IV003 Set 3, problem 1 20 points **due to 7. 5. 2025**

Name: Andrii Bezrukov **UČO:** 570943

0007 *list* 1 *učo* 570943 *body*

0123456789

Suppose you are given a set of n course assignments. All assignments take the same time to finish, so let's say each takes a unit of time to finish. The ith assignment has deadline d_i and reward r_i. You get the reward only if you finish the assignment within the deadline, otherwise the reward is zero. Design a *greedy* algorithm to find the maximum possible reward you can get. Prove the correctness of your algorithm.
The complexity of your algorithm should be in $\mathcal{O}(n \log n)$. To prove the correctness, follow a scheme from the lecture.

A natural greedy strategy is to **sort all assignments by their reward in descending order**, since we want to earn as many points as possible. For each assignment, we try to schedule it in the **latest available time slot that is still before or on its deadline**. This way, we always prioritize high-value work, while preserving earlier time slots for other tasks with tighter deadlines. This rule ensures that at every step we:

1. Choose the assignment with the highest available reward,
2. Schedule it as late as possible (but before the deadline), to leave earlier slots open for other work.

By doing so, we avoid missing deadlines and maximize total reward.

Observations

1. There exists an optimal schedule with no idle time.
2. The reward-first greedy schedule uses all available time slots efficiently.
3. Every scheduled assignment is completed on or before its deadline.

Proof of Correctness

Assume for contradiction that there exists a feasible schedule O whose total reward is greater than that of our greedy schedule S. Both S and O fill up available time slots without idle time, and every scheduled assignment finishes before its deadline.
Let us compare the two schedules. The greedy schedule S picks the highest-reward available assignment at each step and places it in the latest available slot that meets its deadline. If the schedule O chooses a lower-reward assignment in any time slot where a higher-reward assignment could have been scheduled (as in S), we can swap them. This does not violate any deadlines, and the total reward either increases or stays the same.
By repeatedly applying such swaps, we can transform O into the greedy schedule S without decreasing the total reward. This contradicts the assumption that O had a greater reward than S. Therefore, no schedule can produce a higher reward than the greedy one, proving that S is optimal.

Running Time Analysis

Sorting by descending reward takes $O(n \log n)$, and scheduling each task takes at most $O(n)$ total over all assignments. Therefore, the overall running time is:

$$\boxed{O(n \log n)}$$