LECTURE 8
JOB SCHEDULING
Job Scheduling

**Input:** Set of $n$ jobs, each with length $t_j$

- $J_1$: $t_1$
- $J_2$: $t_2$
- $J_n$: $t_n$

**Output:** A schedule of jobs on $m$ processors where $\sum_{j=1}^{n} C_j$ is minimized.

$C_j$ is the completion time of job $j$. 

**Ex:** $J_1$: $t_1$

- $J_2$: $t_2$
- $J_3$: $t_3$
- $J_n$: $t_n$

**Ex Schedule:** $S = [3, 4, 1, 5, 1]$

Cost of $S$: $3 + 4 + 5 + 10 = 26$
Q: What if you had 2 jobs:

\[ J_1: \begin{array}{c} 3 \\ J_2: \begin{array}{c} 5 \end{array} \end{array} \]

Intuition: shorter jobs have less impact on the completion times of future jobs.

Ex: On our example input:

\[ J_1: \begin{array}{c} 1 \\ J_2: \begin{array}{c} 1 \\ J_3: \begin{array}{c} 3 \end{array} \end{array} \]

1 + 2 \times 5 + 10 = 17.
Proof of Optimality:

For a schedule $S$, let $SEI$ be the $i$th job executed by $S$.

Let $S_g$ be the greedy schedule. Let

$$\text{sub-cost}(S_g) = \sum_{i=1}^{j} c_{s_i}$$

Thus by def $\text{cost}(S) = \text{sub-cost}(S, n)$

Claim: Take an arbitrary solution $S$:

$$\text{sub-cost}(S_g, j) \leq \text{sub-cost}(S, j) \text{ for all } j = 1 \ldots n$$

Proof by induction: Base case $j = 1$: true by definition.

$$\text{sub-cost}(S_g, 1) = \text{length}(S_g[1]) \leq \text{sub-cost}(S, 1)$$

length across all $n$ jobs.

Induction hypothesis: for $j = i$ the claim holds. Need to prove for $j = i+1$.

i.e. $\text{sub-cost}(S_g, k) \leq \text{sub-cost}(S, k)$
Need to show:
\[
\text{sub-cost} (S_g, k+1) \leq \text{sub-cost} (S_i, k+1),
\]
\[
\text{sub-cost} (S_g, k) + \sum_{i=1}^{k+1} \text{length} (S_g\Xi_i)
\]
by IH \quad ? \quad \leq k+1 \sum_{i=1}^{k+1} \text{length} (S_L\Xi_i)
true by greedy criteria
(i.e., this quantity \geq he sum of the shortest k+1 jobs in J)
\sum_{g} \text{cost} (S_g) \leq \text{cost} (S)
\Rightarrow S_g \text{ is optimal} \checkmark
\]
Job Scheduling 2: Now each job also has a weight

J1: $l=2, w=1 \Rightarrow 2 \times 1 = 2$

J2: $l=5, w=2 \Rightarrow 2 \times 5 = 10$

J3: $l=1, w=3$

J4: $l=1, w=1$

Output: a schedule that minimizes

$$\sum_{j=1}^{n} C_j$$

Ex:

\[ \begin{array}{ccc}
J_4 & J_3 & J_1 \\
1 & 3 & 1 \\
\end{array} \]

Cost: $1 \times 1 + 3 \times 2 + 1 \times 5 + 2 \times 10 = 32$
Q1: What if the weights were the same?
   ⇒ same as before, so shorter jobs first

Q2: What if lengths were the same.

\[
\begin{align*}
J_1: & \quad W = 1 \\
J_2: & \quad W = 2 \\
J_3: & \quad W = 5 \\
J_4: & \quad W = 3
\end{align*}
\]

\[
5 \quad 3 \quad 2 \quad 1
\]

\[
\begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array}
\]

\[
5 \times 1 + 3 \times 2 + 2 \times 3 + 1 \times 4 = 21
\]

⇒ higher weight jobs first.
Q3 What to do with mixed \( L \& W \)?

Intuition 1: J1 first if \( L \) shorter

Intuition 2: J2 first if \( L \) more important

General Strategy: combine \( L \& W \) into a single score that captures both intuitions and sort (in increasing order) by \( f \).

So combined scoring function \( f(L_i, W_i) \) should satisfy:

1. If weights are same \( \Rightarrow \) shorten length jobs get lower scores
2. If lengths \( U \) \( \Rightarrow \) higher weight \( U \) \( U \) \( U \) \( U \) \( U \) \( U \).
Possible combined Scoring Function: $f_2(l_i, w_i) = \frac{l_i}{w_i}$

Let's eliminate one of these factors on a simple example.

$S_1: \{l = 3, w_1\}$

$S_2: \{l = 5, w_2\}$

<table>
<thead>
<tr>
<th>$J_1$</th>
<th>$J_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Schedule $J_1: J_2$

Cost: $1 \times 3 + 2 \times 8 = 19$ (optimal)

$J_2: J_1$

Cost: $2 \times 5 + 1 \times 8 = 18$

X: suboptimal
Claim: Greedy sort by \( \frac{b_i}{w_i} \) is optimal.

\[ d = O(n \log n) \]

Proof of optimality (W/O an Exchange Argument)

Argue that an arbitrary schedule \( S \) can be transformed into

So step by step without getting worse.

Notational, let's rename jobs so that \( S_g = J_1 J_2 \ldots J_n \).

Change \( J_1 \) i.e. \( J_i \) is the lowest \( \frac{b_i}{w_i} \) score.

\( J_2 \) a a 2nd \( \frac{b_i}{w_i} \) score.
Let $S$ be any arbitrary schedule set $S \notin S_g$. Then it must be the case that in $S$, there is a job $J_k$ right after a job $J_i$ s.t. $i > k$.

Let's exchange $J_i$ & $J_k$.

Q: how does the cost of $S$ change after the change?
Recall cost of a schedule, \( \sum_{j=1}^{n} w_j C_j \)
$$\frac{\Delta \text{cost } J_i}{wi} \uparrow ? \leq \frac{\Delta \text{cost } J_k}{wk \downarrow}$$

$$wi \downarrow \text{ vs } wk \downarrow$$

$$\text{blc } k < i \text{ we know }$$

$$\frac{lk}{wk} \leq \frac{li}{wi} \Rightarrow \text{the cost increase is less than or equal to the cost decrease}$$

$$\Rightarrow \text{cost}(S') \leq \text{cost}(S)$$

\[ \Rightarrow \text{the exchange can only improve the cost of } S.\]
Therefore if we keep finding such "out of order consecutive pairs" and performed exchanges:

1. At some point we have to stop (i.e., there can no longer be out of order consecutive pairs).

2. \( \binom{n}{2} \) possible pairs we can swap, and once a pair \( Ji, Jk \) gets swapped they can never get swapped again.

\( \Rightarrow \) At some point the swaps must stop.

(2) We will eventually arrive at \( S_g \).
\[ \text{cost}(S_g) \leq \text{cost}(S) \]
\[ \implies S_g \text{ is the optimal schedule} \]