CS 331, Fall 2024
Lecture 8 (9/23)

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

Introduction (Part IV, Section 1)

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

$\boxed{\text{Example}}$ Unbounded knapsack

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

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

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

Carefully consider all n options

In greedy, we pick based on a rule.

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

Example

```
GreedyUBKnapsack(W, V, B):
  Sort W, V similarly according to ...
  (i, tot) = (1, 0)
  While i ∈ [n]:
    If W[i] ≤ B:   // take best item until can't
      (tot, B) ← (tot + V[i], B - W[i])
    Else:
      i ← 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: <u>exchange</u>

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

$$f(OPT) \le f(S_1) \le \dots \le f(S_k) \le f(ALG)$$

Method 2: Contradiction

$$OPT = \arg\max_S f(S)$$

$S' =$ OPT modified using info about ALG

Prove that $f(S') > f(OPT)$ $\Rightarrow\Leftarrow$

Fractional unbounded knapsack (Part IV, Section 2.1)

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

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

Simplification: normalize weights

$W[i] \leftarrow 1$

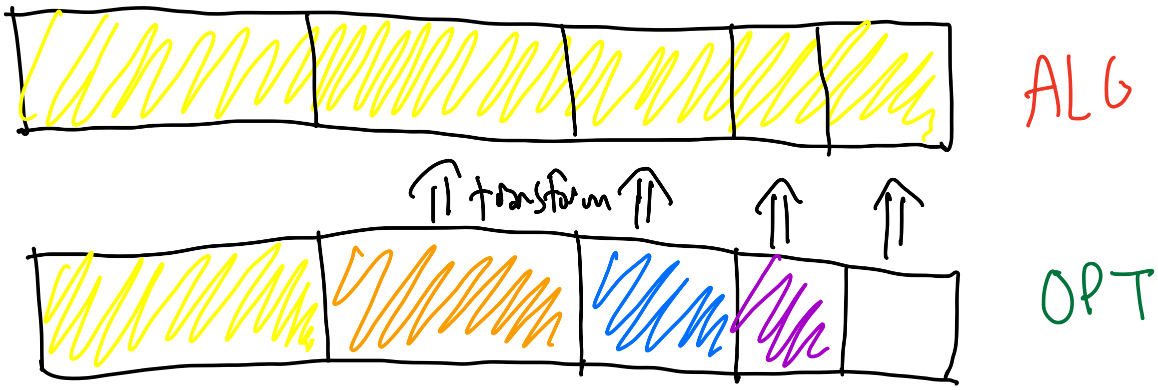
$V[i] \leftarrow \frac{V[i]}{W[i]}$

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

$$\max_{c \in \mathbb{R}^n_{\geq 0}} \sum_{i \in [n]} c_i \frac{V[i]}{W[i]} \quad \text{s.t.} \quad \sum_{i \in [n]} c_i \leq B$$

Claim: c^{ALG} optimal. $c_i^{ALG} = B$ $\quad i = i^* = \operatorname{argmax}_{i \in [n]} \frac{V[i]}{W[i]}$

0 $\quad$ else



Rearrangement Lemma:

If $a \geq b$, $c \geq d$, $ac + bd \geq ad + bc$

amounts values take the bigger thing more!

Proof: $(a-b)(c-d) \geq 0$ + FOIL

Why is c^{ALG} optimal?

$\forall i \neq i^*$, transform $c_i^{OPT} \times V(i) \leq c_i^{OPT} \times V(i^*)$

After all transformations, left with

$$\left(-0-, \sum_{i \in [n]} c_i^{OPT} \leq B, -0- \right)$$

Completion times (Part IV, Section 2.2)

Input: T : list of n #s ≥ 0

(durations of jobs)

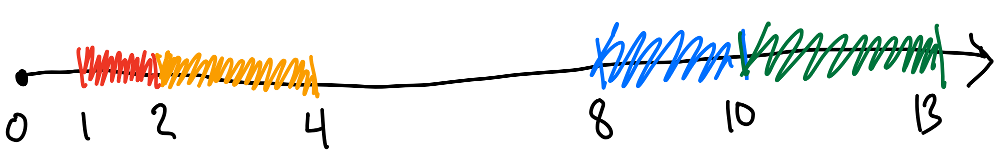
Output: $\min \sum_{i \in [n]} e_i$

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

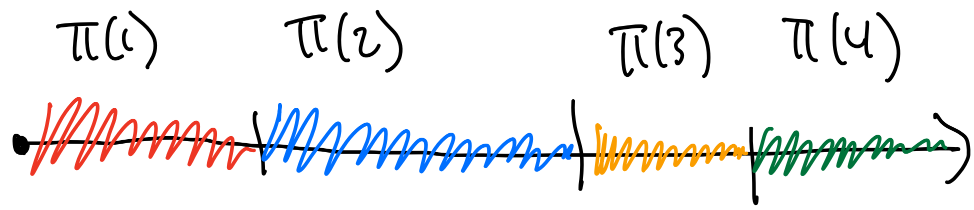
$\boxed{\text{Example}}$ $T = [1, 2, 3, 2]$



$e_1 + e_2 + e_3 + e_4 = 2 + 10 + 13 + 4 = 29$

Observation 1: no idle time

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



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

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

Observation 2: Simplify expression

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

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

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: $\min \sum_{i \in [n]} W(i) e_i$

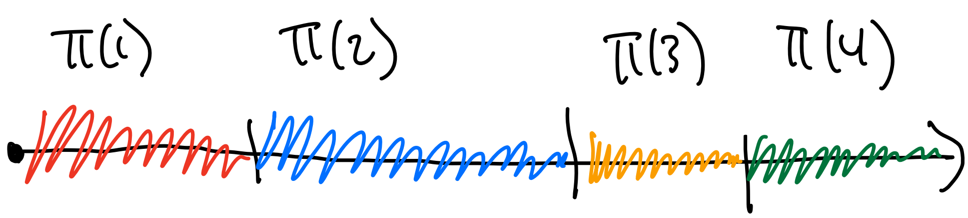
Can we still get away with greedy?

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

Assume $\frac{T(1)}{W(1)} \le \dots \le \frac{T(n)}{W(n)}$

Then, $\pi =$ identity optimal for

$$f(\pi) = \sum_{i \in [n]} W[\pi(i)] \Big(\underbrace{\sum_{j \in [i]} T(\pi(j))}_{= e_{\pi(i)}} \Big)$$



Idea: exchange argument.

How to partially transform?

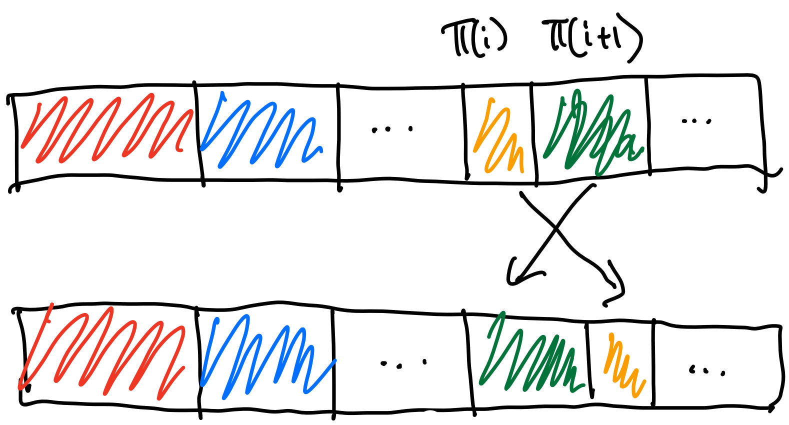
Not so clean to undo inversions:

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

Weights per duration are not fixed.

Cannot directly apply rearrangement inequality.

Fix: undo <u>adjacent inversions</u> only. Always exists!



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

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

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

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

$$f(\pi') - f(\pi) = \sum_{j \in [n]} W[\pi'(j)] \cdot e_{\pi'(j)} - W[\pi(j)] \cdot e_{\pi(j)}$$

$$= W[\pi'(i+1)] \cdot e_{\pi'(i+1)} + W[\pi'(i)] \cdot e_{\pi'(i)}$$

$$- W[\pi(i+1)] \cdot e_{\pi(i+1)} + W[\pi(i)] \cdot e_{\pi(i)}$$

$$= W[\pi(i)] \cdot T[\pi(i+1)]$$

$$- W[\pi(i+1)] \cdot T[\pi(i)] \leq 0.$$

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 $\max\limits_{i \in [n]} \underbrace{e_i - D[i]}_{\text{lateness}}$

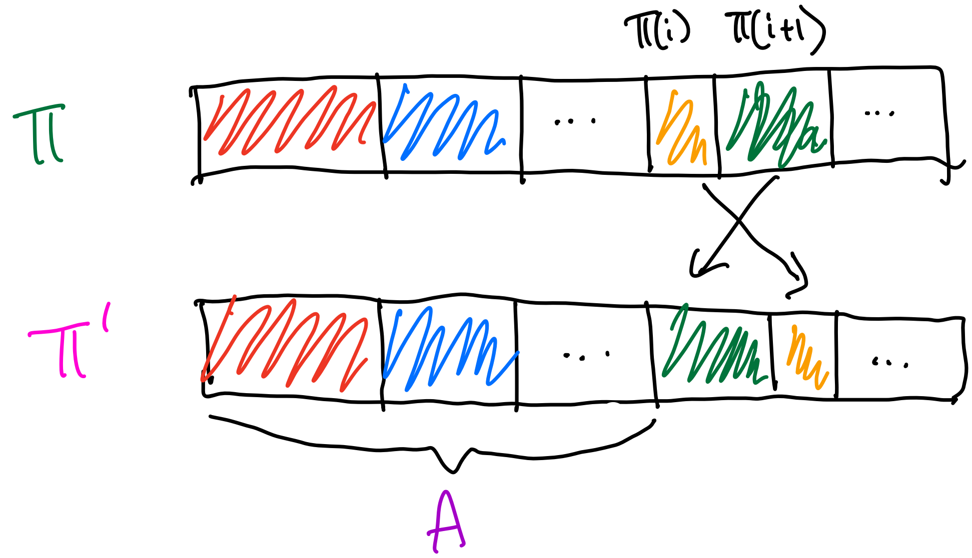
i^{th} job gets $[s_i, e_i = s_i + T[i]] \subset \mathbb{R}_{\geq 0}$

nonoverlapping intervals.

Claim: Sort T, D similarly so D nondecreasing.

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

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



Adjacent inversions again!

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

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

Proof: If argmax for $\pi' \notin \{i, i+1\}$, ✓

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

$$\pi' : \max\{A + T[\pi(j+1)] - D[\pi(j+1)],\ A + T[\pi(j)] + T[\pi(j+1)] - D[\pi(j)]\}$$

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