Tutorial 3, Design and Analysis of Algorithms, 2019

1. (a) Find the number of nodes in the divide and conquer graph for computing FFT of a vector of length n (for simplicity you can assume n to be a power of 2).

   (b) Now find time complexity of the FFT algorithm by only considering the structure of the divide and conquer graph (without solving any recursion).

2. Find 1234×4321 using the FFT algorithm showing its divide and conquer graphs.

3. (a) Describe the generalization of the FFT procedure to the case in which n is a power of 3 (using three subproblems). Give a recurrence for the running time, and solve the recurrence.

   (b) Find 97×68 using the above algorithm showing its divide and conquer graphs.

4. Consider two sets A and B, each having n integers in the range from 0 to $10n$. We wish to compute the Cartesian sum of A and B, defined by
   $C = \{x + y \mid x \in A \wedge y \in B\}$
   Note that the integers in C are in the range from 0 to $20n$. We want to find the elements of C and the number of times each element of C is realized as a sum of elements in A and B. Show how to solve the problem in $O(n \log n)$ time.

5. Suppose you are given two sets A and B, each containing n positive integers. You can choose to reorder each set however you like. After reordering, let a_i be the ith element of set A, and let b_i be the ith element of set B. You then receive a payoff of $\Pi_{i=1}^{n} a_i^{b_i}$. Give an algorithm that will maximize your payoff. Prove that your algorithm maximizes the payoff, and state its running time.

6. Describe an efficient algorithm that, given a set $\{x_1, x_2, ..., x_n\}$ of points on the real line, determines the smallest set of unit-length closed intervals that contains all of the given points. Argue that your algorithm is correct.

7. Suppose you are given two sets A and B, each containing n positive integers. You can choose to reorder each set however you like. After reordering, let a_i be the ith element of set A, and let b_i be the ith element of set B. You then receive a payoff of $\sum_{i=1}^{n} a_i^{b_i}$. Design an algorithm that will maximize your payoff. Give a formal correctness proof for your algorithm, and find its time complexity.