

MCS 401 – Computer Algorithms I  
Spring 2022  
Problem Set 3

Lev Reyzin

**Due:** 3/4/22 by the beginning of class

**1. [10 pts]** Suppose we are given an instance of the Minimum Spanning Tree (MST) problem on an undirected graph  $G$ . We assume that all edge costs are positive and distinct. Let  $T$  be the minimum-cost spanning tree for this instance. Now suppose we replace each edge cost  $c_e$  by its square,  $c_e^2$ , thereby creating a new instance of the problem with the same graph but with different costs. Prove or disprove the following claim:  $T$  must be the minimum-cost spanning tree in this new instance.

**2. [10 pts]** Consider you get as input a very sparse undirected weighted graph  $G = (V, E)$ , in particular for which  $|E| - |V| = 20$ . Give an  $O(|V|)$  time algorithm for finding a minimum spanning tree on  $G$  and prove your algorithm correct.

**3. [10 pts]** 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, in sequence) subarray of numbers which has the largest sum. For example, if the given array is

$$[-2, -5, \mathbf{6}, -2, -3, \mathbf{1}, \mathbf{5}, -6],$$

then the maximum subarray sum is 7 (the subarray is marked in boldface). Argue that your algorithm is correct.

**4. [10 pts]** 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.

$$A[0] \leq A[1] \leq \dots \leq A[n-1],$$

and also

$$B[0] \leq B[1] \leq \dots \leq B[n-1].$$

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. [10 pts] 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. [10 pts] You are given a  $2^k \times 2^k$  board with one missing cell. (See Figure 1 below.)

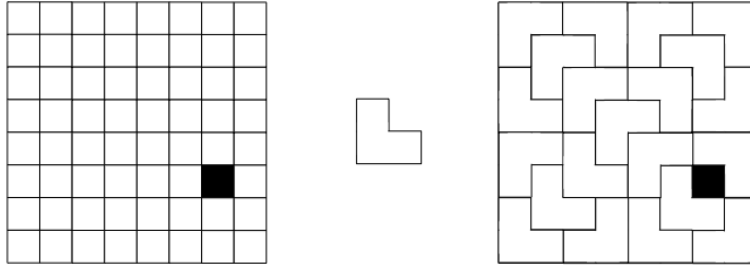


Figure 1: On the left is an example grid with a missing cell, with  $k = 3$ . In the middle is the “L-shaped” tile, to be used for tiling. On the right is an example solution.

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

7. [10 pts] The basic divide-and-conquer technique for multiplying two  $n$ -digit integers cleverly saves one out of four multiplications and yields the recurrence  $T(n) = 3T(n/2) + cn$  and gives an  $O(n^{1.59})$  algorithm. However, it is possible to do much better. Show how the Fast Fourier Transform, which is a method for multiplying two  $n$ -degree *polynomials*, can be used to actually multiply two  $n$ -digit *integers* in time  $O(n \log n)$ . *Hint: as a first step, create polynomials from your integers.*