April 2, 2019 10:59 AM

Greedy Alg'ms

To solve an optimization problem,
incrementally build up sol'n
· at each step, choose what seems best "locally"

adv: Simple, fast
disadv: may not be correct
if correct, needs proof!

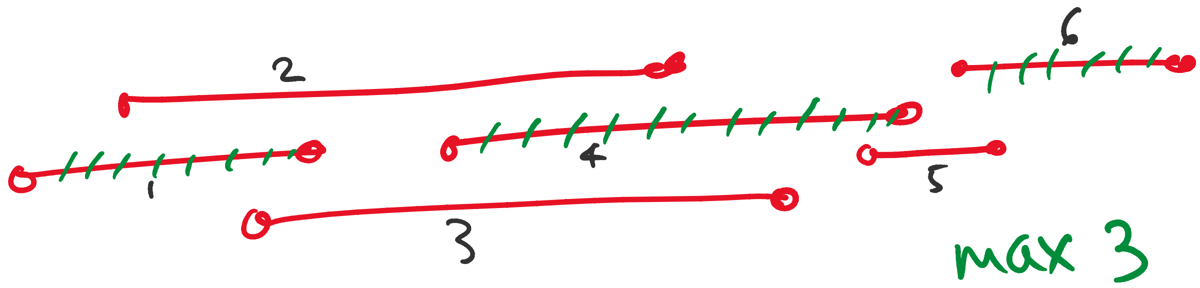
Ex1: Interval Scheduling

Given n intervals $[s_1, f_1], \ldots, [s_n, f_n]$

start time for job 1 — finish time

find largest subset of disjoint intervals

e.g.



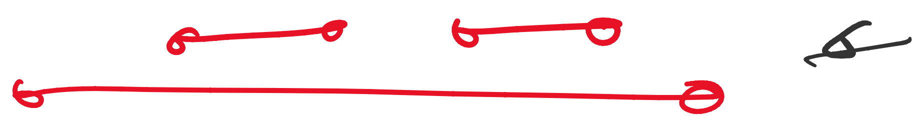
max 3

idea 1 (greedy) · pick shortest job (min $f_i - s_i$)
fail!

idea 2 " - pick job that starts earliest? (min s_i)

fail!

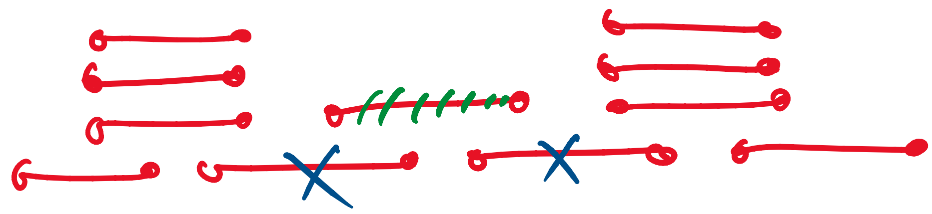
Counterex



idea 3 - pick job that conflicts with fewest other jobs:

fail!

Counterex



greedy 3
opt 4

idea 4 - pick job that finishes earliest (min f_i):

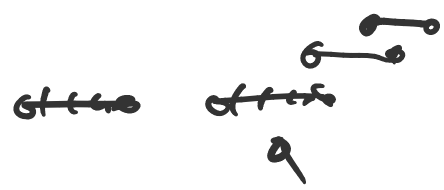
bingo!

Greedy Alg'm:

repeat {
1. pick $[s_i, f_i)$ with smallest f_i
2. remove $[s_i, f_i]$ & all intervals intersecting it
3. } until no intervals left
4.
5. return all intervals picked

Runtime: trivial $O(n^2)$ time

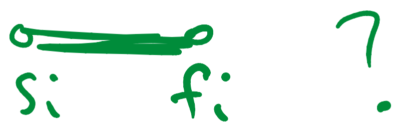
better: sort & scan
$O(n \log n)$ time



Correctness Pf:

Let I^* be an opt sol'n.
Let $[s^*, f^*]$ be leftmost interval in I^*.

Let $[s^*, f^*]$ be leftmost interval in I^*.
Let $[s_i, f_i]$ be first interval chosen by above greedy alg'm.



Know $f_i \leq f^*$ by the way our greedy alg'm works

$$\Rightarrow \quad I^* - \{[s^*, f^*]\} \cup \{[s_i, f_i]\}$$

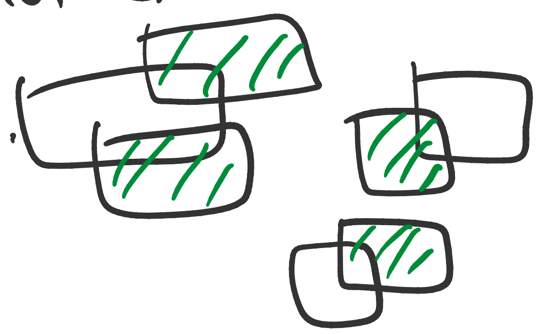
"exchange arg" is a feasible sol'n with same # intervals

Reset $I^* \leftarrow I^* - \{[s^*, f^*]\} \cup \{[s_i, f_i]\}$.

Remove $[s_i, f_i]$ & all intervals intersecting it. & repeat argument (i.e. induction). $\square$

Rmk - does not extend to weighted case (but can do DP)

- does not extend to 2D



Ex 2: Scheduling to minimize avg wait time

Ex2: Scheduling to minimize avg wait time

Given n jobs with process. time $p_1, \dots, p_n$,
find a re-ordering of the jobs
to minimize total wait time

$$\text{cost} = 0 + p_1 + (p_1 + p_2) + (p_1 + p_2 + p_3) + \dots + (p_1 + \dots + p_{n-1})$$

e.g.

jobs	1	2	3	4	5
time	3	4	1	8	2

wait time: $0 + 3 + (3+4) + (3+4+1) + (3+4+1+8)$
$= 34$

better time: $0 + 3 + (3+4) + (3+4+1) + (3+4+1+2)$
$= 28$

Still better: $0 + 3 + (3+1) + (3+1+4) + (3+4+1+2)$
$= 25$

⋮

best: $0 + 1 + (1+2) + (1+2+3) + (1+2+3+4)$
$= 20.$

Greedy Algm:
just sort jobs in increas. p_i.

Correctness Pf:
Let $p_1^*, \dots, p_n^*$ be opt order.
Suppose it is not sorted.
Then $p_i^* > p_{i+1}^*$ for some i.

Then $p_i^* > p_{i+1}$ for some i.

"exchange arg'n" $\Rightarrow$ Now swap i and $i+1$.

$$\text{Old cost} = p_1^* + \dots + (p_1^* + \dots + p_{i-1}^*) + (p_1^* + \dots + p_{i-1}^* + p_i^*) + (p_1^* + \dots + p_{i+1}^*) + \dots + (p_1^* + \dots + p_{n-1}^*)$$

$$\text{New Cost} = p_1^* + \dots + (p_1^* + \dots + p_{i-1}^*) + (p_1^* + \dots + p_{i-1}^* + p_{i+1}^*) + (p_1^* + \dots + p_{i+1}^*) + \dots + (p_1^* + \dots + p_{n-1}^*)$$

$$\text{New cost} - \text{Old cost} = p_{i+1}^* - p_i^* < 0$$

$\Rightarrow$ new sol'n is strictly better.

Contradiction!

Rmk: extends to weighted version

$$\text{cost} = w_1 0 + w_2 p_1 + w_3(p_1 + p_2) + \dots + w_n(p_1 + \dots + p_{n-1})$$

Correct greedy: Sort in increas. $\underline{p_i / w_i}$

$$\text{Old cost} = \dots + w_i^*(p_1^* + \dots + p_{i-1}^*) + w_{i+1}^*(p_1^* + \dots + p_{i-1}^* + p_i^*) + \dots$$

$$\text{New cost} = \dots + w_{i+1}^*(p_1^* + \dots + p_{i-1}^*) + w_i^*(p_1^* + \dots + p_{i-1}^* + p_{i+1}^*) + \dots$$

$$\text{New cost} - \text{Old cost} = w_i^* p_{i+1}^* - w_{i+1}^* p_i^* < 0 \quad \text{if} \quad \frac{p_i^*}{w_i^*} > \frac{p_{i+1}^*}{w_{i+1}^*}$$