Scheduling to minimize lateness. 

Input: \((t_i)_{i=1}^{n} (d_i)_{i=1}^{n}\) 
- \(t_i\): length of job \(i\) 
- \(d_i\): deadline of job \(i\)

Example: \(t_1=1\) \(t_2=2\) \(t_3=3\), \(d_1=3\) \(d_2=3\) \(d_3=6\).

Jobs are done at time \(f_i=1\) \(f_2=3\) \(f_3=6\).

Output: A schedule that minimize the maximum lateness. 

\[L = \max_i l_i\]
\[f_i = \begin{cases} 0 & \text{if } f_i = d_i \\ f_i - d_i & \text{o.w.} \end{cases}\]

Example 1: \(t_1=1\) \(d_1=100\) \(t_2=10\) \(d_2=10\).
\(\text{OPT} = (2, 1).\) \(L = 0.\)

Example 2: \(t_1=1\) \(d_1=2\) \(\text{slack}=1\). \(\text{Slack time} : d_i - t_i\).
\(t_2=10\) \(d_2=10\) \(\text{slack}=0\).
\(\text{OPT} = (1, 2).\) \(L = 1.\)
\[f_1=1\] \(f_2=11\)
\(\text{OPT} = (2, 1)\) \(f_1=11\) \(f_2=10\) \(L = 9\).

Greedy Rule: Schedule the job with earliest deadline first. 
(breaking ties arbitrarily).

Implementation: \(\Theta(n \log n)\). (See page 127-128 of textbook).

The bottleneck is in sorting the jobs by their deadlines.

Correctness:

Claim (4.7): There is an optimal schedule with no idle time.

In Example 1, \(2\) \(\text{wait}\) \(1\) \(L = 0\), eliminating the idle time can only make \(f_i\) smaller.

Theorem (4.19): There is an optimal schedule with no inversions.

Proof: Take any optimal solution \(O\). (If \(O\) has no idle time)

If \(O \neq \text{GREEDY}\), then there exist an inversion 
(see next page)
Inversion in this problem means job \( i \) is scheduled before job \( j \), (\( S_i < S_j \)) but job \( i \) is due after job \( i \) (\( d_i > d_j \)).

\[
\begin{array}{c|c|c|c|c}
0 & \cdots & \text{Job } i & \text{Job } j & \cdots \\
\uparrow & \uparrow & \uparrow & \uparrow & \uparrow \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
\hline
d_j & d_i & f_i & f_j & \text{Swap } i \text{ and } j
\end{array}
\]

\( S_i = \text{start time of job } i \) in \( O \)
\( f_i = \text{finish time} \) of \( i \)
\( L_i = \text{lateness} \)
\( L = \max L_i \)

\[
\begin{array}{c|c|c|c|c}
\overline{0} & \cdots & \text{Job } j & \text{Job } i & \cdots \\
\uparrow & \uparrow & \uparrow & \uparrow & \uparrow \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
\hline
\overline{d_j} & \overline{d_i} & \overline{f_j} & \overline{f_i} & \overline{S_i} = \text{start time of job } i \text{ in } O'
\end{array}
\]

Intuitively, both \( \overline{t_i} \) and \( \overline{t_j} \) are shorter than \( \overline{f_j} \).

Lemma: \( \overline{L} \leq L \) (If \( O \) is optimal, then \( \overline{O} \) must be optimal)

Proof:
\[
\begin{align*}
\overline{f_k} &= \overline{f_k} \quad \forall k \neq i,j \\
\overline{l_k} &= \overline{f_k} \quad \forall k \neq i,j \\
\overline{f_j} &\leq \overline{f_i} = \overline{f_j} \\
\Rightarrow \overline{t_j} &= \overline{f_j} - \overline{d_j} \leq \overline{f_j} - d_j = t_j \\
d_j &\leq d_i \\
\Rightarrow \overline{t_i} &= \overline{f_i} - \overline{d_i} = \overline{f_j} - d_i \leq \overline{f_j} - d_j = t_j
\end{align*}
\]

\[
\overline{L} = \max (\overline{t_i}, \overline{t_j}, \max \overline{l_k}) \leq \max (t_j, \max \overline{l_k}) \\
= \max (t_j, \max \overline{l_k}) \\
\leq \max (\overline{t_i}, t_j, \max \overline{l_k}) = L
\]