A *runtime/online scheduler* maps tasks to processing elements dynamically at runtime.

The map is called a *schedule*.

An *offline scheduler* prepares the schedule prior to the actual execution of the program.
Greedy Scheduling

A strand / task is called *ready* provided all its parents (if any) have already been executed.

- Executed task
- Ready to be executed
- Not yet ready

A *greedy scheduler* tries to perform as much work as possible at every step.
A Centralized Greedy Scheduler

Let $p = \text{number of cores}$

At every step:

- if $\geq p$ tasks are ready:
  execute any $p$ of them
  (complete step)

- if $< p$ tasks are ready:
  execute all of them
  (incomplete step)
A Centralized Greedy Scheduler

Let $p =$ number of cores

At every step:
- if $\geq p$ tasks are ready:
  execute any $p$ of them
  (complete step)
- if $< p$ tasks are ready:
  execute all of them
  (incomplete step)
A Centralized Greedy Scheduler

Let \( p = \) number of cores

At every step:

- if \( \geq p \) tasks are ready:
  execute any \( p \) of them
  (complete step)

- if \( < p \) tasks are ready:
  execute all of them
  (incomplete step)
A Centralized Greedy Scheduler

Let $p = \text{number of cores}$

At every step:

- if $\geq p$ tasks are ready:
  execute any $p$ of them
  ( complete step )

- if $< p$ tasks are ready:
  execute all of them
  ( incomplete step )
A Centralized Greedy Scheduler

Let $p =$ number of cores

At every step:

- if $\geq p$ tasks are ready:
  execute any $p$ of them
  (complete step)

- if $< p$ tasks are ready:
  execute all of them
  (incomplete step)
A Centralized Greedy Scheduler

Let $p = \text{number of cores}$

At every step:

- if $\geq p$ tasks are ready:
  execute any $p$ of them
  (complete step)

- if $< p$ tasks are ready:
  execute all of them
  (incomplete step)
A Centralized Greedy Scheduler

Let $p =$ number of cores

At every step:

- if $\geq p$ tasks are ready:
  execute any $p$ of them
  (complete step)

- if $< p$ tasks are ready:
  execute all of them
  (incomplete step)
Let $p = \text{number of cores}$

At every step:

- if $\geq p$ tasks are ready: execute any $p$ of them (complete step)

- if $< p$ tasks are ready: execute all of them (incomplete step)
A Centralized Greedy Scheduler

Let $p = \text{number of cores}$

At every step:

- if $\geq p$ tasks are ready: execute any $p$ of them (complete step)
- if $< p$ tasks are ready: execute all of them (incomplete step)
Let $p = \text{number of cores}$

At every step:
- if $\geq p$ tasks are ready:
  execute any $p$ of them
  (complete step)
- if $< p$ tasks are ready:
  execute all of them
  (incomplete step)
A Centralized Greedy Scheduler

Let $p =$ number of cores

At every step:

- if $\geq p$ tasks are ready:
  execute any $p$ of them
  (complete step)

- if $< p$ tasks are ready:
  execute all of them
  (incomplete step)
A Centralized Greedy Scheduler

Let $p =$ number of cores

At every step:

- if $\geq p$ tasks are ready:
  execute any $p$ of them
  (complete step)

- if $< p$ tasks are ready:
  execute all of them
  (incomplete step)
A Centralized Greedy Scheduler

Let \( p \) = number of cores

At every step:
- if \( \geq p \) tasks are ready:
  execute any \( p \) of them
  (complete step)
- if \(< p \) tasks are ready:
  execute all of them
  (incomplete step)
Theorem [Graham’68, Brent’74]:

For any greedy scheduler,

\[ T_p \leq \frac{T_1}{p} + T_\infty \]

Proof:

\[ T_p = \#\text{complete steps} + \#\text{incomplete steps} \]

- Each complete step performs \( p \) work:
  \[ \#\text{complete steps} \leq \frac{T_1}{p} \]

- Each incomplete step reduces the span by 1:
  \[ \#\text{incomplete steps} \leq T_\infty \]
Optimality of the Greedy Scheduler

Corollary 1: For any greedy scheduler $T_p \leq 2T_p^*$, where $T_p^*$ is the running time due to optimal scheduling on $p$ processing elements.

Proof:

Work law: $T_p^* \geq \frac{T_1}{p}$

Span law: $T_p^* \geq T_\infty$

∴ From Graham-Brent Theorem:

$$T_p \leq \frac{T_1}{p} + T_\infty \leq T_p^* + T_p^* = 2T_p^*$$
Optimality of the Greedy Scheduler

Corollary 2: Any greedy scheduler achieves $S_p \approx p$ (i.e., nearly linear speedup) provided $\frac{T_1}{T_\infty} \gg p$.

Proof:

Given, $\frac{T_1}{T_\infty} \gg p \Rightarrow \frac{T_1}{p} \gg T_\infty$

∴ From Graham-Brent Theorem:

$$T_p \leq \frac{T_1}{p} + T_\infty \approx \frac{T_1}{p}$$

$$\Rightarrow \frac{T_1}{T_p} \approx p \Rightarrow S_p \approx p$$