CS 3100: Algorithms

Greedy Algorithms
Announcements

- Starting Ch 7 today
- Don't forget those problem session worksheets!
- Oral HW on Friday
Dynamic Programming vs Greedy
Dyn. pro: try all possibilities — but intelligently!

In greedy algorithms, we avoid building all possibilities.

How?
- Some part of the problem’s structure lets us pick a local “best” and have it lead to a global best.

But - be careful!
Most common mistake:

Students often design a greedy strategy, but don't check that it yields the best global one.

Example:

HW question 3

\( (a + b) \)
Problem: Interval Scheduling

Given a set of events (intervals with a start and end time), select as many as possible so that no two chosen will overlap.

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

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

where interval $i$ starts at $S[i]$ and ends at $F[i]$

**Output:** a set $I = \{ i_1, i_2, \ldots, i_k \}$

with $F[i_j] \leq S[i_{j+1}]$

$\forall i \in I$

maximizing $k$
How would we formalize a dynamic programming approach?

Recursive structure:

Consider interval 1:

- In or out

  - remove any overlapping intervals
  - then recurse

Remove any j s.t.

\[ S[i] \leq S[j] \leq F[i] \]

or

\[ S[i] \leq F[i-1] < F[i] \]
Intuition for greedy:
Consider what might be a good first one to choose.

Ideas?
- earliest start time
- shortest interval
- latest end time
- take smallest end time
Key intuition:
If it finishes as early as possible, we can fit more things in!

So - strategy:
Sort by finish time.
Select the first interval.
Remove any that overlap. Continue.
A maximal conflict-free schedule for a set of classes.

The same classes sorted by finish times and the greedy schedule.
Pseudocode

```
GREEDYSCHEDULE(S[1..n], F[1..n]):
sort F and permute S to match
  count ← 1
  X[count] ← 1
  for i ← 2 to n  
    if S[i] > F[X[count]]
      count ← count + 1
      X[count] ← i
  return X[1..count]
```

Runtime: \( O(n \log n) \)
Correctness:

Why does this work?

Note: No longer trying all possibilities or relying on optimal substructure!

So we need to be very careful on our proofs.

(Clearly, intuition can be wrong!)
Lemmas: We may assume the optimal schedule includes the class that finishes first.

pf: by contradiction
then Opt is some intervals:
\(< O_1, O_2, O_3 \ldots, O_k >\)
(sort so \( O_i \) finishes before \( O_{i+1} \) starts, + so on)
\[ \Rightarrow F[O_i] < S[O_{i+1}] \]
Consider \( g \), the interval that finished first:
\[ F[g] < F[O_i] \]
this means \( F[g] < S[O_i] \)
\( \forall i \geq 2 \)
so also optimal is:
\[ \langle g, O_2, \ldots, O_k \rangle \]
Thm: The greedy schedule is optimal.

pf: Suppose not. Then there exists an optimal schedule that has more intervals than the greedy one. Consider first time they differ:

\[
\text{greedy: } \langle g_1, g_2, g_3, \ldots, g_k \rangle \quad \text{vs.} \quad \langle g_1, g_2, \ldots, g_i, o_{i+1}, \ldots, o_k \rangle
\]

(same up to \(i\), \(i + 1\) then not)

(i exists \(i \geq 1\), by lemma)

\[\text{know: } F[o_{i+1}] > F[g_{i+1}]\]

since greedy

also, \(S[o_{i+2}] > F[o_{i+1}]\)

since \(0\) is opt schedule.
pf cont

So: we can replace $o_{i+1}$ with $g_{i+1}$ and still be valid.

OPT was: $<g_1, \ldots, g_i, o_{i+1}, \ldots, o_k>$

$\rightarrow$ $<g_1, g_2, \ldots, g_i, u, o_{i+2}, \ldots, o_k>$

is still valid.

contradiction $\n$
Overall greedy strategy:

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

⇒ there is no "first place" where they must differ, so greedy in fact is an optimal solution.

Another example in notes: storing the most files on a tape

Intuition: (check notes)