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

Typical properties

- Easy to create
- Runtime easy to analyze
- Frequently not optimal
- When optimal, hard to prove correct.

Scheduling

ex: $n = 3$

time needed for task 1

$$t = \begin{array}{c} t_1 \; t_2 \; t_3 \\ \boxed{3 \mid 4 \mid 2} \end{array} \dots$$

1 is most important

$$w = \begin{array}{c} \boxed{5 \mid 1 \mid 2} \\ w_1 \; w_2 \; w_3 \end{array}.$$

Input: n tasks,

- time for each task,
- weight/importance for each task

[Idea: Can only do one task at a time, must complete a task before moving on]

Output: Ordering of tasks $\sigma = (\sigma_1, \sigma_2, \sigma_3 \dots)$ e.g. $\sigma = (3, 1, 2)$

that minimizes

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

← "Objective Function" is a function where your goal (object) is to maximize or minimize its value. Used in "optimization" problems, where goal is to optimize something.

Application: CPU scheduling

What is the runtime of a brute force algorithm?

A) $O(n)$ B) $O(n^2)$ C) $O(n!)$ D) $O(n \cdot n!)$

Input: n tasks,

- time for each task,
- weight/importance for each task

Output: Ordering of tasks $\sigma = (\sigma_1, \sigma_2, \sigma_3 \ldots)$

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

Brute Force:

optσ = $\emptyset$

min $A \leftarrow \infty$

For $\sigma \in$ permutations of $\{1, 2, 3, \ldots, n\}$ $O(n!)$

  $A \leftarrow A(\sigma)$

  If $A <$ min A,

    min $A \leftarrow A$

    opt$\sigma \leftarrow \sigma$

Return optσ

$A(\sigma)$:

Sum $\leftarrow 0$

time $\leftarrow 0$

For $i \in \sigma$ $O(n)$

  time $\leftarrow$ time + $t[i]$

  Sum $\leftarrow$ sum + $w[i] \cdot$ time

Return sum

Objective Function + Completion Time

Is large when important jobs are delayed

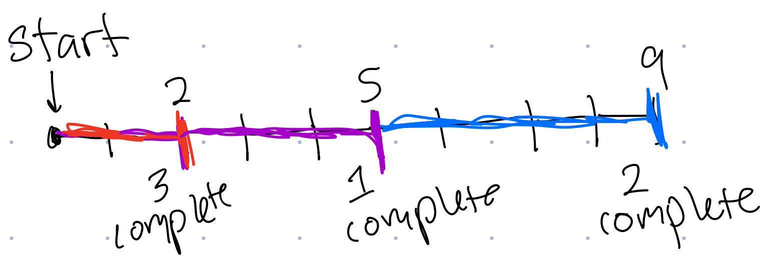
$$\boxed{A(\sigma) = \sum_{i=1}^{n} w_i C_i(\sigma)}$$

Completion Time: $C_i(\sigma)$ time when job i is completed with ordering σ

ex:

t_1	t_2	t_3	
3	4	2	...

$\sigma(3,1,2)$



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

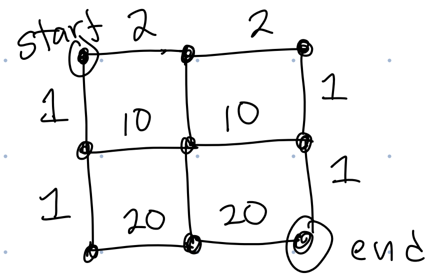
$C_2(3,1,2) = 9$

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

Greedy Alg Warm-Up Grab-Bag

Greedy Algorithm (informal def): an alg that sequentially contstructs a solution through a series of myopic decisions

1. What is a greedy path finding alg on a grid?



2. Why is the following approach "greedy"? (based on def)

$\boxed{\begin{array}{l} \bullet \text{ Calculate a score } w_i - t_i \text{ for each task} \\ \bullet \text{ Schedule in order of decreasing score} \end{array}}$ (highest score first)

3. Provide a counter example that does not always minimize $A(\sigma)$
   - create (2) tasks; calculate scores, determine schedule, show not optimal
     $(A(\sigma) = \sum_i w_i C_i(\sigma))$

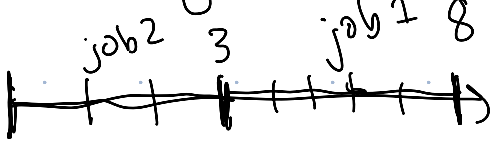
4. What might be a general ethical concern with our scheduling problem, applied to any domain?

Counterexample:

	job1	job2
$t =$	5	3
$w =$	2	1
score $=$	-3	-2

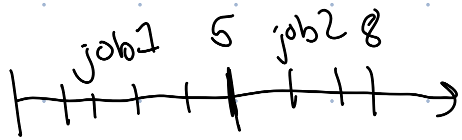
$(w_i - t_i)$

Greedy Ordering $(2, 1)$



$$A = w_1 C_1 + w_2 C_2$$
$$= 2 \cdot 8 + 1 \cdot 3 = 19$$

Other Ordering $(1, 2)$



$$A = w_1 C_1 + w_2 C_2$$
$$= 2 \cdot 5 + 1 \cdot 8 = 18$$

One approach to designing greedy alg:

$w_i - t_i$

w_i / t_i

$w_i^2 - 2t_i$

- create several reasonable scoring functions
- test, try to create counterexamples
- if no counter example, try to prove correct

<u>Thm</u>: Ordering jobs by decreasing value of w_i/t_i is optimal for minimizing $A(\sigma) = \sum_i w_i C_i(\sigma)$ if w_i/t_i are all distinct.

<u>Pf</u>: [Exchange Argument = Type of Pf by Contradiction]

WLOG, relabel so $w_1/t_1 > w_2/t_2 > w_3/t_3$ so greedy ordering is $\sigma = (1, 2, 3, \ldots, n)$. Assume for contradiction that σ is not optimal. Then $\exists\, \sigma^*$ that is optimal.

Since $\sigma^* \neq \sigma$, there must be tasks b, y in σ^* that are next to each other but out of order relative to σ:

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

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

What is b, y in this example?

A) $b = 1$, $y = 3$

B) $b = 3$, $y = 1$

C) $b = 2$, $y = 3$

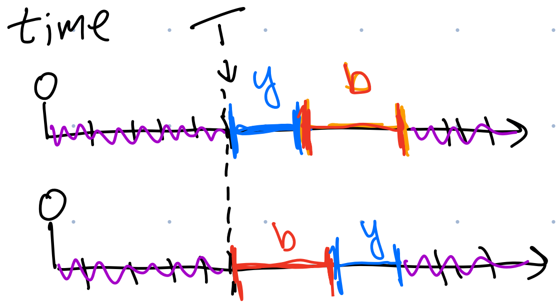
D) There is not a unique b, y in this example

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

$$\sigma^* = (\ldots\ y, b\ \ldots) \qquad \sigma^* = (3, 1, 2)$$

$$\sigma^{*\prime} = (\ldots\ b, y\ \ldots) \qquad \sigma^{*\prime} = (1, 3, 2)$$

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



σ^*

$\sigma^{*\prime}$

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

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

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

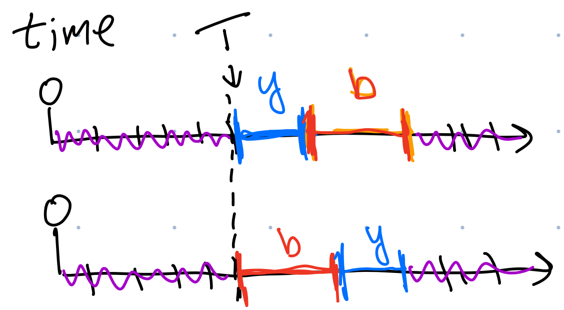
$$= w_b t_y - w_y t_b$$

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

$$\sigma^* = (\ldots\ y,\ b\ \ldots) \qquad \sigma^* = (3, 1, 2)$$

$$\sigma^{*'} = (\ldots\ b,\ y\ \ldots) \qquad \sigma^{*'} = (1, 3, 2)$$

What is $A(\sigma^*) - A(\sigma^{*'})$? $\left(A(\sigma) = \sum_{i=1}^{n} w_i C_i(\sigma) \right)$



σ^*

$\sigma^{*'}$

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

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

$$-A(\sigma^{*'}) = \cancel{w_{\sigma^{*'}(1)} C_{\sigma^{*'}(1)}} + \cancel{w_{\sigma^{*'}(2)} C_{\sigma^{*'}(2)}} + \cdots w_b(\cancel{T} + \cancel{t_b}) + w_y(\cancel{T} + \cancel{t_y} + t_b) + \cancel{\cdots}$$

$$= w_b t_y - w_y t_b$$

$A(\sigma^*) - A(\sigma^{*\prime}) = w_b t_y - w_y t_b$. Divide both sides by $t_y \cdot t_b$:

$$\frac{A(\sigma^*) - A(\sigma^{*\prime})}{t_y t_b} = \frac{w_b}{t_b} - \frac{w_y}{t_y}.$$ But $b < y$, so $\frac{w_b}{t_b} > \frac{w_y}{t_y}$, so

$$\frac{A(\sigma^*) - A(\sigma^{*\prime})}{t_y t_b} > 0,$$ and $t_y, t_b > 0$, so

$A(\sigma^*) - A(\sigma^{*\prime}) > 0$. We said

$A(\sigma^*)$ was optimal so it must have the smallest A-value, but now we've shown $\sigma^{*\prime}$ has a smaller A-value, a contradiction.

Thus our assumption that σ was not optimal was incorrect, and σ must be optimal.

Structure of Exchange Proof

1. Assume greedy strategy σ is not optimal
2. There must exist an optimal strategy $\sigma^* \neq \sigma$
3. Modify σ^* by exchanging/swapping 2 elements $\Rightarrow \sigma^{*'}$
4. Show $\sigma^{*'}$ is better than $\sigma^* \Rightarrow$ contradiction

Assumption incorrect

What is the runtime of our greedy scheduling algorithm?

A) $O(1)$ B) $O(n)$ C) $O(n \log n)$ D $O(n^2)$

Where did we use w_i/t_i are all distinct?

Schedule(n):
- For $i \leftarrow 1$ to n:
  - Calculate w_i/t_i score
- Sort by score

Recall

$A(\sigma^*) - A(\sigma^{*\prime}) = w_b t_y - w_y t_b$. Divide both sides by $t_y \cdot t_b$:

$$\frac{A(\sigma^*) - A(\sigma^{*\prime})}{t_y t_b} = \frac{w_b}{t_b} - \frac{w_y}{t_y}.$$ But $b < y$, so $\frac{w_b}{t_b} > \frac{w_y}{t_y}$, so

$$\frac{A(\sigma^*) - A(\sigma^{*\prime})}{t_y t_b} > 0,$$ and $t_y, t_b > 0$, so

$A(\sigma^*) - A(\sigma^{*\prime}) > 0$. We said

$A(\sigma^*)$ was optimal so it must have the smallest A-value, but now we've shown $\sigma^{*\prime}$ has a smaller A-value, a contradiction.

Thus our assumption that σ was not optimal was incorrect, and σ must be optimal.

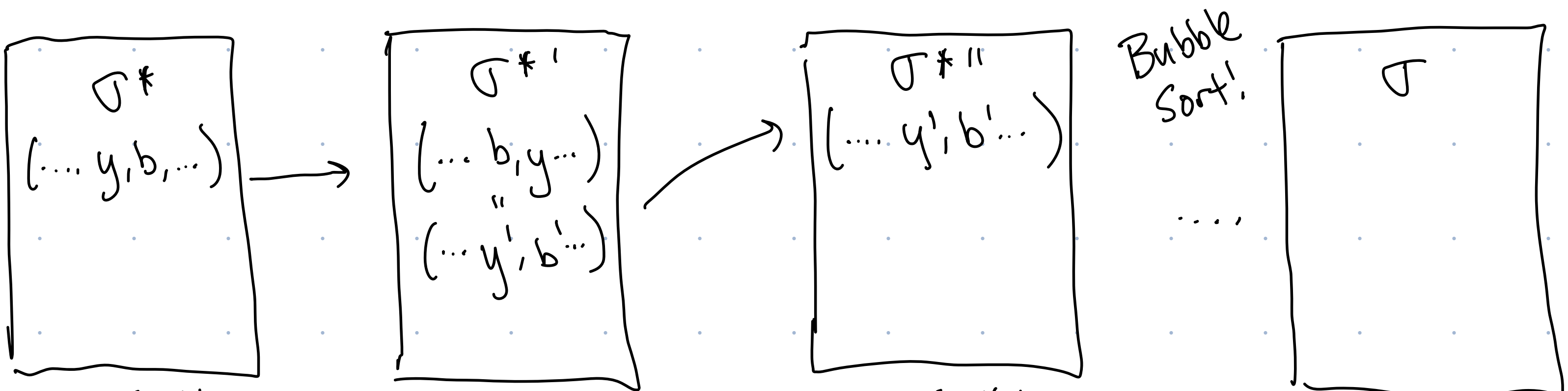
Lose contradiction!

Thm: Ordering jobs by decreasing value of w_i/t_i is optimal for minimizing $A(\sigma) = \sum_i w_i C_i(\sigma)$ ~~if w_i/t_i are all distinct.~~

Pf sketch: Choose some relabelling of tasks so that

$$w_1/t_1 \geq w_2/t_2 \geq w_3/t_3 \cdots \geq w_n/t_n$$

We call $\sigma = (1, 2, 3, \ldots, n)$ this greedy ordering. Let σ^* be any other ordering. We will show $A(\sigma^*) \geq A(\sigma)$. Since for every permution σ^*, $A(\sigma^*) \geq A(\sigma)$, σ must be optimal.



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

What is the runtime of our greedy scheduling algorithm?

A) $O(1)$ B) $O(n)$ C) $O(n \log n)$ D $O(n^2)$

Algorithm has not changed!
We don't do bubblesort; we just
imagine doing bubblesort in the proof!