Yizhou Liu liu773@wisc.edu CS577 HW6.

1. **Definition** let $d(i, s)$ be max profit that can be achieved with packages $\{p_i, \dots p_n\}$, i represent package index, s denote day number.

Equation $d(i, s) = \max \begin{cases} d(i+1, s) & \text{// not deliever } P[i]. \\ d(i+1, s+t_i) + p_i \text{ if } s+t_i \le d_i & \text{// deliever } P[i]. \end{cases}$

Base Case: $d(n+1, s) = \phi$ $\forall s$. // no package to deliever for P_{n+1}

Algo:

```
sort p by di - ti in increasing order,
for s=1 to T.
    D[n+1, s] = 0
for s=1 to T
    for i = n to 1
        if s+ti > di
            M[i, s] = M[i+1, S]
        else
            M[i, s] = max ( M[i+1, S], M[i+1, s+ti] + pi ).
then M[1, 1] contains the max profit.
and use backtrack to rebuild the order of delievery.
```

Running Time: $O(nT)$.

Program Correctness:

Base Case: $i = n+1$, this mean we get no packages, so easy to prove that profit is 0.

Induction Hypothesis: The equations are correct for $i+1$.

Induction Before we analyze the 2 cases, we first prove that sorting is essential for the algo, because every package needs to be able to done on time in order to consider if deliever it or not.

Case I: not deliever $P[i]$

The remaining delievery is $P[i+1 \dots n]$ and profit is just $d(i+1, s)$

By induction, $d(i+1, s)$ is the max profit for $P[i+1 \dots n]$ starting in day

Case II: deliever $P[i]$.

It takes t_i days to deliever $P[i]$. so the next delivery starts on day $s+t_i$, and the profit is $d(i+1, s+t_i) + P_i$ So, by induction, $d(i+1, s+t_i)$ is the max profit for $P[i+1 \dots n]$ starting on day $s+t_i$

Conclusion $d(i, s)$ is optimal, since it's always the larger one

2 Definition: Let findOpt(i,j) be min days needed to deliever j packages in $\{P_1, \dots P_i\}$

Equation: $findOpt(i,j) = \min \begin{cases} findOpt(i-1, j) \\ findOpt(i-1, j-1) + t_i \text{ if quantity} \le d_i \end{cases}$

Base Case: $findOpt(i,j) = 0 \quad \forall i$

$findOpt(i,j) = \infty \quad \forall j > i$

Algo:

```
sort p by di-ti in increasing order.
for i=0 to n
    for j= n to i
        M[i,j]=∞
    M[i,0]=0.
for i=1 to n.
    for j=1 to i:
        We need to calculate M[i-1,j-1], M[i-1,j]
        if findOpt(i-1,j-1)+ti <= di:
            M[i,j]= min(findOpt(i-1,j-1), findOpt(i-1,j))
        else:
            M[i,j]= findOpt(i-1,j).
maxNumD=n. // initialize max num of delivery be n
while M[n,j] >T: // remove ones that exceed due date
    maxNumD--;
Return maxNumD;
```

We use backtrack to rebuild the order of the deliery

扫描全能王 创建

Program Correctness:

$\boxed{\text{Base Case}}$: $i=0$.

if $j=0$, then getOpt $(i,j)=\phi$ since it takes ϕ day to deliver ϕ package. if $j \neq \phi$, (ex: $j>i$), then getOpt $(i,j)=\infty$ since we can't deliver j packages out of ϕ packages

So, the equation is correct for base case.

$\boxed{\text{Induction Hypothesis}}$: Equation is correct for $i=0$ to $i-1-1$.

$\boxed{\text{Case I}}$: not deliver p[i].

We need to deliver j packages out of p[1... i-1]. And we need getOpt(i-1, j) days to do this. By IH, we get the min days.

$\boxed{\text{Case II}}$: deliver p[i].

Easy to get that we need getOpt(i-1, j-1) + t_i days to complete it. When quantity $\leq d_i$, by IH, this approach is feasible

$\boxed{\text{Conclusion}}$ getOpt(i, j) is optimal min delivery day cos it's always the smaller for the above 2 cases.

$\boxed{\text{Time Complexity}}$: $O(n^2)$