COL 351 : Analysis & Design of Algorithms

Lecture 14

Greedy Algorithms I : Job Scheduling

Aug 27, 2024 | Rohit Vaish

ANNOUNCEMENTS & REMINDERS

* Quiz 2 on Aug 30 (Friday)

* Two-sided tutorial answer sheets (from sheet 5 onwards)

* Course feedback on Moodle (due today)

Greedy Algorithms

Greedy Algorithms

Do what looks best right now and hope everything works out at the end

Greedy Algorithms

Do what looks best right now and hope everything works out at the end

Example: Dijkstra's shortest path algorithm

Greedy Algorithms

Do what looks best right now and hope everything works out at the end

Example: Dijkstra's shortest path algorithm

- iteratively and irrevocably "lock in" the estimate for shortest path to one additional vertex

Design Runtime Correctness

Divide and Conquer

Graph algorithms

Greedy algorithms

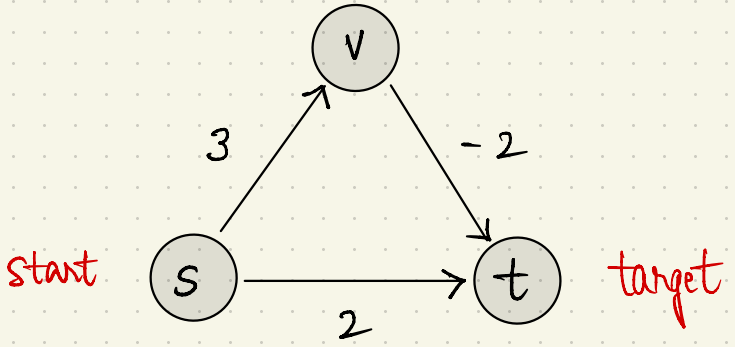
	Design	Runtime	Correctness
Divide and Conquer			
Graph algorithms			
Greedy algorithms			

	Design	Runtime	Correctness
Divide and Conquer			
Graph algorithms			
Greedy algorithms			

	Design	Runtime	Correctness
Divide and Conquer			
Graph algorithms			
Greedy algorithms			

	Design	Runtime	Correctness
Divide and Conquer	🌶🌶🌶🌶	🌶🌶🌶🌶	🌶🌶🌶🌶
Graph algorithms	🌶🌶🌶🌶	🌶🌶🌶🌶	🌶🌶🌶🌶
Greedy algorithms	🌶🌶🌶🌶	🌶🌶🌶🌶	🌶🌶🌶🌶

Most greedy heuristics are incorrect



Dijkstra : $s \rightarrow v$

Correct : $s \rightarrow v \rightarrow t$

Proofs of Correctness

Proofs of Correctness

Method 1: Induction (e.g., Dijkstra) "greedy stays ahead"

Proofs of Correctness

Method 1: Induction (e.g., Dijkstra) "greedy stays ahead"

Method 2: Exchange argument (coming up!)

Proofs of Correctness

Method 1: Induction (e.g., Dijkstra) "greedy stays ahead"

Method 2: Exchange argument (coming up!)

Method 3: Whatever works!

JOB SCHEDULING

JOB SCHEDULING

One shared resource (e.g., a processor, a classroom)

Many jobs to do (e.g., processes, lectures)

Q. In what order should the jobs be scheduled?

JOB SCHEDULING

One shared resource (e.g., a processor, a classroom)

Many jobs to do (e.g., processes, lectures)

Q. In what order should the jobs be scheduled?

Assume: n jobs, each job j has

* length l_j: amount of time needed to process j
* weight w_j: higher weight $\Rightarrow$ higher priority

COMPLETION TIMES

The completion time C_j of job j =

sum of job lengths up to and including j.

COMPLETION TIMES

The completion time C_j of job j =

sum of job lengths up to and including j.

E.g., 3 jobs, lengths $l_1 = 1$, $l_2 = 2$, $l_3 = 3$

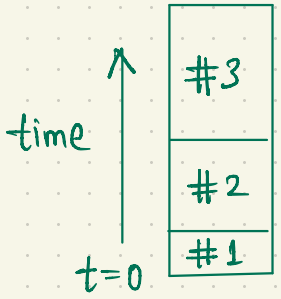
COMPLETION TIMES

The completion time C_j of job j =

sum of job lengths up to and including j.

E.g., 3 jobs, lengths $l_1 = 1$, $l_2 = 2$, $l_3 = 3$

Schedule : an ordering of jobs



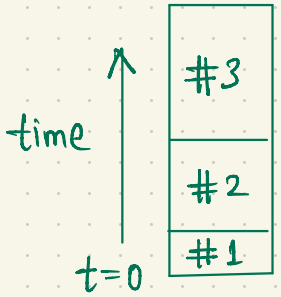
COMPLETION TIMES

The completion time C_j of job j =

sum of job lengths up to and including j.

E.g., 3 jobs, lengths $l_1 = 1$, $l_2 = 2$, $l_3 = 3$

Schedule : an ordering of jobs



$C_1 = ?$ $C_2 = ?$ $C_3 = ?$

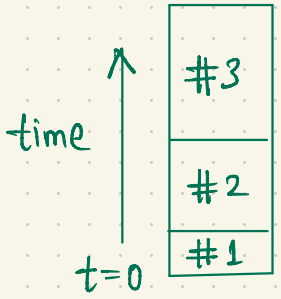
COMPLETION TIMES

The completion time C_j of job j =

sum of job lengths up to and including j.

E.g., 3 jobs, lengths $l_1 = 1$, $l_2 = 2$, $l_3 = 3$

Schedule: an ordering of jobs



$C_1 = 1$ $C_2 = 3$ $C_3 = 6$

OBJECTIVE FUNCTION

OBJECTIVE FUNCTION

goal : minimize the weighted sum of completion times

$$\min \sum_{j=1}^{n} w_j \cdot C_j$$

OBJECTIVE FUNCTION

goal : minimize the weighted sum of completion times

$$\min \sum_{j=1}^{n} w_j \cdot C_j$$

E.g., 3 jobs, lengths $l_1 = 1$, $l_2 = 2$, $l_3 = 3$

weights $w_1 = 3$, $w_2 = 2$, $w_3 = 1$

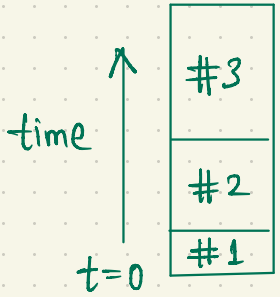
OBJECTIVE FUNCTION

goal : minimize the weighted sum of completion times

$$\min \sum_{j=1}^{n} w_j \cdot C_j$$

E.g., 3 jobs , lengths $l_1 = 1$, $l_2 = 2$, $l_3 = 3$

weights $w_1 = 3$, $w_2 = 2$, $w_3 = 1$



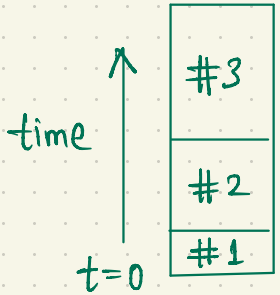
OBJECTIVE FUNCTION

goal : minimize the weighted sum of completion times

$$\min \sum_{j=1}^{n} w_j \cdot C_j$$

E.g., 3 jobs, lengths $l_1 = 1$, $l_2 = 2$, $l_3 = 3$

weights $w_1 = 3$, $w_2 = 2$, $w_3 = 1$



$$\sum_{j=1}^{n} w_j \cdot C_j =$$

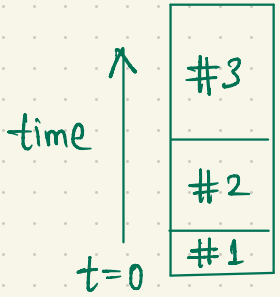
OBJECTIVE FUNCTION

goal : minimize the weighted sum of completion times

$$\min \sum_{j=1}^{n} w_j \cdot C_j$$

E.g., 3 jobs, lengths $l_1 = 1$, $l_2 = 2$, $l_3 = 3$

weights $w_1 = 3$, $w_2 = 2$, $w_3 = 1$



$$\sum_{j=1}^{n} w_j \cdot C_j = \underset{w_1}{3} \times \overset{C_1}{1} + \underset{w_2}{2} \times \overset{C_2}{3} + \underset{w_3}{1} \times \overset{C_3}{6}$$

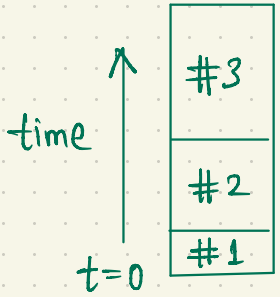
OBJECTIVE FUNCTION

goal : minimize the weighted sum of completion times

$$\min \sum_{j=1}^{n} w_j \cdot C_j$$

E.g., 3 jobs, lengths $l_1 = 1$, $l_2 = 2$, $l_3 = 3$

weights $w_1 = 3$, $w_2 = 2$, $w_3 = 1$



$$\sum_{j=1}^{n} w_j \cdot C_j = \underset{w_1}{3} \times \overset{C_1}{1} + \underset{w_2}{2} \times \overset{C_2}{3} + \underset{w_3}{1} \times \overset{C_3}{6}$$

$$= 15$$

(verify Optimality !)

JOB SCHEDULING

input : a set of n jobs with positive lengths $l_1, l_2, \ldots, l_n$ and positive weights $w_1, w_2, \ldots, w_n$

output : a job schedule (or sequence) that minimizes the sum of weighted completion times $\sum_{j=1}^{n} w_j \cdot C_j$.

JOB SCHEDULING

input : a set of n jobs with positive lengths $l_1, l_2, \ldots, l_n$
and positive weights $w_1, w_2, \ldots, w_n$

output : a job schedule (or sequence) that minimizes the sum of weighted completion times $\sum_{j=1}^{n} w_j \cdot C_j$.

Brute force : ?

JOB SCHEDULING

input : a set of n jobs with positive lengths $l_1, l_2, \ldots, l_n$ and positive weights $w_1, w_2, \ldots, w_n$

output : a job schedule (or sequence) that minimizes the sum of weighted completion times $\sum_{j=1}^{n} w_j \cdot C_j$.

Brute force : $O(n!)$ ☹

DEVELOPING A GREEDY ALGORITHM

DEVELOPING A GREEDY ALGORITHM

Recall : minimize $\sum_{j=1}^{n} w_j \cdot C_j$.

Approach : Use special cases to guide us to the algorithm

DEVELOPING A GREEDY ALGORITHM

Recall : minimize $\sum_{j=1}^{n} w_j \cdot C_j$.

Approach : Use special cases to guide us to the algorithm

(I) All lengths equal : schedule heavier or lighter job earlier ?

DEVELOPING A GREEDY ALGORITHM

Recall : minimize $\sum_{j=1}^{n} w_j \cdot C_j$.

Approach : Use special cases to guide us to the algorithm

(I) All lengths equal : schedule heavier or lighter job earlier ?

DEVELOPING A GREEDY ALGORITHM

Recall : minimize $\sum_{j=1}^{n} w_j \cdot C_j$.

Approach : Use special cases to guide us to the algorithm

(I) All lengths equal : schedule heavier or lighter job earlier ?

Completion times are same for any schedule !

DEVELOPING A GREEDY ALGORITHM

Recall : minimize $\sum_{j=1}^{n} w_j \cdot C_j$.

Approach : Use special cases to guide us to the algorithm

(I) All lengths equal : schedule heavier or lighter job earlier ?

Completion times are same for any schedule !

(II) All weights equal : schedule shorter or longer job earlier ?

DEVELOPING A GREEDY ALGORITHM

Recall : minimize $\sum_{j=1}^{n} w_j \cdot C_j$.

Approach : Use special cases to guide us to the algorithm

(I) All lengths equal : schedule heavier or lighter job earlier ?

Completion times are same for any schedule !

(II) All weights equal : schedule shorter or longer job earlier ?

DEVELOPING A GREEDY ALGORITHM

Recall : minimize $\sum_{j=1}^{n} w_j \cdot C_j$.

Approach : Use special cases to guide us to the algorithm

(I) All lengths equal : schedule heavier or lighter job earlier ?

Completion times are same for any schedule !

(II) All weights equal : schedule shorter or longer job earlier ?

each job negatively impacts all future jobs

DEVELOPING A GREEDY ALGORITHM

Recall : minimize $\sum_{j=1}^{n} w_j \cdot C_j$.

Approach : Use special cases to guide us to the algorithm

(I) All lengths equal : schedule heavier or lighter job earlier ?

Completion times are same for any schedule !

(II) All weights equal : schedule shorter or longer job earlier ?

each job negatively impacts all future jobs

$l_1 = 1$, $l_2 = 3$ → (1,2) $C_1 = 1$ $C_2 = 4$

→ (2,1) $C_1 = 3$ $C_2 = 4$

RESOLVING CONFLICTING ADVICE

RESOLVING CONFLICTING ADVICE

Q. What if heavier jobs are not shorter ($w_i > w_j$ but $l_i > l_j$)?

RESOLVING CONFLICTING ADVICE

Q. What if heavier jobs are not shorter ($w_i > w_j$ but $l_i > l_j$)?

Schedule heavier jobs earlier regardless of length?

RESOLVING CONFLICTING ADVICE

Q. What if heavier jobs are not shorter ($w_i > w_j$ but $l_i > l_j$)?

Schedule heavier jobs earlier regardless of length?

$l_1 = 1$ $l_2 = 100$

$w_1 = 1$ $w_2 = 2$

#1
#2

#2
#1

$1 \times 101 + 2 \times 100 > 1 \times 1 + 2 \times 101$

RESOLVING CONFLICTING ADVICE

Q. What if heavier jobs are not shorter ($w_i > w_j$ but $l_i > l_j$)?

Schedule heavier jobs earlier regardless of length?

$l_1 = 1$ $l_2 = 100$

$w_1 = 1$ $w_2 = 2$

#1
#2

#2
#1

$1 \times 101 + 2 \times 100 > 1 \times 1 + 2 \times 101$

Schedule shorter jobs earlier regardless of weight?

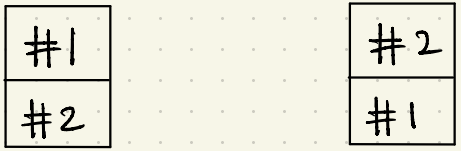
RESOLVING CONFLICTING ADVICE

Q. What if heavier jobs are not shorter ($w_i > w_j$ but $l_i > l_j$)?

Schedule heavier jobs earlier regardless of length?

$l_1 = 1$ $l_2 = 100$

$w_1 = 1$ $w_2 = 2$

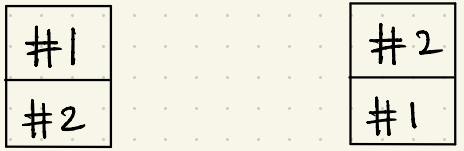


$1 \times 101 + 2 \times 100 > 1 \times 1 + 2 \times 101$

Schedule shorter jobs earlier regardless of weight?

$l_1 = 2$ $l_2 = 1$

$w_1 = 100$ $w_2 = 1$



$100 \times 3 + 1 \times 1 > 100 \times 2 + 1 \times 3$

RESOLVING CONFLICTING ADVICE

Q. What if heavier jobs are not shorter ($w_i > w_j$ but $l_i > l_j$)?

Idea: Aggregate the weight and length parameters into a single "score"
(higher score $\Rightarrow$ scheduled earlier)

RESOLVING CONFLICTING ADVICE

Q. What if heavier jobs are not shorter ($w_i > w_j$ but $l_i > l_j$)?

Idea: Aggregate the weight and length parameters into a single "score"
(higher score $\Rightarrow$ scheduled earlier)

Consistent: Score function should be

* increasing in weight
* decreasing in length

RESOLVING CONFLICTING ADVICE

Q. What if heavier jobs are not shorter ($w_i > w_j$ but $l_i > l_j$)?

Idea: Aggregate the weight and length parameters into a single "score"
(higher score $\Rightarrow$ scheduled earlier)

Consistent: score function should be

* increasing in weight
* decreasing in length

Proposal 1:

$$\text{score}(j) = w_j - l_j$$

Proposal 2:

$$\text{score}(j) = \frac{w_j}{l_j}$$

RESOLVING CONFLICTING ADVICE

Q. What if heavier jobs are not shorter ($w_i > w_j$ but $l_i > l_j$)?

Idea: Aggregate the weight and length parameters into a single "score"
(higher score $\Rightarrow$ scheduled earlier)

Consistent: score function should be

* increasing in weight
* decreasing in length

Proposal 1:

$$\text{score}(j) = w_j - l_j$$

Proposal 2:

$$\text{score}(j) = \frac{w_j}{l_j}$$

different $\Rightarrow$ at least one of these has to be wrong!

Breaking a greedy algorithm

BREAKING A GREEDY ALGORITHM

Proposal 1 : schedule in decreasing order of $w_j - l_j$

Proposal 2 : schedule in decreasing order of w_j / l_j.

BREAKING A GREEDY ALGORITHM

Proposal 1 : schedule in decreasing order of $w_j - l_j$

Proposal 2 : schedule in decreasing order of w_j / l_j.

E.g., $l_1 = 5$ $l_2 = 2$

$w_1 = 3$ $w_2 = 1$

BREAKING A GREEDY ALGORITHM

Proposal 1 : schedule in decreasing order of $w_j - \ell_j$

Proposal 2 : schedule in decreasing order of w_j / ℓ_j.

E.g., $\ell_1 = 5$ $\ell_2 = 2$

$w_1 = 3$ $w_2 = 1$

larger ratio

larger difference

BREAKING A GREEDY ALGORITHM

Proposal 1 : schedule in decreasing order of $w_j - l_j$

Proposal 2 : schedule in decreasing order of w_j / l_j.

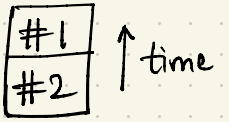
E.g., $l_1 = 5$ $l_2 = 2$

$w_1 = 3$ $w_2 = 1$

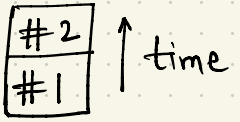
larger ratio

larger difference

order by difference :



order by ratio :



BREAKING A GREEDY ALGORITHM

Proposal 1 : schedule in decreasing order of $w_j - l_j$

Proposal 2 : schedule in decreasing order of w_j / l_j.

E.g., $l_1 = 5$ $l_2 = 2$

$w_1 = 3$ $w_2 = 1$

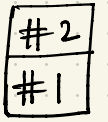
larger ratio

larger difference

order by difference :

$$\sum_{j=1}^{n} w_j \cdot C_j = 1 \times 2 + 3 \times 7 = 23$$

order by ratio :



$$\sum_{j=1}^{n} w_j \cdot C_j = 3 \times 5 + 1 \times 7 = 22$$

BREAKING A GREEDY ALGORITHM

Proposal 1 : schedule in decreasing order of $w_j - l_j$

Proposal 2 : schedule in decreasing order of w_j / l_j.

E.g., $l_1 = 5$ $l_2 = 2$

$w_1 = 3$ $w_2 = 1$

larger ratio

larger difference

order by difference :

$$\sum_{j=1}^{n} w_j \cdot C_j = 1 \times 2 + 3 \times 7 = 23$$

order by ratio :

$$\sum_{j=1}^{n} w_j \cdot C_j = 3 \times 5 + 1 \times 7 = 22$$

Order-by-ratio does better!

CORRECTNESS OF ORDER-BY-RATIO

CORRECTNESS OF ORDER-BY-RATIO

Theorem : Scheduling the jobs in decreasing order of ratios w_j/l_j minimizes the weighted completion time.

CORRECTNESS OF ORDER-BY-RATIO

Theorem: Scheduling the jobs in decreasing order of ratios w_j/l_j minimizes the weighted completion time.

Proof: (by an exchange argument)

CORRECTNESS OF ORDER-BY-RATIO

Theorem: Scheduling the jobs in decreasing order of ratios w_j/l_j minimizes the weighted completion time.

Proof: (by an exchange argument)

Two variants:

* by contradiction

* by massaging

CORRECTNESS OF ORDER-BY-RATIO

Theorem: Scheduling the jobs in decreasing order of ratios w_j/l_j minimizes the weighted completion time.

Proof: (by an exchange argument)

Two variants:

* by contradiction: if greedy is suboptimal, then optimal can be improved

* by massaging

CORRECTNESS OF ORDER-BY-RATIO

Theorem: Scheduling the jobs in decreasing order of ratios w_j/l_j minimizes the weighted completion time.

Proof: (by an exchange argument)

Two variants:

* by contradiction: if greedy is suboptimal, then optimal can be improved
* by massaging: convert any feasible solution into output of greedy algo while improving the objective

CORRECTNESS OF ORDER-BY-RATIO

Theorem: Scheduling the jobs in decreasing order of ratios w_j/l_j minimizes the weighted completion time.

Proof: (by an exchange argument)

Two variants:

* by contradiction: if greedy is suboptimal, then optimal can be improved

* by massaging: convert any feasible solution into output of greedy algo while improving the objective

CORRECTNESS OF ORDER-BY-RATIO

Theorem: Scheduling the jobs in decreasing order of ratios w_j/ℓ_j minimizes the weighted completion time.

Proof: Suppose, for contradiction, that order-by-ratios is not optimal on some input.

CORRECTNESS OF ORDER-BY-RATIO

Theorem: Scheduling the jobs in decreasing order of ratios w_j/ℓ_j minimizes the weighted completion time.

Proof: Suppose, for contradiction, that order-by-ratios is not optimal on some input.

Let σ = greedy schedule and σ^* = optimal schedule.

CORRECTNESS OF ORDER-BY-RATIO

Theorem: Scheduling the jobs in decreasing order of ratios w_j/ℓ_j minimizes the weighted completion time.

Proof: Suppose, for contradiction, that order-by-ratios is not optimal on some input.

Let σ = greedy schedule and σ^* = optimal schedule.

We will construct a schedule better than σ^*, contradicting purported optimality of σ^*.

CORRECTNESS OF ORDER-BY-RATIO

Theorem: Scheduling the jobs in decreasing order of ratios w_j/l_j minimizes the weighted completion time.

Proof: Two assumptions:

CORRECTNESS OF ORDER-BY-RATIO

Theorem: Scheduling the jobs in decreasing order of ratios w_j/ℓ_j minimizes the weighted completion time.

Proof: Two assumptions:

① jobs are indexed in decreasing order of ratios

$$\frac{w_1}{\ell_1} \geq \frac{w_2}{\ell_2} \geq \cdots \geq \frac{w_n}{\ell_n}.$$

CORRECTNESS OF ORDER-BY-RATIO

Theorem: Scheduling the jobs in decreasing order of ratios w_j/ℓ_j minimizes the weighted completion time.

Proof: Two assumptions:

1. jobs are indexed in decreasing order of ratios

$$\frac{w_1}{\ell_1} \geq \frac{w_2}{\ell_2} \geq \cdots \geq \frac{w_n}{\ell_n}.$$

2. no ties between ratios

$$\frac{w_i}{\ell_i} \neq \frac{w_j}{\ell_j} \quad \text{whenever } i \neq j$$

CORRECTNESS OF ORDER-BY-RATIO

Theorem: Scheduling the jobs in decreasing order of ratios w_j/ℓ_j minimizes the weighted completion time.

Proof: Two assumptions:

Without loss of generality (1) jobs are indexed in decreasing order of ratios

$$\frac{w_1}{\ell_1} \geq \frac{w_2}{\ell_2} \geq \cdots \geq \frac{w_n}{\ell_n}.$$

Will remove this assumption later (2) no ties between ratios

$$\frac{w_i}{\ell_i} \neq \frac{w_j}{\ell_j} \quad \text{whenever} \quad i \neq j$$

CORRECTNESS OF ORDER-BY-RATIO

Theorem: Scheduling the jobs in decreasing order of ratios w_j/ℓ_j minimizes the weighted completion time.

Proof: By the two assumptions:

greedy schedule σ is $(1, 2, \cdots, n)$ where

$$\frac{w_1}{\ell_1} > \frac{w_2}{\ell_2} > \cdots > \frac{w_n}{\ell_n}$$

CORRECTNESS OF ORDER-BY-RATIO

Theorem: Scheduling the jobs in decreasing order of ratios w_j/l_j minimizes the weighted completion time.

Proof: By the two assumptions:

greedy schedule σ is $(1, 2, \cdots, n)$ where

$$\frac{w_1}{l_1} > \frac{w_2}{l_2} > \cdots > \frac{w_n}{l_n}$$

By contradiction assumption: $\sigma^* \neq \sigma$

CORRECTNESS OF ORDER-BY-RATIO

Theorem: Scheduling the jobs in decreasing order of ratios w_j/l_j minimizes the weighted completion time.

Proof:

Key observation

Since $\sigma^* \neq \sigma$, there must exist a consecutive inversion

CORRECTNESS OF ORDER-BY-RATIO

Theorem: Scheduling the jobs in decreasing order of ratios w_j/ℓ_j minimizes the weighted completion time.

Proof:

Key observation

Since $\sigma^* \neq \sigma$, there must exist a consecutive inversion i.e., jobs i and j such that

i and j are consecutive in σ^*, and $i > j$.

CORRECTNESS OF ORDER-BY-RATIO

Theorem: Scheduling the jobs in decreasing order of ratios w_j/ℓ_j minimizes the weighted completion time.

Proof:

Key observation

Since $\sigma^* \neq \sigma$, there must exist a consecutive inversion i.e., jobs i and j such that

i and j are consecutive in σ^*, and $i > j$.

[only schedule where indices always go up is $\sigma = (1, 2, \cdots, n)$]

CORRECTNESS OF ORDER-BY-RATIO

Theorem: Scheduling the jobs in decreasing order of ratios w_j/ℓ_j minimizes the weighted completion time.

Proof:

Thought experiment

Suppose we exchange the order of i and j in σ^* without making any other changes.

CORRECTNESS OF ORDER-BY-RATIO

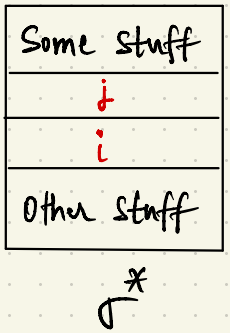
Theorem: Scheduling the jobs in decreasing order of ratios w_j/ℓ_j minimizes the weighted completion time.

Proof:

Thought experiment

Suppose we exchange the order of i and j in σ^* without making any other changes.

$i > j$



σ^*

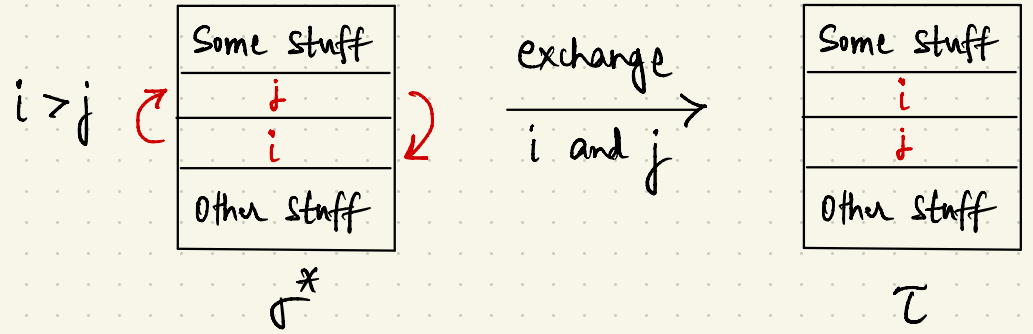
CORRECTNESS OF ORDER-BY-RATIO

Theorem: Scheduling the jobs in decreasing order of ratios w_j/ℓ_j minimizes the weighted completion time.

Proof:

Thought experiment

Suppose we exchange the order of i and j in σ^* without making any other changes.



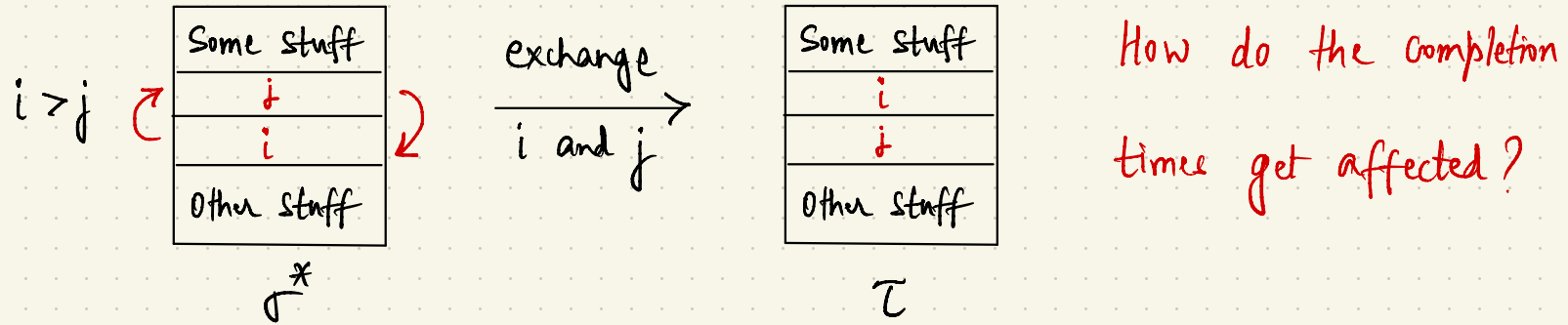
CORRECTNESS OF ORDER-BY-RATIO

Theorem: Scheduling the jobs in decreasing order of ratios w_j/l_j minimizes the weighted completion time.

Proof:

Thought experiment

Suppose we exchange the order of i and j in σ^* without making any other changes.



How do the completion times get affected?

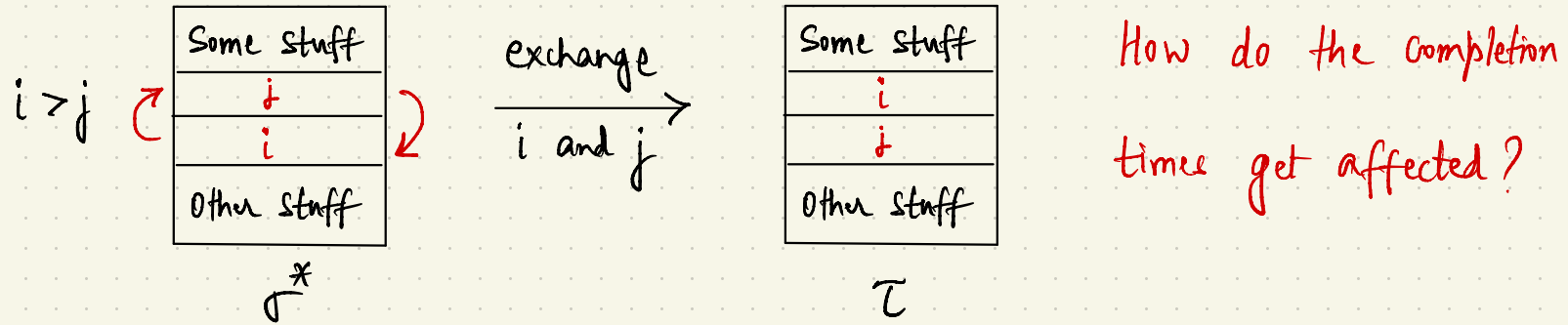
CORRECTNESS OF ORDER-BY-RATIO

Theorem: Scheduling the jobs in decreasing order of ratios w_j/l_j minimizes the weighted completion time.

Proof: Completion time of job i ?

" " " j ?

" " " $k \neq i, j$?



How do the completion times get affected?

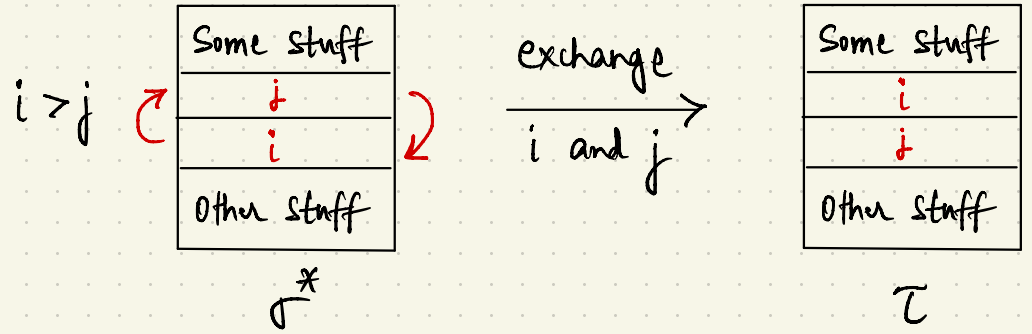
CORRECTNESS OF ORDER-BY-RATIO

Theorem: Scheduling the jobs in decreasing order of ratios w_j/l_j minimizes the weighted completion time.

Proof: Completion time of job i increases by l_j

" " " j ?

" " " $k \neq i, j$?



How do the completion times get affected?

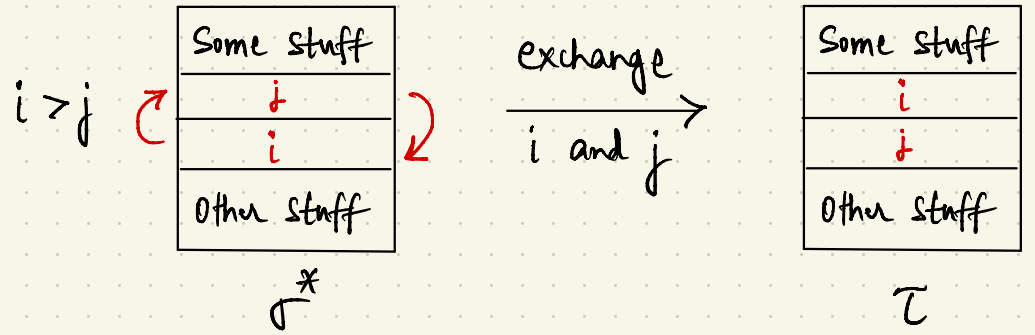
CORRECTNESS OF ORDER-BY-RATIO

Theorem: Scheduling the jobs in decreasing order of ratios w_j/l_j minimizes the weighted completion time.

Proof: Completion time of job i increases by l_j

" " " j decreases by l_i

" " " $k \neq i, j$?



σ^* τ

How do the completion times get affected?

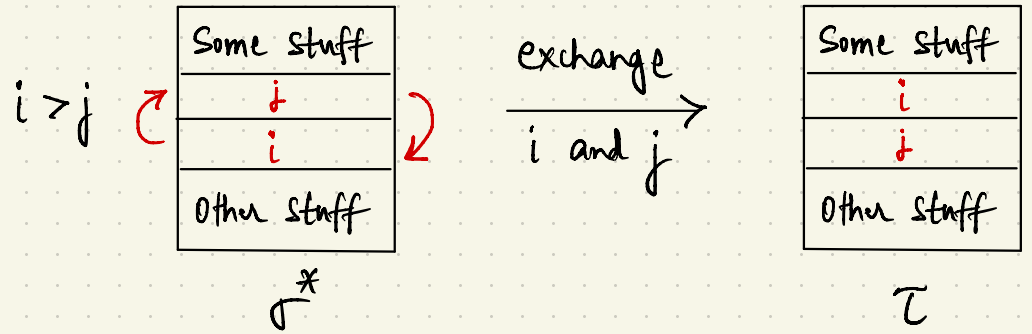
CORRECTNESS OF ORDER-BY-RATIO

Theorem: Scheduling the jobs in decreasing order of ratios w_j/ℓ_j minimizes the weighted completion time.

Proof: Completion time of job i increases by ℓ_j

" " " j decreases by ℓ_i

" " " $k \neq i, j$ stays the same



How do the completion times get affected?

CORRECTNESS OF ORDER-BY-RATIO

Theorem : Scheduling the jobs in decreasing order of ratios w_j/l_j minimizes the weighted completion time.

Proof : Cost-benefit analysis

CORRECTNESS OF ORDER-BY-RATIO

Theorem: Scheduling the jobs in decreasing order of ratios w_j/ℓ_j minimizes the weighted completion time.

Proof:

Cost-benefit analysis

cost of exchange $w_i \cdot \ell_j$ (C_i goes up by ℓ_j)

CORRECTNESS OF ORDER-BY-RATIO

Theorem: Scheduling the jobs in decreasing order of ratios w_j/ℓ_j minimizes the weighted completion time.

Proof:

Cost-benefit analysis

cost of exchange $w_i \cdot \ell_j$ (C_i goes up by ℓ_j)

benefit of exchange $w_j \cdot \ell_i$ (C_j goes down by ℓ_i)

CORRECTNESS OF ORDER-BY-RATIO

Theorem: Scheduling the jobs in decreasing order of ratios w_j/ℓ_j minimizes the weighted completion time.

Proof:

Cost-benefit analysis

cost of exchange $w_i \cdot \ell_j$ (C_i goes up by ℓ_j)

benefit of exchange $w_j \cdot \ell_i$ (C_j goes down by ℓ_i)

$$\sum_{k=1}^{n} w_k \cdot C_k(\tau) = \sum_{k=1}^{n} w_k \cdot C_k(\sigma^*) + w_i \ell_j - w_j \ell_i$$

objective for τ | objective for σ^* | effect of exchange

CORRECTNESS OF ORDER-BY-RATIO

Theorem: Scheduling the jobs in decreasing order of ratios w_j/l_j minimizes the weighted completion time.

Proof:

$$\sum_{k=1}^{n} w_k \cdot C_k(\tau) = \sum_{k=1}^{n} w_k \cdot C_k(\sigma^*) + w_i l_j - w_j l_i$$

CORRECTNESS OF ORDER-BY-RATIO

Theorem: Scheduling the jobs in decreasing order of ratios w_j/ℓ_j minimizes the weighted completion time.

Proof:

$$\sum_{k=1}^{n} w_k \cdot C_k(\tau) = \sum_{k=1}^{n} w_k \cdot C_k(\sigma^*) + w_i \ell_j - w_j \ell_i$$

Recall: $i > j \Rightarrow \frac{w_i}{\ell_i} < \frac{w_j}{\ell_j}$

CORRECTNESS OF ORDER-BY-RATIO

Theorem: Scheduling the jobs in decreasing order of ratios w_j/l_j minimizes the weighted completion time.

Proof:

$$\sum_{k=1}^{n} w_k \cdot C_k(\tau) = \sum_{k=1}^{n} w_k \cdot C_k(\sigma^*) + w_i l_j - w_j l_i$$

Recall: $i > j \Rightarrow \frac{w_i}{l_i} < \frac{w_j}{l_j} \Rightarrow w_i \cdot l_j < w_j \cdot l_i$

CORRECTNESS OF ORDER-BY-RATIO

Theorem: Scheduling the jobs in decreasing order of ratios w_j/ℓ_j minimizes the weighted completion time.

Proof:

$$\sum_{k=1}^{n} w_k \cdot C_k(\tau) = \sum_{k=1}^{n} w_k \cdot C_k(\sigma^*) + w_i \ell_j - w_j \ell_i$$

Recall: $i > j \Rightarrow \frac{w_i}{\ell_i} < \frac{w_j}{\ell_j} \Rightarrow w_i \cdot \ell_j < w_j \cdot \ell_i$

$\Rightarrow$ cost < benefit

CORRECTNESS OF ORDER-BY-RATIO

Theorem: Scheduling the jobs in decreasing order of ratios w_j/l_j minimizes the weighted completion time.

Proof:

$$\sum_{k=1}^{n} w_k \cdot C_k(\tau) = \sum_{k=1}^{n} w_k \cdot C_k(\sigma^*) + w_i l_j - w_j l_i$$

Recall: $i > j \Rightarrow \frac{w_i}{l_i} < \frac{w_j}{l_j} \Rightarrow w_i \cdot l_j < w_j \cdot l_i$

$\Rightarrow$ cost < benefit

$\Rightarrow$ τ is better than σ^*

CORRECTNESS OF ORDER-BY-RATIO

Theorem: Scheduling the jobs in decreasing order of ratios w_j/l_j minimizes the weighted completion time.

Proof:

$$\sum_{k=1}^{n} w_k \cdot C_k(\tau) = \sum_{k=1}^{n} w_k \cdot C_k(\sigma^*) + w_i l_j - w_j l_i$$

Recall: $i > j \Rightarrow \frac{w_i}{l_i} < \frac{w_j}{l_j} \Rightarrow w_i \cdot l_j < w_j \cdot l_i$

$\Rightarrow$ cost < benefit

$\Rightarrow$ τ is better than σ^*

Contradiction!

CORRECTNESS OF ORDER-BY-RATIO

Theorem: Scheduling the jobs in decreasing order of ratios w_j/l_j minimizes the weighted completion time.

Proof:

$$\sum_{k=1}^{n} w_k \cdot C_k(\tau) = \sum_{k=1}^{n} w_k \cdot C_k(\sigma^*) + w_i l_j - w_j l_i$$

Recall: $i > j \Rightarrow \frac{w_i}{l_i} < \frac{w_j}{l_j} \Rightarrow w_i \cdot l_j < w_j \cdot l_i$

$\Rightarrow$ cost < benefit

$\Rightarrow$ τ is better than σ^*

Contradiction!

∎