19.4
Scheduling to Minimize Lateness
Scheduling to Minimize Lateness

1. Given jobs $J_1, J_2, \ldots, J_n$ with deadlines and processing times to be scheduled on a single resource.

2. If a job $i$ starts at time $s_i$ then it will finish at time $f_i = s_i + t_i$, where $t_i$ is its processing time. $d_i$: deadline.

3. The lateness of a job is $\ell_i = \max(0, f_i - d_i)$.

4. Schedule all jobs such that $L = \max \ell_i$ is minimized.

<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>
</tr>
</thead>
<tbody>
<tr>
<td>$t_i$</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>$d_i$</td>
<td>6</td>
<td>8</td>
<td>9</td>
<td>9</td>
<td>14</td>
<td>15</td>
</tr>
</tbody>
</table>

$\ell_1 = 2$, $\ell_5 = 0$, $\ell_4 = 6$
Scheduling to Minimize Lateness

1. Given jobs $J_1, J_2, \ldots, J_n$ with deadlines and processing times to be scheduled on a single resource.

2. If a job $i$ starts at time $s_i$ then it will finish at time $f_i = s_i + t_i$, where $t_i$ is its processing time. $d_i$: deadline.

3. The lateness of a job is $\ell_i = \max(0, f_i - d_i)$.

4. Schedule all jobs such that $L = \max \ell_i$ is minimized.

<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>
</tr>
</thead>
<tbody>
<tr>
<td>$t_i$</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>$d_i$</td>
<td>6</td>
<td>8</td>
<td>9</td>
<td>9</td>
<td>14</td>
<td>15</td>
</tr>
</tbody>
</table>

\[ \ell_1 = 2 \quad \ell_5 = 0 \quad \ell_4 = 6 \]

- $J_3$: 0
- $J_2$: 1
- $J_6$: 2
- $J_1$: 3
- $J_5$: 4
- $J_4$: 5
- $J_1$: 6
- $J_5$: 7
- $J_4$: 8
- $J_1$: 9
- $J_5$: 10
- $J_4$: 11
- $J_1$: 12
- $J_5$: 13
- $J_4$: 14
- $J_4$: 15
Greedy Template

Initially \( R \) is the set of all requests
\[
\text{curr\_time} = 0 \\
\text{max\_lateness} = 0 \\
\text{while } R \text{ is not empty do} \\
\quad \text{choose } i \in R \\
\quad \text{curr\_time} = \text{curr\_time} + t_i \\
\quad \text{if } (\text{curr\_time} > d_i) \text{ then} \\
\quad \quad \text{max\_lateness} = \max(\text{curr\_time} - d_i, \text{max\_lateness}) \\
\text{return } \text{max\_lateness}
\]

Main task: Decide the order in which to process jobs in \( R \)
Greedy Template

Initially $R$ is the set of all requests
\[
\begin{align*}
\text{curr\_time} &= 0 \\
\text{max\_lateness} &= 0 \\
\text{while } R \text{ is not empty do} \\
\quad \text{choose } i \in R \\
\quad \text{curr\_time} &= \text{curr\_time} + t_i \\
\quad \text{if } (\text{curr\_time} > d_i) \text{ then} \\
\quad \quad \text{max\_lateness} &= \max(\text{curr\_time} - d_i, \text{max\_lateness}) \\
\end{align*}
\]

return $\text{max\_lateness}$

Main task: Decide the order in which to process jobs in $R$
Three Algorithms

1. Shortest job first — sort according to $t_i$.
2. Shortest slack first — sort according to $d_i - t_i$.
3. EDF = Earliest deadline first — sort according to $d_i$.

Counter examples for first two: exercise
Three Algorithms

1. Shortest job first — sort according to $t_i$.
2. Shortest slack first — sort according to $d_i - t_i$.
3. EDF = Earliest deadline first — sort according to $d_i$.

Counter examples for first two: exercise
Theorem 19.1.  
*Greedy with EDF rule minimizes maximum lateness.*

Proof via an exchange argument.

Idle time: time during which machine is not working.

Lemma 19.2.  
*If there is a feasible schedule then there is one with no idle time before all jobs are finished.*
Earliest Deadline First

**Theorem 19.1.**

*Greedy with EDF rule minimizes maximum lateness.*

Proof via an exchange argument.

Idle time: time during which machine is not working.

**Lemma 19.2.**

*If there is a feasible schedule then there is one with no idle time before all jobs are finished.*
Earliest Deadline First

**Theorem 19.1.**

*Greedy with EDF rule minimizes maximum lateness.*

Proof via an exchange argument.

Idle time: time during which machine is not working.

**Lemma 19.2.**

*If there is a feasible schedule then there is one with no idle time before all jobs are finished.*
Earliest Deadline First

**Theorem 19.1.**

*Greedy with EDF rule minimizes maximum lateness.*

Proof via an exchange argument.

Idle time: time during which machine is not working.

**Lemma 19.2.**

*If there is a feasible schedule then there is one with no idle time before all jobs are finished.*
Inversions

EDF = Earliest Deadline First

Assume jobs are sorted such that \( d_1 \leq d_2 \leq \ldots \leq d_n \). Hence EDF schedules them in this order.

**Definition 19.3.**

A schedule \( S \) is said to have an **inversion** if there are jobs \( i \) and \( j \) such that \( S \) schedules \( i \) before \( j \), but \( d_i > d_j \).

**Claim 19.4.**

If a schedule \( S \) has an inversion then there is an inversion between two adjacent scheduled jobs.

Proof: exercise.
Inversions

EDF = Earliest Deadline First

Assume jobs are sorted such that $d_1 \leq d_2 \leq \ldots \leq d_n$. Hence EDF schedules them in this order.

**Definition 19.3.**

A schedule $S$ is said to have an inversion if there are jobs $i$ and $j$ such that $S$ schedules $i$ before $j$, but $d_i > d_j$.

**Claim 19.4.**

*If a schedule $S$ has an inversion then there is an inversion between two adjacent scheduled jobs.*

Proof: exercise.
Proof sketch of Optimality of EDF

- Let $S$ be an optimum schedule with smallest number of inversions.
- If $S$ has no inversions then this is same as EDF and we are done.
- Else $S$ has two adjacent jobs $i$ and $j$ with $d_i > d_j$.
- Swap positions of $i$ and $j$ to obtain a new schedule $S'$

Claim 19.5.

Maximum lateness of $S'$ is no more than that of $S$. And $S'$ has strictly fewer inversions than $S$. 
THE END

...(for now)