Task 1: EDD

Check whether the Earliest Due Date (EDD) algorithm produces a feasible schedule for the following task set (all tasks are synchronous and start at time $t = 0$):

<table>
<thead>
<tr>
<th></th>
<th>$J_1$</th>
<th>$J_2$</th>
<th>$J_3$</th>
<th>$J_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_i$</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>$D_i$</td>
<td>9</td>
<td>16</td>
<td>5</td>
<td>10</td>
</tr>
</tbody>
</table>

Solution - Task 1

EDD (Earliest Due Date) is a scheduling algorithm that minimizes the maximum lateness. The Jackson’s rule says that, given a set of $n$ independent tasks, any algorithm that executes the tasks in order of nondecreasing deadlines is optimal with respect to minimizing the maximum lateness.

The EDD produces a feasible schedule, given in the next figure.

![Figure 1: EDD schedule.](image)

The tasks ordered with respect to the nondecreasing order of the deadlines is as follows: $J_3, J_1, J_4, J_2$. 
Task 2: LDF

Given the precedence graph in Figure 2 and the following table of task execution times and deadlines, determine the Latest Deadline First (LDF) schedule. Is the schedule feasible?

<table>
<thead>
<tr>
<th></th>
<th>$J_1$</th>
<th>$J_2$</th>
<th>$J_3$</th>
<th>$J_4$</th>
<th>$J_5$</th>
<th>$J_6$</th>
<th>$J_7$</th>
<th>$J_8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_i$</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>$D_i$</td>
<td>6</td>
<td>8</td>
<td>10</td>
<td>14</td>
<td>12</td>
<td>15</td>
<td>18</td>
<td>20</td>
</tr>
</tbody>
</table>

Figure 2: Precedence graph.

Solution - Task 2

LDF (Latest Deadline First) is a scheduling algorithm that minimizes the maximum lateness (it assumes the synchronous tasks’ activations and preemptive scheduling).

The algorithm to produce an LDF schedule proceeds in two stages: firstly, a precedence graph is constructed. Going from tail to head: among the tasks without successors or whose successors have been all selected, LDF selects the tasks with the latest deadline to be scheduled last. At runtime, tasks are extracted from the head of the queue: the first task inserted in the queue will be executed last.

The queue created is as follows: $1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 4 \rightarrow 6 \rightarrow 7 \rightarrow 8$. The resulting schedule produced is presented below:

Figure 3: LDF schedule.
Task 3: EDF

Given are five tasks with arrival times, execution times and deadlines according to following table. Determine the Earliest Deadline First (EDF) schedule. Is the schedule feasible?

<table>
<thead>
<tr>
<th></th>
<th>J_1</th>
<th>J_2</th>
<th>J_3</th>
<th>J_4</th>
<th>J_5</th>
</tr>
</thead>
<tbody>
<tr>
<td>a_i</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>8</td>
<td>13</td>
</tr>
<tr>
<td>C_i</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>D_i</td>
<td>16</td>
<td>7</td>
<td>8</td>
<td>11</td>
<td>18</td>
</tr>
</tbody>
</table>

At time $t = 3$, a new task $J_x$ arrives with execution time $C_x = 2$ and deadline $D_x = 10$. Can you guarantee this new task or do you have to reject it?
Solution - task 3

EDF (Earliest Deadline First) is optimal in the sense of feasibility (minimizes the maximum lateness, is preemptive). The Horn’s rule says that given a set of \( n \) independent tasks with arbitrary arrival times any algorithm that at any instant executes the task with earliest absolute deadlines among the ready tasks is optimal with respect to the maximum lateness.

The EDF schedule is feasible, and the respective schedule is shown in the Figure 4.

![Figure 4: EDF schedule.](image1)

The arrival of a new task \( X \) at time point 3 still maintains the schedule feasible. The schedule now looks like:

![Figure 5: EDF schedule.](image2)

The \( a \) priori verification of the feasibility of the schedule, at the time point \( t = 3 \) is sketched in the following. The tasks ordered by their deadlines (nondecreasing), which execution time has not been completed is \( J_3, J_X, J_4, J_1 \) and \( J_5 \). For each of the tasks we check the condition, as described in the EDF-Guarantee algorithm (\( f_0 = 0 \)).

1. \( f_1 = f_0 + c_3(3) = 0 + 4 = 4 \ [4 \not\geq 8 = d_3] \)
2. \( f_2 = f_1 + c_X(3) = 4 + 2 = 6 \ [6 \not\geq 10 = d_X] \)
3. \( f_3 = f_2 + c_4(3) = 6 + 2 = 8 \ [8 \not\geq 11 = d_4] \)
4. \( f_4 = f_3 + c_1(3) = 8 + 3 = 11 \ [11 \not\geq 16 = d_1] \)
5. \( f_5 = f_4 + c_5(3) = 11 + 3 = 14 \ [14 \not\geq 18 = d_5] \)

The schedule is feasible.
Task 4: Aperiodic Scheduling with Precedence Constraints

Given are seven tasks $A, B, C, D, E, F, G$ with following precedence constraints:

\[ A \rightarrow C, \quad B \rightarrow C, \quad C \rightarrow E, \quad D \rightarrow F, \quad B \rightarrow D, \quad C \rightarrow F, \quad D \rightarrow G \]

All tasks arrive at time $t_0 = 0$, have deadlines $d_i = 20$ and following execution times:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_i$</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

Construct the precedence graph and modify arrival times and deadlines in a way such that EDF can be used to schedule the task set.
Solution - task 4

EDF* (with precedence constraints) with the concurrent activation can be solved polynomialy if tasks are preemptable. It minimizes the maximum lateness in the case of tasks with precedence constraints if the timing constraints are modified adequately.

There are several point to take into consideration, when attempting to modify the release times and the deadlines. They are summarized below:

modify the release times
1. task must start the execution not earlier than its release time
2. task must start the execution not earlier than the minimum finishing time of its predecessors

modify the deadlines
1. task must finish the execution time within its deadline
2. task must finish the execution not later than the maximum start time of its successors

The precedence graph is depicted:

![Precedence graph](image)

Figure 6: Precedence graph.

The modified released times and deadlines are

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r^*_i$</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>6</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>$d^*_i$</td>
<td>15</td>
<td>11</td>
<td>18</td>
<td>16</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
</tbody>
</table>
The algorithm to modify the release times proceeds as in the following:

1) for initial nodes \( \{A, B\} \) set \( r^*_A = r_A, r^*_B = r_B \)
2) select a task in \( \{C, D\} \)
   \[ r^*_C = \max\{r_c, \max\{(r^*_A + c_A), (r^*_B + c_B)\} = \max\{0, \max\{2, 3\}\} = 3 \]
   \[ r^*_D = \max\{0, 3\} = 3 \]
   \[ r^*_F = \max\{0, \max\{3 + 3, 3 + 4\}\} = 7 \]
   \[ r^*_E = \max\{0, 3 + 3\} = 6 \]
   \[ r^*_G = \max\{0, 3 + 4\} = 7 \]

The modification of the deadlines

1) start from the terminal nodes \( \{E, F, G\} \)
   \[ d^*_E = d^*_F = d^*_G = 20 \]
2) select one task from \( \{C, D\} \)
   \[ d^*_C = \min\{d_C, \min\{(d^*_E - c_E), (d^*_F - c_F)\}\} = \min\{20, 19, 18\} = 18 \]
   \[ d^*_D = 16 \]
   \[ d^*_A = 15 \]
   \[ d^*_B = 11 \]

The resulting EDF* schedule:

![Figure 7: EDF* schedule.](image-url)
**Task 5: EDF with Precedence Constraints**

Given is a set of 6 tasks named A, B, C, D, E and F with the following precedence relations:

\[
A \rightarrow B, \quad A \rightarrow C, \quad B \rightarrow D, \quad B \rightarrow E, \quad C \rightarrow F
\]

The execution times $C$, arrival times $r$ and absolute deadlines $d$ are as follows:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C$</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>$d$</td>
<td>4</td>
<td>10</td>
<td>8</td>
<td>6</td>
<td>10</td>
<td>12</td>
</tr>
<tr>
<td>$r$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>7</td>
<td>9</td>
</tr>
</tbody>
</table>

1. Draw the schedule in case of an EDF policy that ignores all precedence constraints.
2. Draw the schedule in case of an EDF policy that releases a task only after its predecessor has finished execution.
3. Determine the modified release times and deadlines for an EDF* policy, i.e. EDF with precedence constraints.
4. Draw the schedule for EDF* and the parameters determined in 3.
Solution - task 5

(1)

Figure 8: EDF schedule.

(2) the schedule is not feasible!
(3) The precedence graph and the modified release times and deadlines are

![Precedence graph](image)

**Figure 9: Precedence graph.**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r^*_i$</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td>$d^*_i$</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>6</td>
<td>10</td>
<td>12</td>
</tr>
</tbody>
</table>

(4)

![EDF* schedule](image)

**Figure 10: EDF* schedule.**