Name:

Student ID #: 

• You are expected to work on this assignment on your own

• Use pseudocode, Python-like or English to describe your algorithms. Absolutely no C++/C/Java

• When designing an algorithm, you are allowed to use any algorithm or data structure we explained in class, without giving its details, unless the question specifically requires that you give such details

• Always remember to analyze the time complexity of your algorithms

• Homework has to be submitted electronically on Gradescope by the deadline. No late assignments will be accepted
Problem 1. (25 points)

Suppose you have $k$ sorted arrays, each with $n$ elements, and you want to combine them into a single sorted array of $kn$ elements. Describe a divide and conquer algorithm that takes $O(kn \log k)$ time. Explain carefully why your algorithm takes $O(kn \log k)$ time.

Answer:
Problem 2. (25 points)

A CS 141 student has been trying to speed-up Strassen’s divide-and-conquer for matrix multiplication algorithm. Recall that Strassen’s algorithm computes the product of two $n \times n$ matrices in $O(n^{2.81})$, and uses the fact that one can multiply two $n/2 \times n/2$ matrices with only 7 multiplications instead of 8 with the naive matrix-multiplication algorithm. Suppose the student came up with a variant of Strassen’s algorithm based on the fact that the product of two $n/3 \times n/3$ can be found with only $m$ multiplication instead of the normal 27. How small would $m$ have to be for this algorithm to be asymptotically faster than Strassen’s algorithm covered in class? You can assume $m > 9$. Justify your answer.

Answer:
Problem 3. (25 points)

Give a divide-and-conquer algorithm for multiplying two polynomials of degree $n$ in time $O(n^{\log_2 3})$. This algorithm is similar to Karatsuba’s integer multiplication algorithm that we covered in class.

Answer:
Problem 4. (25 points)

An array $A$ is said to have a majority element if more than half of the entries in $A$ are exactly the same. Describe an $O(n \log n)$ divide-and-conquer algorithm that determines whether an array $A$ of $n$ items has a majority element, and if so, returns that item. The only comparison operation allowed on the items is equality. That is, your algorithm can determine whether “$A[i] == A[j]$” or not in $O(1)$ time, but it cannot, for example, compare the items to sort them, or hash the items into buckets. Explain why your algorithm takes $O(n \log n)$ time.

Answer: