Directions: Use a separate page for each problem, and label each solution according to the problem attempted. Please make sure your pages are in order when you scan your work. Also make sure to upload your two-sided, handwritten notes page as well as any scratch work you used (but please make sure to clearly identify which work you want graded and which work is scratch).

You will have 90 minutes to complete this exam. Upon finishing, you may use up to 20 additional minutes to scan and upload your work to T-Square. You may not modify your exam once you have begun to scan and upload it. If you encounter any problems submitting your exam to T-Square, email your answers to `cpryby3@gatech.edu` within this 20-minute period.

1. Let T be a rooted binary tree with n leaves. Let $w_0, w_1, ..., w_{n-1}$ be n weights (positive integers) such that w_i is associated with the leaf ℓ_i, $0 \le i \le n-1$, ordered from left to right. Define the *weighted external path length* (WEPL) of T to be $\sum_{i=0}^{n-1} w_i d_i$, where d_i is the number of edges from the root to the ith leaf.

   Use dynamic programming to solve the *minimum weighted external path length* (Min-WEPL) problem, defined as follows: Given n weights $w_0, ..., w_{n-1}$, construct a binary tree T with leaves $\ell_0, ..., \ell_{n-1}$ such that WEPL(T) is the minimum among all such trees.

   [Note that you may not reorder the left-to-right order of the weights: the leftmost leaf must have weight w_0, the next leftmost leaf must have weight w_1, and so on.]

   (a) Define the subproblems to be solved in English.

   (b) Define an appropriate recurrence for the subproblems.

   (c) Implement the dynamic programming algorithm in pseudocode using an iterative, bottom-up method (not recursion or memoization).

   (d) Analyze the worst-case time performance of the algorithm in terms of n.

2. A *matching* in an undirected graph $G = (V, E)$ is a subset of the edge set E no two of whose edges have a vertex in common.

   Let G be an undirected graph with weight function w on its edges. Consider the problem of selecting a matching in G of maximum total weight.

   Can this problem can be solved in general using the greedy strategy of selecting, at each step, the heaviest edge in G not adjacent to any currently selected edge? Prove why or why not.

3. Consider the binary counter described in the lecture on amortized analysis, and suppose that, in addition to the `increment` operation, we also require a `reset` operation which resets all bits in the counter to zero. Suppose that the cost of examining or setting a bit is $\Theta(1)$.

   (a) Implement the `reset` operation and re-implement the `increment` operation so that the total amortized cost of any sequence of n operations is $O(n)$.

   (b) Define a potential function Φ and use it to show that your implementations of these operations have the desired amortized cost.

   [Hint: Keep a pointer to the highest-order bit set to `1`.]