Suppose you have one machine and a set of *n* jobs $a_1, a_2, \ldots, a_n$ to process on that machine. Each job a_j has a processing time t_j , a profit p_j and a deadline d_j . The machine can only one job at a time, and job a_j must run uninterruptedly for t_j consecutive time units. If job a_j is completed by its deadline d_j, you receive a profit p_j, but if it's completed after its deadline, you receive a profit of 0. Give a dynamic programming algorithm to find the schedule that obtains the maximum amount of profit, assuming that all processing times are integers between 1 and *n*.

STEP 1 Characterize the structure of the optimal solution

We obtain the profit when adding a job into the sequence, using the following equations:

$$h_k\left(\sum_{i\in S} t_i + t_k\right) = \sum_{i\in S} p_i + p_k$$

where h_k defines if:

$$p_k = p_k \qquad if \sum_{i\in S} t_i + t_k \le d_k$$

$$p_k = 0 \qquad if \sum_{i\in S} t_i + t_k > d_k$$

Where k is the job that is currently being added to the schedule, and elements in S refers to each combination of jobs that have been already processed. t_i is the processing time of each job, and h_k is the function that determines if profit p_k is added or not. In other words, h_k defines if profit p_k remains with its original value (considering that the total processing times are completed by the deadline d_k), or if it turns to 0, (considering that the sum of all processing times is higher than the deadline d_k).

STEP 2 Recursively define the value of an optimal solution

Next, we obtained the recurrence relation, where if the size of the problem n is equal to 1, we would only have one way of scheduling the jobs e.g. $\left(p_1\right)$. However, if we have a size problem of $n \ge 2$ the recursive solution would be:

$$f[S] = max\left\{f[i] + h_k\left(\sum_{i\in S} t_i\right)\right\} \qquad \forall i \in C\ if\ k \ne C\ if\ n \ge 2$$

$$f[S] = p_k \qquad if\ n = 1$$

Where C are the possible combinations, considering that $length(S) - 1$ elements have been scheduled and k is currently being added. For instance, if $S = [a_1, a_2, a_3]$ then $C = [a_1, a_2],\ [a_1, a_3],\ [a_2, a_3]$; if we want to add a_1, then k = 1, and $f[C]$ would be equal to $f[a_2, a_3]$. This happens because, we would add the profit of scheduling a_1 to the already obtained profit of scheduling a_2 and a_3. Thus, the number of possible combinations C is equal to $\binom{S}{d}$ (where d is the $length(s) - 1$).

In addition, the maximum number of combinations is equal to $\binom{n}{l}$, where l is the current iteration number. For example, if we have a size problem $n = 4$ and we are developing the first iteration ($l = 1$), then the number of combinations would be $\binom{4}{1}$ which is equal to 4 (S: $\{s_1, s_2, s_3, s_4\}$). Where $s_1 = a_1$, $s_2 = a_2$, $s_3 = a_3$, $s_4 = a_4$, thus the evaluations done are for: $f[a_1]$, $f[a_2]$, $f[a_3]$, and $f[a_4]$.

To illustrate this better we will follow up in the case where we have developed the second iteration; thus, $l = 2$ and so the number of evaluations is $\binom{4}{2}$. This would result in 6 evaluations which would have the following form: $f[a_1, a_2]$, $f[a_1, a_3]$, $f[a_1, a_4]$, $f[a_2, a_3]$, $f[a_2, a_4]$ and $f[a_3, a_4]$.

STEP 3 Calculate the value of the optimal solution in a bottom-up way
Now we provide the solution for the problem, considering that we have the following data:

j	t_j	d_j	p_j
1	4	10	2
2	4	15	5
3	5	5	3
4	6	10	4

FIRST ITERATION

$f_1[a_1] = 2$ $\quad k[1] = 1$

$f_1[a_2] = 5$ $\quad k[2] = 2$

$f_1[a_3] = 3$ $\quad k[3] = 3$

$f_1[a_4] = 4$ $\quad k[4] = 4$

SECOND ITERATION

$$\begin{aligned}
f_2[a_1, a_2] &= max\{(f_1[a_1] + h_2(t_1 + t_2)); (f_1[a_2] + h_1(t_1 + t_2))\} \\
&= max\{(2 + 5); (5 + 2)\} \\
&= 7 \qquad k[1, 2] = 1, 2 \\
f_2[a_1, a_3] &= max\{(f_1[a_1] + h_3(t_1 + t_3)); (f_1[a_3] + h_1(t_1 + t_3))\} \\
&= max\{(2 + 0); (3 + 2)\} \\
&= max\{2; 5\} \\
&= 5 \qquad k[1, 3] = 1 \\
f_2[a_1, a_4] &= max\{(f_1[a_1] + h_4(t_1 + t_4)); (f_1[a_4] + h_1(t_1 + t_4))\} \\
&= max\{(2 + 4); (4 + 2)\} \\
&= max\{6; 6\}
\end{aligned}$$

$= 6 \qquad k[1,4] = 1,4$

$f_2[a_2, a_3] = max\{(f_1[a_2] + h_3(t_2 + t_3)); (f_1[a_3] + h_2(t_2 + t_3))\}$

$= max\{(5 + 0); (3 + 5)\}$

$= max\{5; 8\}$

$= 8 \qquad k[2,3] = 2$

$f_2[a_2, a_4] = max\{(f_1[a_2] + h_4(t_2 + t_4)); (f_1[a_4] + h_2(t_2 + t_4))\}$

$= max\{(5 + 4); (4 + 5)\}$

$= max\{9; 9\}$

$= 9 \qquad k[2,4] = 2,4$

$f_2[a_3, a_4] = max\{(f_1[a_3] + h_4(t_3 + t_4)); (f_1[a_4] + h_3(t_3 + t_4))\}$

$= max\{(3 + 0); (4 + 0)\}$

$= max\{3; 4\}$

$= 4 \qquad k[3,4] = 3$

THIRD ITERATION

$f_3[a_1, a_2, a_3] = max\{(f_2[a_1, a_2] + h_3(t_1 + t_2 + t_3)); (f_2[a_1, a_3] + h_2(t_1 + t_2 + t_3)); (f_2[a_2, a_3] + h_1(t_1 + t_2 + t_3))\}$

$= max\{(7 + 0); (5 + 5); (8 + 0)\}$

$= max\{7; 10; 8\}$

$= 10 \qquad k[1,2,3] = 2$

$f_3[a_1, a_2, a_4] = max\{(f_2[a_1, a_2] + h_4(t_1 + t_2 + t_4)); (f_2[a_1, a_4] + h_2(t_1 + t_2 + t_4)); (f_2[a_2, a_4] + h_1(t_1 + t_2 + t_4))\}$

$= max\{(7 + 0); (6 + 5); (9 + 0)\}$

$= max\{7; 11; 9\}$

$= 11 \qquad k[1,2,4] = 2$

$f_3[a_1, a_3, a_4] = max\{(f_2[a_1, a_3] + h_4(t_1 + t_3 + t_4)); (f_2[a_1, a_4] + h_3(t_1 + t_3 + t_4)); (f_2[a_3, a_4] + h_2(t_1 + t_3 + t_4))\}$

$= max\{(5 + 0); (6 + 0); (4 + 0)\}$

$= max\{5; 6; 4\}$

$= 6 \qquad k[1,3,4] = 3$

$f_3[a_2, a_3, a_4] = max\{(f_2[a_2, a_3] + h_4(t_2 + t_3 + t_4)); (f_2[a_2, a_4] + h_3(t_2 + t_3 + t_4)); (f_2[a_3, a_4] + h_2(t_2 + t_3 + t_4))\}$

$= max\{(8 + 0); (9 + 0); (4 + 5)\}$

$= max\{8; 9; 9\}$

$= 9 \qquad k[2,3,4] = 2,3$

FOURTH ITERATION

$f_4[a_1, a_2, a_3, a_4] = max\{(f_3[a_1, a_2, a_3] + h_4(t_1 + t_2 + t_3 + t_4)); (f_3[a_1, a_3, a_4] + h_2(t_1 + t_2 + t_3 + t_4));$

$(f_3[a_1, a_2, a_4] + h_3(t_1 + t_2 + t_3 + t_4)); (f_3[a_2, a_3, a_4] + h_1(t_1 + t_2 + t_3 + t_4))$

$= max\{(10 + 0); (6 + 0); (11 + 0); (9 + 0)\}$

$= max\{10; 6; 11; 9\}$

$= 11 \qquad k[1,2,3,4] = 3$

STEP 4 Construct the optimal solution with the information gathered.