Quiz 3 Solution

Fatemeh Keshavarz

1.(a).
The greedy algorithm would be sorting *n* items regarding their shipping time (t_i) in ascending order.
Utilizing algorithms such as Merge Sort will cause the algorithm to run in O(*n.logn*).

(b).
My suggested algorithm sort items by increasing order of their t_i . So, consider the proposed order of the items to be shipped is: $I_1, I_2, \ldots, I_i, I_j, \ldots, I_n$ where $t_1 < t_2 < \ldots < t_i < t_j < \ldots < t_n$.
In this case, total delivery time is $D = \sum d_i = d_1 + d_2 + \ldots + d_i + d_j + \ldots + d_n = (t_1)+ (t_1 + t_2) + \ldots + (t_1 + t_2 + \ldots + t_i) + (t_1 + t_2 + \ldots + t_i + t_j) + \ldots + (t_1 + t_2 + \ldots + t_n)$.

Assume that there exists an optimal solution, which does not sort items as I do. It means that at least there is one inversion in the sequence it gives. Say, it gives this order: $I_1, I_2, \ldots, I_j, I_i, \ldots, I_n$ in which $t_j > t_i$.
Calculating D in this case, we have $D = \sum d_i = d_1 + d_2 + \ldots + d_j + d_i + \ldots + d_n = (t_1)+ (t_1 + t_2) + \ldots + (t_1 + t_2 + \ldots + t_j) + (t_1 + t_2 + \ldots + t_j + t_i) + \ldots + (t_1 + t_2 + \ldots + t_n) = nt_1 + (n-1)\, t_2 + \ldots + (n-i+1)\, t_j + (n-i)\, t_i + \ldots + t_n$.

Now, I want to swap I_i and I_j in the above sequence in order to make it similar to my algorithm's output and see if total delivery time will become worse or not.
Consequently, new total delivery time known as D′ would be:
$D' = \sum d_i = d_1 + d_2 + \ldots + d_i + d_j + \ldots + d_n = (t_1)+ (t_1 + t_2) + \ldots + (t_1 + t_2 + \ldots + t_i) + (t_1 + t_2 + \ldots + t_i + t_j) + \ldots + (t_1 + t_2 + \ldots + t_n) = nt_1 + (n-1)\, t_2 + \ldots + (n-i+1)\, t_i + (n-i)\, t_j + \ldots + t_n$.

Comparing D and D′, their difference is only for $(n-i+1)\, t_j + (n-i)\, t_i$ and $(n-i+1)\, t_i + (n-i)\, t_j$ respectively. Since we have $t_i < t_j$, so $(n-i+1)\, t_i + (n-i)\, t_j$ is smaller than $(n-i+1)\, t_j + (n-i)\, t_i$ which means $D' \leq D$. As can be seen, swapping did not increase the total delivery time. Thus, it just looks like my algorithm and is still optimal.
By expanding this approach in the optimal solution for any pair of items which are not in the same order as in my algorithm, I can transform the optimal algorithm to mine without jeopardizing the optimality.

Finally, this is a proof of optimality for the suggested algorithm.