CS314 - Greedy Algorithms

Announcements

- HW due Friday
- Office hours tomorrow 9-10am
Scheduling to minimize lateness

Single resource + set of n requests for resource J (like last time)

But here, request i has: \( E_1, \ldots, E_n \) deadline \( d_i \), time \( t_i \) to run

(Think jobs on a computer requesting processor time)
Many different ways to optimize.

Here: Allowed to run past deadline, but want to minimize the maximum lateness.

Formally:

Assign $s_i$ to each $i$.

Let $f_i = s_i + t_i$.

Then lateness $l_i = f_i - d_i$.

Want to minimize $\max_i l_i$. 
Example:

Job 1  

Job 2

Job 3

Length 1

Length 2

Length 3

$d_1 = 2$

$d_2 = 4$

$d_3 = 6$

How can we schedule?

\[ l_1 = 2 \]
\[ l_2 = 2 \]
\[ l_3 = 0 \]
Greedy Strategies?

- Shortest to longest $t_i$:
  1: $t_1 = 6$  $d_1 = 6$
  2: $t_2 = 1$  $d_2 = 9$

- Shortest Slack Time $d_i - t_i$:
  1: $t_1 = 10$  $d_1 = 10$  (slack = 0)
  2: $t_2 = 1$  $d_2 = 2$
Instead - earliest deadline first \((EDF)\)

\[
\begin{align*}
\text{Sort by } d_i \text{ (reorder } t_i \text{ accordingly)} \\
f \leftarrow 0 \\
\text{for } i \leftarrow 1 \text{ to } n \\
\quad s[i] \leftarrow f \\
\quad f \leftarrow s[i] + f[i] \\
\text{return } S[1...n] \\
\end{align*}
\]
\(O(n)\)

Runtime? \(O(n \log n)\)

Note - no inverted pairs in our algorithm.
Proof of correctness:

Lemma: All schedules with no inversions and no idle time have same maximum lateness.

pf: Two schedules, both have no inversions and no idle time. Only possible difference is jobs with same deadline in different order. So consider all jobs w/ a deadline d. No matter of order last one finishes at same time. So lateness is same.
Lemma 2: There is an optimal schedule with no idle time.

(obvious)

Lemma: There is an optimal schedule with no inversions (≠ no idle time).

pf: Suppose optimal schedule \( S \) has inversions

\[
\begin{align*}
\text{i} & \quad \text{k} & \quad \text{l} & \quad \text{j} \\
\text{d}_i & > & \text{d}_j
\end{align*}
\]

Want to find an adjacent inversion
If there is inverted pair \( i \) \( j \), can step up job by job from \( i \) to \( j \).
If first adjacent pair along way, done.
If not, eventually get to \( j \) \( \Rightarrow \) must have reached adjacent inversion.

\[
\begin{align*}
&\text{Swap } a \leftrightarrow b; \text{ fewer inversions} \\
&\text{haven't changed any lateness} \\
&\text{besides } a \leftrightarrow b.
\end{align*}
\]

Change only \( f_a \) \( f_b \).

\( a \) finishes at "old" \( f_a \)
\( b \) finishes at time \( f_b \)
\[ f_b > f_a \]

Had: \[ f_a - d_a \quad d_a > d_b \]

Now: \[ f_b - d_a < f_a - d_a \]

Rewritten: a now finishes at \( f_b \) so its lateness is \( f_b - d_a \)
We had \( f_b - d_b \) and \( d_a > d_b \)

So a can't be more late in new schedule than b was in old.
Since \( f_b - d_a < f_b - d_b \)
End of argument:

Swapping inverted pairs can not hurt.

Take O and start swapping at most O(n^2) inverted pairs.

⇒ end with optimal schedule w/ no inverted pairs (no idle time)

apply lemma 1.