Greedy Approximation Algorithms
(§ 11.1, 11.3)

Announcement:
Set 17 regrade deadline May 14, 2021.

Scheduling Jobs on Identical Machines.

Given \( n \) jobs with processing times \( t_1, t_2, \ldots, t_n \) on \( m \) machines.

Goal: Partition jobs into sets \( J(1), \ldots, J(m) \).

Processing time of machine \( k \) is

\[
T(k) = \sum_{j \in J(k)} t_j
\]

Minimize \( T = \max \{ T(1), \ldots, T(m) \} \),

"makespan".
**Greedy Algorithm 1**

Initialize $J(k) = \emptyset$ and $T(k) = 0$, for all $k$.

for $j = 1, 2, \ldots, n$:

Find a machine $k$ with minimum $T(k)$.

Assign job $j$ to machine $k$:

$J(k) = J(k) \cup \{j\}$

$T(k) = T(k) + t_j$

endfor

output $J(1), \ldots, J(n)$.
How could this ever output a suboptimal plan?

Hint: there is an example with 3 jobs, 2 machines, where output is suboptimal.

\[ t_1 = 1, \quad t_2 = 1, \quad t_3 = 2, \quad m = 2. \]

Optimum is \( J(1) = \{1, 2\}, \quad J(2) = \{3\} \)
\( T = T(1) = T(2) = 2. \)

Greedy will assign job 1 to \( M_1 \), job 2 to \( M_2 \), job 3 to \( M_1 \):

\( J(1) = \{1, 3\}, \quad J(2) = \{2\} \)
\( T(1) = 3, \quad T(2) = 1 \quad \text{SUBOPTIMAL!} \)

This job partitioning problem is NP-Complete even when \( m = 2 \), by reduction from \text{SUBSET SUM}.

How to compare greedy solution with optimum?

Reasoning is analogous to "greedy stays ahead."

At the time job \( n \) is placed on some machine \( k \), we know that

\[ T(\hat{a}) > T(k) \quad \forall \hat{a} = 1, 2, \ldots, n. \]
\[ \sum_{i=1}^{k} T(i) \geq m \cdot T(k). \]

\[ \Rightarrow \sum_{j=1}^{n} t_j \geq \frac{1}{m} \sum_{j=1}^{n} t_j \Rightarrow m \cdot T(k). \]

\[ T(k) \leq \frac{1}{m} \sum_{j=1}^{n} t_j. \]

Before last job was placed on machine k, we managed to put T(k) work on each machine.

The optimum solution partitions all jobs into sets \( J^*(1), \ldots, J^*(m) \) such that

\[ \forall i : \sum_{j \in J^*(i)} t_j \leq T^* \]

where \( T^* \) = makepan of optimal solution.

\[ \sum_{j=1}^{n} t_j = \sum_{i=1}^{m} \sum_{j \in J^*(i)} t_j \leq m \cdot T^* \]

\[ T^* \geq \frac{1}{m} \sum_{j=1}^{n} t_j \]

"optimal solution can't do better than perfect load balancing, \( \frac{1}{m} \) of work on each machine."

The last job we place, job \( n \), adds its processing time to machine \( k \).
So \( k \) ends up with load \( T(k) + t_n \).

Above, we saw \( T(k) \leq \frac{1}{m} \sum_{j=1}^{n} t_j \leq T^* \).

Finally, \( t_n \leq T^* \).

(For solution must place job \( n \) on some machine. That machine’s proc time is at least \( t_n \).)

Summing up, \( T(k) + t_n \leq 2 \cdot T^* \).

If we were guaranteed that the heaviest loaded machine in the greedy solution is the one with job \( n \), we’d be done now: we have shown \( T(k) + t_n \leq 2 \cdot T^* \) so greedy is a 2-approx to \( \text{OPT} \).

Maybe some other machine \( k' \) has higher load than \( k \) at the end of running Greedy.

\[ \begin{array}{c}
\text{k} \\
\text{t}_3 \\
\text{t}_1 \\
\end{array} \quad \begin{array}{c}
k' \\
\text{t}_1 \\
\end{array} \]
Suppose $k'$ is the most heavily loaded machine in the greedy solution and $j'$ is the last job placed on it. Then just before $j'$ was assigned to $k'$,

\[ \forall i, \quad T(i) \leq T(k') \quad \text{greedy rule.} \]

\[
\sum_{j=1}^{j'-1} t_j = \sum_{i=1}^{n} T(i) \geq m \cdot T(k').
\]

\[ \downarrow \]

\[ T(k') \leq \frac{1}{m} \sum_{j=1}^{j'-1} t_j \leq T^*. \]

\[ t_{j'} \leq T^* \]

\[ T(k') + t_{j'} \leq 2 \cdot T^* \]

= final load on $k'$

= makespan of greedy solution
**SET COVER:** Given n-element set $U$. Subsets $S_1, \ldots, S_m \subseteq U$. Assume $U = \bigcup_{i=1}^{m} S_i$.

Find min # of subsets whose union is $U$.

**Greedy Set Cover**

Initialize $T = \{ \text{uncovered elements} \} = U$.

Initialize $J = \emptyset$ (indices of chosen sets) = $\emptyset$.

While $T$ non-empty:

1. find $i$ such that $|S_i \cap T|$ is maximum.
2. $J = J \cup \{i\}$
3. $T = T \setminus S_i$.

Endwhile

Output $\{ S_j \mid j \in J \}$.

Greedy picks $S_3, S_4, S_5$. 

Diagram:

- Set $S_1$ covers elements $\bullet$.
- Set $S_2$ covers elements $\bullet$.
- Set $S_3$ covers elements $\bullet$.
- Set $S_4$ covers elements $\bullet$.
- Set $S_5$ covers elements $\bullet$.
Generalize this to $2 \cdot (2^k - 1)$ elements such that $S_1$ & $S_2$ cover them all, but greedy can be fooled into picking $k$ sets rather than $2$.

$$k \leq \log_2(n) - 1.$$ 

We'll see (next lecture) greedy set cover is a $O(\log n)$ approximation in the worst case.