Announcements

- HW 8 is out
  due next Wed. in class
Approximation

- Problems can be hard to solve
- Exponential algorithms take too long

So: sometimes we can get (in polynomial time) an answer that is “close” to optimal
Load Balancing (Sec. 11.1 in book, Sec. 1 in notes)

\[ m \text{ machines } M_1, ..., M_m \]

n jobs given in array, \( t_j \) = job j's run time

[Want a balanced assignment:

So if \( A_i \) = jobs assigned to \( M_i \)

\[ \frac{1}{T_i} = \sum_{j \in A_i} t_j \text{ is load on } M_i, \]

Minimize \[ T = \max_i T_i \]

(called make span)
Example: 3 machines, jobs: 2, 3, 4, 6, 2, 2

How small can make span be?
I won't prove it, but this is NP-Hard.
(from partition)
What if we need to solve it anyway?

We'll use approximation:
What is a natural greedy idea?
Pseudo code

Greedy-Balance:

\[ T_i \leftarrow 0, A_i \leftarrow \emptyset \] for all \( M_i \)

\[ \text{for } j \leftarrow 1 \text{ to } n \]

Let \( M_i \) be machine minimizing \( T_i \)

Assign job \( j \) to \( M_i \)

\[ A_i \leftarrow A_i \cup J_j \]

\[ T_i \leftarrow T_i + t_j \]
Greedy Balance:

Example: 3 machines, jobs: 2, 3, 4, 6, 2, 2

How small can make span be?

\[ 8 \rightarrow \begin{array}{c}
6 \\
2 \\
\end{array} \]

\[ \begin{array}{c}
2 \\
3 \\
\end{array} \]

\[ \begin{array}{c}
2 \\
4 \\
\end{array} \]

M_1  M_2  M_3
Would like to argue that we are close to optimal value $T^*$.

But we don't know $T^*$!

⇒ We can, however, get bounds on $T^*$.

our makespan = $T$

(in prev. example, $T^* = 7$)
For example: \[ T^* = \text{average} \]

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

Why?

This is what value would be if things could be spread perfectly evenly over all machines.

If \( T^* \) were less, then \( m \times T^* < \sum t_i \)
Another:

2. \( T^* \geq \max_j t_j \geq t_j \)

Why?

Some machine has to run the longest job.
Thm: Greedy - Balance gives a schedule with makespan $T \leq 2 \cdot T^*$.

pf: Consider the machine $M_i$ with the largest load, so $T_i = \max_k T_k$.

Consider the last job that $M_i$ runs, job $j$.

$T_i - t_j$ was lowest "load" on a machine $\Rightarrow$ every other machine had $\geq T_i - t_j$. 
pf cont: If we add load on all machines,

$$\sum_{k} T_k \geq m (T_i - t_j)$$

rewrite: $T_i - t_j \leq \frac{1}{m} \sum_{k} T_k = \frac{1}{m} \sum_{k} t_k$

we know $T^* = \frac{1}{m} \sum_{k} t_k$ looks familiar,

$$\Rightarrow T_i - t_j \leq \frac{1}{m} \sum_{k} t_k \leq T^*$$

so $M_i$ had $\leq T^*$ on it before it got its last job.

\[\text{and gives us} \quad t_j \leq T^* \]

so $T_i = (T_i - t_j) + t_j \leq T^* + T^*$ \[\Box\]
Can we get this bad?

Bad case: large job at the end
If the jobs are sorted largest to smallest, we can do better!

\[ T_i \leftarrow 0 \quad A_i \leftarrow \emptyset \quad \text{for all } M_i \]

Sort jobs so that \( t_1 \geq t_2 \geq \ldots \geq t_n \)

\[ T_i \text{ for } j \in 1 \text{ to } n \]

Let \( M_j \) be machine minimizing

Assign job \( j \) to \( M_i \)

\[ A_i \leftarrow (A_i \cup \{ j \}) \quad T_i \leftarrow T_i + t_j \]
We'll use a slightly better bound on $T^*$

Lemma: If there are more than $m$ jobs,
then $T^* \geq 2t_{m+1}$.

pf:
First $m$ jobs go on empty machines. Job $m+1$ gets put with a job whose processing time is $\geq t_{m+1}$.
So $T^*$ has to take at least $2 + t_{m+1}$ time.
Theorem: Algorithm Sorted-Balance gives $T \leq \frac{3}{2} \cdot T^*$. 

Proof: Consider worst machine $M_i$, and let $j$ be the last job on it.

\[ \frac{T^*}{2} \implies \begin{cases} T^* \geq \sum_{i}^{T^*} T_i - t_j \\ M_i \end{cases} \]

Still have $T_i - t_j \leq T^*$.

If $j \leq m$, done

If $j > m$, use (3).

So $T^* \geq 2t_{m+1} \geq 2t_j$.

Rewrite: $t_j \leq \frac{T^*}{2}$.