3: Scheduling (Greedy)

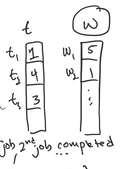
Wednesday, March 3, 2021
8:54 PM

Goals:
1. Understand Greedy Approach ✓
2. Practice applying to scheduling problem ✓

Greedy: simple method to make choices

Problem: Scheduling

Input: n tasks,
- time for each task
- importance (weight) for each
(larger weight = more important)



Output: Ordering of jobs: sigma $\sigma = (\sigma_1, \sigma_2, \dots, \sigma_n)$

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

fast, important jobs scheduled earlier

Ethical Concern? Weights!

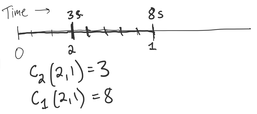
Applications: Jobs on CPU

Completion time:

ex

job	time
1	5
2	3

$\sigma = (2, 1)$



$C_2(2,1) = 3$
$C_1(2,1) = 8$

$C_i(\sigma)$ = completion time of job i under ordering σ

What is $C_2(1,2)$?

A) 2 B) 3 C) 5 D) 8

Order Should Minimize:

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

job	time	weight
1	5	2
2	3	1

What is $A(1,2)$?

A) 3
B) 8
C) 13
D) 18

$w_1 \cdot C_1(1,2) + w_2 \cdot C_2(1,2)$
$2 \cdot 5 + 1 \cdot 8 = 18$

$w_1 \cdot C_1(2,1) + w_2 \cdot C_2(2,1)$
$2 \cdot 8 + 1 \cdot 3 = 19$

How to Create a Greedy Algorithm

1. Choose a function that assigns a score to each job
ex $f(i) = w_i + t_i$

job	time	weight	f-score	
1	5	2	7	2
2	3	1	4	1
3	9	4	13	3

2. Sort jobs by f-score (decide increasing or decrea
3. Choose σ = sorted order

Pros
- Easy to implement
- Fast ?
- Flexible

Cons
- Ties?
- Not best ordering
- Long runtime (?)

Hopefully optimal

How to Create a Greedy Algorithm

1. Choose several f-functions that seem reasonable, given desired output
2. Test on several inputs to see whether give correct ordering → Try to find examples where is incorrect to rule out
3. If an f-function always seems to give correct ordering, then try to prove correct.

In Groups
- Come up with at least 2 reasonable f-functions to minimize A. (use +, −, ÷, or * of w_i, t_i)
- Test on

job	time	weight
1	5	2
2	3	1

(To be consistent, will order large f-value jobs first)

or otherwise try to rule out

$f_1 = w_i / t_i$ $f_2 = w_i^{1/t_i}$ $f_3 = w^{(d)} + t^{(q)}$

job	time	weight	f_1	f_2
1	5	2		
		1		

order	A
1 2	