Greedy Algorithms

A greedy algorithm you all know:

Make change for $3.47

1 * $2
1 * $1
1 * 25¢
2 * 10¢
2 * 1¢

7 coins

Claim This is the min. no. of coins.

EX1 (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" [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>9 AM</th>
<th>10</th>
<th>11</th>
<th>12</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>CS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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>

Greedy approach
- pick one activity greedily
- remove conflicts
- repeat
There are several possible greedy approaches

1. Select activity that starts earliest
   
   1. 

2. Select the shortest interval
   
   1. 

3. Select the interval with fewest conflicts
   
   1. 

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

   There is a slick way to implement this:

   Sort activities 1...n by end time
   
   \[ A = \emptyset \]

   for \( i = 1 \) to \( n \)

   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.
Here we use method 1.

**Lemma** This alg. returns a max size set $A$ of disjoint intervals.

**Proof** Let $A = \{a_1, \ldots, a_k\}$ 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** $a_1, a_2, \ldots b_i$ be is an opt. soly $\forall i$

**Proof by induction**

**Basis** $i = 1$. $a_1$ had earliest end time of all intervals so $\text{end}(a_1) \leq \text{end}(b_1)$

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

**Induction step** Suppose $a_1, a_2, \ldots a_{i-1}, b_i$ be is an opt. soly.

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

so $\text{end}(a_i) \leq \text{end}(b_i)$

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

This proves the claim. To finish proving the lemma:

If $k < l$ then $a_1, a_2, \ldots a_k, b_{k+1}, \ldots$ be is an opt. soly

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 version** (more general)
fined 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)

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 — not correct

- Do jobs with less slack first, slack = \( d_i - t_i \) (works above) — not correct

- Jobs in order of deadline.
  i.e. order jobs s.t. \( d_1 \leq d_2 \leq \cdots \leq d_n \) and do them in that order. Check that this works on above examples.

Next day: proof that this works.