CS314 - Greedy Algorithms

Note Title 9/18/2013

- Oral grading - done later today, passed back Monday
- Next HW - up later today

Dynamic Programming versus Greedy

- Try all possibilities, but intelligently.

- With greedy algorithms, we can avoid trying all possibilities.

  How?

  - Some part of structure lets us pick ~~a~~ local "best" + have it lead to global "best".

Sound suspiscious?

Greedy is dangerous!

- Often, students design a "greedy" strategy, but it doesn't yield the best solution.

<u>Example</u>: HW2, Q3

<u>The key</u>: Proof of correctness

Interval Scheduling

A maximal conflict-free schedule for a set of classes.

Goal: Select as many intervals as possible so that no two overlap.

Notation:

Input: Two arrays $S[1..n]$ and $F[1..n]$

Stat times   finish times

- So interval i starts at $S[i]$ & ends at $F[i]$.

Output: Largest subset $X \subseteq \{1..n\}$ s.t. $\forall i, j \in X$,

$F[i] < S[j]$ or $S[i] > F[j]$

Dynamic Programing

Recursive Structure?

i in X → remove all overlappings + recurse

i not in X → recurse w/out i

Run time: exponential?

Intuition for greedy strategy

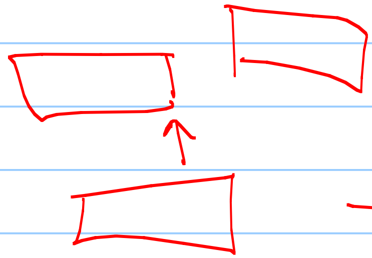
Consider first class we'd pick.

What would be a good choice?

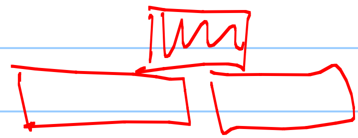
- earliest end time

$\Updownarrow$

latest start time



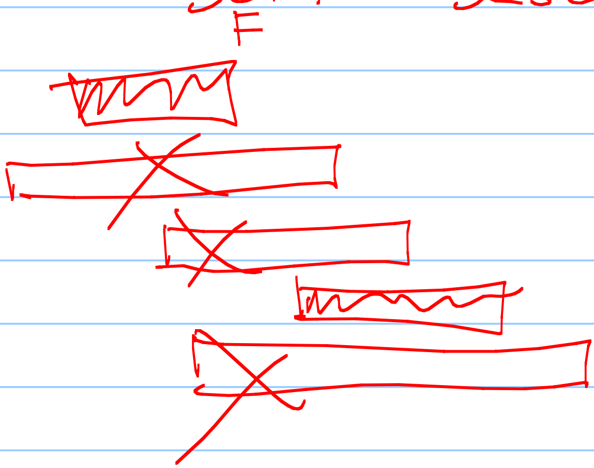
- smallest interval first



Idea: If it finishes as early as possible, that's good!

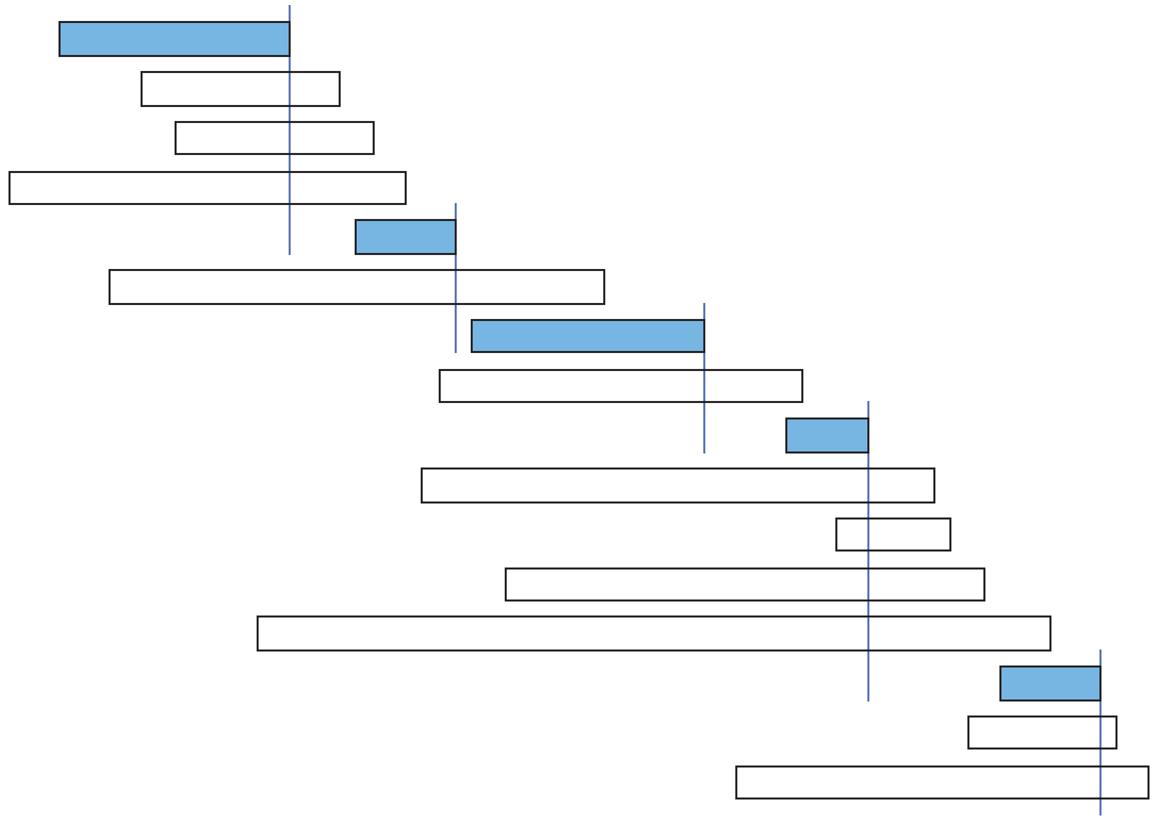
So strategy:

- Sort based on end times



Picture:

(same schedule, but sorted by F[i])



Pseudocode

Run time?

$\textsc{GreedySchedule}(S[1..n], F[1..n])$:
  sort F and permute S to match ← O(nlogn)
  $count \leftarrow 1$
  $X[count] \leftarrow 1$
  for $i \leftarrow 2$ to n
    if $S[i] > F[X[count]]$
      $count \leftarrow count + 1$
      $X[count] \leftarrow i$
  return $X[1..count]$

O(n)

$\boxed{O(n \log n)}$

<u>Correctness:</u>

But <u>why</u> does this work?

(Note: No longer trying all possibilities!)

So we <u>need</u> to be very careful on our proofs for greedy strategies.

Our intuition from before is the start of our proof...

Lemma: Can assume the optimal schedule contains the class that finishes first.

pf: Spps not: $\langle o_1, o_2, o_3, \ldots, o_k \rangle$ ← sorted "in order"

$\langle g_1, g_2, \ldots, g_\ell \rangle$

$F[g_1] \leq F[o_1]$ since g_1 was first thing to finish

also: $S[o_2] > F[o_1]$

$\Rightarrow F[g_1] < S[o_2]$

Since g_1 finishes before 2^{nd} element in optimal, $\langle g_1, o_2, \ldots, o_k \rangle$ is also valid.

Thm: The greedy schedule is optimal.

proof: Suppose not. So there's an optimal schedule with more intervals in it.

Look at first time they are different:

greedy schedule: $\langle g_1, g_2, \ldots, g_\ell \rangle$

optimal schedule: $\langle g_1, g_2, \ldots g_i, o_{i+1}, \ldots o_k \rangle$

Know: $F[o_{i+1}] \geq F[g_{i+1}]$

also: $S[o_{i+2}] > F[o_{i+1}]$ since it is a valid schedule pf: (cont) So replace o_{i+1} with g_{i+1}

OPT: $\langle g_1, g_2 \ldots, g_i, g_{i+1}, o_{i+2}, \ldots, o_k \rangle$

This works for any i!

Strategy for greedy proofs:

- Assume optimal solution is different from the greedy solution.
- Find the "first" place they differ.
- Argue that we can exchange the two without making optimal any worse

$\Rightarrow$ there is no "first place" they differ, so greedy is optimal.