# 29 January — Greedy Stays Ahead

## Today's Plan

1. Interval Scheduling Problem
2. Announcements
3. Greedy Stays Ahead

Classic Motivation:

* single central processor
* many job requests

Question: How do we schedule the jobs?

## Classic Motivation:

* single central processor
* many job requests

**Question:** How do we schedule the jobs?

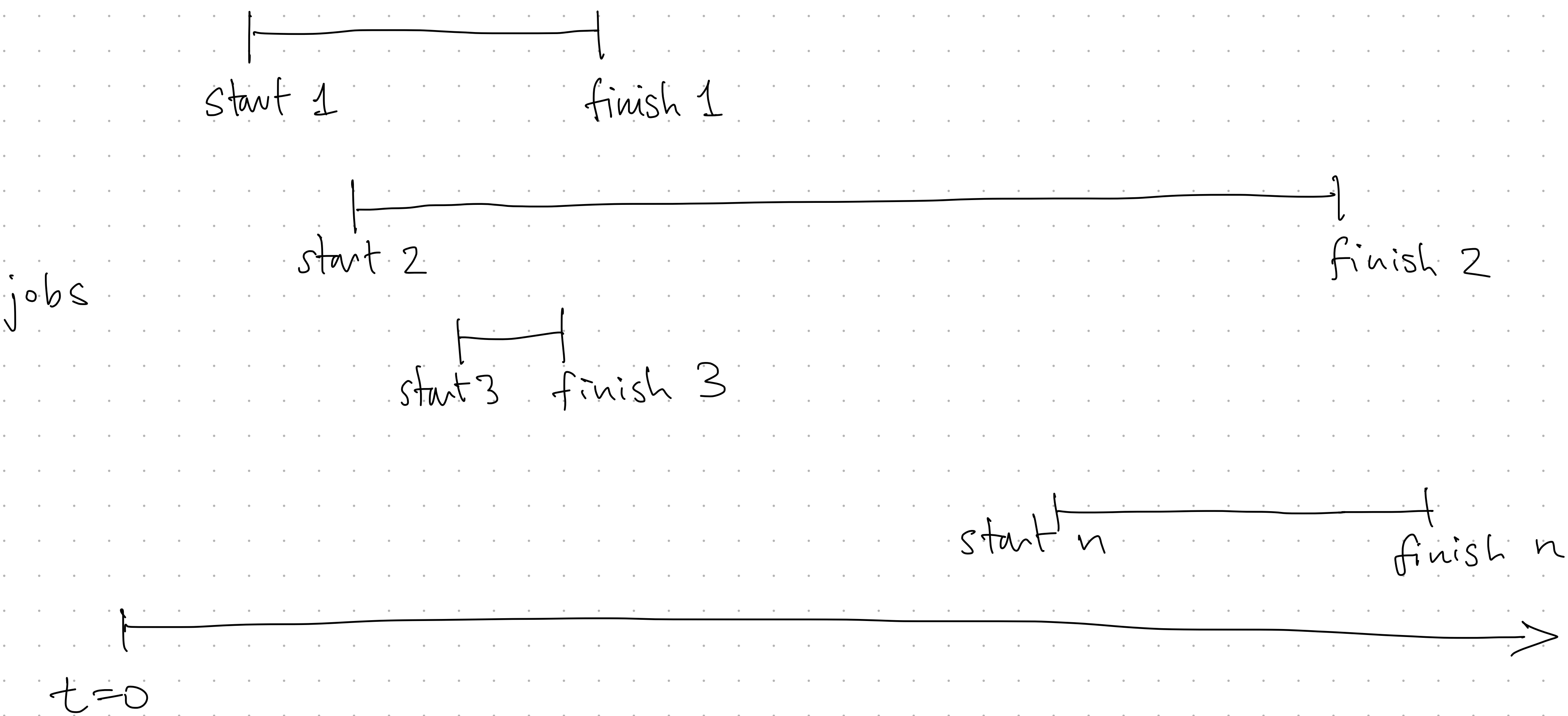
## Details

- Each job has a proposed start time & finish time
- Processor can handle at most 1 job at a time
- Assumption: jobs have equal priority

## Classic Motivation:

* single central processor
* many job requests

**Question:** How do we schedule the jobs?



# Interval Scheduling Problem.

Given: List of $n$ jobs, specified by [start, finish] time

$$\langle [s_1, f_1], [s_2, f_2], \ldots, [s_n, f_n] \rangle$$

Goal: Return a set of non-conflicting jobs of maximum cardinality

# Interval Scheduling Problem

Given: List of $n$ jobs, specified by [start, finish] time

$$\langle [s_1, f_1], [s_2, f_2], \ldots, [s_n, f_n] \rangle$$

Goal: Return a set of non-conflicting jobs of maximum cardinality

Equal priority

# Interval Scheduling Problem

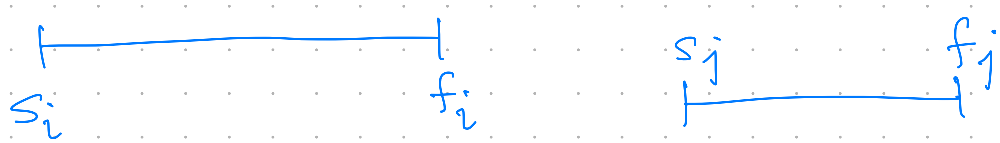
<u>Given</u>. List of $n$ jobs, specified by [start, finish] time

$$\langle [s_1, f_1], [s_2, f_2], \ldots, [s_n, f_n] \rangle$$

<u>Goal</u>. Return a set of <u>non-conflicting</u> jobs of maximum cardinality

<u>Defn</u>. A set of intervals $S$ is <u>non-conflicting</u> if for all $i \neq j \in S$

$$s_i \leq s_j \implies f_i < s_j$$




non-conflicting

# Interval Scheduling Problem.

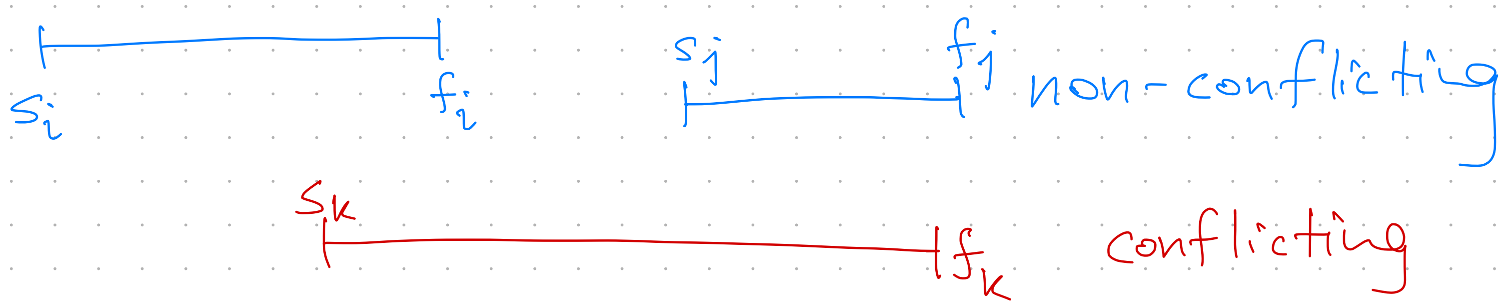
**Given.** List of $n$ jobs, specified by [start, finish] time

$$\langle [s_1, f_1], [s_2, f_2], \ldots, [s_n, f_n] \rangle$$

**Goal.** Return a set of non-conflicting jobs of maximum cardinality

**Defn.** A set of intervals $S$ is non-conflicting if for all $i \neq j \in S$

$$s_i \leq s_j \implies f_i < s_j$$



# Announcements

* HW 0 due (solutions by Thurs)
* HW 1 Released After Lecture

# Interval Scheduling Problem

Given: List of $n$ jobs, specified by [start, finish] time

$$\langle [s_1, f_1], [s_2, f_2], \ldots, [s_n, f_n] \rangle$$

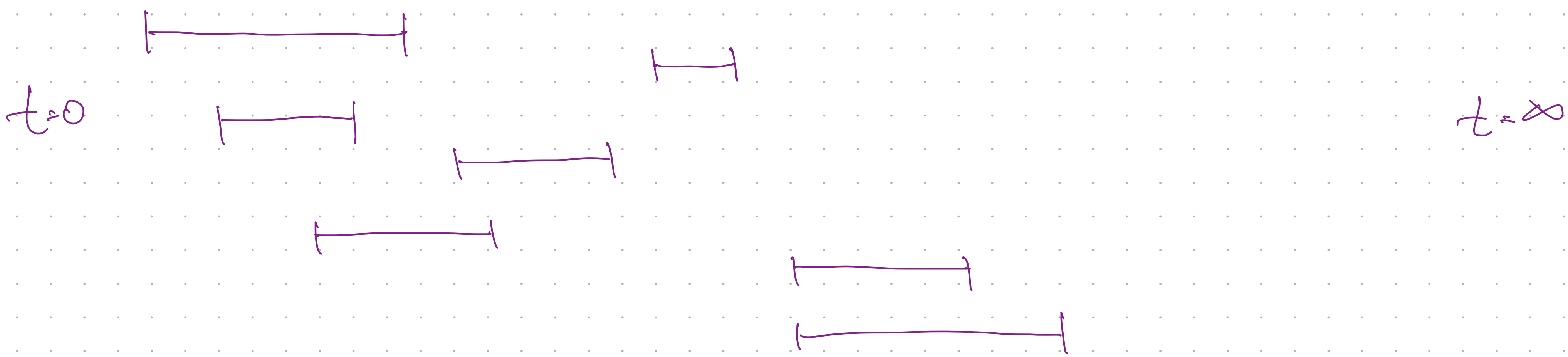
Goal: Return a set of non-conflicting jobs of maximum cardinality

# Interval Scheduling Problem.

Given: List of $n$ jobs, specified by [start, finish] time

$$\langle [s_1, f_1], [s_2, f_2], \ldots, [s_n, f_n] \rangle$$

## Candidate "Priority" Functions:



# Earliest Finish Time

1. Sort jobs by finish time
2. Schedule = $\{\}$

Iterate through jobs in sorted order $j = 1 \dots n$

- if job $j$ does not conflict w/ Schedule
  - ↳ Schedule $\leftarrow$ Schedule $\cup \{j\}$

Return Schedule.

# Earliest Finish Time

① Sort jobs by finish time

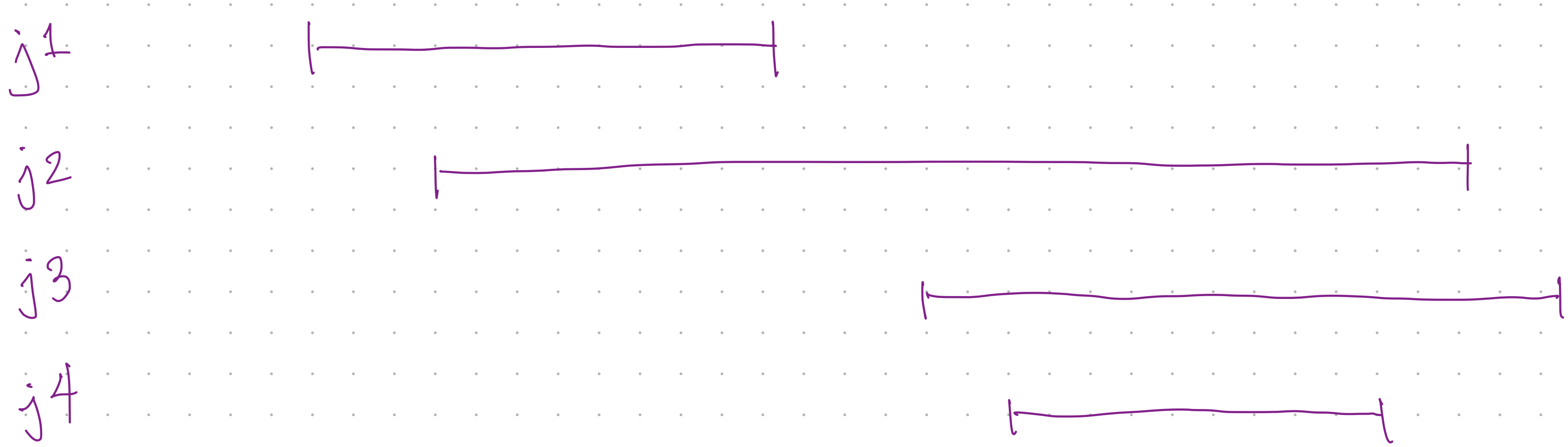
② Schedule = { }

Iterate through jobs in sorted order $j = 1 \dots n$

– if job $j$ does not conflict w/ Schedule

↳ Schedule ← Schedule $\cup \{j\}$

Return Schedule.



# Earliest Finish Time

1. Sort jobs by finish time
2. Schedule = { }

   Iterate through jobs in sorted order $j = 1 \dots n$
   - if job $j$ does not conflict w/ Schedule
     ↳ Schedule ← Schedule $\cup \{j\}$

   Return Schedule.

Claim. EFT can be implemented with RT $O(n \log n)$.

**Theorem**. EFT returns a maximum cardinality set of non-conflicting jobs.

* Non-conflicting → By design

* Maximum Cardinality → "Greedy stays ahead"

# Greedy Stays ahead

- Imagine <u>some</u> optimal schedule $S^*$
- Compare output of EFT to $S^*$

## Greedy Stays ahead

- Imagine some optimal schedule $S^*$
- Compare output of EFT to $S^*$

Argue that:

(a) if EFT "stays ahead" of $S^*$,
then EFT is also optimal

(b) EFT "stays ahead"

## Greedy Stays ahead

- Imagine some optimal schedule $S^*$
- Compare output of EFT to $S^*$

Argue that:

(a) if EFT "stays ahead" of $S^*$, then EFT is also optimal

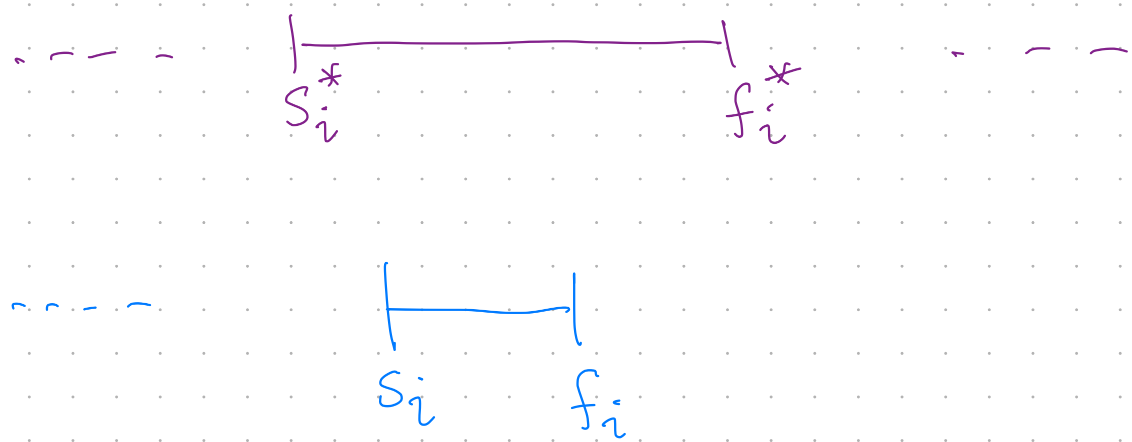
(b) EFT "stays ahead"

WARNING: Need to define "stays ahead" per problem

**Claim.** EFT "stays ahead" of any $S^*$ in the finish time of the $j^{th}$ job.

$S^*$ :

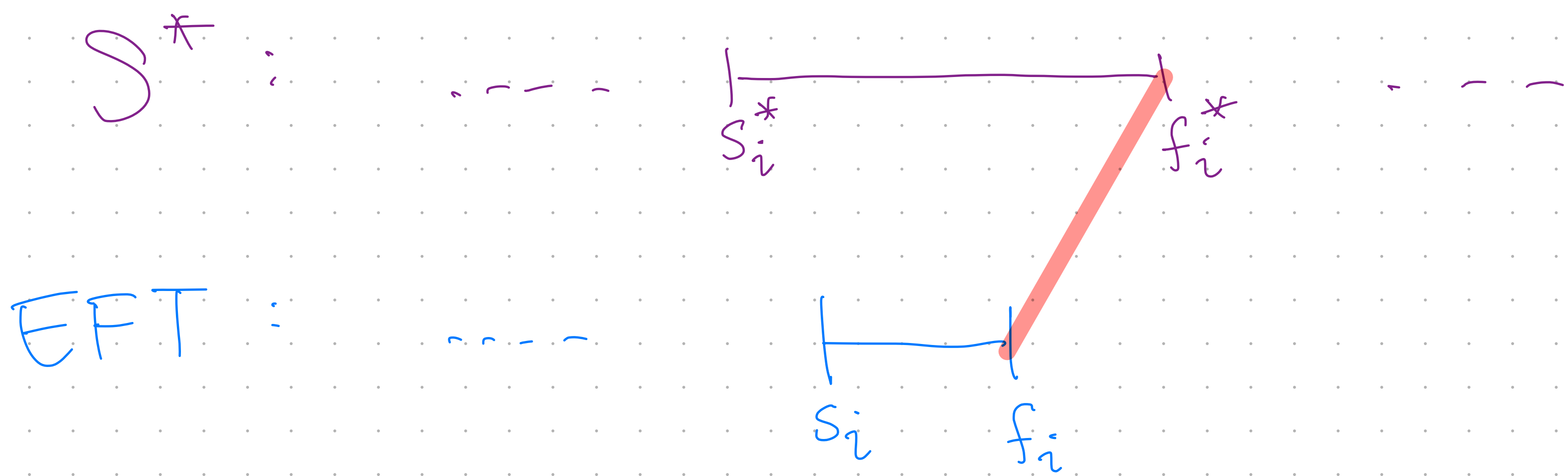
EFT :




**Convenient Notation.** Assume jobs are sorted by finishing time.

$f_1^* < f_2^* < \cdots < f_n^*$ $\qquad$ $f_1 < f_2 < \cdots < f_\ell$

**Claim.** EFT "stays ahead" of any $S^*$ in the finish time of the $j^{th}$ job.




For each $i$, the $i^{th}$ scheduled jobs satisfy $f_i \le f_i^*$

# Greedy Stays Ahead Lemma.

Suppose EFT returns a schedule of $k$ jobs.

For any optimal schedule $S^*$,

For all $i \in \{1, ..., k\}$ $\quad f_i \leq f_i^*$

"EFT stays ahead of $S^*$"

## Greedy Stays Ahead Lemma.

Suppose EFT returns a schedule of $k$ jobs.

For any optimal schedule $S^*$,

For all $i \in \{1, \ldots, k\}$ $\quad f_i \leq f_i^*$

"EFT stays ahead of $S^*$"

---

## Earliest Finish Time Lemma

$S^*$ is optimal

AND $\qquad \Longrightarrow$ EFT is also <u>optimal</u>.

$\forall i \; f_i \leq f_i^*$

"If EFT stays ahead of $S^*$, then EFT is optimal"

Greedy Stays Ahead Lemma. $\underbrace{\text{EFT vs. } S^*}$

For all $i \in \{1, \dots, k\}$ $\quad f_i \preceq f_i^*$

Pf. By induction on intervals added by EFT

## Greedy Stays Ahead Lemma.

$\underbrace{\text{EFT vs. } S^*}$

For all $i \in \{1, \dots, k\}$ $\quad f_i \leq f_i^*$

---

Pf. By induction on intervals added by EFT

Base case. $i = 1$.

* By EFT rule, first job selected because,

$$f_1 \leq f_j$$

for all jobs $j \in [n]$,

$$\Longrightarrow \quad f_1 \leq f_1^*$$

for any choice of $S^*$

## Greedy Stays Ahead Lemma.

EFT vs. $S^*$

For all $i \in \{1, \ldots, k\}$ $\quad f_i \leq f_i^*$

---

Inductive Step:

$$\left( \begin{array}{c} \forall i \in \{1, \ldots, k-1\} \\ f_i \leq f_i^* \end{array} \right) \Longrightarrow f_k \leq f_k^*$$

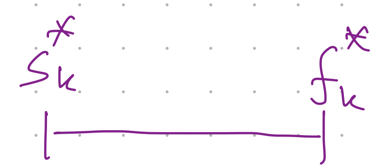
Inductive Hypothesis

# Greedy Stays Ahead Lemma. EFT vs. $S^*$

For all $i \in \{1, \dots, k\}$ $\quad f_i \le f_i^*$

**Inductive Step:** $\underbrace{\left( \forall i \in \{1, \dots, k-1\} \; f_i \le f_i^* \right)}_{\text{Inductive Hypothesis}} \implies f_k \le f_k^*$

* Consider the $k^{th}$ job of $S^*$



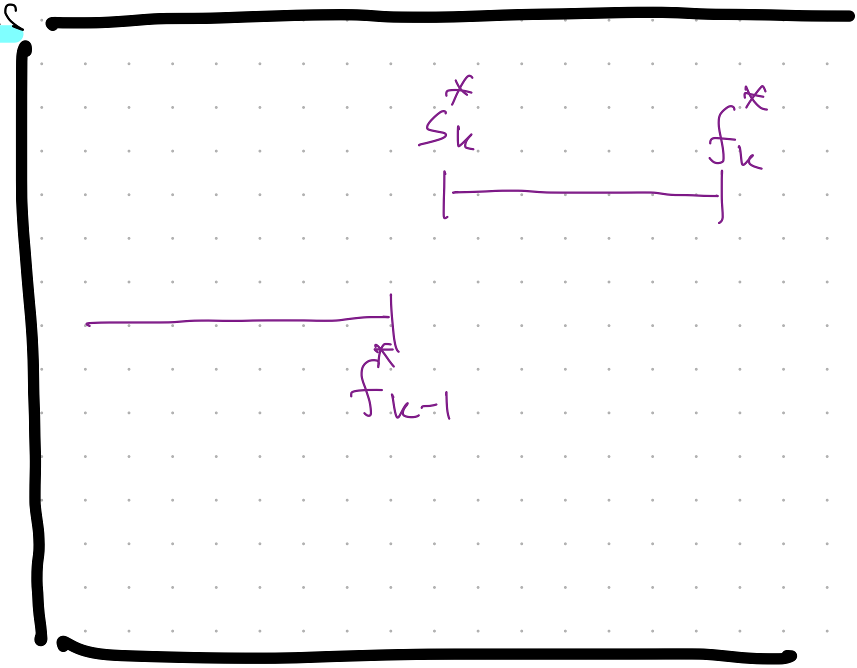
# Greedy Stays Ahead Lemma.

EFT vs. $S^*$

For all $i \in \{1, \dots, k\}$ $\quad f_i \le f_i^*$

**Inductive Step:** $\left( \forall i \in \{1, \dots, k-1\} \ \ f_i \le f_i^* \right) \Longrightarrow f_k \le f_k^*$

Inductive Hypothesis




* Consider the $k^{th}$ job of $S^*$
* By non-overlapping: $f_{k-1}^* < s_k^*$

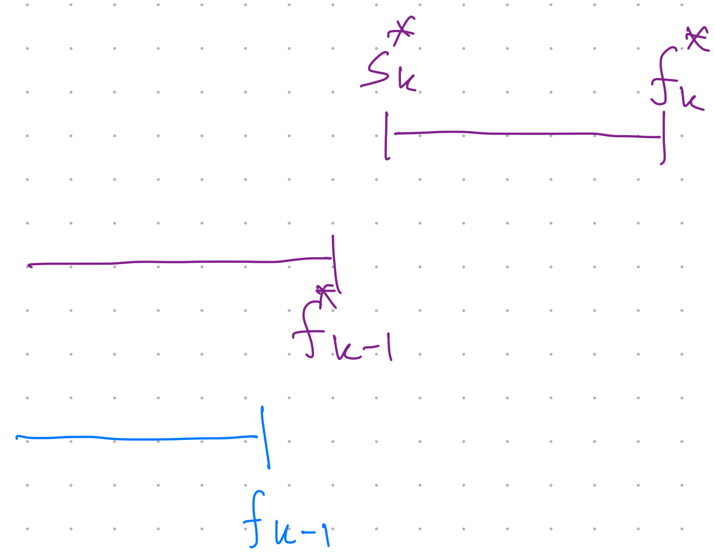
# Greedy Stays Ahead Lemma. EFT vs. $S^*$

For all $i \in \{1, \dots, k\}$ $\quad f_i \le f_i^*$

Inductive Step: $\underbrace{\left( \forall i \in \{1, \dots, k-1\} \; f_i \le f_i^* \right)}_{\text{Inductive Hypothesis}} \Longrightarrow f_k \le f_k^*$

* Consider the $k^{th}$ job of $S^*$
* By non-overlapping: $f_{k-1}^* < s_k^*$
* By IH, $f_{k-1} \le f_{k-1}^*$

$\Longrightarrow [s_k^*, f_k^*]$ non-conflicting w/ $[s_{k-1}, f_{k-1}]$



# Greedy Stays Ahead Lemma.

$\underbrace{\text{EFT vs. } S^*}$

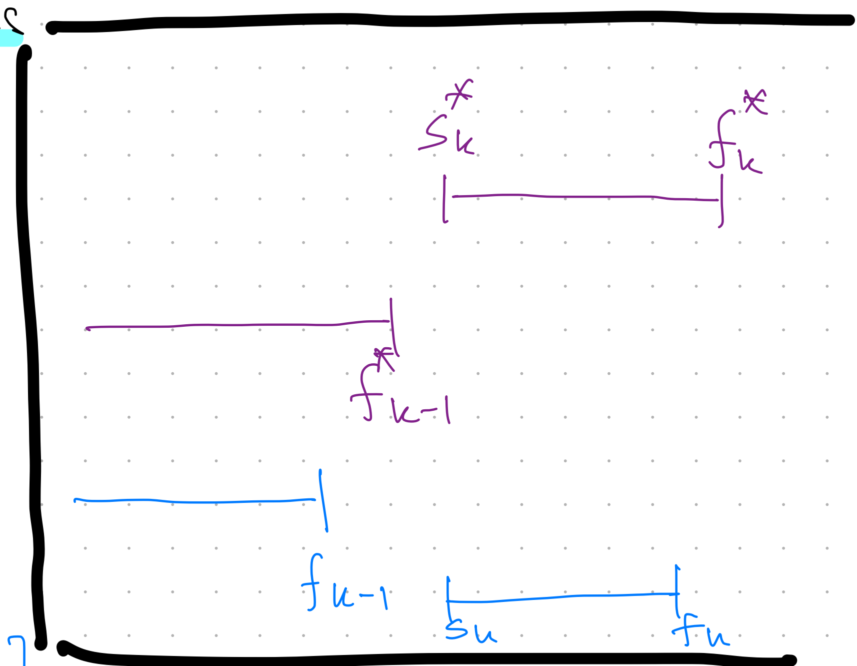
For all $i \in \{1, \dots, k\}$ $\quad f_i \le f_i^*$

---

Inductive Step: $\left( \forall i \in \{1, \dots, k-1\} \; f_i \le f_i^* \right) \implies f_k \le f_k^*$

Inductive Hypothesis

* Consider the $k^{th}$ job of $S^*$
* By non-overlapping: $f^*_{k-1} < s^*_k$
* By IH, $f_{k-1} \le f^*_{k-1}$




$\implies [s_k^*, f_k^*]$ non-conflicting w/ $[s_{k-1}, f_{k-1}]$

But $[s_k, f_k]$ is non-conflicting job of earliest finish time: $f_k \le f_k^*$ ∎

# Earliest Finish Time Lemma

$S^*$ is optimal

AND $\Longrightarrow$ EFT is also optimal.

$\forall i \quad f_i \leq f_i^*$

# Earliest Finish Time Lemma

$S^*$ is optimal
AND $\implies$ EFT is also optimal.
$\forall i \; f_i \leq f_i^*$

---

Pf. Consider any non-overlapping set $S^*$

We show:

EFT "stays ahead" of $S^* \implies |EFT| \geq |S^*|$.

$*$ Consider any prefix of indices $1, \ldots, i \leq |EFT|$

# Earliest Finish Time Lemma

$S^*$ is optimal

AND $\implies$ EFT is also optimal.

$\forall i \; f_i \leq f_i^*$

---

Pf. Consider any non-overlapping set $S^*$

We show:

EFT "stays ahead" of $S^* \implies |EFT| \geq |S^*|$.

* Consider any prefix of indices $1, \dots, i \leq |EFT|$

* $f_i \leq f_i^* \implies$ if $|S^*| \geq i+1$, then

$[s_{i+1}^*, f_{i+1}^*]$ is feasible for EFT.

$\implies |EFT| \geq |S^*|$ ∎

# Recap

* Earliest Finish Time algorithm returns maximum non-overlapping schedule

* Runs in $O(n \log n)$ time

* Proof of Correctness:
  - ↳ EFT "stays ahead" of optimal
  - ↳ Staying ahead $\Longrightarrow$ optimality.

Food for thought ...

What if the jobs had different priorities?