Algorithms. Compulsory Assignments

We are given n jobs. The ith job has size s_i, deadline d_i, and value v_i. All these numbers are positive integers. Here, "size" means that it takes s_i time units to execute the ith job. Once a job has started, its execution must not be interrupted. The time intervals in which the jobs are executed must be pairwise disjoint (but the end of an interval may equal the start of the next interval). Not all jobs must be done, but if the ith job is executed at all, it must end before its deadline, that is, at time d_i at the very latest. Hence it must start at a time no later than $d_i - s_i$.

The problem is to select a subset of jobs and to schedule them, in such a way that all selected jobs are finished before their deadlines, and the sum of values v_i of the selected jobs is maximized. The schedule starts at time 0.

As a "playful" application example, imagine that you want to watch videos from a multimedia database. Each video has a known duration and a subjective value (measuring how important it is for you), but unfortunately they are provided only for limited periods and will be removed after their respective deadlines.

1. We claim that the special case when all deadlines are equal is equivalent to the Knapsack problem. Motivate this statement in detail. (Do not only say "it is obvious", but explain in which sense they are equivalent.)

2. Back to the general problem with different deadlines: Prove that there always exists an optimal solution such that the selected jobs are done in the order of their deadlines. (For any two selected jobs, the job with earlier deadline is scheduled earlier.) Hint: Consider any optimal solution and reorder the jobs by a careful exchange argument. A slight difficulty is that the jobs have, in general, different sizes.

3. Now the most heavy part: Use the property from point 2 to design a dynamic programming algorithm for the problem. Make sure that you provide all ingredients: First define an OPT function and explain its intended meaning, then specify how you compute this OPT function and an optimal solution, argue why your computation is correct, and analyze the time.