Greedy Algorithms

A greedy algorithm you all know:

Make change for $3.47:

- $2
- $1
- 25¢
- 10¢
- 1¢

7 coins

Claim: This is the min. no. of coins.

EX 1 (not easy) Prove that the greedy method of making change works for the Canadian coin system.

Does the greedy method work for every possible coin system?

1¢, 6¢, 7¢ coins. Make change for 12¢.

Greedy: 7¢ + 5×1¢ better 2×6¢

Claim: The greedy change algorithm can be implemented in polynomial time using quotients and remainders.
Interval Scheduling or "Activity selection" \[\text{[CLRS 16.1 but not easy to read]}\]

Given a set of activities, each with a specified time interval, select a maximum set of disjoint (=non-intersecting) intervals.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Math</td>
<td></td>
<td></td>
<td></td>
<td>Math, News</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Science</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Bike-Trip</td>
<td></td>
</tr>
</tbody>
</table>

\[\text{Diagram shows activities from 9 AM to 7 PM, with Math, Lunch, Math, News, Science, Soccer, Bike-Trip intervals.}\]
There are several possible greedy approaches

1. Select activity that starts earliest
   \[ \text{No.} \]

2. Select the shortest interval
   \[ \text{No.} \]

3. Select the interval with fewest conflicts
   \[ \text{No,} \]

4. Select the interval that ends earliest.
   For above we get math, seminar, lunch, science, soccer.

There is a slick way to implement this:

Sort activities \( 1 \ldots n \) by end time

\( A \leftarrow \emptyset \)

For \( i = 1 \ldots n \)

if activity \( i \) does not overlap with any activities in \( A \)

then \( A \leftarrow A \cup \{i\} \)

end.

Analysis: \( O(n \log n) \) to sort and \( O(n) \) for the loop. Thus \( O(n \log n) \)

Correctness: We will see two basic ways to show greedy algs. are correct:

1. Greedy stays ahead all the time
2. "exchange" proof.
Lemma: This alg. returns a max size set $A$ of disjoint intervals.

Proof: Let $A = \{a_1, \ldots, a_k\}$ be sorted by end time.

Compare to an optimum solution $B = \{b_1, \ldots\}$ be sorted by end time. Thus $l \geq k$ and we want to prove $l = k$.

Idea: At every step we can do better with the $a_i$'s.

Claim: $a_1, \ldots, a_i, b_i, \ldots$ be is an opt. soln.

Proof by induction

- Basis: $i = 1$. $a_1$ had earliest end time of all intervals so $end(a_1) \leq end(b_1)$.

  so replacing $b_1$ by $a_1$ gives disjoint intervals.

- Induction step: Suppose $a_1, \ldots, a_{i-1}, b_i, \ldots$ be is an opt. soln.

  $b_i$ does not intersect $a_{i-1}$ so the greedy alg. could have chosen it. Instead, it chose $a_i$ so

  $end(a_i) \leq end(b_i)$

  and replacing $b_i$ by $a_i$ leaves disjoint intervals.

This proves the claim. To finish proving the lemma:

If $k < l$ then $a_{k+1}, a_{k+2}, \ldots$ be is an opt. soln. But then the greedy alg. had more choices after $a_k$. 

Another example of greedy alg.
Scheduling to minimize lateness.

<table>
<thead>
<tr>
<th>assignments</th>
<th>time required</th>
<th>deadline</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS 341</td>
<td>4 hrs</td>
<td>in 9 hrs</td>
</tr>
<tr>
<td>Math</td>
<td>2 hrs</td>
<td>in 6 hrs</td>
</tr>
<tr>
<td>Philosophy</td>
<td>3 hrs</td>
<td>in 14 hrs</td>
</tr>
<tr>
<td>CS 350</td>
<td>10 hrs</td>
<td>in 25 hrs</td>
</tr>
</tbody>
</table>

Can you do everything by its deadline (ignoring sleep?)
How? (no parallel processing!)

Optimization Problem (more general)
find a schedule, allowing some jobs to be late
but minimizing the maximum lateness.

Note: this is different from minimizing sum of lateness
(= min average lateness) (can do this too.)

Why is the opt. problem more general? A schedule
completes all jobs on time iff its max lateness is 0.
Job i takes time ti and has deadline di.
Observation 1. You might as well finish a job once you start. This is at least as good: the other jobs finish earlier and job $i$ finishes at same time.

Thus each job should be done contiguously.

Observation 2. There’s never any value in taking a break. What are some greedy approaches?

- do short jobs first

\begin{tabular}{|l|c|c|}
\hline
No. & $d_1$ & $d_2$ \\
\hline
05. & 2 & 1 \\
\hline
\end{tabular}

[Well, we should take deadlines into account!]

- do jobs with less slack first, slack = $d_i - t_i$

\begin{tabular}{|l|c|c|}
\hline
No. & $d_2$ & $d_1$ \\
\hline
05. & 1 & 2 \\
\hline
\end{tabular}

- jobs in order of deadline.

i.e. order jobs s.t. $d_1 \leq d_2 \leq \ldots \leq d_n$ and do them in that order.

check that this works on above examples.
A general approach to proofs

Don't be general at first! Try special cases!

What is a good special case here?

\( n = 2 \quad d_1 < d_2 \)

- the \( \{ \text{wrong, other} \} \) solution, \( O \)
- the greedy solution, \( G \)

The greedy solution makes both jobs late

\[
\begin{align*}
    l_G &= \max \text{lateness of greedy schedule} = \max \{ l_G(1), l_G(2) \} \\
    l_O &= \max \text{lateness of other schedule} \\
    l_G(1) &\leq l_O(1) \text{ because we moved 1 earlier} \\
    l_G(2) &\leq l_O(1) \text{ because } d_1 \leq d_2
\end{align*}
\]

Therefore: \( l_G \leq l_O(1) \leq l_O \)

Can we generalize?

This idea allows us to swap a pair of consecutive jobs
if their deadlines are out of order, making the solution better (or at least not worse).

Next: a proof that greedy gives opt. soln using exchange proof.
Theorem. This greedy alg. gives an optimal solution, i.e. one that minimizes the maximum lateness.

Proof - an “exchange proof” that converts any solution to the greedy one without increasing max. lateness.

Let \(1, \ldots, n\) be ordering of jobs by greedy alg., i.e. \(d_1 \leq d_2 \leq \ldots \leq d_n\). Consider any other ordering.

There must be two jobs that are consecutive in this ordering but in wrong order for greedy: \(i, j\) with \(d_j \leq d_i\).

[Aside: We can sort by swapping consecutive-pairs]

1 5 3 4 2
1 3 5 4 2
1 3 5 2 4, etc.

How do you justify?

What improves at each step?

Consider swapping jobs \(i\) and \(j\).

\[
\begin{array}{c|c|c}
& i & j \\
\hline
i & 1 & \cancel{j} \\
\hline
\cancel{j} & i & \cancel{j}
\end{array}
\]

old \(l_N(j) \leq l_0(j)\) because \(d_j \leq d_i\) now we do \(j\) first

new \(l_N(i) \leq l_0(j)\) because \(d_j \leq d_i\)

And all other jobs have same lateness.

Thus \(l_N \leq l_0\)

So we can swap until we get the greedy order and \(l\) goes down or is unchanged.

Therefore the greedy solution is at least as good as any other.