Scheduling Continued

Review: digraph, processors, tasks, priority lists

Processors: can be busy or idle

Tasks: can be

✦ ready- task could be started
✦ in execution- task is being worked on
✦ completed- finished
✦ ineligible- cannot be started

We explored keeping track of precedence relations and processors, now look at keeping track of progress on the priority list:

Use a circle to represent that a task is ready.

One line through the task represents that it is being executed.

Two lines represent a completed task.

An unmarked task is ineligible.

\[
\begin{array}{c}
T(2.5), B(3), M(4), N(0.5), F(3.5), S(1), D(2), C(1.5)
\end{array}
\]

This means that tasks T and M are completed, while task F is being worked on.

Tasks N and C are ready to be started and tasks B, S and D are ineligible.
Try scheduling according to this priority list:
T(2.5), B(3), M(4), N(0.5), F(3.5), S(1), D(2), C(1.5)

P_1
|--------------------------------------------------|
P_2
|--------------------------------------------------|

T(2.5), N(0.5), F(3.5), C(1.5), S(1), D(2), M(4), B(3)

P_1
|--------------------------------------------------|
P_2
|--------------------------------------------------|
B(3), M(4), D(2), S(1), C(1.5), F(3.5), N(0.5), T(2.5)

Is there a better way to pick a priority list?
Remember that with 8 tasks there are 8! or 40320 different priority lists.

**Decreasing Time Algorithm**
Prioritizes the longer tasks over the shorter tasks.
Write down the tasks in order of completion time, longer tasks first. This gives you the priority list.
(in case of a tie, flip a coin or try it both ways)
Schedule the tasks according to the decreasing time priority list.
Pro: we know we’re probably doing better than we would by picking a random priority list.

Con: not guaranteed to find optimal solution

The decreasing time algorithm is an approximate algorithm.

For the wedding example:
Use the decreasing time algorithm to schedule the following tasks.

![Diagram showing task dependencies]

Write down the priority list:

D(12), C(9), A(8), E(6), B(5), G(2), F(1)

What if we had three processors?

![Diagram showing task dependencies with three processors]

D(12), C(9), A(8), E(6), B(5), G(2), F(1)