

CS 331, Fall 2026
Lecture 8 (9/21)

- Today:
- Fractional knapsack
 - Rearrangement
 - Completion times
 - Minimizing lateness

Introduction (Part IV, Section 1)

In DP, we carefully made decisions via recursion + memoization.

Example Unbounded knapsack

$$S(B) = \max_{c \in \mathbb{Z}_{\geq 0}^n} \sum_{i \in [n]} c_i V(i) \quad \text{s.t.}$$

$$\sum_{i \in [n]} c_i W(i) \leq B$$

$$S(B) = \max_{\substack{i \in [n] \\ W(i) \leq B}} S(B - W(i)) + V(i)$$

Carefully consider all n options

In greedy, we pick based on a rule.

"Myopic" = near-sighted: might regret later!

Example

Greedy 0/1 Knapsack (W, V, B):

Sort W, V similarly according to ...

$(i, tot) = (1, 0)$

While $i \in [n]$:

If $W(i) \leq B$: // take best item until can't

$(tot, B) \leftarrow (tot + V(i), B - W(i))$

Else:

$i \leftarrow i + 1$

Return tot

Very simple and intuitive. Is it optimal?

How to sort?

... = by value (highest first)?

Nope. $W = [1, 3]$, $V = [2, 3]$, $B = 4$

... = by value density $\frac{V(i)}{W(i)}$ (highest first)?

Nope. $W = [2, 3]$, $V = [4, 7]$, $B = 4$

General rules about greedy.

- Sort inputs + repeatedly take current "best"
- Runtime analysis: usually easy.
- Correctness analysis: much harder.

Beware of counterexamples!

Don't trust until you've tried to break it.

When does greedy work?

Key idea: exchange

- 1) Define greedy method. Produces solution **ALG**
- 2) Suppose there's some other solution **OPT** that is optimal for your objective function f
- 3) Transform **OPT** $\rightarrow$ **ALG**

Method 1: partial transform

$$f(\text{OPT}) \leq f(s_1) \leq \dots \leq f(s_k) \leq f(\text{ALG})$$

Method 2: Contradiction

Assume $\text{OPT} = \underset{s}{\operatorname{argmax}} f(s) \neq \text{ALG}$

$s' = \text{OPT}$ modified using info about **ALG**

Prove that $f(s') > f(\text{OPT}) \Rightarrow \Leftarrow$

Fractional unbounded knapsack (Part IV, Section 2.1)

$$S(B) = \max_{c \in \mathbb{R}_{\geq 0}^n} \sum_{i \in [n]} c_i V(i) \quad \text{s.t.}$$

don't need
to be integers

$$\sum_{i \in [n]} c_i W(i) \leq B$$

Simplification: normalize weights

$$W(i) \leftarrow 1$$

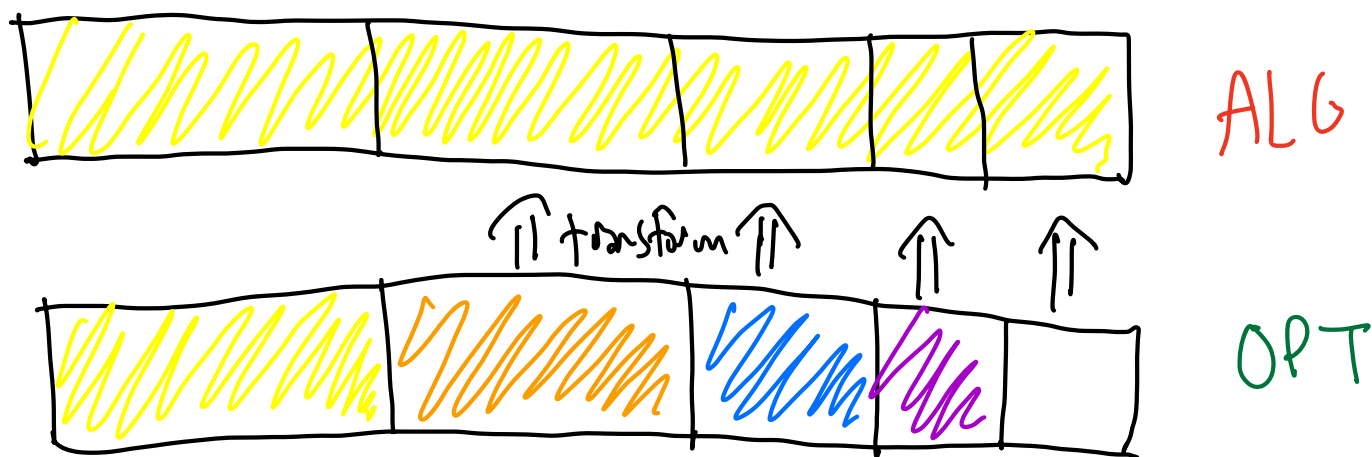
(just take $W(i) \times$ more)

$$V(i) \leftarrow \frac{V(i)}{W(i)} = D(i)$$

"density"

$$\max_{c \in \mathbb{R}_{\geq 0}^n} \sum_{i \in [n]} c_i D(i) \quad \text{s.t.} \quad \sum_{i \in [n]} c_i \leq B$$

Claim: c_i^{ALG} optimal. $c_i^{\text{ALG}} = \begin{cases} B & i = i^* = \arg\max_{i \in [n]} D(i) \\ 0 & \text{else} \end{cases}$



Rearrangement Lemma:

$$\text{If } a \geq b, c \geq d, \quad ac + bd \geq ad + bc$$

amounts values take the bigger thing more!

Proof: $(a-b)(c-d) \geq 0 \rightarrow \text{FOIL}$

Why is C^{ALG} optimal?

$\forall i \neq i^*, \text{ transform } C_i^{OPT} \times D(i) \leq C_i^{OPT} \times D(i^*)$

After all transformations, left with

$$\left(-0-, \sum_{i \in G} C_i^{OPT} \leq B, -0- \right)$$

Completion times (Part IV, Section 2.2)

Input: T : list of n #s ≥ 0

(durations of jobs)

Output: $\min \sum_{i \in [n]} e_i$

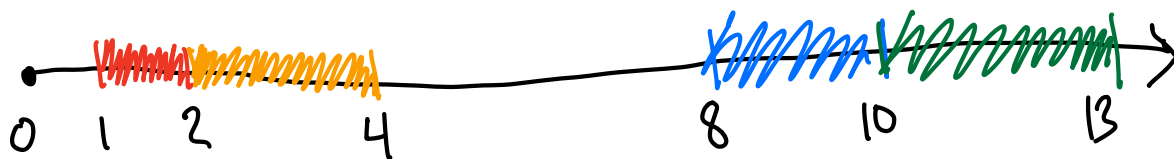
e_i = end time of i th job

i th job gets interval $[s_i, e_i = s_i + T(i)] \subset \mathbb{R}_{\geq 0}$

Non-overlapping intervals

Example

$$T = [1, 2, 3, 2]$$



$$e_1 + e_2 + e_3 + e_4 = 2 + 4 + 13 + 8 = 27$$

Observation 1: no idle time

Pick a permutation $\pi: [n] \rightarrow [n]$



$$e_{\pi(i)} = \sum_{j \in [i)} T[\pi(j)] \quad (\text{total duration of preceding jobs})$$

$$\text{Objective: } \min_{\pi: [n] \rightarrow [n]} f(\pi), \quad f(\pi) = \sum_{i \in [n]} e_{\pi(i)}$$

Observation 2: Simplify expression

$$f(\pi) = T[\pi(1)] + (T[\pi(1)] + T[\pi(2)]) + \dots$$

$$= n \cdot T[\pi(1)] + (n-1) \cdot T[\pi(2)] + \dots + T[\pi(n)]$$

Aside

Rearrangement inequality

Suppose $a_1 \geq a_2 \geq \dots \geq a_n$ (counts)

$b_1 \geq b_2 \geq \dots \geq b_n$ (values)

Then, $a_1 b_1 + a_2 b_2 + \dots + a_n b_n$

$\geq a_1 b_{\pi(1)} + a_2 b_{\pi(2)} + \dots + a_n b_{\pi(n)}$

$\geq a_1 b_n + a_2 b_{n-1} + \dots + a_n b_1$

Proof: Suppose π not identity ("ALG")

It has inversion: $\pi(i) > \pi(j), i < j$

Apply rearrangement lemma. Value goes up

Repeat until no inversions. $ALG \geq \pi$

Other way similar, max inversions rather than min.

Punchline

$$f(\pi) = n \cdot T(\pi(1)) + (n-1) \cdot T(\pi(2)) + \dots + T(\pi(n))$$

Minimal if $T(\pi)$ sorted backwards from counts

$(n, n-1, \dots, 1) \Rightarrow T$ nondecreasing, optimal.

Weighted completion times (Part IV, Section 3.1)

Same idea. One more input:

W , list of n #s ≥ 0 (weights of jobs)

Output: $\min \sum_{i \in \pi} w(i) e_i$

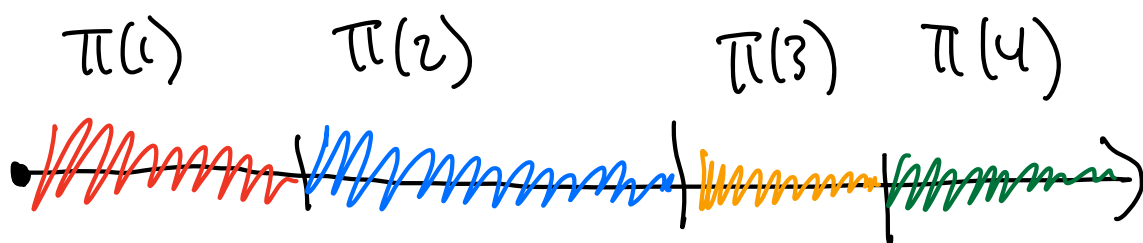
Can we still get away with greedy?

Claim: Sorting by $\frac{T(i)}{w(i)}$ is optimal.

Assume $\frac{T(1)}{w(1)} \leq \dots \leq \frac{T(n)}{w(n)}$

Then, $\pi = \text{identity}$ optimal for

$$f(\pi) = \sum_{i \in [n]} w(\pi(i)) \underbrace{\left(\sum_{j \in (i)} T(\pi(j)) \right)}_{= r_{\pi(i)}}$$



Idea: exchange argument.

How to partially transform?

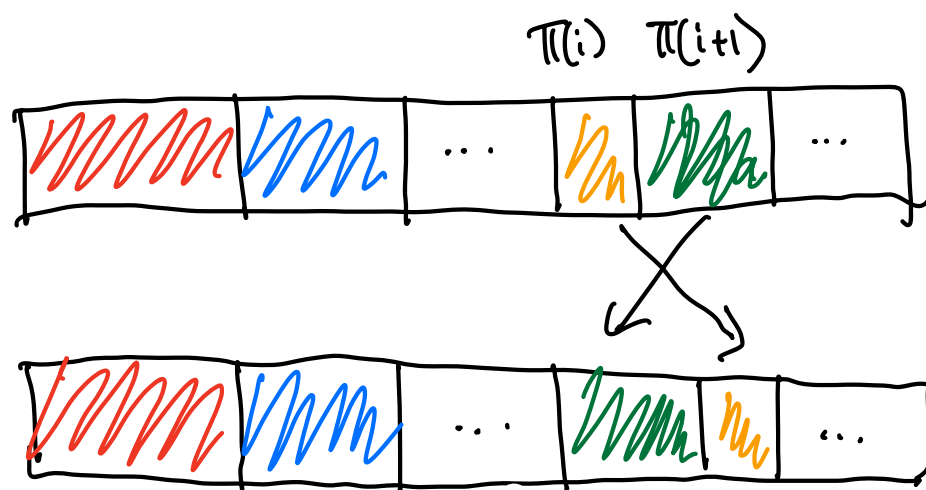
Not so clear to undo inversions:

$$f(\pi) = \left(\underbrace{w(\pi(1)) + \dots + w(\pi(n))}_{\text{red underline}} \right) \cdot T[\pi(1)] \\ + \left(\underbrace{w(\pi(2)) + \dots + w(\pi(n))}_{\text{red underline}} \right) \cdot T[\pi(2)] \\ + \dots + \underbrace{w(\pi(n))}_{\text{red underline}} \cdot T[\pi(n)]$$

Weights per duration are not fixed.

Cannot directly apply rearrangement inequality.

Fix: undo adjacent inversions only. Always exists!



All $e_{\pi(j)}$ stay same except $j=i, i+1$.

Let $i, i+1$ be adjacent inversion.

$$\pi(i) > \pi(i+1) \Rightarrow \frac{T[\pi(i)]}{W[\pi(i)]} \geq \frac{T[\pi(i+1)]}{W[\pi(i+1)]}$$

$$\text{Let } \pi'(j) = \begin{cases} \pi(i+1) & j=i \\ \pi(i) & j=i+1 \\ \pi(j) & \text{else} \end{cases}$$

$$\begin{aligned} f(\pi') - f(\pi) &= \sum_{j \in [n]} W[\pi'(j)] \cdot e_{\pi'(j)} - W[\pi(j)] \cdot e_{\pi(j)} \\ &= W[\pi'(i+1)] \cdot e_{\pi'(i+1)} + W[\pi'(i)] \cdot e_{\pi'(i)} \\ &\quad - W[\pi(i+1)] \cdot e_{\pi(i+1)} + W[\pi(i)] \cdot e_{\pi(i)} \\ &= W[\pi(i)] \cdot T[\pi(i+1)] \\ &\quad - W[\pi(i+1)] \cdot T[\pi(i)] \leq 0. \end{aligned}$$

Hence exchange improves. $f(\text{identity}) \leq f(\pi)$

Minimizing lateness (Part IV, Section 3.2)

Input: T : list of n #s ≥ 0

(durations of jobs)

D : list of n #s ≥ 0

(deadlines of jobs)

Output: minimize $\max_{i \in [n]} \underbrace{e_i - D[i]}_{\text{lateness}}$

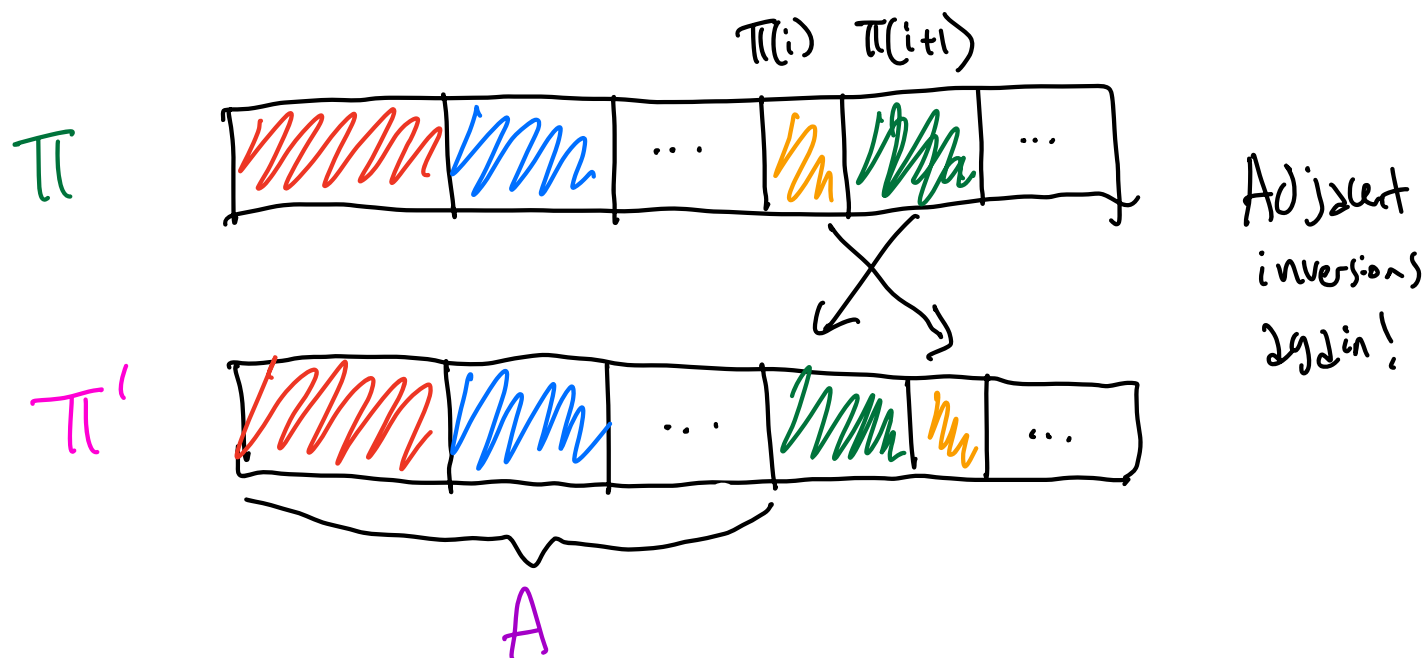
i^{th} job gets $(s_i, e_i = s_i + T[i]) \subset \mathbb{R}_{\geq 0}$

nonoverlapping intervals.

Claim: Sort T, D similarly so D nondecreasing.

$$D[1] \leq D[2] \leq \dots \leq D[n]$$

Then optimal to have $e_i = \sum_{j \in [i]} T[j]$.



$$f(\pi) = \max_{i \in [n]} \sum_{j \in [i]} T[\pi(j)] - D[\pi(i)]$$

Claim: $f(\pi') \leq f(\pi)$ so identity optimal.

Proof: If $\arg \max$ for $\pi' \notin \{i, i+1\}$, $\checkmark$

Else, let $A = \sum_{j \in [i-1]} T[\pi(j)]$

$$\pi': \max \left\{ \begin{aligned} &A + T[\pi(i+1)] - D[\pi(i+1)], \\ &A + T[\pi(i)] + T[\pi(i+1)] - D[\pi(i)] \end{aligned} \right\}$$

$$\pi: A + T[\pi(i)] + T[\pi(i+1)] - \underbrace{D[\pi(i+1)]}_{\leq D[\pi(i)]}$$