

# Lecture05

---



# CSE 421 Introduction to Algorithm

Greedy Algorithms

Winter 2024

Lecture 5

1

# Announcements

- Reading
  - For today, sections 4.1, 4.2,
  - For Wednesday, sections 4.4, 4.5, 4.7, 4.8
- No class on Monday

# Greedy Algorithms

- Solve problems with the simplest possible algorithm
- The hard part: showing that something simple actually works
- Pseudo-definition
  - An algorithm is **Greedy** if it builds its solution by adding elements one at a time using a simple rule



# Scheduling Theory

- **Tasks**
  - Processing requirements, release times, deadlines
- **Processors**
- **Precedence constraints**
- **Objective function**
  - Jobs scheduled, lateness, total execution time

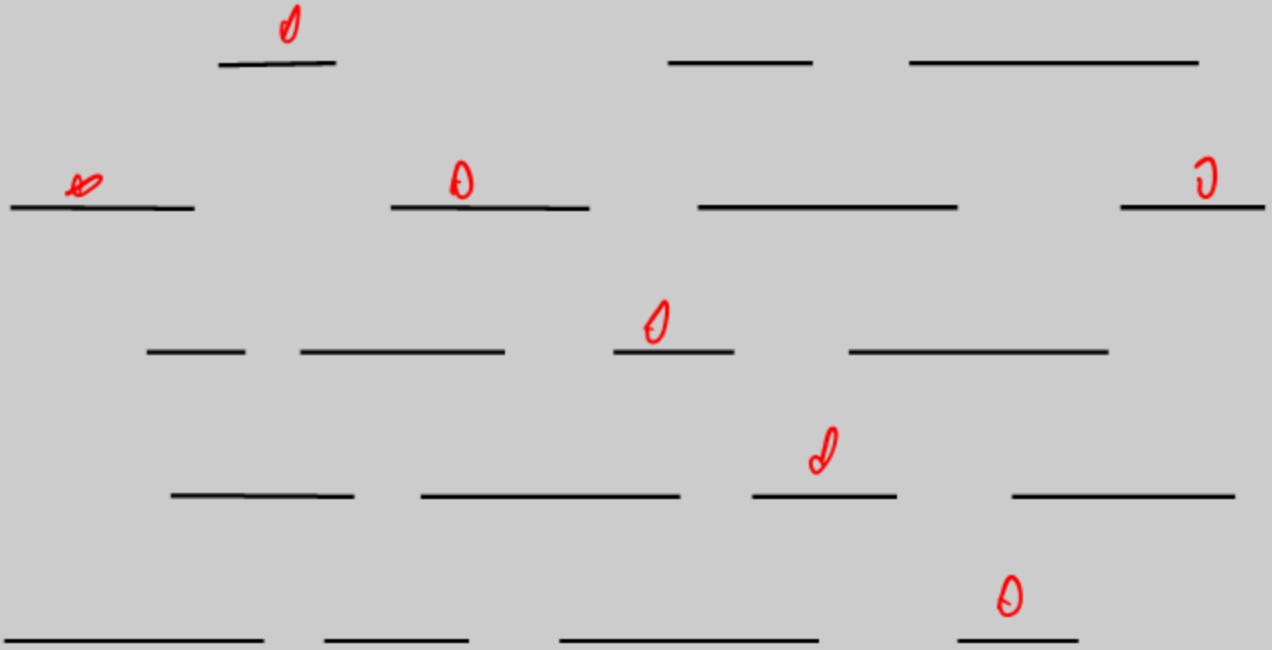
# Interval Scheduling

- Tasks occur at fixed times
- Single processor
- Maximize number of tasks completed



- Tasks  $\{1, 2, \dots, n\}$
- Start and finish times,  $s(i)$ ,  $f(i)$

# What is the largest solution?



# Greedy Algorithm for Scheduling

Let  $T$  be the set of tasks, construct a set of independent tasks  $I$ ,  $A$  is the rule determining the greedy algorithm

$I = \{\}$

While ( $T$  is not empty)

    Select a task  $t$  from  $T$  by a rule  $A$

    Add  $t$  to  $I$

    Remove  $t$  and all tasks incompatible with  $t$  from  $T$

# Interval Scheduling Heuristics

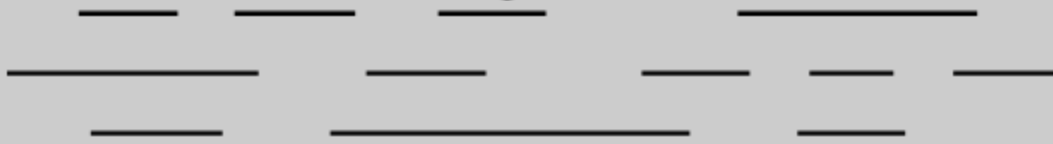
- Earliest starting time first
- Shortest interval first
- Smallest number of conflicting tasks
- Earliest finishing time first



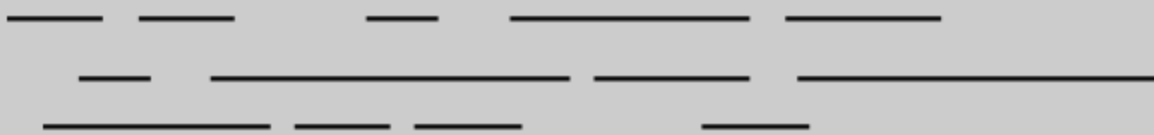


# Simulate the greedy algorithm for each of these heuristics

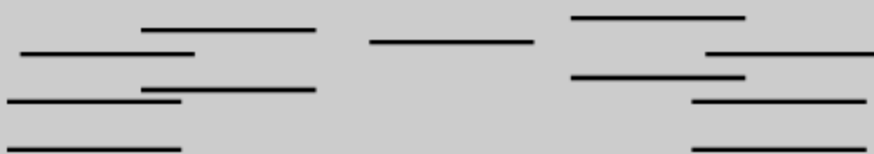
Schedule earliest starting task



Schedule shortest available task

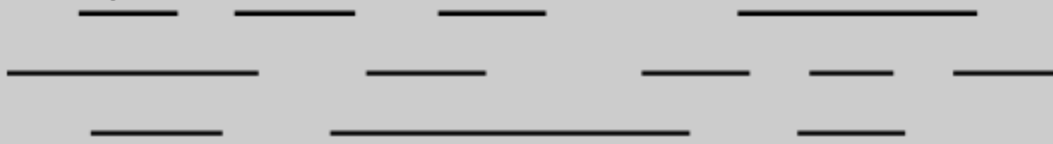


Schedule task with fewest conflicting tasks

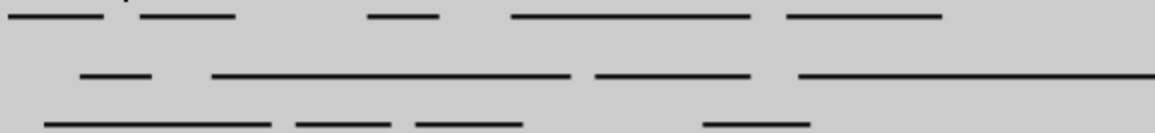


# Greedy solution based on earliest finishing time

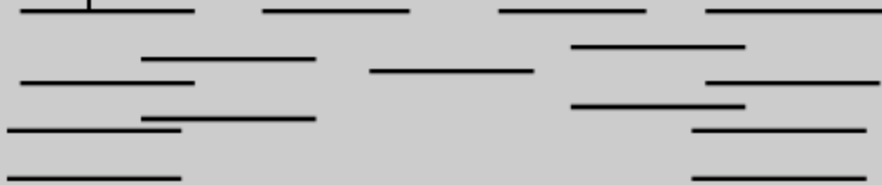
Example 1



Example 2



Example 3



## Theorem: Earliest Finish Algorithm is Optimal

- Key idea: Earliest Finish Algorithm stays ahead
- Let  $A = \{i_1, \dots, i_k\}$  be the set of tasks found by EFA in increasing order of finish times
- Let  $B = \{j_1, \dots, j_m\}$  be the set of tasks found by a different algorithm in increasing order of finish times
- Show that for  $r \leq \min(k, m)$ ,  $f(i_r) \leq f(j_r)$

11

# Stay ahead lemma

- A always stays ahead of B,  $f(i_r) \leq f(j_r)$
- Induction argument
  - $f(i_1) \leq f(j_1)$  ✓
  - If  $f(i_{r-1}) \leq f(j_{r-1})$  then  $f(i_r) \leq f(j_r)$  ✓

$$\begin{array}{cc}
 \underline{i_{r-1}} & \underline{i_r} \\
 \underline{j_{r-1}} & \underline{j_r}
 \end{array}$$

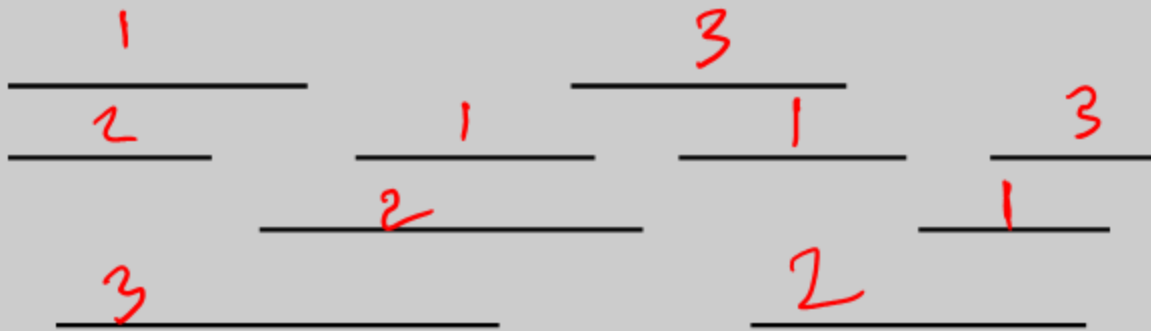
# Completing the proof

- Let  $A = \{i_1, \dots, i_k\}$  be the set of tasks found by EFA in increasing order of finish times
- Let  $O = \{j_1, \dots, j_m\}$  be the set of tasks found by an optimal algorithm in increasing order of finish times
- If  $k < m$ , then the Earliest Finish Algorithm stopped before it ran out of tasks

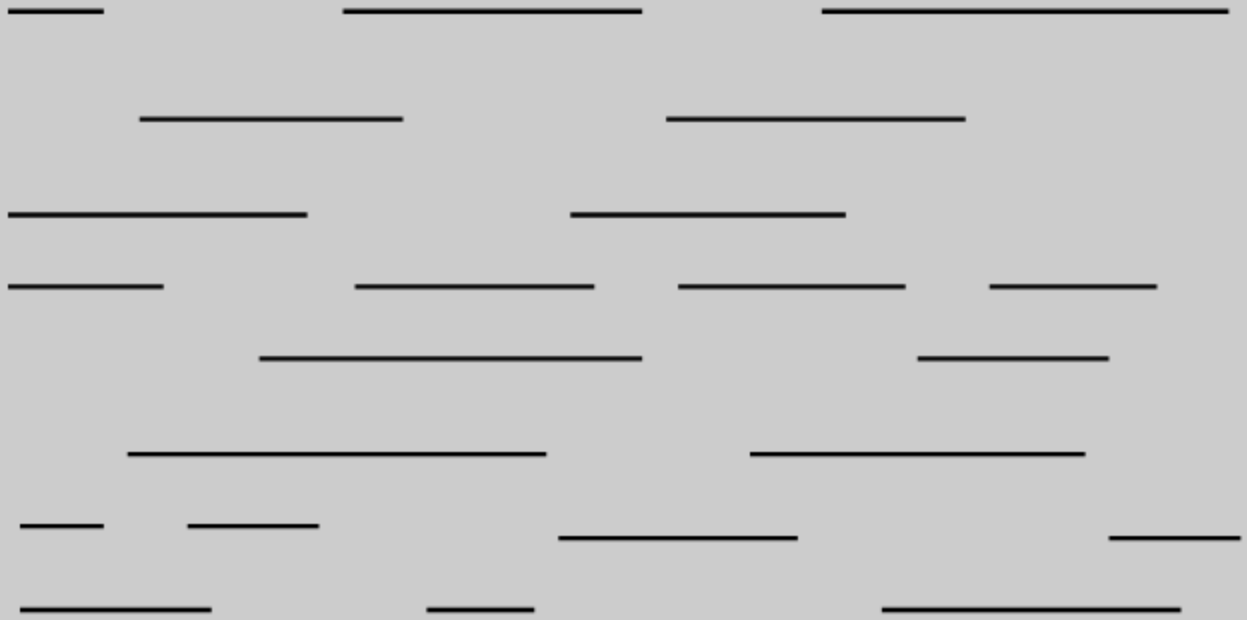
$$\begin{array}{c} i_k \\ \hline j_k \end{array} \quad \begin{array}{c} j_{k+1} \end{array}$$

# Scheduling all intervals

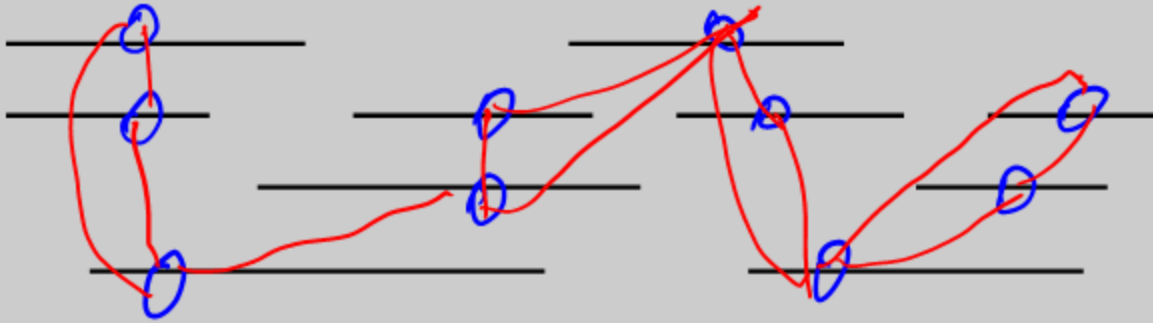
- Minimize number of processors to schedule all intervals



# How many processors are needed for this example?

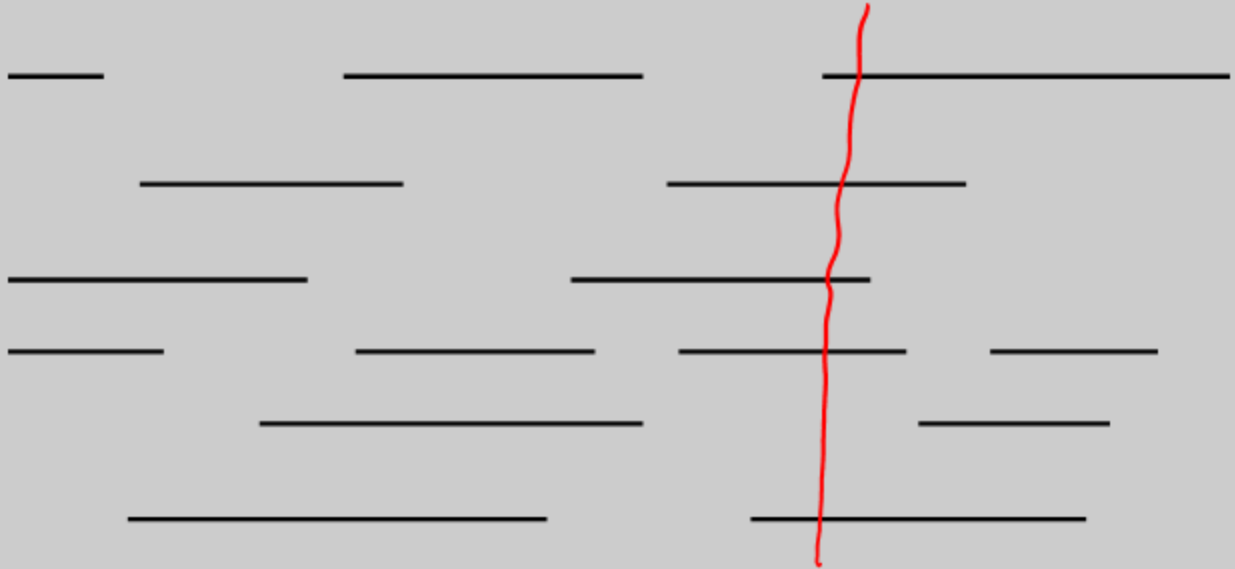


Prove that you cannot schedule this set of intervals with two processors





# Depth: maximum number of intervals active



# Algorithm

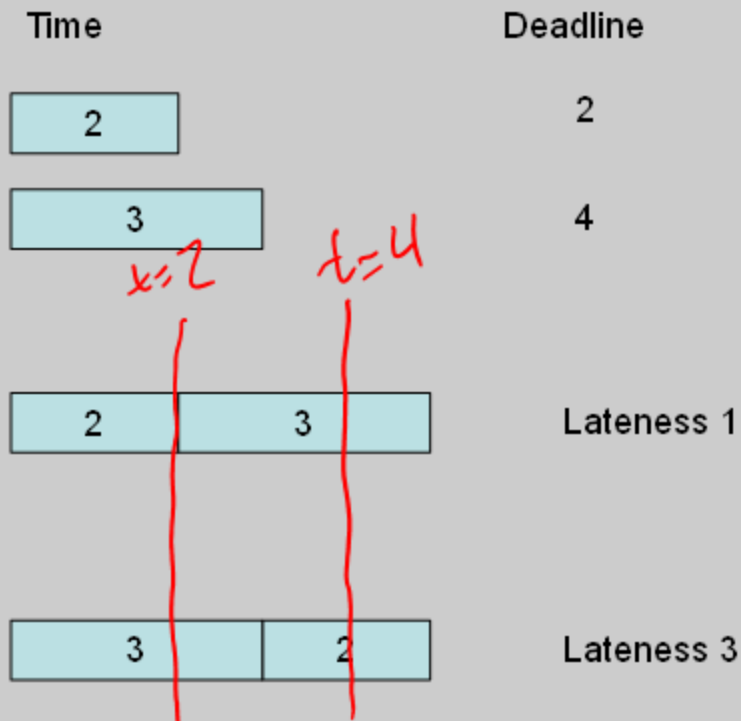
- Sort by start times
- Suppose maximum depth is  $d$ , create  $d$  slots
- Schedule items in increasing order, assign each item to an open slot
- Correctness proof: When we reach an item, we always have an open slot

By contradiction — if no slot  
for task  $t$ ,  
then  $d$  tasks conflict  
with  $t$ , and depth is  
 $d+1$

# Scheduling tasks

- Each task has a length  $t_i$  and a deadline  $d_i$
- All tasks are available at the start
- One task may be worked on at a time
- All tasks must be completed
  
- Goal minimize maximum lateness
  - Lateness =  $f_i - d_i$  if  $f_i \geq d_i$

# Example



# Determine the minimum lateness

Time

Deadline

2

6

3

4

4

5

5

12



21