Problem 1. Consider the optimal BST problem on $S = \{1, 2, 3, 4\}$ and the weight array $W = (10, 20, 30, 40)$.

- Give the values of $\text{optcost}(a, b)$ for all $a, b$ satisfying $1 \leq a \leq b \leq 4$. Recall that $\text{optcost}(a, b)$ is the smallest average cost of all BSTs on $\{a, a + 1, \ldots, b\}$.
- Give the value of $\text{optcost}(1, 4 \mid 3)$. Recall that this is the smallest average cost of a BST on $\{1, 2, 3, 4\}$ on condition that 3 must be the root of the BST.
- Show an optimal BST on $S$ with the smallest average cost.

Problem 2. For the optimal BST problem, we have derived in the class $\text{optavg}(a, b)$ as follows:

$$\text{optavg}(a, b) = \begin{cases} 0 & \text{if } a > b \\ \sum_{i=a}^{b} W[i] + \min_{r=a}^{b} \{ \text{optavg}(a, r-1) + \text{optavg}(r+1, b) \} & \text{otherwise} \end{cases}$$

Give an algorithm to evaluate $\text{optavg}(1, n)$ in $O(n^3)$ time.

Problem 3. Continuing on the previous problem, although we are now able to compute $\text{optavg}(1, n)$, we have not constructed any optimal BST yet. Describe an algorithm to do so in $O(n^3)$ time.

Hint: For any such $a, b$ satisfying $1 \leq a \leq b \leq n$, define $\text{bestroot}(a, b)$ to be the $r \in [a, b]$ $\text{optavg}(a, r-1) + \text{optavg}(r+1, b)$.

Problem 4. Consider again the optimal BST problem on set $S = \{1, 2, \ldots, n\}$ and a weight array $W$, as defined in the class. Prof. Goofy proposes the following greedy algorithm for finding an optimal BST $T$:

- $r$ = the integer $i \in [1, n]$ with the largest $W[i]$.
- Make $r$ the root of $T$.
- Apply the above strategy to build a tree $T_1$ on $\{1, 2, \ldots, r-1\}$, and a tree $T_2$ on $\{r+1, r+2, \ldots, n\}$.
- Make the root of $T_1$ the left child of $r$, and the root of $T_2$ the right child of $r$.

Prove: the above algorithm does not always return an optimal BST.

Problem 5. Consider again the set $S = \{1, 2, \ldots, n\}$ and a weight array $W$ as in the optimal BST problem. This time, we want to find instead the most terrible BST: the one with the largest average cost. Give an algorithm to do so in $O(n^3)$ time.