Lecture08

CSE 417
Algorithms and Complexity

Greedy Algorithms

Autumn 2024

Lecture 8

1

Announcements

- Reading
  - For today, sections 4.1, 4.2,
  - For next week sections 4.4, 4.5, 4.7, 4.8
- Homework 3 is available
  - Graph algorithms
  - Programming problem: Random Graphs

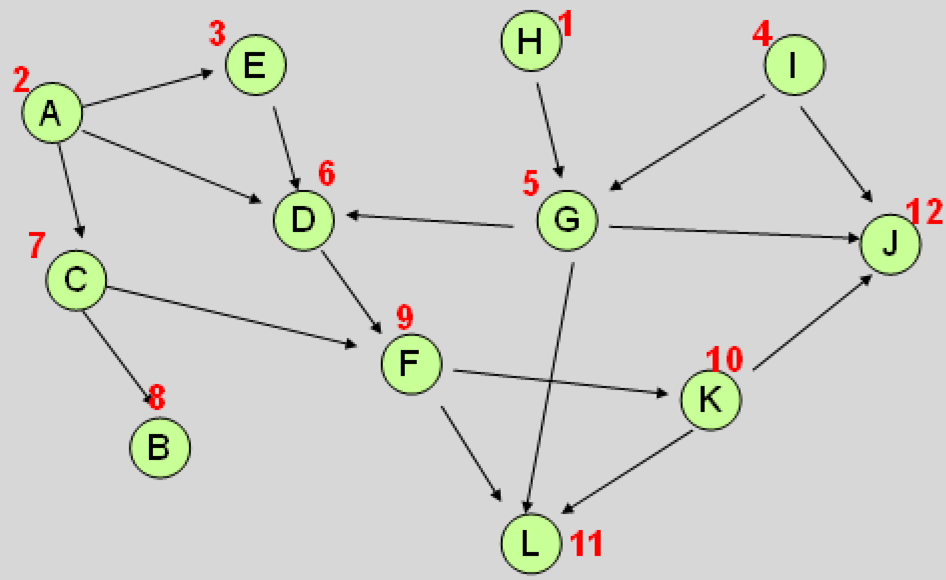
2

Highlight from last lecture: Topological Sort Algorithm

While there exists a vertex v with in-degree 0

Output vertex v

Delete the vertex v and all out going edges



3

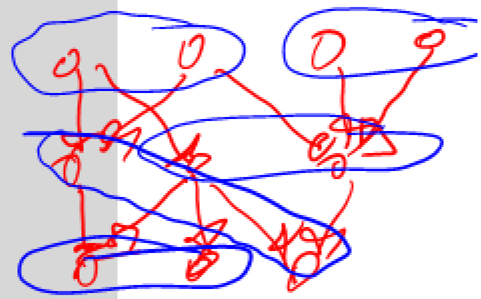
Greedy Algorithms

- 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

4

Scheduling Theory

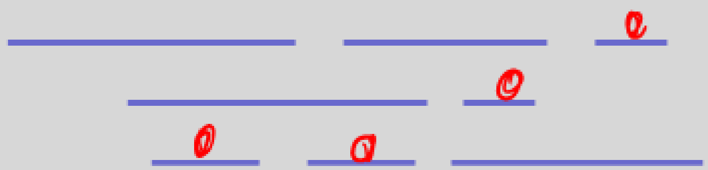
- Tasks
  - Processing requirements, release times, deadlines
- Processors
- Precedence constraints
- Objective function
  - Jobs scheduled, lateness, total execution time



5

Interval Scheduling

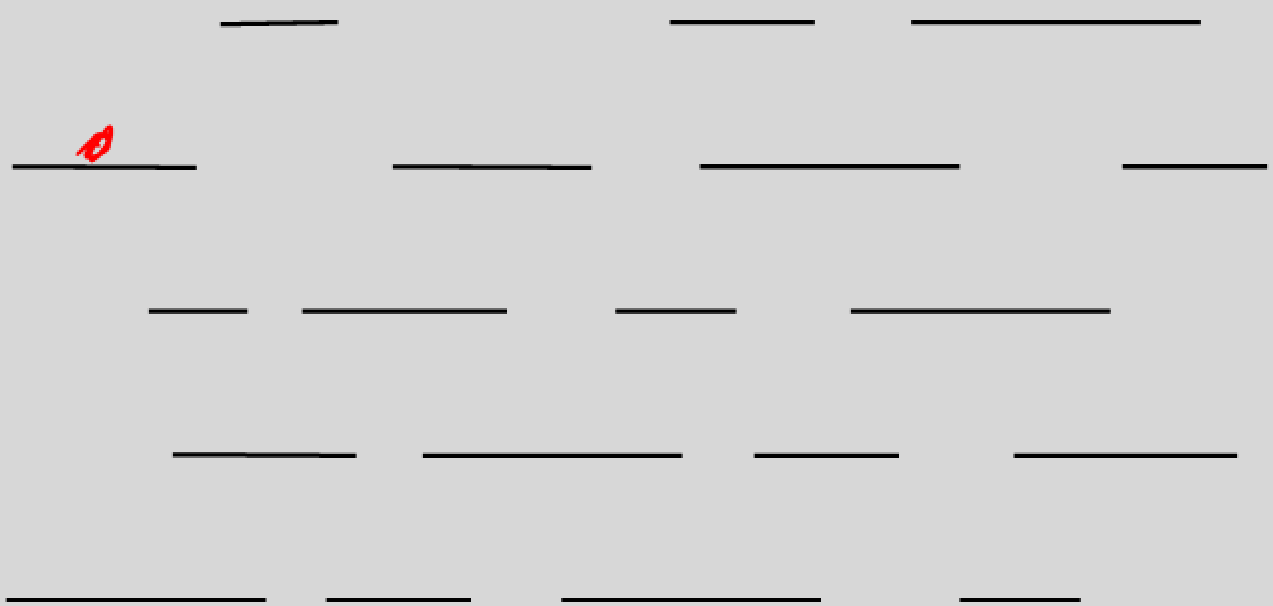
- Tasks occur at fixed times
- Single processor
- Maximize number of tasks completed



- Tasks {1, 2, . . . N}
- Start and finish times: s(i), f(i)

6

What is the largest solution?



7

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

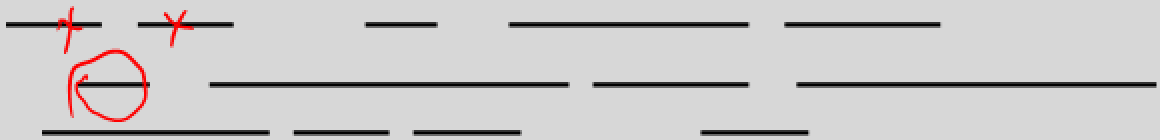
    Remove t and all tasks incompatible with t from T

8

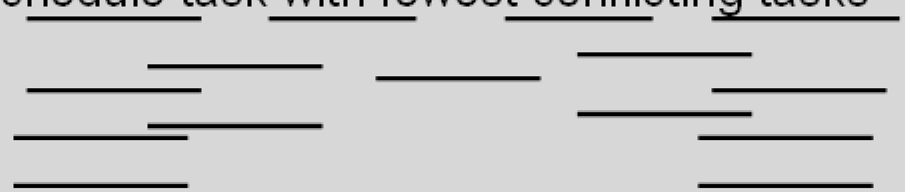
Simulate the greedy algorithm for each of these heuristics

Schedule earliest starting task

Schedule shortest available task

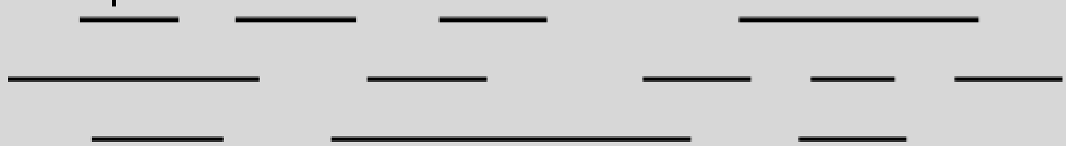


Schedule task with fewest conflicting tasks

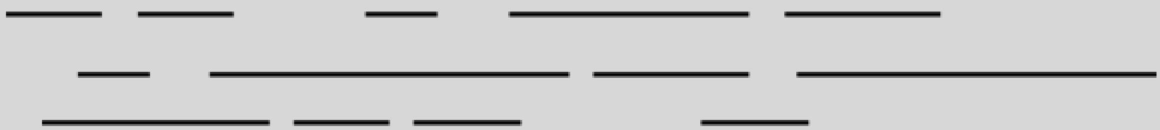


Greedy solution based on earliest finishing time

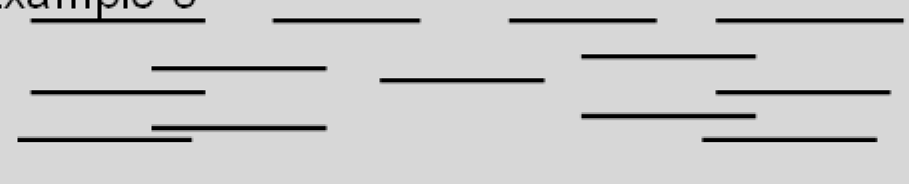
Example 1



Example 2



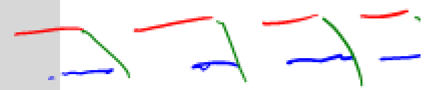
Example 3



10

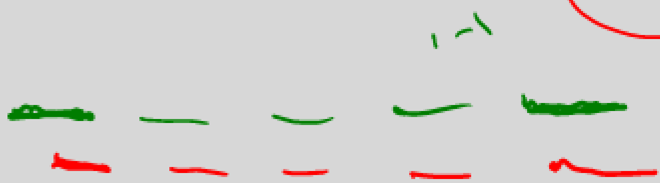
Theorem: Earliest Finish Algorithm is Optimal

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



Stay ahead lemma

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



12

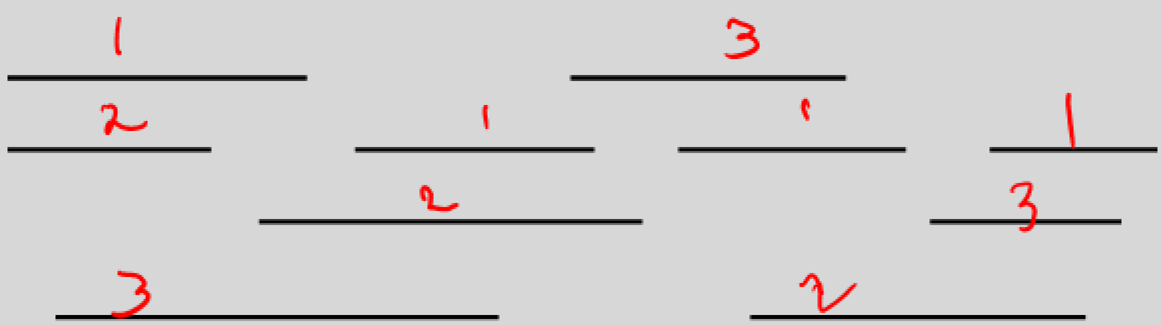
Completing the proof

- Let A = $\{i_1, \ldots, i_k\}$ be the set of tasks found by EFA in increasing order of finish times
- Let O = $\{j_1, \ldots, j_m\}$ 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

13

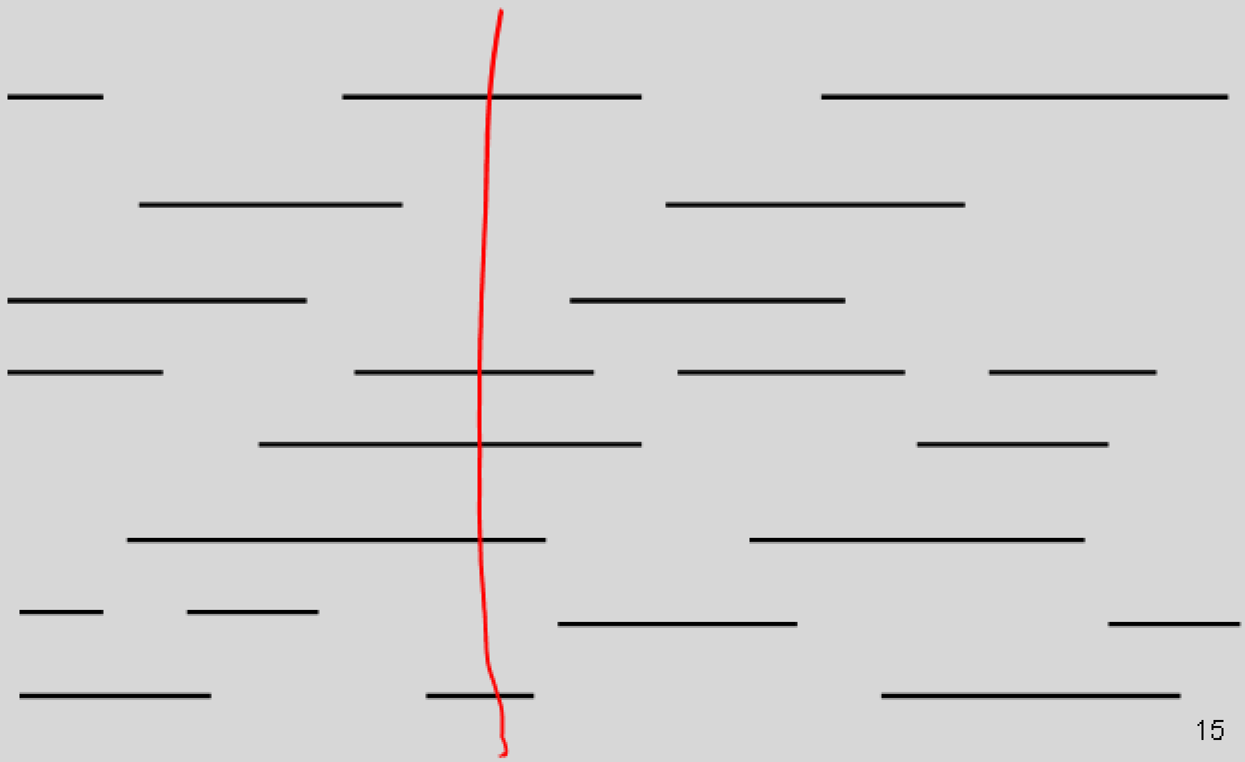
Scheduling all intervals

- Minimize number of processors to schedule all intervals



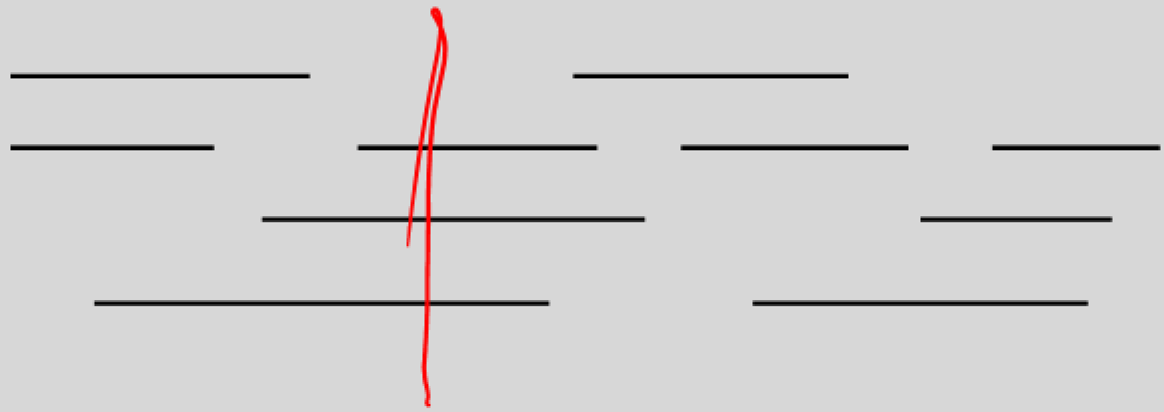
14

How many processors are needed for this example?



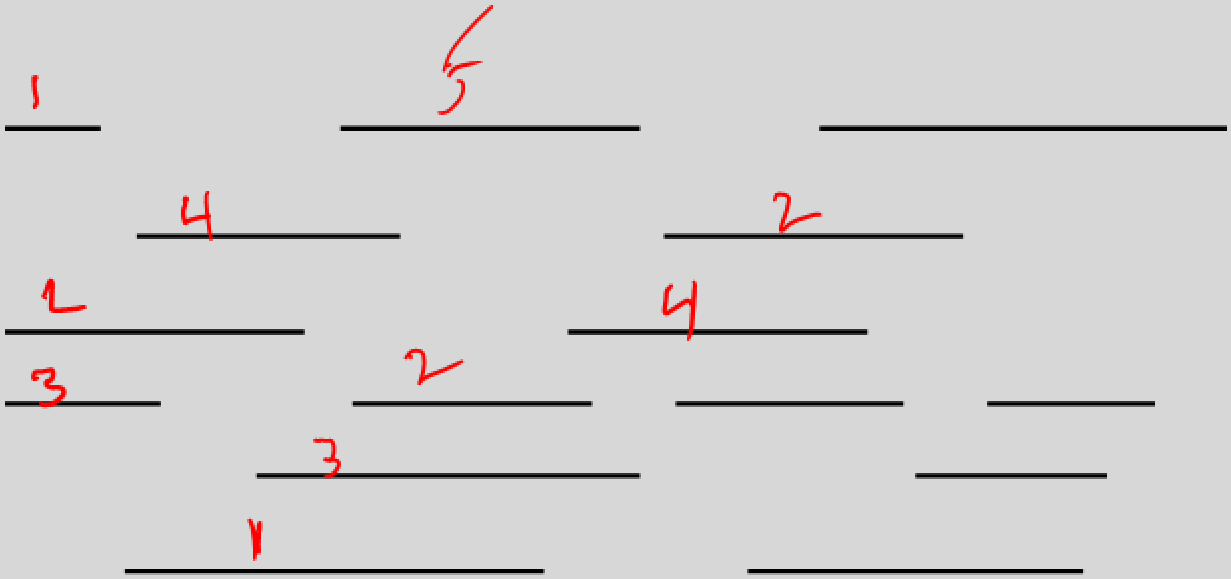
15

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



16

Depth: maximum number of intervals active



17

Algorithm

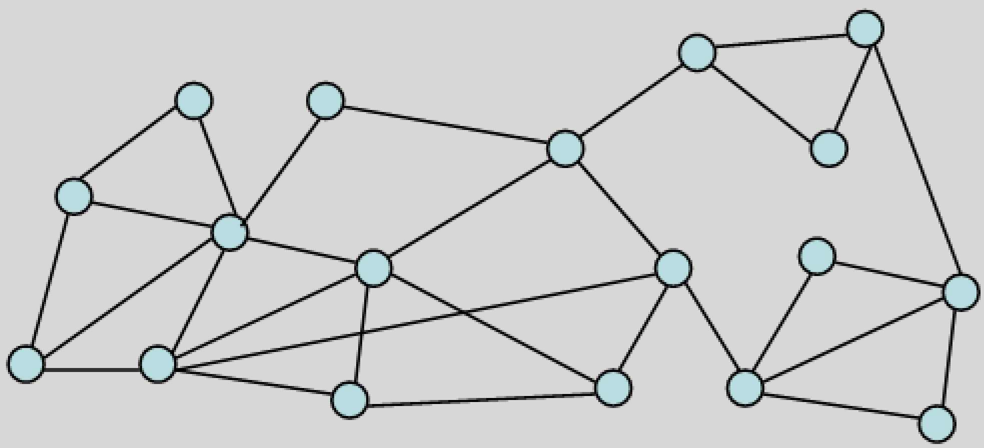
- Sort by start times
- Suppose maximum depth is d, create d slots
- Schedule items in increasing order, assign each item to an open slot

- Correctness proof: When we reach an item, we always have an open slot

18

Greedy Graph Coloring

Theorem: An undirected graph with maximum degree K can be colored with K+1 colors



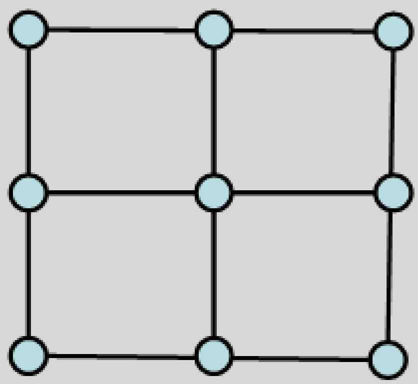
19

Coloring Algorithm, Version 1

```
Let k be the largest vertex degree
Choose k+1 colors

for each vertex v
      Color[v] = uncolored

for each vertex v
      Let c be a color not used in N[v]
      Color[v] = c
```

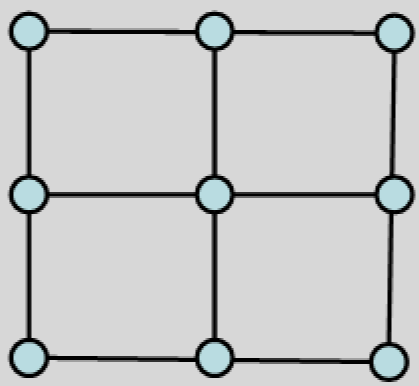


20

Coloring Algorithm, Version 2

```
for each vertex v
       Color[v] = uncolored

for each vertex v
       Let c be the smallest color not used in N[v]
       Color[v] = c
```



21

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 $f_i \geq d_i$

22

Example

Time	Deadline
2	2
3	4

2 | 3 — Lateness 1

3 | 2 — Lateness 3

23

Determine the minimum lateness

Time	Deadline
2	6
3	4
4	5
5	12