

Assignment 1

Assignments from here on out are graded according to completion.

1. This problem concerns the maximal interval scheduling problem. Recall that in this problem we are given a list of intervals $I_1, \dots, I_n$ where each interval I_j has a start time s_j and an end time e_j , and want to find a maximal collection of intervals that are non-overlapping.

As we showed in class, the algorithm **EEF** that sorts by earliest end time first and sequentially picks intervals in that order that do not overlap with any intervals already chosen produces an optimal solution.

- (a) Consider the following instance:

Interval	Start Time	End Time
I_1	0	2
I_2	1	4
I_3	2	7
I_4	3	6
I_5	5	9
I_6	6	9
I_7	8	10
I_8	10	12
I_9	11	12
I_{10}	12	13

What collection is produced by **EEF** on this input?

Answer: **EEF** picks I_1, I_4, I_6, I_8 , and I_{10} , in that order.

- (b) Consider the following optimal collection: $I_1, I_3, I_7, I_9, I_{10}$. Trace through applying the exchange argument described in class, to arrive at the collection produced by **EEF**.

Answer: **OPT:** $I_1, I_3, I_7, I_9, I_{10}$

OPT': $I_1, I_4, I_7, I_9, I_{10}$

OPT'': $I_1, I_4, I_6, I_9, I_{10}$

OPT''': $I_1, I_4, I_6, I_8, I_{10}$

Note that at each step, each solution is still feasible (the intervals do not overlap), and that the number of intervals stays the same each time.

Assignment 1

2. This problem concerns the job scheduling with deadlines problem. Recall that in this problem we are given a list of jobs $j_1, \dots, j_n$ where each job j_i has an associated start time s_i , deadline d_i , and processing time p_i .

Consider the following algorithm **SSF**: first, sort the jobs by slack, where the slack of a job j_i is the amount of leeway in time available to schedule it: $d_i - s_i - p_i$. Schedule jobs as late as possible, in order of their slack (from least to greatest).

- (a) First, what would it mean for **SSF** to be incorrect? How could you convince someone that this algorithm is incorrect?

Answer: It means that there is some input on which **SSF** fails. I could convince someone by giving them input and showing that **SSF** fails on that input. In this case, that means finding an instance where **SSF** fails to find a feasible schedule, despite one existing.

- (b) Convince me that **SSF** is incorrect.

Answer: Consider the input:

	J_1	J_2
Start	0	1
End	4	4
Processing Time	3	1

A feasible schedule is possible:

0 – 1	1 – 2	2 – 3	3 – 4
J_1	J_1	J_1	J_2

On this input, J_1 has a slack of 1 and J_2 has a slack of 2. **SSF** fails because it tries to schedule J_1 as late as possible first (from 1-4), but now J_2 has no possible place available to schedule. Thus **SSF** fails on this input, showing that it is incorrect.