(40 points) Problem 15-7. (Manan Sanghi’s solution)

**Claim 1:** There exists an optimal schedule such that all the jobs in the schedule are in the increasing order of their deadlines.

**Lemma 1:** If \( S = < J_1, J_2, \ldots, J_k > \) is a valid* schedule and there exists \( J_i, J_{i+1} \) in \( S \) such that \( d_i > d_{i+1} \), then \( S' = < J_1, \ldots, J_{i-1}, J_{i+1}, J_i, J_{i+2}, \ldots, J_k > \) is a valid schedule.

Proof of Lemma 1: (left as an exercise)

**Lemma 2:** If \( S = < J_1, J_2, \ldots, J_k > \) is a valid schedule, then there exists a permutation \( \Pi \) such that \( S' = < J_{\Pi(1)}, J_{\Pi(2)}, \ldots, J_{\Pi(k)} > \) is a valid schedule and \( d_{\Pi(1)} \leq d_{\Pi(2)} \leq \ldots \leq d_{\Pi(k)} \).

Proof of Lemma 2: (left as an exercise)

(Hint: Use Lemma 1 and do something akin to bubble-sort)

Proof of Claim 1 follows from Lemma 1 and Lemma 2.

On the strength of Claim 1, all we need to do is to find an optimal schedule such that the jobs are processed in an increasing order of their deadlines. This reduces our search space to \( 2^n \) from \( n! \).

Now we can use Dynamic Programming to solve the problem in polynomial time.

Let \( W[j,k] \) be the maximum profit achievable by scheduling jobs \( (j, j+1, \ldots, n) \) starting at time \( k \). Note that the order of jobs is not the original \( (a_1, a_2, \ldots, a_n) \), the jobs are in the increasing order of their deadlines. Therefore, \( d_n \) is the maximum of all \( d_i \)'s.

\[
W[j,k] = \begin{cases} 
0 & \text{if } k \geq d_n \\
0 & \text{if } j > n \\
W[j+1,k] & \text{if } k > d_j - t_j \\
\max\{W[j+1,k], W[j+1,k+t_j] + P_j\} & \text{otherwise}
\end{cases}
\]

Using this we can fill a table of size \( n \times d_n \), since \( j \) varies from 1 to \( n \) and \( k \) from 1 to \( d_n \). The cell corresponding to \( W[1,1] \) will give us the answer.

So the running time will be \( O(n.d_n) \)