1.

Prove that a postorder traversal run in time $\Theta(n)$ ***when the input is an n-vertex binary tree.***

To prove that a postorder traversal has a time complexity of $\Theta(n)$ when the input is an n-vertex binary tree, we can use a combination of upper and lower bounds.

First, we will establish the upper bound of $O(n)$. Since the algorithm visits each vertex exactly once and spends a constant amount of time at each vertex, the running time of the algorithm is proportional to the number of vertices in the binary tree, which is n. Therefore, the algorithm has an upper bound of $O(n)$. Next, we will establish the lower bound of $\Omega(n)$. Since the algorithm visits each vertex exactly once, it must take at least $\Omega(n)$ time to visit all n vertices in the worst case.

This observation is used to establish both the upper and lower bounds of a postorder traversal. Therefore, since we have established an upper bound of $O(n)$ and a lower bound of $\Omega(n)$, we can conclude that the postorder traversal algorithm has a time complexity of $\Theta(n)$ when the input is an n-vertex binary tree.

2.

To derive the Bellman equation for this problem, we need to define the subproblems and the optimal substructure property. Let $T(i, j)$ be the minimum cost of shipping the first i weeks, where the last j weeks are shipped by company B (assuming $j = 0$ means that no weeks are shipped by company B). The optimal substructure property holds because the optimal schedule for the first i weeks with the last j weeks shipped by company B is the concatenation of the optimal schedule for the first $i - j$ weeks and the optimal schedule for the last j weeks shipped by company B.

To derive the Bellman equation, we consider the two cases for the last j weeks: either they are shipped by company A, or they are shipped by company B. If they are shipped by company A, then the optimal cost is $T(i - 1, 0) + r * s_i$. If they are shipped by company B, then the optimal cost is $T(i - 1, 4) + r * s_i$. For j that are between 2 and 4, we add c to keep track of the cost for the unfulfilled company B shipment.

Therefore, the Bellman equation is:

$$T(i,j) = \{$$
$$\quad min(T(i-1,0) + r * s_i,\ T(i-1,4) + r * s_i),\ for\ 0 \le j \le 1$$
$$\quad T(i-1,\ j-1) + c \qquad\qquad\qquad\qquad\qquad\qquad ,\ for\ 2 \le j \le 4$$
$$\}$$

The base cases are

$$T(1,0) = r * s_1, T(1,1) = c, T(1,2) = T(1,3) = T(1,4) = inf.$$

The final answer is $min(T(n,0),\ T(n,4))$, which is the minimum cost of shipping all n weeks while the last week is shipped by company A or the 4th week by company B.

3.

The problem of finding the segmentation of a given string that maximizes the total quality of its constituent words can be formulated as a dynamic programming problem, where we construct an optimal solution bottom-up by solving subproblems and storing their solutions in a table. Let $T(i)$ be the maximum total quality that can be achieved for the first i letters of the string, and let $quality(p_1, p_2)$ be the quality of the substring $y[p_1 : p_2]$. Then we have:

$T(0) = -Inf$ (empty string has the lowest quality)
$T(i) = max(T(i-j) + quality(i-j, i)) \; for \; all \; j < i$

The first equation initializes the base case, where the maximum quality for an empty string is zero. The second equation computes the maximum quality for all possible substrings of the input string. It does this by considering all possible partitions of the substring $y[1: i]$ into two substrings $y[1: i-j]$ and $y[i-j+1: i]$, and taking the maximum of $max(T(i-j) + quality(i-j, i)) \; for \; all \; 1 \leq j < i$.

The optimal solution for the entire string y can be obtained as $T(n)$, where n is the length of the input string. We can also keep track of the positions where the maximum quality is achieved, and then reconstruct the optimal segmentation of the string using these positions.

To prove the correctness of this algorithm, we use mathematical induction. The base case $T(0) = 0$ is true by definition. For the inductive step, assume that $T(i-j)$ is the maximum total quality that can be achieved for the first i letters of the string. Then we need to show that $T(i)$ computed by the Bellman equation is also the maximum total quality that can be achieved for the first i letters of the string.

Consider any optimal segmentation of the string $y[1: i]$. Let k be the position of the last word in the segmentation. Then the maximum total quality for the first i letters can be obtained by adding the quality of the last word $y[k+1: i]$ to the maximum total quality for the first k letters $y[1: k]$. By the induction hypothesis, the maximum total quality for the first k letters is $T(k)$. Therefore, the maximum total quality for the first i letters is $T(k) + quality(k+1, i)$, which is the value computed by the Bellman equation.

The time complexity of this algorithm is $O(n^2)$, where n is the length of the input string, because we need to compute the quality of all possible substrings, and there are $O(n^2)$ substrings, each of which requires $O(n)$ time to compute the quality. However, this complexity can be reduced to $O(n^2)$ by precomputing the quality of all possible substrings in $O(n^2)$ time using dynamic programming, and then using these precomputed values in the Bellman equation.

4.

(a) Consider the case where K = 5 and the weights of containers are as follows: {1, 2, 3, 3}. The greedy algorithm will first load the containers {1, 2}, {3}, and then {3}. This will require three trucks. However, a better solution is to load the containers {1, 3} and {2, 3}, which only requires two trucks.

(b) Let the optimal solution use OPT trucks. We know that there will only be at most one truck less than half full since if there is more than one half-full truck we can get a better solution by loading two truck's containers into one truck. Hence, we can have that

$OPT <= 2 * \sum w_i/K + 1.$

For a greedy algorithm, we can not have two consecutive trucks that are less than half full since the loadings should be selected to the same truck by the rule. Hence, we can have the maximum trucks used by the greedy algorithm be $G \leq 2 * 2 * \sum w_i/K.$

By combine the two inequality we can have $G \leq 2 * 2 * \sum w_i/K < 2 * (2 * \sum w_i/K + 1)$ and hence $G < 2 * OPT$.