Another Application of Greed

CLRS Section 17.5: A task scheduling problem.

Input: - a set $S = \{1, 2, 3, \ldots, n\}$ of unit-time tasks

- a set of n integer deadlines $d_1, d_2, d_3, \ldots, d_n$

Such that $1 \leq d_i \leq n \quad \forall i, 1 \leq i \leq n$

(task i is supposed to be done by time d_i)

- a set of n non-negative weights or penalties

$w_1, w_2, w_3, \ldots, w_n$ such that penalty w_i

occurs if task i is not completed by d_i.

Output: a permutation L of the tasks that

minimizes $\sum_{\substack{L[j] = i \\ \text{and } j < d_i}} w_i$

[i.e. minimize the weights of jobs not completed by their deadline]

1 2 3 4 5 6 7

Eg:

task i	1	2	3	4	5	6	7
d_i	4	2	4	6	7	2	1
w_i	14	3	9	6	11	2	8

task i	7	6	2	1	3	4	5
d_i	1	2	2	4	4	6	7
w_i	8	2	3	14	9	6	11

task i	1	5	3	7	4	2	6
d_i	4	7	4	1	6	2	2
w_i	14	11	9	8	6	3	2

7	2	3	1		4	5

Problem: Come up with an algorithm that uses and prove that it always finds the min-cost schedule.

L: