Announcements

- HW 3 Due Today
- Exam Next Friday
  - Basically the same rules as the last exam.
  - Complete exam instructions assignment on gradescope.
  - If you cannot make the usual time (and could last time), please email me by Wednesday.
  - Review video on course webpage.

Exam Topics

- Chapters 4 and 2
  - BFS
  - Dijkstra
    - Priority Queues
  - Bellman-Ford
  - Shortest Paths in DAGs
  - Karatsuba
  - Master Theorem
  - Merge Sort
  - Order Statistics
  - Binary Search
  - Closest Pair of Points

Greedy Algorithms (Ch 5)

- Basics
- Change making
- Interval scheduling
- Exchange arguments
- Optimal caching
- Huffman codes
- Minimal spanning trees

Greedy Algorithms

Often when trying to find the optimal solution to some problem you need to consider all your possible choices and how they might interact with other choices down the line.

But sometimes you don’t. Sometimes you can just take what looks like the best option for now and repeat.

Greedy Algorithms

General Algorithmic Technique:

1. Find decision criterion
2. Make best choice according to criterion
3. Repeat until done

Surprisingly, this sometimes works.

Example: Making Change

Problem: How do you make exact change for $12.83 using the fewest number of bills/coins?

Idea: Want to use biggest denominations possible.

Bills used:	Amount Left:
$10 $1 $1 ¢25 ¢25	$2.83 $1.83 $0.83 $0.58 $0.33
¢25 ¢5 ¢1 ¢1 ¢1	$0.08 $0.03 $0.02 $0.01 $0.00
10 bills/coins	This is the best possible!

Change Making

Note: This technique *always* works for making change when using American currency.

Note: It does not always work when using other currencies.

Example: Weirdtopia has $1, $5, $7 bills.

Problem: Make change for $10 in Weirdtopia.

Greedy:	Optimal:
$7+$1+$1+$1 = $10	$5+$5 = $10

Things to Keep in Mind about Greedy Algorithms

- Algorithms are very natural and easy to write down.
- However, not all greedy algorithms work.
- Proving correctness is important.

Interval Scheduling

Imagine that you are trying to schedule *as many* classes as possible without any conflicting lectures.

Problem: Given a collection C of intervals, find a subset $S \subseteq C$ so that:

1. No two intervals in S overlap.
2. Subject to (1), $|S|$ is as large as possible.

Question: Interval Scheduling

What is the greatest number of non-overlapping intervals that can be picked from the below?

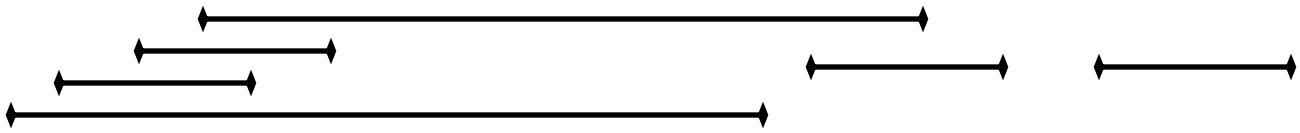
A) 0

B) 1

C) 2

D) 3

E) 4



First Interval

- Note: S must consist of intervals
  $J_1 = [x_1,y_1]$, $J_2 = [x_2,y_2]$, …
  where $y_i < x_{i+1}$.
- What is the best J_1?
  - Only way J_1 matters is that $y_1 < x_2$.
  - Want y_1 as small as possible.
- Once, we've picked J_1, have another interval cover problem among the intervals that don't overlap J_1.
- Algorithm: repeatedly pick non-overlapping interval with smallest max.

Algorithm

```
IntervalScheduling(C)
   S ← {}
   While(some interval in C
         doesn't overlap any in S)
     Let J be the non-overlapping
       interval with smallest max
     Add J to S
   Return S
```

O(n) iterations

O(n) time

Runtime: $O(n^2)$

Proof of Correctness

- Algorithm produces $J_1, J_2, \ldots, J_s$ with $J_i = [x_i, y_i]$.
- Consider some other solution
  $K_1, K_2, \ldots, K_t$ with $K_i = [w_i, z_i]$.

Claim: For each $m \leq t$, $y_m \leq z_m$.

In particular, $s \geq t$.

Proof of Claim

Use Induction on m.

Base Case: m = 1.

J_1 is the interval with smallest max, so $y_1 \leq z_1$.

Inductive Step: Assume $y_m \leq z_m$.

- J_{m+1} has smallest y for any [x,y] with $x > y_m$.
- $K_{m+1} = [w_{m+1}, z_{m+1}]$ has

$$w_{m+1} > z_m \geq y_m$$

- Therefore, $y_{m+1} \leq z_{m+1}$.

Optimization

- Original algorithm checks all intervals every time.
- Only need to consider intervals in increasing order of y.
- Sort once.

Optimized Algorithm

```
IntervalScheduling(C)
   Sort C by y-value
   S ← {}
   y_max ← -∞
   For J = [x,y] in C
      If x > y_max
         Add J to S
         y_max ← y
   Return S
```

O(n log(n))

O(n)

Runtime:
O(n log(n))

Question: Other Greedy Candidates

What are other possible candidate greedy decision procedures for interval scheduling?

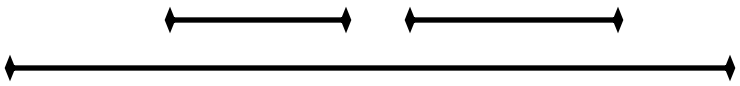
- Smallest max
- Shortest
- Fewest overlaps
- Largest min
- Smallest min
- Largest max

Only these work!

Smallest Min

Greedy

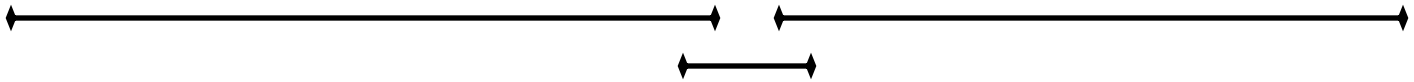
Optimal



Shortest

Greedy

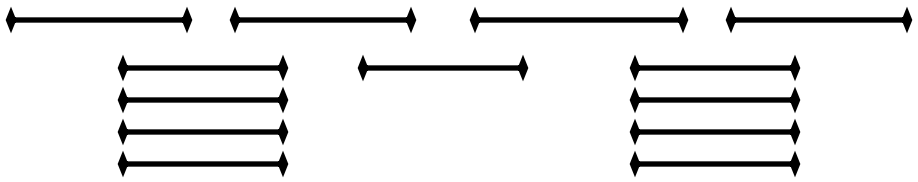
Optimal



Fewest Overlaps

Greedy

Optimal



Proofs

As it is very easy to write down plausible greedy algorithms for problems, but more difficult to find correct ones, it is very important to be able to *prove* that your algorithm is correct.

Fortunately, there is a standard proof technique for greedy algorithms.

Exchange Argument

- Greedy algorithm makes a sequence of decisions D_1, D_2, D_3,…,D_n eventually reaching solution G.
- Need to show that for arbitrary solutions A that G ≥ A.
- Find sequence of solutions
  $A=A_0, A_1, A_2,\ldots,A_n = G$
  so that:
  - $A_i \leq A_{i+1}$
  - A_i agrees with $D_1,D_2,\ldots,D_i$

Exchange Argument

In particular, we need to show that given any A_i consistent with $D_1,\ldots,D_i$ we can find an A_{i+1} so that:

- A_{i+1} is consistent with $D_1,\ldots,D_{i+1}$
- $A_{i+1} \geq A_i$

Then we inductively construct sequence

$A=A_0 \leq A_1 \leq A_2 \leq \ldots \leq A_n = G$

Thus, $G \geq A$ for any A. So G is optimal.

Exchange Argument for Interval Packing

- What decisions does greedy algorithm make?
  - D_i, i^{th} interval equals J_i.
- Need to show that IF we have a solution that agrees with first i decisions, can make it agree with i+1 without making it worse.
- Have solution: $J_1,J_2,\ldots,J_i,K_{i+1},\ldots,K_m$
  - Need to modify to use interval J_{i+1}.

Inductive Step

Greedy solution: $J_1, J_2, \ldots, J_n$ $\quad J_i = [x_i, y_i]$

Current solution: $J_1, J_2, \ldots, J_i, K_{i+1}, \ldots, K_m$ $\quad K_i = [w_i, z_i]$

Claim: $J_1, J_2, \ldots, J_i, J_{i+1}, K_{i+2}, \ldots, K_m$ is a valid solution.

Proof: Need to verify:

- $x_{i+1} > y_i$: This is because J_i, J_{i+1} don't overlap.
- $w_{i+2} > y_{i+1}$: This is because $w_{i+2} > z_{i+1} \geq y_{i+1}$.

Example

Greedy Solution

Arbitrary Solution