**Greedy Algorithms:**

To solve some optimization problems, incrementally building the solution at each step, what seems best "locally".

Adv: Simple & fast

Disadv: May not be correct

If correct, then needs a proof.

---

**Ex. 1: Interval Scheduling.**

Given $n$ intervals $[s_i, t_i], [s_2, t_2], ..., [s_n, t_n]$.

Find the largest subset of non-overlapping intervals.

Minimum # of jobs

---

**eq.**

- Opt: 3

---

**idea 1 (greedy):** Pick job by earliest start time ($\min s_i$)

- $\{3, 5\}$ → 2 jobs fails!

**idea 2 (greedy):** Pick the smallest job ($\min (t_i - s_i)$)

- $\{5, 1\}$ fails!
\[ \{5, 13\} \text{ fails!} \]

idea 3: Pick the job that overlaps \#8 other jobs.

\[ \{6, 2, 13\} \text{ yay!} \]

opt: 4

Greedy: 3

idea 4: Pick job with earliest finish time (\(\min t_i\))

bingo! Works.

Greedy Alg: \[ \text{Greedy solm.} \]

1. repeat \[ \] \[ \Rightarrow \]

2. Pick \([s_i, t_i]\) of smallest \(t_i\).

3. Remove \([s_i, t_i]\) of all intervals overlapping with it.

4. \[ \text{Until intervals left.} \]

5. Output picked intervals.

Running Time: Naive \(O(n^2)\)

Better: Sort in increasing order \(s_i, t_i\), scan, \(O(n \log n)\).

Correctness Proof: \[ I = \{[s_1, b_1], \ldots, [s_m, b_m]\} \]

* \[ \text{or soln} \ (\text{unknown}) \]
Coarseness Proof: \( I^* \) is an opt soln (unknown).

- let \( I^* \) be an opt soln
- let \([s^*, b^*]\) be the left most interval in \( I^* \)
- let \([s_i, t_i]\) be the first interval picked by Greedy Alg.

\[ I^* = \{[s^*, b^*]\} \cup \{[s_i, t_i]\} \]

Know: \( t_i \leq b^* \)

\[ I^* = \{[s^*, b^*]\} \cup \{[s_i, t_i]\} \]

is feasible.

AND has the same # intervals as

\[ I^*. \]

Reset \( I^* = I^* - \{[s^*, b^*]\} \cup \{[s_i, t_i]\} \)

is an optimal soln.

exchange argument.

Remove \([s_i, t_i]\) & all intervals overlapping w/ it.

Repeat the argument on smaller instance

\[ \Rightarrow \]

Induction.

idea 5: Pick latest start time. (max \( s_i \)).

Note: This must extend to weighted.
Ex. 2: Job scheduling to minimize average wait time.

Given n jobs and processing times $p_1, p_2, \ldots, p_n$.
Find an ordering that minimizes total wait time.

Ordering: $p_1, p_2, \ldots, p_n$

Cost = $0 + p_1 + (p_1 + p_2) + \ldots + (p_1 + p_2 + \ldots + p_{n-1})$

(total time) = $(n-1)p_1 + (n-2)p_2 + \ldots + (n-i)p_i + \ldots + p_{n-1}$

<table>
<thead>
<tr>
<th>jobs</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proces</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>8</td>
<td>2</td>
</tr>
</tbody>
</table>

Order: 3, 4, 1, 8, 2

Cost = $0 + 3 + (3 + 4) + (3 + 4 + 1) + (3 + 4 + 1 + 8) = 36$

\[ \downarrow \]
Better order: 3, 4, 1, 2, 8

Cost = $0 + 3 + (3 + 4) + (3 + 4 + 1) + (3 + 4 + 1 + 2) = 38$

\[ \downarrow \]
Even better: 3, 1, 4, 2, 8

Cost = $0 + 3 + (3 + 1) + (3 + 1 + 4) + (3 + 1 + 4 + 2) = 35$

Best: 1, 2, 3, 4, 8

\[ \ldots (1+2) + (1+2+3) + (1+2+3+4) \]
best: 1, 2, 3, 4, 8
\[
\text{cost} = 0 + 1 + (1+2) + (1+2+3) + (1+2+3+4) \\
= 20
\]

Greedy Alg’nm: Order in increasing \( P_i \).

Correctness Proof:

Let \( P_1^*, P_2^*, \ldots, P_n^* \) be the optimal order.

Suppose it is not sorted.

Then \( 3i: P_i > P_{i+1} \) \( \Rightarrow (P_1^*, \ldots, P_{i-1}^*, P_{i+1}^*, P_i^*, P_{i+2}^*, \ldots, P_n^*) \)

Swap \( i \leftrightarrow i+1 \) to get a new order.

Old cost = \( 0 + P_1^* + (P_2^*+P_3^*) + \ldots + (P_{i-1}^* + P_i^* + P_{i+1}^*) + \ldots + (P_n^* + P_{n-1}^*) \)

Opt cost = \( P_1^* + P_2^* + \ldots + P_n^* \)

New cost = \( 0 + P_1^* + \ldots + (P_i^* + \ldots + P_{i-1}^*) + \ldots + (P_{i+1}^* + \ldots + P_n^*) \)

After swap: \( (P_1^* + \ldots + P_{i-1}^*) + (P_{i+1}^* + \ldots + P_n^*) \)

New cost - Old cost = \( P_i^* - P_i < 0 \) (\( P_i > P_{i+1}^* \))

We get a better sol’n than opt! \( \exists \)

Note: Works for weighted version.

\( \cdots \) for jobs
Note: Works for weighted versions also, given weight \( w_i \) for job \( i \).

Want to minimize weighted total wait-time

\[
\text{cost} = w_1 D + w_2 (P_i) + w_3 (P_i + P_{i+1}) + \ldots + w_n (P_i + \ldots + P_{n-1})
\]

Greedy strategy: want increasing \( P_i \)

and decreasing \( w_i \)

Sort in increasing order of \( \frac{P_i}{w_i} \)

Connectors \( P_i^* \): (similar)

Opt: \( P_1^*, P_2^*, \ldots, P_i^*, P_{i+1}^*, \ldots, P_n^* \)

After swap: \( P_1^*, P_2^*, \ldots, P_i^*, P_{i+2}^*, P_{i+1}^*, \ldots, P_n^* \)

Old cost:

\[
= w_1^* D + \ldots + w_i^* (P_i^* + \ldots + P_{i-1}^*) + w_{i+1}^* (P_{i+1}^* + \ldots + P_{i+2}^*) + \ldots + w_n^* (P_{i+1}^* + \ldots + P_n^*)
\]

New cost:

\[
= w_1^* D + \ldots + w_i^* (P_i^* + \ldots + P_{i-1}^*) + w_{i+1}^* (P_{i+1}^* + \ldots + P_{i+2}^*) + \ldots + w_n^* (P_{i+1}^* + \ldots + P_n^*)
\]

New cost - old cost:

\[
\geq w_i^* (P_{i+1}^* - P_i^*) < 0
\]

Thus, \( \frac{P_i^*}{w_i^*} > \frac{P_{i+1}^*}{w_{i+1}^*} \)
\[ \frac{r_i}{\omega_*} > \frac{\omega_i^*}{\omega_{i+1}^*} \]