Algorithmic Strategies

- Divide and Conquer
- Greedy
- Dynamic Programming
- Brute Force
- Reduction
- Randomization

Asymptotic Analysis

- Focus on the behavior of algorithms as the problem size approaches infinity.

- $f(n) \in O(g(n))$ Asymptotic Upper Bound

  $\exists n_0, c > 0 \quad f(n) \leq c\, g(n) \quad \forall n \geq n_0$

- $f(n) \in \Omega(g(n))$ Asymptotic Lower Bound

  $\exists n_0, c > 0 \quad c\, g(n) \leq f(n) \quad \forall n \geq n_0$

- $f(n) \in \Theta(g(n))$ Asymptotic ~~Bound~~ Tight Bound

  $f(n) \in O(g(n))$ and $f(n) \in \Omega(g(n))$

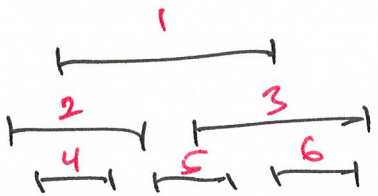
Rates of Growth

Starting with input of size 10^6 processed in one hour. If I double the speed of the machine how large of an input can I solve?

Algorithm Speed	Old Machine	New Machine
$\lg n$	10^6	$\approx 1{,}100{,}000{,}000{,}000$
$\sqrt{n}$	10^6	4,000,000
n	10^6	2,000,000
$n \lg n$	10^6	$\approx 1{,}920{,}000$
n^2	10^6	$\approx 1{,}414{,}213$
2^n	10^6	1,000,001
$n!$	10^6	1,000,000

Interval Scheduling

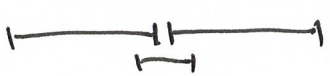
Given a collection of n requests and a single resource. Each request has a start time, s_i, and a finish time, f_i, where $s_i < f_i$. Two requests are compatible if they don't overlap, i.e. $f_i \leq s_j$ or $f_j \leq s_i$. Select a compatible subset of maximum size.



Claim: this problem can be solved with a Greedy Algorithm.

Greedy Strategies

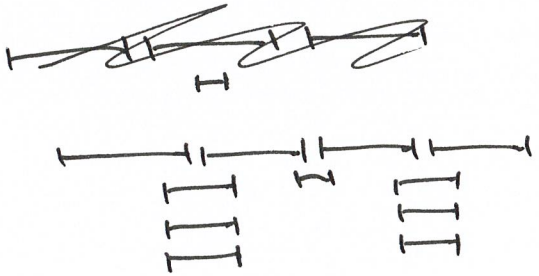
- shortest interval first



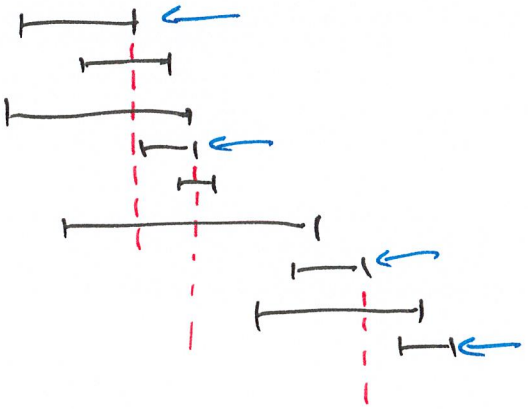
← counterexample

- earliest start time

- fewest incompatible requests



- earliest finish time
  - works



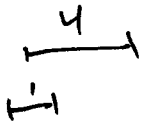
Weighted Interval Scheduling

- Each request i has a weight w_i.

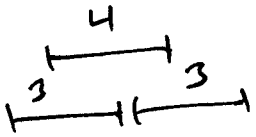
Schedule the subset of compatible requests with maximum weight.

Greedy Counterexample

- earliest finish first



- highest weight first



Dynamic Programming ← works in $O(n^2)$

Multiple Resources Weighted Interval

- Instead of one resource ~~at~~ we have 3 or more
- there is no known efficient algorithm for maximizing weights
- This problem is NP-Hard