Homework 6: Divide and Conquer Algorithms

You may work in groups, but you must write solutions yourself. List your collaborators at the top of your submission.

Instructions for submission: Submit your homework as a single PDF file to the course Gradescope page. Either scan your *legible* handwritten solutions or type them up in LaTeX.

Setting: Suppose you are consulting for a high-performance computing center. They have a single supercomputer that can run one job at a time and have n jobs to run. Each job i has a running time of $t_i > 0$ and a strict deadline d_i.

Problem 1. Consider the following divide-and-conquer algorithm, which takes an array A of n numbers as input and returns a single number. The algorithm is given below.

Algorithm 1 Mystery Divide-and-Conquer Algorithm

```
1:  procedure MYSTERY(A, n)
2:      if n ≤ 1 then
3:          return A[1]                                  ▷ Base case
4:      k ← ⌊n/3⌋                                        ▷ Find the one-third point

5:      left ← MYSTERY(A[1..k], k)                       ▷ Recur on the left third
6:      right ← MYSTERY(A[k + 1..n], n − k)              ▷ Recur on rest

7:      total ← 0
8:      for i ← 1 to n do
9:          total ← total + (A[i] ∗ left ∗ right)        ▷ Combine results
10:     return total
```

Let's analyze the running time of this algorithm. Let $T(n)$ be the running time of MYSTERY on an input of size n. Write down a recurrence relation for the worst-case running time $T(n)$. You may assume n is a power of 3 for simplicity. **Note:** You do not need to solve the recurrence relation, just write it down.

Problem 2. Suppose you have developed a new divide-and-conquer algorithm. You analyze your pseudocode and determine that its worst-case running time $T(n)$ satisfies the following recurrence relation:

$$T(n) \leq 3T\left(n/2\right) + cn^2 \quad \text{for } n > 1, \quad \text{and} \quad T(1) \leq c,$$

where c is a positive constant.
Use the recursion tree method to find a closed-form expression for an upper bound on $T(n)$ in terms of n.

(a) Find the amount of work done at the root level (level 0) of the recursion tree.

(b) Find the amount of work done at level 1 and 2 of the recursion tree.

(c) Find a general formula for the total work down at level i of the recursion tree.

(d) Find the height of the recursion tree.

(e) Use the above to find a closed-form expression for an upper bound on $T(n)$.