Learning Goals

- Describe typical properties of greedy algorithms + key words: "Objective Function", "Completion Time", "Optimization Problem", "Greedy Ordering," "Optimal ordering", "Scoring Function"
- Create counterexample for a greedy ordering [G1]
- Prove optimality of a greedy algorithm using an exchange argument [G2]

Greedy Algorithm (informal def): an alg that sequentially constructs a solution through a series of myopic (short-sighted = local = not global = not thinking about future) decisions

Typical properties

- Easy to create
- Runtime analysis easy
- Not optimal
- When optimal, hard to prove correct

Scheduling Tasks "Optimization Problem"

task	1	2	3
time	3	4	2
weight	5	1	2

Input: n tasks: time, weight for each ↑ larger = more important

Output: Order σ ("sigma") that minimizes

"objective function" → $A(\sigma) = \sum_{i=1}^{n} w_i C_i(\sigma)$ (number)

$\sigma \in \{(123), (213), \ldots\}$ — $3!$

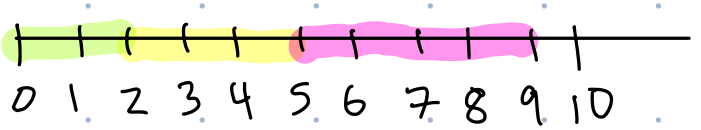
Applications: CPU scheduling, hw schedule

$C_i(\sigma)$ = completion time of task i w/ ordering σ

ex:

task	1	2	3
time	3	4	2

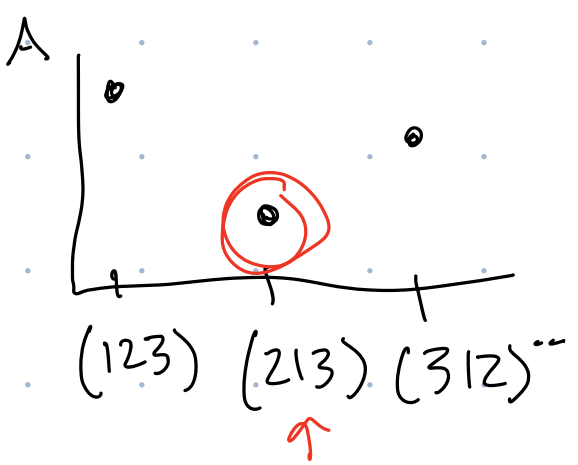
$\sigma = (3,1,2)$



$C_1(3,1,2) = 5$

$C_2(3,1,2) = 9$

$C_3(3,1,2) = 2$



Brute Force

$\Omega(n!)$

What is $C_3(2,1,3)$?

A) 2 B) 3 C) 7 D) 9

Calculate $A(\sigma)$ + determine best order: (Brute force)

task	1	2
time	3	4
weight	5	1

Order	(1,2)	(2,1)
$A(\sigma)$	22	39

Order	(1,2)	(2,1)
$C_1(\sigma)$	3	7
$C_2(\sigma)$	7	4

(1,2)
$5 \cdot 3 + 1 \cdot 7 = 22$
$5 \cdot 7 + 1 \cdot 4 = 39$

$$A(\sigma) = \sum_{i=1}^{n} w_i C_i(\sigma)$$

What is this objective function optimizing? What time/weight jobs is it prioritizing?

Output: Ordering σ that minimizes

$$A(\sigma) = \sum_{i=1}^{n} w_i C_i(\sigma)$$

Creating Scoring Functions + Counterexamples

Creating a greedy algorithm

To create, come up with any function $f(i)$ of i, w_i, t_i

$f(i) = t_i - 6$ (order in increasing f-value)

↖ scoring function

task	1	2	
time	3	4	
weight	5	1	
f	-3	-2	→ (1,2)

We did brute force above, found (1,2) is best

Optimal Order:

(1,2)

↑

Greedy Ordering

But will this greedy ordering always be correct?

Try to find a counterexample:
- Think extreme
- Create task close in f-score value but different otherwise

$f(i) = t_i - 6$ (increasing)

task	1	2
time	1	2
weight	1	100
f	-5	-4

→ Greedy (1,2)

Order	(1,2)	(2,1)
$A(\sigma)$	301	203

Optimal Order (2,1)

Group Exercise [G1]

Create counterexamples for scoring functions

A) $f(i) = w_i$ (decreasing) B) $f(i) = w_i - t_i$ (decreasing)

A)

task	1	2
time	1000	1
weight	2	1
f	2	1

→ (1, 2)

Brute force:

Order	(1,2)	(2,1)
A(σ)	3003	2002

?

Optimal: (2, 1)

B)

task	1	2
time		
weight		
f		

Brute force

Order	(1,2)	(2,1)
A(σ)		

Optimal:

One approach to designing greedy alg:

- create several reasonable scoring functions ($w_i - t_i$, w_i / t_i, $w_i^2 - 2t_i$)
- test, try to create counterexamples
- if no counter example, try to prove correct, or just use (heuristic)

Thm: Ordering jobs by $f = \frac{w_i}{t_i}$ (decreasing) is optimal for minimizing $A(\sigma) = \sum_i w_i C_i(\sigma)$.

Pf: [Exchange Argument]

WLOG (without loss of generality) relabel jobs $w_1/t_1 \geq w_2/t_2 \geq w_3/t_3 \cdots \geq w_n/t_n$ so greedy ordering is $\sigma = (1, 2, 3, \ldots, n)$. Suppose you have another ordering $\sigma^* \neq \sigma$. Then there must be tasks b, y that are next to each other in σ^*, but out of order

$$\sigma^* = (\ldots\ldots, y, b, \ldots) \quad \text{but} \quad b < y$$

ex: $\sigma^* = (3, 1, 2)$.

What is b, y in this example?

A) b = 1, y = 3

B) b = 3, y = 1

C) b = 2, y = 3

D) No unique b, y

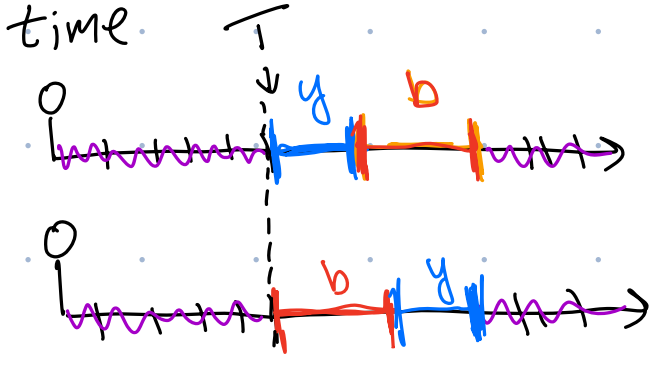
Let $\sigma^{*\prime}$ be the same sequence as σ^*, but with b, y exchanged to be in the correct order

$\sigma^* = (\ldots, y, b, \ldots)$

$\sigma^{*\prime} = (\ldots, b, y, \ldots)$

$\sigma^* = (\boxed{3, 1,} 2)$ → $\sigma^{*\prime} = (1, 3, 2)$

What is $A(\sigma^*) - A(\sigma^{*\prime})$? $\left(A(\sigma) = \sum_{i=1}^{n} w_i C_i(\sigma)\right)$ Show ≥ 0.



σ^*

$\sigma^{*\prime}$

$\sigma^*(i) = i^{th}$ job in ordering σ^*

w_b, w_y, t_b, t_y

$$A(\sigma^*) = w_{\sigma^*(1)} C_{\sigma^*(1)} + w_{\sigma^*(2)} C_{\sigma^*(2)} + \cdots + w_y(T + t_y) + w_b(T + t_y + t_b) + \cdots$$

$$-A(\sigma^{*\prime}) = w_{\sigma^*(1)} C_{\sigma^*(1)} + w_{\sigma^*(2)} C_{\sigma^*(2)} + \cdots + w_b(T + t_b) + w_y(T + t_b + t_y) + \cdots$$

$$= w_b t_y - w_y t_b$$

Note $b < y$, so $\frac{w_b}{t_b} \geq \frac{w_y}{t_y}$

Multiply both sides by $t_b t_y$ to get $w_b t_y \geq w_y t_b$.

Thus $A(\sigma^*) - A(\sigma^{*'}) \geq 0$.

Now if $\sigma^{*'}$ is not the greedy ordering σ, we can exchange two out-of-order jobs to get $\sigma^{*''}$ s.t. $A(\sigma^{*'}) \geq A(\sigma^{*''})$. Continuing in this way, we will eventually get to σ (by bubble sort process). At each swap, we have

$$A(\sigma^*) \geq A(\sigma^{*'}) \geq A(\sigma^{*''}) \geq \cdots \geq A(\sigma)$$

Thus

$$A(\sigma^*) \geq A(\sigma)$$

But σ^* was <u>any</u> ordering that was not σ, so the greedy ordering is optimal.

Structure of Exchange Proof

1. Relabel items so greedy strategy σ, is "simple"
2. Consider any other strategy $\sigma^* \neq \sigma$
3. Modify σ^* by exchanging/swapping 2 elements to get $\sigma^{*\prime}$ (make $\sigma^{*\prime}$ a little bit more like σ)
4. Show $A(\sigma^*)$ is worse (not better) than $A(\sigma^{*\prime})$
5. Claim if repeat steps 2-4 repeatedly, end up at σ. At each step A-value is not worse, so σ is optimal.