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.

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?

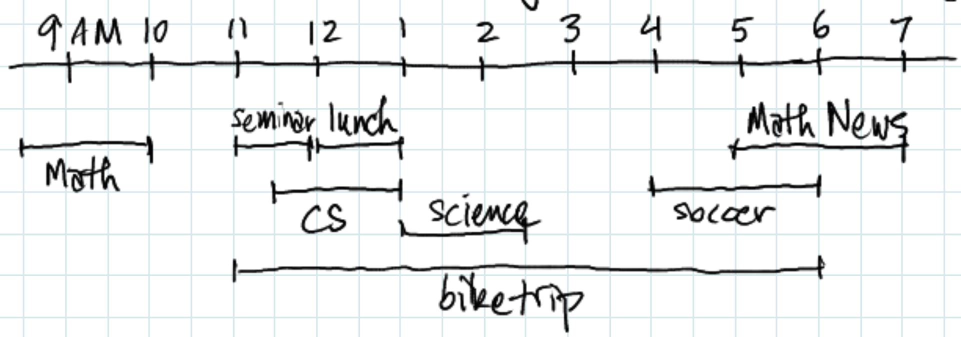
1¢ 6¢ 7¢ coins. Make change for 12¢

greedy: 7¢ + 5 × 1¢ better 2 × 6¢

Claim The greedy change algorithm can be implemented in polynomial time using quotients and remainders.

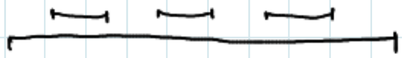
Interval Scheduling or "Activity selection" [CLRS 16.1 but not easy to read]

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

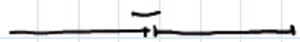


There are several possible greedy approaches

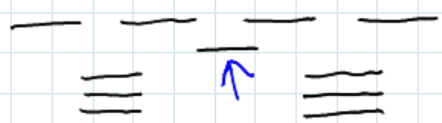
1. select activity that starts earliest

   No. 

2. select the shortest interval

   No. 

3. Select the interval with fewest conflicts

   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..n by end time
A ← ∅
for i = 1..n
    if activity i does not overlap with any activities in A
    then A ← A ∪ {i}                  just need to check last one
end.
```

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

Thus $O(n \log n)$

<u>Correctness</u> 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 $B = \{b_1 \cdots b_\ell\}$ ordered by end time.

Thus $\ell \geq k$ and we want to prove $\ell = k$.

Idea At every step we can do better with the a's.

Claim $a_1 \cdots a_i\ b_{i+1} \cdots b_\ell$ is an opt. soln $\forall i$

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 $a_1 \cdots a_{i-1}\ b_i \cdots b_\ell$ 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

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

and replacing b_i by a_i leaves disjoint intervals.

This proves the claim. To finish proving the lemma:
If $k < \ell$ then $a_1 \cdots a_k\ b_{k+1} \cdots b_\ell$ is an opt. soln
But then the greedy alg. had more choices after a_k.

Another example of greedy alg.

Scheduling to minimize lateness.

assignments	time required	deadline
CS 341	4 hrs	in 9 hrs
Math	2 hrs	in 6 hrs
Philosophy	3 hrs	in 14 hrs
CS 350	10 hrs	in 25 hrs

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

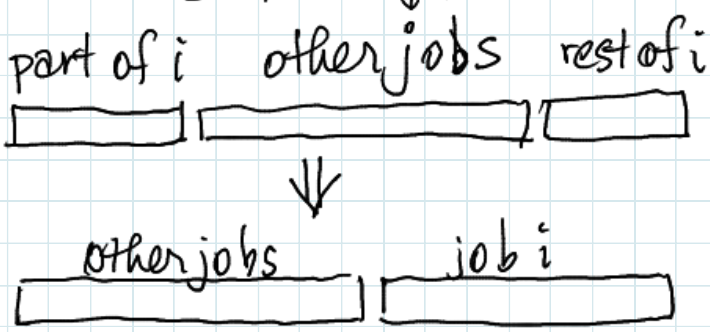
Optimization Problem (more general)
find a schedule, allowing some jobs to be late
but minimizing the maximum lateness.

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 t_i and has deadline d_i.

Observation 1. You might as well finish a job once you start.



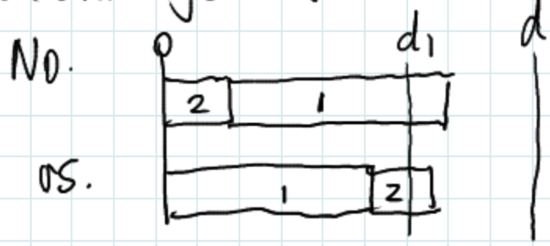
This is at least as good: the other jobs finish earlier and job i finishes at same time.

Thus each job should be done contiguously.

Observation 2 There's never any value in taking a break.

What are some greedy approaches?

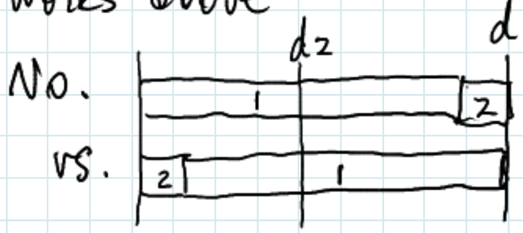
[
- do short jobs first



[Well, we should take deadlines into account!]

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

works above



]

- jobs in order of deadline.

  i.e. order jobs s.t. $d_1 \le d_2 \le \cdots d_n$

  and do them in that order.

Check that this works on above examples.

A general approach to proofs

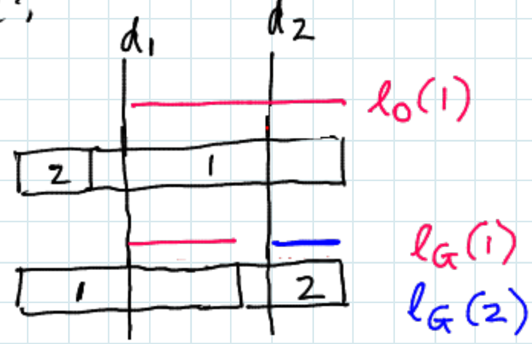
Don't be general at first! Try special cases!

What is a good special case here?

$n = 2 \quad d_1 < d_2$

the {wrong / other} solution, O

the greedy solution, G



the greedy solution makes both jobs late

l_G – max lateness of greedy schedule $= \max\{l_G(1), l_G(2)\}$

l_O – max lateness of other schedule $\qquad \le l_O(1) \quad \le l_O(1)$

$l_G(1) \le l_O(1)$ because we moved 1 earlier

$l_G(2) \le l_O(1)$ because $d_1 \le d_2$

Therefore $l_G \le l_O(1) \le l_O$

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, …, n be ordering of jobs by greedy alg., i.e. $d_1 \le d_2 \le \dots \le d_n$. Consider any other ordering.

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

[Aside: We can sort by swapping consecutive pairs

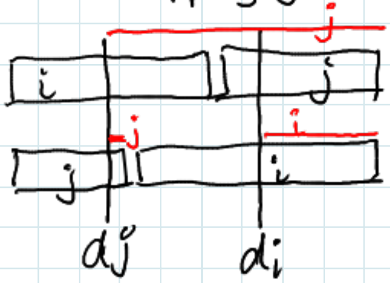
1 5 3 4 2

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.



old $\ell_N(j) \le \ell_O(j)$ because now we do j first

new $\ell_N(i) \le \ell_O(j)$ because $d_j \le d_i$

And all other jobs have same lateness.

Thus $\ell_N \le \ell_O$

So we can swap until we get the greedy soln and ℓ goes down or is unchanged.

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