Exercise 1

Given is the following task set with an overall deadline $D = 10$.

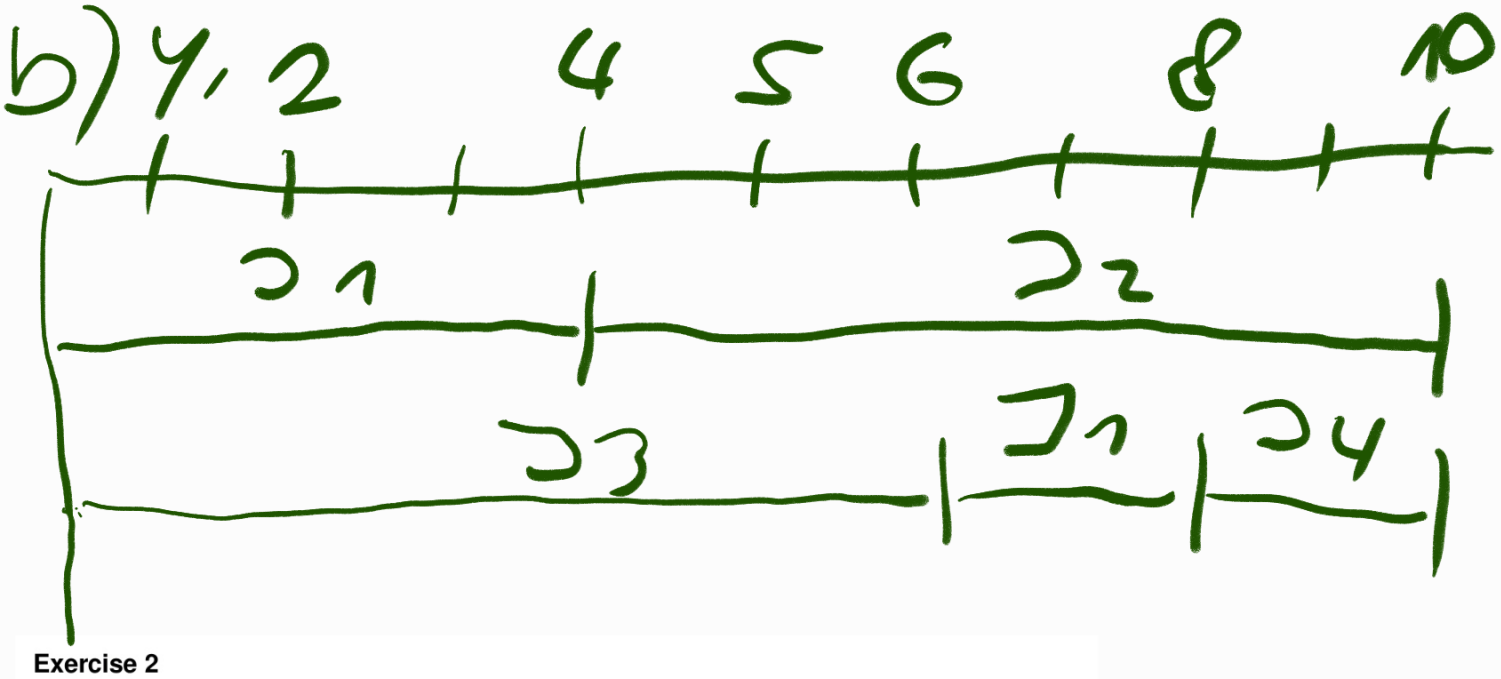
task i	arrival time a_i	deadline d_i	computation time C_i
J_1	0	10	6
J_2	0	10	6
J_3	0	10	6
J_4	0	10	2

a) Does a feasible non-preemptive schedule on a two processor architecture exist?

b) Does a feasible preemptive schedule on a two processor architecture exist?

a)

No, bc one of the processors will have to execute 2 jobs with execution time 6, as a non-preemptive schedule is searched.



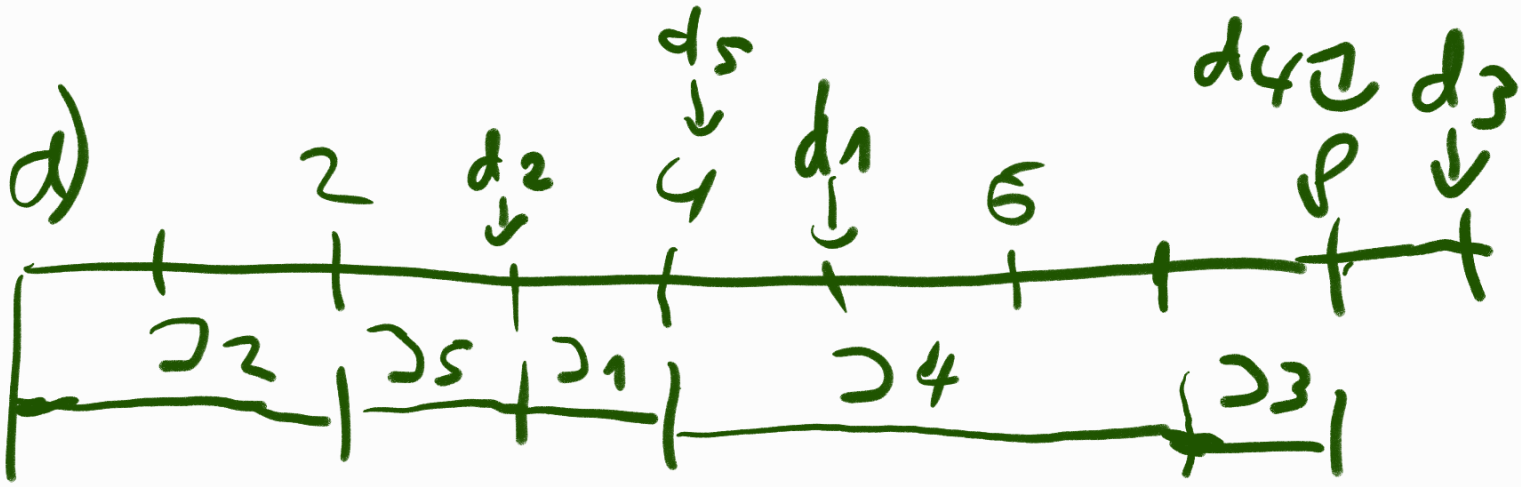
Exercise 2

Given is the following schedule.

task i	arrival time a_i	deadline d_i	computation time C_i
J_1	0	5	1
J_2	0	3	2
J_3	0	9	1
J_4	0	8	3
J_5	0	4	1

a) Schedule this task set so that the task with earliest deadline is scheduled first (without preemption).

b) Is the resulting schedule optimal regarding the maximum lateness?



b)

Theorem (Jackson '55):
Given a set of n independent tasks with synchronous arrival times, any algorithm that executes the tasks in order of non-decreasing deadlines is optimal with respect to minimizing the maximum lateness.

→ increasing deadlines

f.a. $J_i: L_i = f_i - d_i \leq -1$

Exercise 3

Given is the following schedule with five a-periodic tasks.

→ $L_{max} = -1$

task i	arrival time a_i	deadline d_i	computation time C_i
J_1	0	3	1
J_2	0	2	1
J_3	0	5	3
J_4	0	7	2
J_5	0	9	3

a) Check wether there exists a feasible schedule by conducting the test of schedulability with EDD from the lecture.

b) Why does schedulability/non-schedulability with EDD prove schedulability/non-schedulability in general for this particular set of tasks?

a) EDD-schedule: J_2, J_1, J_3, J_4, J_5

$\forall i = 1, \ldots 5: \sum_{k=1}^{i} C_k \leq d_i$: $1 \leq 2$, $1+1=2 \leq 3$

$1+1+3 = 5 \leq 5$, $1+1+3+2 = 7 \leq 7$,

$1+1+3+2+3 = 10 > 9$ → non-sched. by EDD and in general bc. optimal

b)

- If the conditions of the EDD algorithm are fulfilled, schedulability can be checked in the following way:
  - Sort task wrt. non-decreasing deadline. Let w.l.o.g. $J_1, \ldots, J_n$ be the sorted list.
  - Check whether in an EDD schedule $f_i \leq d_i \; \forall i = 1, \ldots, n$.
  - Since $f_i = \sum_{k=1}^{i} C_k$, we have to check:
    $\forall i = 1, \ldots, n: \sum_{k=1}^{i} C_k \leq d_i$
- (Since EDD is optimal, non-schedulability by EDD implies non-schedulability in general.)

→ max Lateness

→ optimal doesn't work already → then sth. less optimal won't work for sure

- A set of (a-periodic) tasks $\{J_1, \ldots, J_n\}$ with
  - arrival times $a_i = 0 \; \forall 1 \leq i \leq n$, i.e. "synchronous" arrival times
  - deadlines d_i,
  - computation times C_i
  - no precedence constraints, no resource constraints, i.e. "independent tasks"
- non-preemptive
- single processor
- Optimal
- Find schedule which minimizes maximum lateness (variant: find feasible solution)

Exercise 4

Given is the following schedule.

→ not same arrival times

→ EDD

task i	arrival time a_i	deadline d_i	computation time C_i
J_1	1	3	1
J_2	1	10	1
J_3	1	7	1
J_4	0	8	3
J_5	1	5	2

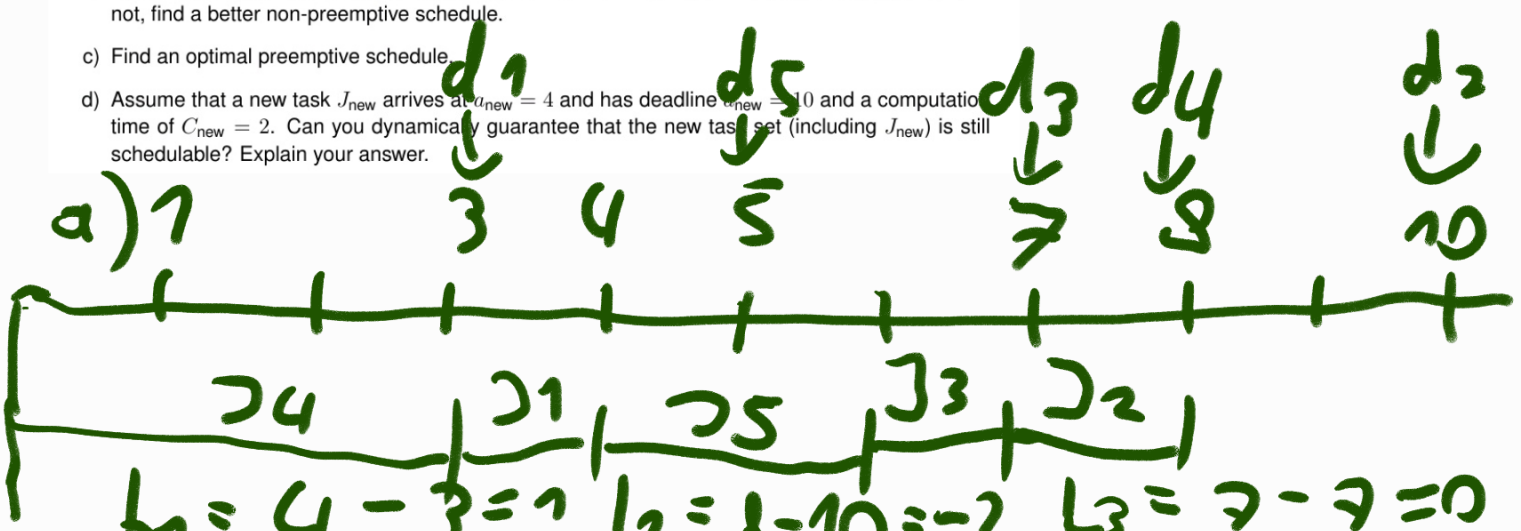
a) Schedule this task set so that the task with earliest deadline is scheduled first (without preemption and without using idle times, if there are tasks that are ready to be executed).

b) Is the resulting schedule optimal regarding the maximum lateness? If yes, prove its optimality. If not, find a better non-preemptive schedule.

c) Find an optimal preemptive schedule.

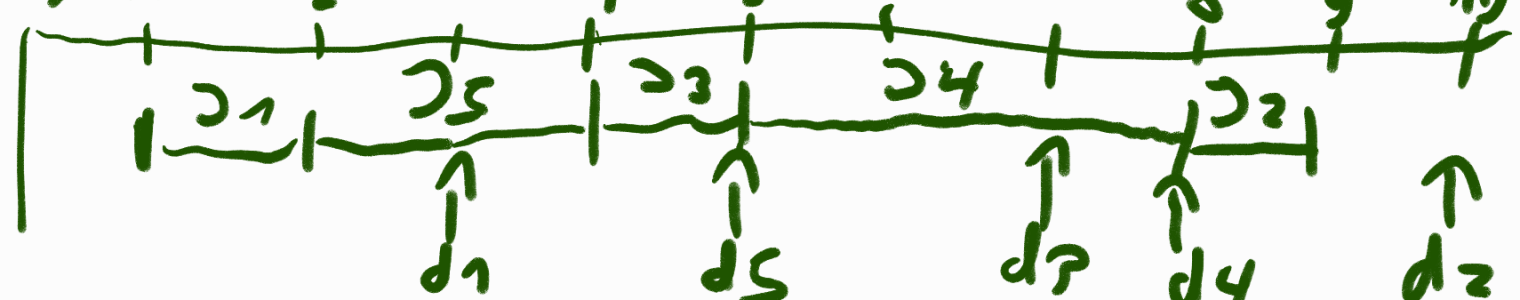
d) Assume that a new task J_{new} arrives at $a_{new} = 4$ and has deadline $d_{new} = 10$ and a computation time of $C_{new} = 2$. Can you dynamically guarantee that the new task set (including J_{new}) is still schedulable? Explain your answer.

a)



$L_1 = 4 - 3 = 1 \quad L_2 = 9 - 10 = -2 \quad L_3 = 7 - 7 = 0$

b) $L_4 = 3 - 8 = -5 \quad L_5 = 6 - 5 = 1 \rightarrow L_{max} = 1$



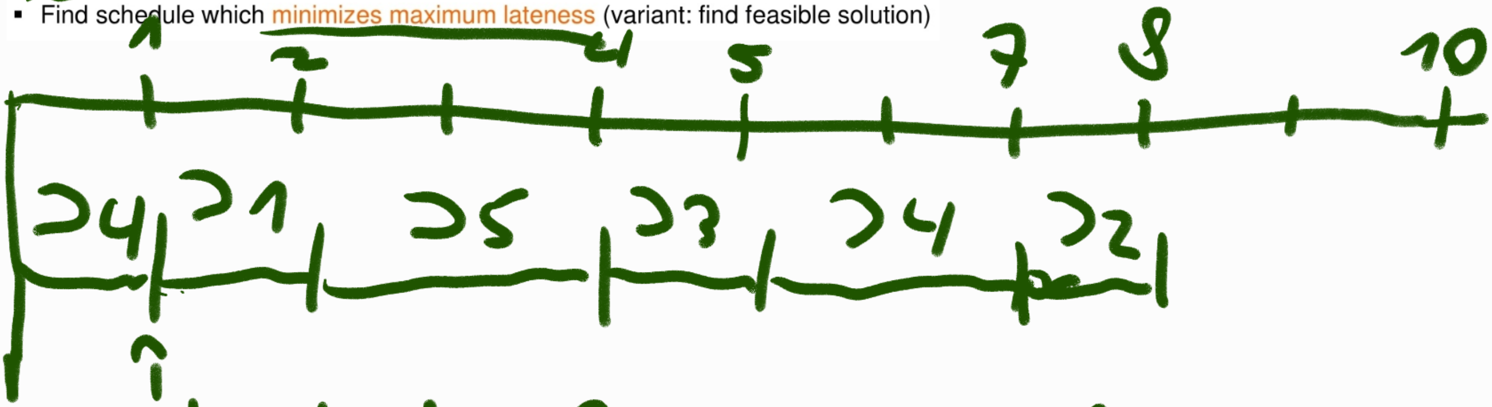
$L_1 = 2 - 3 = -1$ $L_2 = 9 - 10 = -1$

$L_3 = 5 - 7 = -2$ $L_4 = 8 - 8 = 0$ $L_5 = 4 - 5 = -1$

c) → not same arrival times → $L_{max} = 0$

→ not EDD (not optimal)

- A set of (a-periodic) tasks $\{J_1, \ldots, J_n\}$ with
  - **arbitrary** arrival times a_i
  - deadlines d_i,
  - computation times C_i
  - no precedence constraints, no resource constraints, i.e. "independent tasks" → Theorem Horn
- **preemptive** (this simplifies the problem!!!)
- Single processor
- Optimal
- Find schedule which minimizes maximum lateness (variant: find feasible solution)



d) all arrived → EDD order; anew 1 2 1 C_i (E) (3 - 1)

- Guarantee-based scheduling:
  - Let J be set of not yet finished tasks.
  - Whenever new task J_{new} enters the system: Guarantee test, whether $J' = J \cup \{J_{new}\}$ is still schedulable.
  - If EDF conditions are fulfilled:
    - Assume w.l.o.g. that tasks in J' are ordered by increasing deadlines.
    - When J_{new} arrives at time t, some tasks may be partially executed.
    - Let $c_i(t)$ be the remaining worst-case execution time of task J_i at time t.
    - Assume w.l.o.g. $J' = \{J_1, \ldots, J_n\}$ and $d_1 \le d_2 \le \ldots \le d_n$.
    - Worst-case finishing time of J_i: $f_i = t + \sum_{k=1}^{i} c_k(t)$.
    - Schedulability guaranteed, if
      $\forall$ i=1, …, n $f_i \le d_i$, i.e. if
      $\forall$ i=1, …, n $t + \sum_{k=1}^{i} c_k(t) \le d_i$.

    ⇒ One dynamic guarantee test can be performed in O(n).

    ⇒ EDF with n tasks and dynamic guarantee tests can be performed in $O(n^2)$.

$J_3: C_3 = 1 \le 3 = d'_3$ (7-4)

$J_4: C_3 + C_4 = 1 + 2 \le 4 = d'_4$ (8-4)

$J_2: C_3 + C_4 + C_2 = 1 + 2 + 1 \le 6 = d'_2$ (10-4)

$J_{new}: C_3 + C_4 + C_2 + C_{new} = 1 + 2 + 1 + 2 \le 6 = d'_{new}$ (10-4)

→ can guarantee schedulable

Theorem (Horn ’74):
Given a set of n independent tasks with arbitrary arrival times, any algorithm that at any instant executes the task with the earliest deadline among all the ready tasks is optimal with respect to minimizing the maximum lateness.