Rui Yin
2018/10/9
HW4

1. To prove the correctness of 'if and only if', we show it from the two following aspects:
   a) If it is a feasible set, each package in the subset will cost one full day, so for all t ($t \leqslant n$), the number of packages in S due within t days is no more than t;
   b) If for all t ($t \leqslant n$), the number of packages in S due within t days is no more than t. Considering sorting the deliveries in increasing order of the due date. Then there exists a way to order the deliveries so that each one is made on time. Specifically, between the day that deliver the last one and at day t, make t-th delivery. So the set S is feasible.

2. According to 1, we know that a set of deliveries is infeasible once there exists some t ($t \leqslant n$), the number of deliveries in S due within t days is **GREATER** than t.

   A direct idea is that, if there is only one due date at date t, it can be made on time, and the maximum profit is itself. However, if there is not only one, the maximum can be made on time, and the remaining packages still have chances if their profits is larger than deliveries with earlier due date.

   This idea leads to the policy here. Sorting the deliveries in increasing order of the due date, we then decide which item to deliver from the last due date by a **GREEDY** policy. At each date t, we choose the item with maximum profit among the **remaining** packages which have a due date no smaller than t.

   A formal description is that, denoting the due dates: $t_1 \leqslant t_2 \leqslant \ldots \leqslant t_n$. Traverse date t from t_n back to 1 with a step -1, for each date t, choose the package by:

$$\text{argmax}\{p_i, p_{i+1}, \ldots, p_j\}$$
$$\text{s.t. } t_i, t_{i+1}, \ldots, t_j \geqslant t$$

   then remove the package because its delivery is finished.

   Correctness:

   (1) *feasibility*: by this way, the set of deliveries is initially feasible, and adding each delivery to the schedule remains the feasibility;

   (2) *optimal*: use an 'exchange' argument to prove that.

   Denote the solution generated by this schedule is $A=\{a_1, a_2, \ldots, a_k\}$, but the optimal solution is $O=\{o_1, o_2, \ldots, o_k\}$. Then there is at least an element of O not in A.

   Exchange the elements in O to make it more similar to A, because the other deliveries will be still on time, so the new solution O' is feasible too. And as our algorithm picks the package with maximal profits at each date, so the new solution will be not worse than O. If the arguments have polynomial differences, then exchange polynomial times can reduce all differences. And the new solution will always be as good as O. That is to say, A is as good as O. So A is optimal too.

   Time complexity:

The algorithm traverse all dates from t_n back to 1 (it is necessary to make each date's schedule) only once. At each date, if there is no new packages due, there is no need to sort the profits. So the algorithm is efficient.

In the worst case, at each date t, sorting the profits of size $n-(t_n-t)$, because one delivery per day is made on time from date t to the last due date t_n. And the whole running time is:

$$\sum_{t=1}^{t_n} n-(t_n-t) = nt_n + \frac{t_n(t_n+1)}{2} - t_n^2 = O(nt_n)$$