Greedy Algorithms

A Scheduling Application: Problem Definition

Algorithms: Design and Analysis, Part II

A Scheduling Problem

Setup:

- One shared resource (e.g., a processor).
- Many "jobs" to do (e.g., processes).

Question: In what order should we sequence the jobs?

Assume: Each job has a:

- weight w_j ("priority")
- length l_J

Completion Times

Definition: The completion time C_j of job j = Sum of job lengths up to and including j.

Example: 3 jobs, $l_1 = 1, l_2 = 2, l_3 = 3$.

Schedule:

#1	#2	#3

$0 \rightarrow$
(time)

Question: What is C_1, C_2, C_3?

A) $1, 2, 3$ C) $1, 3, 6$

B) $3, 5, 6$ D) $1, 4, 6$

The Objective Function

Goal: Minimize the weighted sum of completion times:
$\min \sum_{j=1}^{n} w_j C_j$.

Back to example: If $w_1 = 3, w_2 = 2, w_3 = 1$, this sum is
$3 \cdot 1 + 2 \cdot 3 + 1 \cdot 6 = 15$.

Algorithms: Design and Analysis, Part II

Greedy Algorithms

A Scheduling Application: The Algorithm

Intuition for Algorithm

Recall: Want to $\boxed{\min \sum_{j=1}^{n} w_j}$. Cj

Goal: Devise correct greedy algorithm.

Question:

1. With equal lengths, schedule larger or smaller-weight jobs earlier?
2. With equal weights, schedule shorter or longer jobs earlier?

A) Larger/shorter
C) Larger/longer
B) Smaller/shorter
D) Smaller/longer

Resolving Conflicting Advice

Question: What if $w_i > w_j$ but $l_i > l_j$?

Idea: Assign "scores" to jobs that are:

- inscreasing in weight
- decreasing in length

Guess (1): Order jobs by decreasing value of $w_j - l_j$.

Guess (2): Order w_j / l_j.

Breaking a Greedy Algorithm

To distinguish (1) & (2): Find example where the two algorithms produce different outputs. (At least one will be incorrect.)

Example:

$l_1 = 5, w_1 = 3$ (longer ratio) w1 = 3, l1 = 5

$l_1 = 2, w_1 = 1$ (larger difference) w2 = 1, l2 = 2

Question: What is the sum of weighted completion times of algorithms (1) & (2) respectively?

A) 22 and 23

B) 23 and 22

C) 17 and 17

D) 17 and 11

Alg#1: $\boxed{\#2}\boxed{\#1} \rightarrow 1 \cdot 2 + 3 \cdot 7 = 23$

Alg#2: $\boxed{\#1}\boxed{\#2} \rightarrow 3 \cdot 5 + 1 \cdot 7 = 22$

The Story So Far

So: Alg#1 not (always) correct.

Claim: Alg#2 (order by decreasing ratio w_j/l_j's) is always correct.
[not obvious! - proof coming up next]

Running time: $O(n \log n)$. [just need to sort]

Greedy Algorithms

A Scheduling Application: Correctness Proof Part I

Algorithms: Design and Analysis, Part II

Correctness Claim

Claim: Algorithm #2 (order jobs according to decreasing ratios w_j/l_j) is always correct.

Proof: By an Exchange Argument.

Plan: Fix arbitrary input of n jobs. Will proceed by contradiction. Let $\sigma =$ greedy schedule, $\sigma^* =$ optimal schedule. (With σ^* better than σ.)
Will produce schedule even better than σ^*, contradicting purported optimality of σ^*.

Correctness Proof

Assume: All w_j/l_j's distinct.

Assume: [Just by renaming jobs] $w_1/l_1 > w_2/l_2 > \ldots > w_n/l_n$.

Thus: Greedy schedule σ is just $1, 2, 3, \ldots, n$.

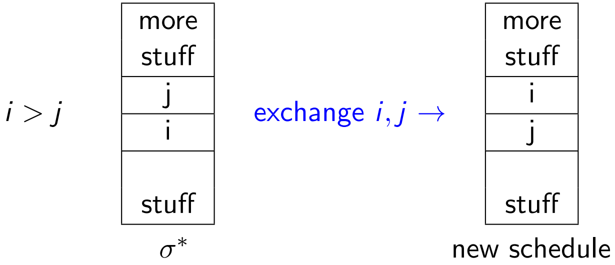
Thus: If optimal schedule $\sigma^* \neq \sigma$, then there are consecutive jobs i, j with $i > j$.
[Only schedule where indices always go up is $1, 2, 3, \ldots, n$]

Correctness Proof (con'd)

So far:

1. $w_1/l_1 > w_2/l_2 > \ldots > w_n/l_n$
2. In optimal σ^*, $\exists$ consecutive jobs i, j with $i > j$.

Thought experiment: Suppose we exchange order of $i \& j$ in σ^* (leaving other jobs unchanged):



Greedy Algorithms

A Scheduling Application: Correctness Proof Part II

Algorithms: Design and Analysis, Part II

Cost-Benefit Analysis, Part I

$i > j$	σ^*	exchange $i, j \rightarrow$	new schedule
	more stuff		more stuff
	j		i
	i		j
	stuff		stuff

Question: What is the effect of this exchange on the completion time of (1) a job k other than i or j, (2) the job i, (3) the job j?

A) Not enough info/goes up/goes down

B) Not enough info/goes down/goes up

C) Unaffected/ goes up / goes down

D) Unaffected/goes down/goes up

by l_j

by l_i

Cost-Benefit Analysis, Part II

Upshot:

1. Cost of exchange $w_i l_j$. [C_i goes up by l_j]
2. Benefit of exchange is $w_j l_i$. [C_j goes down by l_i]

Note: $i > j \Rightarrow w_i/l_i < w_j/l_j \Rightarrow w_i l_j < w_j l_i \Rightarrow COST < BENEFIT$
$\Rightarrow$ Swap improves σ^*, contradicts optimality of σ^*.

QED!

Greedy Algorithms

A Scheduling Application: Handling Ties

Algorithms: Design and Analysis, Part II

Correctness Claim

Claim: Algorithm #2 (order jobs in nonincreasing order of ratio w_j/l_j) is always correct. [Even with ties]

New Proof Plan: Fix arbitrary input of n jobs. Let σ = greedy schedule, let σ^* = any other schedule.

Will show σ at least as good as $\sigma^* \Rightarrow$ Implies that greedy schedule is optimal.

Correctness Proof

Assume: [Just by renaming jobs] Greedy schedule σ is just $1, 2, 3, \ldots, n$ (and so $w_1/l_1 \geq w_2/l_2 \geq \ldots \geq w_n/l_n$).

Consider arbitrary schedule σ^*. If $\sigma^* = \sigma$, done.

Else recall $\exists$ consecutive jobs i, j in σ^* with $i > j$. (From last time)

Note: $i > j \Rightarrow w_i/l_i \leq w_j/l_j \Rightarrow w_i l_j \leq w_j l_i$.

Recall: Exchanging $i \& j$ in σ^* has net benefit of $w_j l_i - w_i l_j \geq 0$.

$i > j$

more stuff
j
i
stuff

σ^*

exchange $i, j \rightarrow$

more stuff
i
j
stuff

new schedule

Correctness Proof

Upshot: Exchanging an "adjacent inversion" like i, j only makes σ^* better, and it decreases the number of inverted pairs.

Jobs i, j with $i > j$ and i scheduled earlier

$\Rightarrow$ After at most $\binom{n}{2}$ such exchanges, can transform σ^* into σ.

$\Rightarrow$ σ at least as good as σ^*.

$\Rightarrow$ Greedy is optimal.

QED!