X$ of $n$ elements. A majority element of $X$ is any element occurring in more than $n/2$ positions. The only access you have to the array is to compare any two of its elements for equality; hence you cannot sort the array, nor add up its values, etc. Design an $O(n \log n)$ divide-and-conquer algorithm to find a majority element in $X$ (or determine that no majority element exists).

**6.** You are given a $2^k \times 2^k$ board with one missing cell. (See Figure 1 below.)

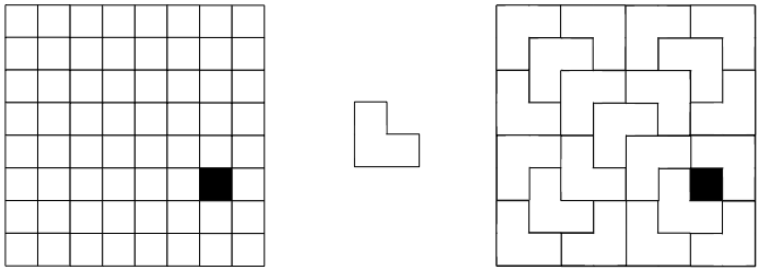

Figure 1: On the left is an example grid with a missing cell, with $k = 3$. In the middle is the "L-shaped" tile, to be used for tiling. On the right is an example solution.

Give an $O(2^{2k})$-time algorithm for filling the board with "$L$-shaped" tiles.