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

- Given $n$ items with values $v_1, \ldots, v_n$.
- 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 with maximum value among remaining items.

**Proof of correctness**, the Exchange Argument

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

Now consider item $i_{R+1}$ chosen by the algorithm.

**Case 1**

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

$\{i_1, i_2, \ldots, i_R, i_{R+1}\} \subseteq S'$. Done.

**Case 2**

$i_{R+1} \notin S'$.

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

Since $i_{R+1}$ was the maximum value item after choosing $i_1, i_2, \ldots, i_R$, it must be
that \( V_j \leq V_{i_{r+1}} \).

\[ S = (S' \setminus \{j\}) \cup \{i_{r+1}\} \]

has value at least that of \( S' \), and since \( S' \) is already optimal, so is \( S \).

\[ \{i_1, i_2, \ldots, i_r, i_{r+1}\} \subseteq S, \quad S \text{ optimal}. \]

Done.

2. **Infinitely Divisible Items**

<table>
<thead>
<tr>
<th>Item</th>
<th>Copper</th>
<th>Sand</th>
<th>Gold</th>
<th>Silver</th>
<th>Iron</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>$20</td>
<td>$1</td>
<td>$500</td>
<td>$100</td>
<td>$900</td>
</tr>
<tr>
<td>Weight</td>
<td>2 kg</td>
<td>1 kg</td>
<td>0.1 kg</td>
<td>0.5 kg</td>
<td>300 kg</td>
</tr>
</tbody>
</table>

Goal: To carry 5 kg, maximize value.

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

\[
\begin{array}{cccccc}
\text{value} & $10 & $1 & $5000 & $200 & $3 \\
\text{weight} & \end{array}
\]
- gold silver copper iron
- $0.1 \text{ kg} + 0.5 \text{ kg} + 2 \text{ kg} + 2.4 \text{ kg}$

3) Indivisible Items

<table>
<thead>
<tr>
<th></th>
<th>$I_1$</th>
<th>$I_2$</th>
<th>$I_3$</th>
<th>$I_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>$3$</td>
<td>$7$</td>
<td>$5$</td>
<td>$2$</td>
</tr>
<tr>
<td>weight</td>
<td>$5 \text{ kg}$</td>
<td>$2 \text{ kg}$</td>
<td>$7 \text{ kg}$</td>
<td>$1 \text{ kg}$</td>
</tr>
</tbody>
</table>

- Each item: take it or leave it.
- You can carry at most $M \text{ kg}$.
- Maximize value.

- NP-complete! No polytime algo. known.
  (or believed to exist).
- In particular, greedy does not work.

Counter-ex

<table>
<thead>
<tr>
<th></th>
<th>value</th>
<th>$80$</th>
<th>$45$</th>
<th>$36$</th>
</tr>
</thead>
<tbody>
<tr>
<td>weight</td>
<td>$8 \text{ kg}$</td>
<td>$5 \text{ kg}$</td>
<td>$4 \text{ kg}$</td>
<td></td>
</tr>
<tr>
<td>val/weight</td>
<td>$10$</td>
<td>$9$</td>
<td>$9$</td>
<td></td>
</tr>
</tbody>
</table>
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 overlapping (conflicting) jobs.

\[
\begin{align*}
&\quad s_1 \quad f_1 \quad f_2 \quad s_2 \quad f_6 \quad s_6 \quad f_6 \\
&\text{Answer} = 4.
\end{align*}
\]

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.
  \[ f_1 \leq f_2 \leq f_3 \leq \ldots \leq f_n \].

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

Ex

```
\[ \begin{array}{c}
\times \\
1 \\
\times \\
\times \\
2 \\
\times \\
3 \\
\times \\
4 \\
\end{array} \]
```
Analysis - Let $i_1, i_2, \ldots, i_l$ be jobs selected by the algorithm.
- Let $O = \{j_1, j_2, \ldots, j_m\}$ be hypothetical optimal set/solution.
- Goal: to prove that $l = m$.

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 $O: \quad j_1 \quad j_2 \quad j_3 \quad \ldots \quad j_m$

\[ \vdash \{i_1, j_2, j_3, \ldots, j_m\} \text{ is also a solution.} \]

Claim For $0 \leq r \leq l$, there is a hypothetical optimal solution $O$, $|O| = m$, such that $\{i_1, i_2, \ldots, i_r\} \subseteq O$.

(Moreover all other jobs in $O$ start after $i_r$)
NoteApplying the claim for $r=1$, there is an optimal solution that includes $i_1, i_2, \ldots, i_r$. But the algorithm stopped after including $i_r$, so every other job overlaps with one of these $l$ jobs. Hence optimum $\leq l$.

Proof $r=0$. Nothing to prove.

Inductive Assume that there is an optimal solution $\Theta' = \{i_1, i_2, \ldots, i_r, j_{r+1}, \ldots, j_m\}$.

Since algo. chose $i_{r+1}$ over $j_{r+1}$, finish time of $i_{r+1} \leq$ finish time of $j_{r+1}$.

$\therefore \{i_1, \ldots, i_r, i_{r+1}, j_{r+2}, \ldots, j_m\} = \emptyset$

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 $k$.

- Observation.

$$\text{depth} = \max \frac{\text{# jobs alive at any time instant}}{\text{active m/c}} \geq \text{depth}.$$  

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:
  \[ I_1, I_2, \ldots, I_n \]
  \[ s_1 \leq s_2 \ldots \leq s_n \]
- 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.

\[ I_j \]

1.

2.

3.

4.

\( t^\star \) \( I_i_d \) \( I_i_1, \ldots, I_i_d \) are jobs on \( d \) machines that overlap with \( I_j \) and have earlier start time.
\[ \text{depth} \geq d+1 \text{ at time } t^\star. \] Contradiction!