Part 1 – Package Prioritization: The greedy choice is always to select the consistent delivery that finishes earliest. Then, sort all deliveries by end time; afterward, pick the next non-overlapping one. This choice is optimal because finishing early maximizes the remaining time for future deliveries. The greedy choice ensures an optimal schedule that includes the earliest finishing activity, and the optimal substructure allows solving the remaining problems. Choosing the shortest duration/earliest start can fail a short delivery in the middle of the day, blocking two longer ones on either side, whereas an earliest finish never does.

Part 2 – Truck Loading: The greedy choice is to sort packages by descending weight ratio and take as many as possible until capacity is spent. Because fractions are allowed, the highest items should always be taken first. Any other ordering would leave higher value weights on the table. If fractions were disallowed, greedy could fail. An example of this would be a 10-pound item worth $60 and two 6-pound items worth $40 each.

Part 3 – Driver Assignment: The greedy choice is to sort deliveries by start time and assign each to the first free driver. This reuses drivers as soon as they become available, minimizing the total number required. This differs from package prioritization because package prioritization maximizes activities with a single fixed resource. The driver assignment minimizes the resources needed to cover all activities.

All three implementations use O(n log n).

1. For package prioritization, we use an O(n log n) sort followed by an O(n) linear scan.
2. For truck loading, it's an O(n log n) sort followed by an O(n) greedy fill.
3. For driver assignment, O(n log n) sort is also used.

The brute force alternatives are worse and far less practical:

1. Specific activity selection requires checking all 2^n subsets, or all n! orderings.
2. The 0/1 knapsack using DP is O(n·W), but can become huge for realistic weight values in large instances.
3. Optimal driver assignment by trying all possible partitions would be factorial in the number of overlapping intervals, making it completely intractable.

Ultimately, the greedy versions run in milliseconds even on the larger package size scenarios, while brute force approaches would take much longer to complete, potentially hours or more. This is exactly the kind of performance the CEO needs before deciding on whether to invest in an expensive third-party system, making sure that the in-house solution remains viable and cost-effective.