

COL 351 : ANALYSIS & DESIGN OF ALGORITHMS

LECTURE 14

GREEDY ALGORITHMS I : JOB SCHEDULING

AUG 25, 2026

|

ROHIT VAISH

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



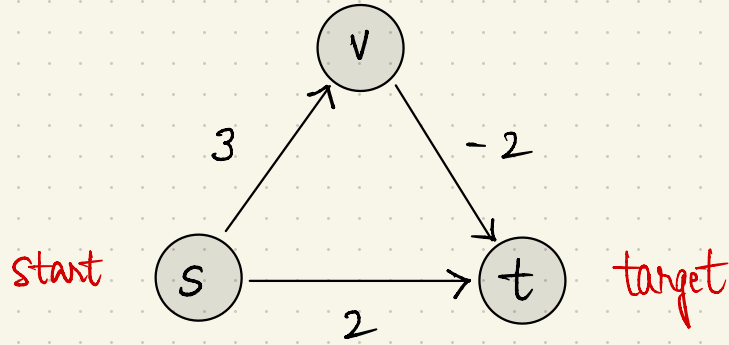
Greedy algorithms



	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 .

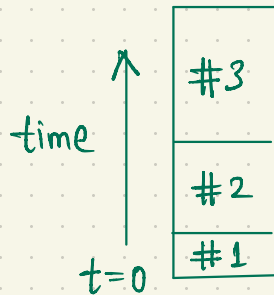
E.g., 3 jobs, lengths $l_1 = 1$, $l_2 = 2$, $l_3 = 3$

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



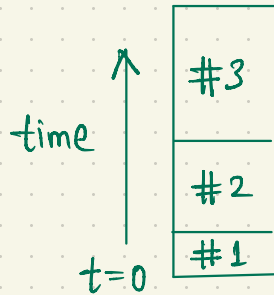
COMPLETION TIMES

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



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

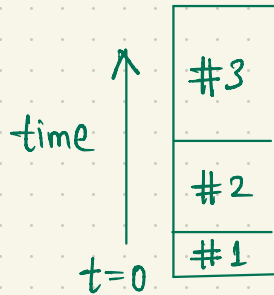
COMPLETION TIMES

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

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

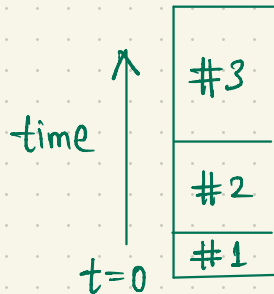
E.g., 3 jobs , lengths $l_1 = 1$, $l_2 = 2$, $l_3 = 3$
weights $w_1 = 3$, $w_2 = 2$, $w_3 = 1$

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

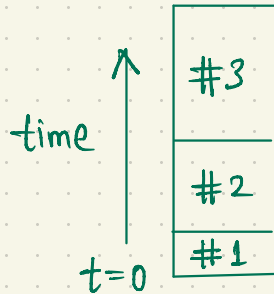


OBJECTIVE FUNCTION

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



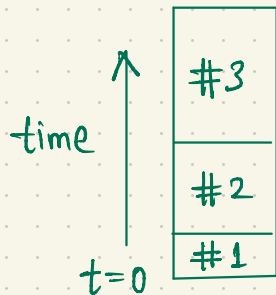
$$\sum_{j=1}^n w_j \cdot C_j =$$

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



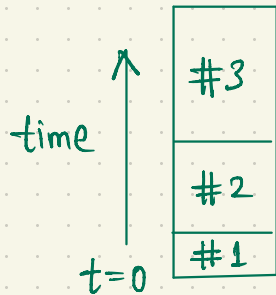
$$\sum_{j=1}^n w_j \cdot C_j = \underset{w_1}{3} \times \underset{C_1}{1} + \underset{w_2}{2} \times \underset{C_2}{3} + \underset{w_3}{1} \times \underset{C_3}{6}$$

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



$$\begin{aligned} \sum_{j=1}^n w_j \cdot C_j &= \underset{w_1}{3} \times \underset{C_1}{1} + \underset{w_2}{2} \times \underset{C_2}{3} + \underset{w_3}{1} \times \underset{C_3}{6} \\ &= 15 \quad (\text{verify optimality!}) \end{aligned}$$

JOB SCHEDULING

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

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 $l_1, l_2, \dots, l_n$
and positive weights $w_1, w_2, \dots, w_n$

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 $l_1, l_2, \dots, l_n$
and positive weights $w_1, w_2, \dots, w_n$

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

$l_1 = 1, l_2 = 3$
 $\nearrow (1, 2) \quad C_1 = 1 \quad C_2 = 4$
 $\searrow (2, 1) \quad C_1 = 3 \quad C_2 = 4$

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*?

$$l_1 = 1 \quad l_2 = 100$$

$$w_1 = 1 \quad w_2 = 2$$

#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*?

$$l_1 = 1 \quad l_2 = 100$$

$$w_1 = 1 \quad w_2 = 2$$

#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*?

$$l_1 = 1 \quad l_2 = 100$$

$$w_1 = 1 \quad w_2 = 2$$

#1
#2

#2
#1



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

Schedule shorter jobs earlier *regardless of weight*?

$$l_1 = 2 \quad l_2 = 1$$

$$w_1 = 100 \quad w_2 = 1$$

#1
#2

#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 $w_j - l_j$

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

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., $l_1 = 5$ $l_2 = 2$
 $w_1 = 3$ $w_2 = 1$

order by difference :

#1	↑ time
#2	

larger
ratio

larger
difference

order by ratio :

#2	↑ time
#1	

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:

#1	↑ time
#2	

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

order by ratio:

#2	↑ time
#1	

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

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:

#1	↑ time
#2	

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

order by ratio:

#2	↑ time
#1	

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

Order-by-ratio does better!