Problem 1. Consider the following recurrence equation for the runtime of some algorithm:

\[ T(n) = 3T(n/5) + 2n + 1, \quad T(1) = 4 \]

(a) Calculate \( T(25) \) by hand. Show your work.

(b) Use the tree method to solve the recurrence exactly, assuming \( n \) is a power of 5. For each subpart briefly justify and/ or show your work when appropriate.

1. Draw the tree. You should show at least three levels at the top and at least two levels at the bottom (as done in class).
2. What is the height of the tree? (Note that a tree with one node has height 0, a tree with a root and some children are at level 1, etc.)
3. How many leaves are there?
4. What is the total work done by the leaves?
5. What is the size of each subproblem at level \( i \)?
6. How much work does each subproblem at level \( i \) do?
7. What is the total work for level \( i \)?
8. Write a summation for the total work not including the leaves?
9. Simplify the summation.
10. What is the total work for the entire algorithm?

Problem 2. A company wants to determine the highest floor of its \( n \)-story headquarters from which a gadget can fall with no impact on the gadget’s functionality. The company has two identical gadgets to experiment with. Design an optimal algorithm to solve this problem. You may write high level pseudo-code / steps in structured English.

Problem 3. Collaborative filtering is a technique for generating recommendations, such as suggestions for products, songs, movies, news stories, and so on. The idea is to identify other users who have similar preferences, and recommend to you things that have been popular with them. It requires a formal notion of “similarity” between users, and some of the essence of this is captured by the problem of counting the number of inversions in an array.

An inversion in an array \( A[1, \ldots, n] \) is a pair of indices \((i, j)\) such that \( i < j \) and \( A[i] > A[j] \).

For example, in the array \( A = [1, 3, 5, 2, 4, 6] \), there are three inversions, 5 and 2 (corresponding to \( i = 3 \) and \( j = 4 \)), 3 and the 2, and the 5 and the 4. For this problem we will count the number of inversions in an array of size \( n \).

(a) Write pseudo-code for an efficient brute force algorithm to find the total number of inversions in the worst-case. Find the exact number of inversions. Show your work.

(b) Write pseudo-code for an optimal algorithm to find the total number of inversions in the worst-case. What is the asymptotic runtime of your algorithm?