Question #1: Scheduling unit jobs with penalties and deadlines

---

You are given n job the ith of which has deadline d_i and penalty p_i. Each job takes precisely one hour to finish and can not be interrupted once started. All deadlines are integers and $d_i \geq 1$. If you finish job i before its deadline that is great if you can't you will have to pay p_i no matter if/when you finish it. Also, all the deadlines are less than or equal to n (but you can assume that without loss of generality anyway, otherwise you can always complete that job because there will be sometime before $t = n$ when you are idle.)

Example: If you have $n = 4$ with $p_1 = 5, d_1 = 2$, $p_2 = 5, d_2 = 2$ $p_3 = 100, d_3 = 1$, $p_4 = 10, d_4 = 4$, then one possible optimal solution is $p_3\ p_2\ p_4$ with penalty 5.

Solution: Think about the time line as 0 to n and being divided to n one hour intervals. Originally the time line is empty, i.e. we haven't scheduled anything. Sort jobs according to decreasing penalty, then for each job if there is some time before its deadline when it could be done do it at the latest such time otherwise we will not do that job.

Run time: $\Theta(n^2)$ if we do the search for the last possible time trivially.

Proof of optimality: Define

$P(i) \equiv$ There is an optimal schedule that makes the same choices as the greedy algorithm up to and including job i.

Prove $P(i)$ by induction. $P(0)$ is trivial: It just says there is an optimal solution.
Assume that $P(i-1)$ holds; we will show $P(i)$ follows. Assume that O_{i-1} is the optimal solution that agrees with all the choices that greedy makes up to but not including on job i. This exists by the induction hypothesis $P(i-1)$.
There are two cases:

Case 1: If the choice greedy made was to not schedule job i it must be that just by making the decisions it made on the first $i-1$ jobs there is no place to schedule it before its deadline. But O_{i-1} agrees with the first $i-1$ choices of greedy so it can't finish job $i-1$ either and must pay its penalty. If O_{i-1} does schedule job i we just remove it from the schedule and call that O_i otherwise we just define $O_i = O_{i-1}$. In both cases O_i has the same penalty as O_{i-1} and is consistent with greedy's choices up to and including on the ith job so $P(i)$ is proved and we are done.

Case 2: If the choice greedy made was to schedule job i at time $[j-1, j]$. Then we have 3 cases depending on what the optimal solution O_{i-1} did with job i:

Case 2.1: O_{i-1} did not schedule job i or scheduled it after its deadline. In this case O_i will be exactly like O_{i-1} except that it schedules job i at time $[j-1, j]$ and if O_{i-1} scheduled some other job (call it i') at that interval O_i will not schedule that job. Notice that i' is a job that O_{i-1} is scheduling differently than the greedy algorithm so by induction hypothesis $i' \geq i$.

In this case penalty of O_i is either penalty of O_{i-1} minus p_i or penalty of O_{i-1} minus p_i plus $p_{i'}$. Given that $i' \geq i$ it must be that $p_{i'} \leq p_i$ so the penalty of O_i is less than or equal to penalty of O_{i-1} so O_i is also an optimal solution and given that it agrees with the greedy algorithm

Case 2.2: O_{i-1} scheduled job i before its deadline. Notice that because O_{i-1} agrees with greedy on the first $i-1$ jobs and by the way the greedy algorithm chooses the time j to schedule job i it must be that O_{i-1} has scheduled job j no later than the slot $[j-1, j]$. If O_{i-1} has scheduled job i in this interval we choose $O_i = O_{i-1}$ and are done. Otherwise, lets say that O_{i-1} has scheduled job i in the earlier slot $[j'-1, j']$. Let the job O_{i-1} schedules in the interval $[j-1, j]$ to be job i' (if it exists). We set O_i to be exactly like O_{i-1} except that we move job i to interval $[j-1, j]$ and job i' (if it exists) to earlier interval $[j'-1, j']$. Notice that the penalty of O_i is exactly the same as penalty of O_{i-1} because job i' has been moved earlier in the schedule so it can't contribute to the penalty of O_i unless it also contributes to the penalty of O_{i-1} and job i is move to some place before its deadline so it does not contribute to the penalty of O_{i-1}. So O_i must also be optimal and it agrees with the greedy algorithm on all i jobs so we have completed the proof.

If you had extra time: If you had extra time at the end talk about how you can implement this using disjoint sets in time $\Theta(n \log n)$. To do this one thinks of all the n possible time slots and makes a disjoint sets data structure on top of them. The idea is that if i and j are in the same set then the latest available time slot before $t = i$ and $t = j$ are the same. There will also be an array `latest_available[0...n]` with `latest_available[get_set(i)]` being the latest available time slot before i that is available. This way whenever the time slot $[i-1, i]$ gets filled we do a `join_set(i-1, i)` because now the latest available time slot before $i-1$ and i are the same and so is the latest available time slot before any place which was in the same set as i or $i-1$. And then we update the `latest_available[get_set(i)]` to the old value of `latest_available[get_set(i-1)]`. Overall:

Input: Integer n, penalties $p_1, \dots, p_n$ and deadlines $d_1, \dots, d_n$.
Output: Schedule with minimum penalty.

int *latest_available*$[0...n]$
`The 0th item is a dummy and will be 0 for ‘‘nothing available before that time’’`
for $i \leftarrow 0$ **to** n **do**
    latest_available$[i] \leftarrow i$
    `make_set`(i)
end
`Sort`(*Jobs according to decreasing* p_i*'s*)
for $i \leftarrow 1$ **to** n **do**
    $j \leftarrow$ *latest_available*[`get_set`(d_i)]
    if $j \neq 0$ **then**
        schedule job i at time interval $[j-1, j]$
        old_available $\leftarrow$ *latest_available*[`get_set`(*j-1*)]
        `join_set`$(j-1, j)$
        latest_available[`get_set`(j)] $\leftarrow$ *old_available*
    end
end
