Chapter 4

Greedy Algorithms
4.1 Interval Scheduling
Interval Scheduling

Interval scheduling.
- Job j starts at $s_j$ and finishes at $f_j$.
- Two jobs compatible if they don't overlap.
- Goal: find maximum subset of mutually compatible jobs.
Interval Scheduling: Greedy Algorithms

**Greedy template.** Consider jobs in some natural order. Take each job provided it's compatible with the ones already taken.

- [Earliest start time] Consider jobs in ascending order of $s_j$.
- [Earliest finish time] Consider jobs in ascending order of $f_j$.
- [Shortest interval] Consider jobs in ascending order of $f_j - s_j$.
- [Fewest conflicts] For each job $j$, count the number of conflicting jobs $c_j$. Schedule in ascending order of $c_j$. 
Greedy template. Consider jobs in some natural order.
Take each job provided it's compatible with the ones already taken.

counterexample for earliest start time

counterexample for shortest interval

counterexample for fewest conflicts
Interval Scheduling: Greedy Algorithm

**Greedy algorithm.** Consider jobs in increasing order of finish time. Take each job provided it's compatible with the ones already taken.

\[
\text{Sort jobs by finish times so that } f_1 \leq f_2 \leq \ldots \leq f_n.\\
\text{set of jobs selected}\\
A \leftarrow \emptyset\\
\text{for } j = 1 \text{ to } n \{\\
\quad \text{if (job } j \text{ compatible with } A)\\
\quad \quad A \leftarrow A \cup \{j\}\\
\}\text{return } A
\]

**Implementation.** \(O(n \log n)\).
- Remember job \(j^*\) that was added last to \(A\).
- Job \(j\) is compatible with \(A\) if \(s_j \geq f_j^*\).
Theorem. Greedy algorithm is optimal.

Pf. (by contradiction)
- Assume greedy is not optimal, and let's see what happens.
- Let $i_1, i_2, \ldots, i_k$ denote set of jobs selected by greedy.
- Let $j_1, j_2, \ldots, j_m$ denote set of jobs in the optimal solution with $i_1 = j_1, i_2 = j_2, \ldots, i_r = j_r$ for the largest possible value of $r$. 

![Diagram of Greedy vs. Optimal Scheduling]

<table>
<thead>
<tr>
<th>Greedy:</th>
<th>OPT:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i_1$</td>
<td>$j_1$</td>
</tr>
<tr>
<td>$i_2$</td>
<td>$j_2$</td>
</tr>
<tr>
<td>$i_r$</td>
<td>$j_r$</td>
</tr>
<tr>
<td>$i_{r+1}$</td>
<td>$j_{r+1}$</td>
</tr>
</tbody>
</table>

Why not replace job $j_{r+1}$ with job $i_{r+1}$?
Theorem. Greedy algorithm is optimal.

Pf. (by contradiction)
- Assume greedy is not optimal, and let's see what happens.
- Let $i_1, i_2, \ldots, i_k$ denote set of jobs selected by greedy.
- Let $j_1, j_2, \ldots, j_m$ denote set of jobs in the optimal solution with $i_1 = j_1, i_2 = j_2, \ldots, i_r = j_r$ for the largest possible value of $r$.

**Diagram:**
- Greedy: $i_1, i_2, i_r, i_{r+1}$
- OPT: $j_1, j_2, j_r, i_{r+1}$

job $i_{r+1}$ finishes before $j_{r+1}$

solution still feasible and optimal, but contradicts maximality of $r$. 


4.2 Scheduling to Minimize Lateness
Scheduling to Minimizing Lateness

Minimizing lateness problem.

- Single resource processes one job at a time.
- Job $j$ requires $t_j$ units of processing time and is due at time $d_j$.
- If $j$ starts at time $s_j$, it finishes at time $f_j = s_j + t_j$.
- Lateness: $\ell_j = \max \{0, f_j - d_j\}$.
- Goal: schedule all jobs to minimize maximum lateness $L = \max \ell_j$.

Ex:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_j$</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>$d_j$</td>
<td>6</td>
<td>8</td>
<td>9</td>
<td>9</td>
<td>14</td>
<td>15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_3 = 9$</td>
<td>$d_2 = 8$</td>
<td>$d_6 = 15$</td>
<td>$d_1 = 6$</td>
<td>$d_5 = 14$</td>
<td>$d_4 = 9$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Lateness = 2   Lateness = 0   Max lateness = 6
Minimizing Lateness: Greedy Algorithms

**Greedy template.** Consider jobs in some order.

- **[Shortest processing time first]** Consider jobs in ascending order of processing time $t_j$.

- **[Earliest deadline first]** Consider jobs in ascending order of deadline $d_j$.

- **[Smallest slack]** Consider jobs in ascending order of slack $d_j - t_j$. 
Minimizing Lateness: Greedy Algorithms

**Greedy template.** Consider jobs in some order.

- **[Shortest processing time first]** Consider jobs in ascending order of processing time $t_j$.

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_j$</td>
<td>1</td>
</tr>
<tr>
<td>$d_j$</td>
<td>100</td>
</tr>
</tbody>
</table>

  counterexample

- **[Smallest slack]** Consider jobs in ascending order of slack $d_j - t_j$.

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_j$</td>
<td>1</td>
</tr>
<tr>
<td>$d_j$</td>
<td>2</td>
</tr>
</tbody>
</table>

  counterexample
**Minimizing Lateness: Greedy Algorithm**

*Greedy algorithm.* Earliest deadline first.

Sort $n$ jobs by deadline so that $d_1 \leq d_2 \leq \ldots \leq d_n$

$t \leftarrow 0$

for $j = 1$ to $n$

Assign job $j$ to interval $[t, t + t_j]$

$s_j \leftarrow t$, $f_j \leftarrow t + t_j$

$t \leftarrow t + t_j$

output intervals $[s_j, f_j]$

max lateness $= 1$
**Minimizing Lateness: No Idle Time**

*Observation.* There exists an optimal schedule with no *idle time.*

<table>
<thead>
<tr>
<th>d = 4</th>
<th>d = 6</th>
<th>d = 12</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2</td>
<td>3 4 5</td>
<td>6 7 8 9 10 11</td>
</tr>
</tbody>
</table>

*Observation.* The greedy schedule has no idle time.
**Def.** Given a schedule $S$, an **inversion** is a pair of jobs $i$ and $j$ such that: $i < j$ but $j$ scheduled before $i$.

![Diagram showing an inversion in a schedule]

*Observation.* Greedy schedule has no inversions.

*Observation.* If a schedule (with no idle time) has an inversion, it has one with a pair of inverted jobs scheduled consecutively.
Minimizing Lateness: Inversions

**Def.** Given a schedule \( S \), an inversion is a pair of jobs \( i \) and \( j \) such that: \( i < j \) but \( j \) scheduled before \( i \).

**Claim.** Swapping two consecutive, inverted jobs reduces the number of inversions by one and does not increase the max lateness.

**Pf.** Let \( \ell \) be the lateness before the swap, and let \( \ell' \) be it afterwards.

- \( \ell'_k = \ell_k \) for all \( k \neq i, j \)
- \( \ell'_i \leq \ell_i \)
- If job \( j \) is late:

\[
\ell'_j = f'_j - d_j \quad \text{(definition)}
\]
\[
= f_i - d_j \quad \text{\( (j \) finishes at time \( f_i \) )}
\]
\[
\leq f_i - d_i \quad \text{\( (i < j) \)}
\]
\[
\leq \ell_i \quad \text{(definition)}
\]
Theorem. Greedy schedule $S$ is optimal.

Pf. Define $S^*$ to be an optimal schedule that has the fewest number of inversions, and let's see what happens.

- Can assume $S^*$ has no idle time.
- If $S^*$ has no inversions, then $S = S^*$.
- If $S^*$ has an inversion, let $i-j$ be an adjacent inversion.
  - swapping $i$ and $j$ does not increase the maximum lateness and strictly decreases the number of inversions
  - this contradicts definition of $S^*$
Greedy Analysis Strategies

Greedy algorithm stays ahead. Show that after each step of the greedy algorithm, its solution is at least as good as any other algorithm's.

Exchange argument. Gradually transform any solution to the one found by the greedy algorithm without hurting its quality.

Other greedy algorithms. Kruskal, Prim, Dijkstra, Huffman, ...