Basic Algo. Techniques

- Divide & conquer
- Greedy
- Dynamic Pgm.
- Amortized Analysis

Greedy Algorithms

- Algorithm makes a sequence of choices.
- At each step, algo. makes a choice that looks best at that step (without look-ahead).

  w.r.t. specific criterion.

- Such an algo. often would not work.
- It does work on some problems.
- Emphasis on polytime algorithm than optimizing the run-time. Proof of correctness

Simplest example

① - Given n items with values $v_1, \cdots, v_n$.
- You are allowed to take k items.
- Maximize your total value.

- Trivial $\binom{n}{k}$ time algo.
- Greedy Algo:
    - k steps.
    - At each step, pick the item w/ maximum value among remaining items.

Proof of correctness, the $\boxed{\text{Exchange Argument}}$

Claim Let $i_1, i_2, \ldots, i_k$ be indices of items picked by the algorithm. Then for every $0 \le r \le k$, there exists an optimal (best) set $S \subseteq \{1, 2, \ldots, n\}$, $|S| = k$ s.t. $\{i_1, i_2, \ldots, i_r\} \subseteq S$.

Note When the claim is applied with $r = k$, it follows that $\{i_1, i_2, \ldots, i_k\}$ is optimal (best) set.

Proof By induction.

$r = 0$ Clearly $\phi \subseteq S$ for the hypothetical optimal set S.

Assume the claim holds for some $0 \leq r \leq k-1$.

I.e. for some hypothetical optimal set S' we have $\{i_1, i_2, .., i_r\} \subseteq S'$.

Now consider item i_{r+1} chosen by the algorithm

case 1 $i_{r+1} \in S'$. Then

$\{i_1, i_2, .., i_r, i_{r+1}\} \subseteq S'$. Done.

case 2 $i_{r+1} \notin S'$.

Since $r \leq k-1$ and $|S'| = k$, there exists some $j \in S' \setminus \{i_1, i_2, .., i_r\}$.

Since i_{r+1} was the maximum value item after choosing $i_1, i_2, .., i_r$, it must be that $v_j \le v_{i_{r+1}}$. $\boxed{\text{Exchange}}$

$\therefore \; S = (S' \setminus \{j\}) \cup \{i_{r+1}\}$ has value at least that of S', and since S' is already optimal, so is S.

$\therefore \; \{i_1, i_2, \dots, i_r, i_{r+1}\} \subseteq S$, S optimal.

Done. ∎

② Infinitely Divisible Items.

	copper	sand	gold	silver	iron
value	\$20	\$1	\$500	\$100	\$900
weight	2 kg	1 kg	0.1 kg	0.5 kg	300 kg

Goal: To carry 5 kg, maximize value.

"Clearly" be greedy on $\frac{\text{value}}{\text{weight}}$.

$\frac{\text{value}}{\text{weight}}$	\$10	\$1	\$5000	\$200	\$3

-	gold		silver		copper		iron
-	0.1 kg	+	0.5 kg	+	2 kg	+	2.4 kg

③ Indivisible Items

	I_1	I_2	I_3	I_4
value	\$3	\$7	\$5	\$2
weight	5 kg	2 kg	7 kg	1 kg

- Each item : take it or leave it.
- You can carry at most M kg.
- Maximize value.

- NP- complete! No polytime algo. known. (or belived to exist).

- In particular, greedy does not work.

counter-ex

M = 9 kg

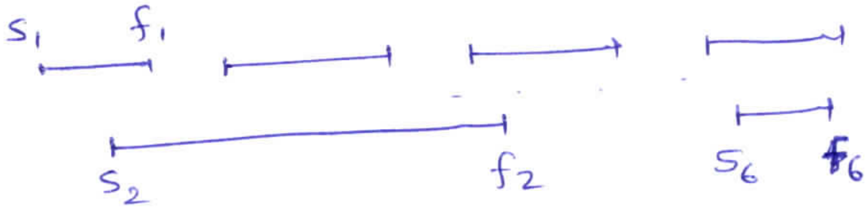
value	\$80	\$45	\$36
weight	8 kg	5 kg	4 kg
val/wt	\$10	\$9	\$9

Interval Scheduling

- n jobs to be scheduled on a machine.
- i^{th} job $[s_i, f_i]$
  - s_i start time
  - f_i finish time
- I.e. collection of intervals on real line.
- How many jobs (non-overlapping) can be scheduled?

Algo

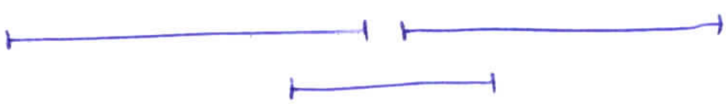
- Seq. of choices.
- Once a job is chosen, delete all over-lapping (conflicting) jobs.



Answer = 4.

Shortest job first?

Nope!



Job with minimum starting time first?

Nope!

Job with min # conflicts first?

Nope! See [Kleinberg Tardos] for counter example.

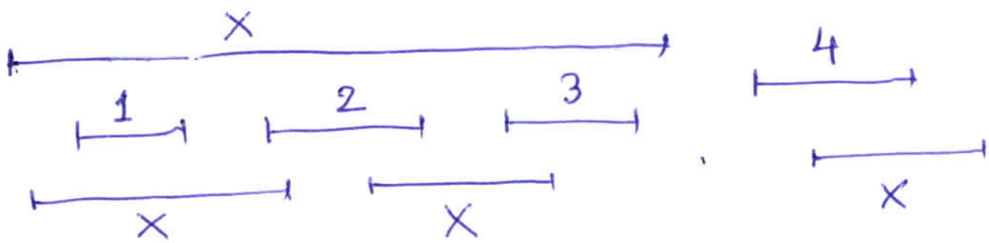
Algorithm   Min finish time first

- Sort jobs according to their finish times.

  $f_1 \leq f_2 \leq f_3 \leq \cdots \leq f_n$.

- Repeat
  - Among the remaining jobs, select one with min finish time.
  - Delete all jobs conflicting with it.

Ex



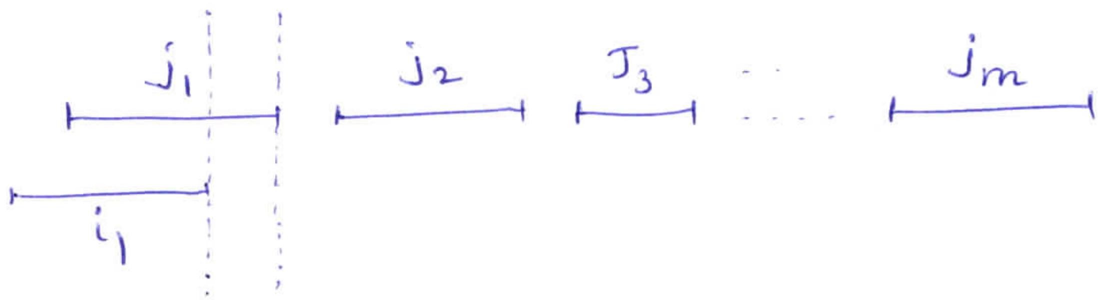
Analysis - Let $i_1, i_2, \ldots, i_\ell$ be jobs selected by the algorithm.

- Let $\mathcal{O} = \{j_1, j_2, \ldots, j_m\}$ be hypothetical optimal set/solution.
- Goal: to prove that $\ell = m$.

Warm-up

We show that there is a solution with m jobs that includes i_1, so we didn't make a mistake by selecting i_1.

Proof $\mathcal{O}$:



$\therefore \{i_1, j_2, j_3, \ldots, j_m\}$ is also a solution. ∎

Claim For $0 \le r \le \ell$, there is a hypothetical optimal solution $\mathcal{O}$, $|\mathcal{O}| = m$, such that

$$\{i_1, i_2, \ldots, i_r\} \subseteq \mathcal{O}.$$

(Moreover all other jobs in $\mathcal{O}$ start after i_r)

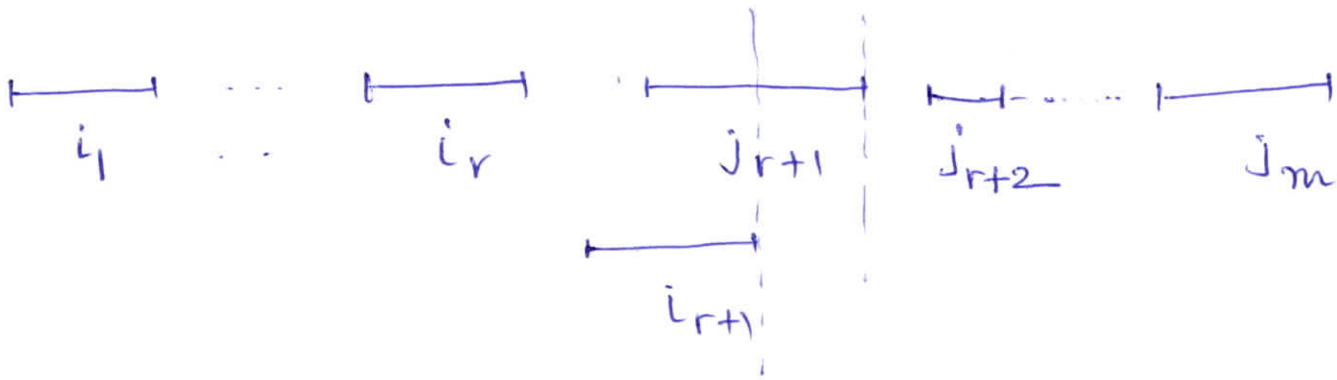
<u>Note</u> Applying the claim for $r = l$, there is optimal solution that includes $i_1, i_2, \dots, i_l$. But the algorithm stopped after including i_l, so every other job overlaps with one of these l jobs. Hence optimum $\le l$.

m

∎

<u>Proof</u> <u>$r = 0$.</u> Nothing to prove.

<u>Inductive</u> Assume that there is optimal solution $\mathcal{O}' = \{i_1, i_2, \dots, i_r, j_{r+1}, \dots, j_m\}$.



Since algo. chose i_{r+1} over j_{r+1},

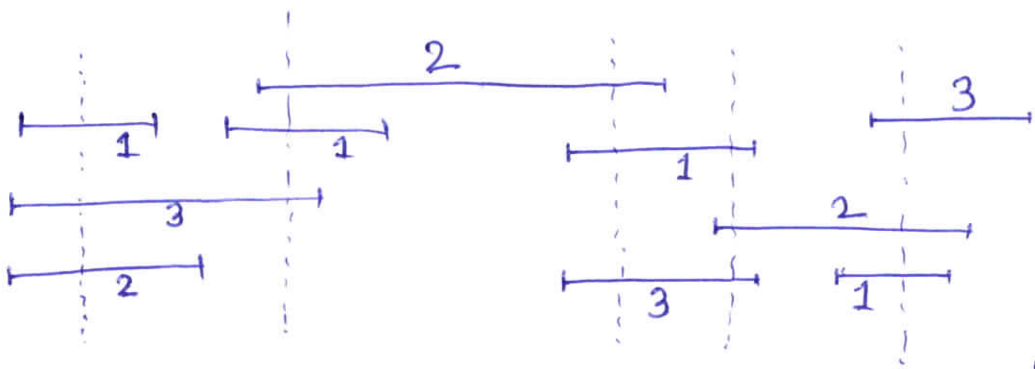
finish-time of i_{r+1} $\le$ finish-time of j_{r+1}

$\therefore \{i_1, \dots, i_r, i_{r+1}, j_{r+2}, \dots, j_m\} = \mathcal{O}$ is also optimal solution.

∎

Interval Partitioning

- Given n jobs : $[s_1, f_1], [s_2, f_2], \ldots, [s_n, f_n]$.
- Schedule all jobs on k machines s.t. jobs assigned to each m/c are disjoint
- Minimize k.



depth = 3

- observation.

  depth = max # jobs alive/active at any time instant.

  # machines needed $\geq$ depth.

Theorem In any instance of I.P. of depth d, there is a schedule with d machines (optimal).

Algorithm

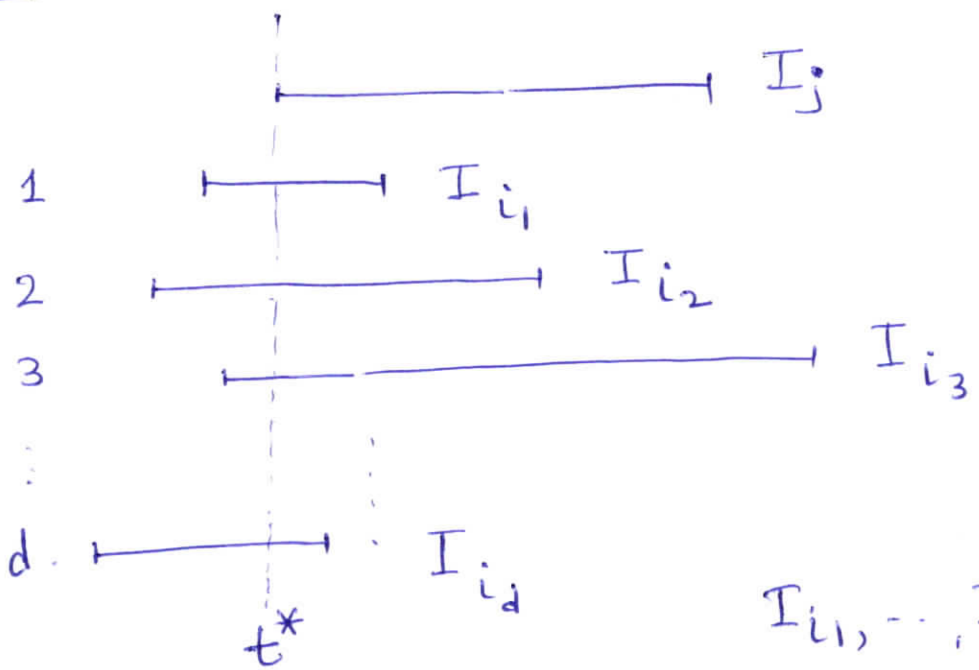
- Sort jobs according to their start time.

$$I_1 \quad I_2 \quad \cdots \quad I_n$$

$$s_1 \leq s_2 \cdots \leq s_n.$$

- Take d machines.
- For $j = 1, 2, \ldots, n$, schedule I_j on any machine that is free (during the entire interval $[s_j, f_j]$).

Proof By contradiction. Suppose no mk is free at j^{th} step.



$I_{i_1}, \ldots, I_{i_d}$ are jobs on d machines that

- overlap with I_j and have earlier start time.

$\therefore$ depth $\geq d+1$ at time t^*. Contradiction! ▨