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

Solution.

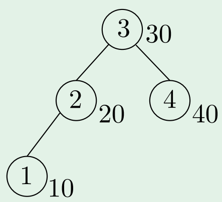
(a)

$optcost(a, b)$	$b = 1$	$b = 2$	$b = 3$	$b = 4$
$a = 1$	10	40	100	180
$a = 2$	0	20	70	150
$a = 3$	0	0	30	100
$a = 4$	0	0	0	40

(b)

$$optcost(1, 4|3) = \left(\sum_{i=1}^{4} W[i]\right) + optcost(1, 2) + optcost(4, 4) = 100 + 40 + 40 = 180.$$

(c) Optimal BST:



Problem 2. For the optimal BST problem, recall that

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

Give an algorithm to compute $optavg(1, n)$ in $O(n^3)$ time.

Solution. Maintain a look-up table storing the value all $optavg(a, b)$'s, initialized to infinity for all entries, except for the case where $a > b$ is set to zero.

Every $optavg(a, b)$ depends on $optavg(a, r-1)$ and $optavg(r+1, b)$ with $a \leq r \leq b$. We order the subproblems such that all the $f(a, b)$ where $1 \leq a \leq b \leq n$ and $b = a + i$ is calculated ahead of all the $f(a, b)$ satisfying $b = a + i + 1$. Additionally, we order the calculation of $optavg(a, a + i)$'s such that smaller values of a's come first. Namely, in round i ($i \in [0, n]$), all dependencies of $optavg(a, a + i)$ are resolved and the computation of $optavg(a, a + i)$ for all $1 \leq a \leq n$ only carries an additional cost of $O(n)$ due to the summation and min operations.

Problem 3. Describe an algorithm to construct an optimal BST in $O(n^3)$ time after computing $optavg(a, b)$ for all $1 \leq a \leq b \leq n$.

Solution. Define $bestroot(a, b)$ to be the $r \in [a, b]$ minimizing $optavg(a, r-1) + optavg(r+1, b)$. As all $optavg(a, b)$ have been computed, such r can be found in $O(n)$ time.

Compute $bestroot(a, b)$ for all $1 \leq a \leq b \leq n$ in $O(n^3)$ time. Using the piggyback technique, the optimal BST on $S = \{1, 2, \ldots, n\}$ is rooted on $r = bestroot(a, b)$ with left subtree as the optimal BST on $S_1 = \{1, 2, \ldots, r-1\}$ and the right subtree as the optimal BST on $S_2 = \{r+1, \ldots, n\}$, which can be obtained recursively.

We obtain the recursive formula for computing the optimal BST:

$$g(n) \leq g(r-1) + g(n-r) + 1,$$

which is evaluated to $O(n)$.

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

- Let r be 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.

Solution. Let $S = \{1, 2, 3, 4\}$ and $W = (10, 20, 30, 40)$.

$$cost = 40 + 30 \times 2 + 20 \times 3 + 10 \times 4 = 200 > optcost(1, 4) = 180.$$

Hence, the greedy strategy does not guarantee optimality.

Problem 5. Describe an algorithm to find the most terrible BST: the one with the largest average cost, in $O(n^3)$ time.

Solution.

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

Similarly, $optavg(a,b)$ for all $1 \le a \le b \le n$ can be computed in $O(n^3)$ time and the optimal BST can be constructed in $O(n^3)$ time using the piggyback technique.