CSE 417
Algorithms and Complexity

Greedy Algorithms

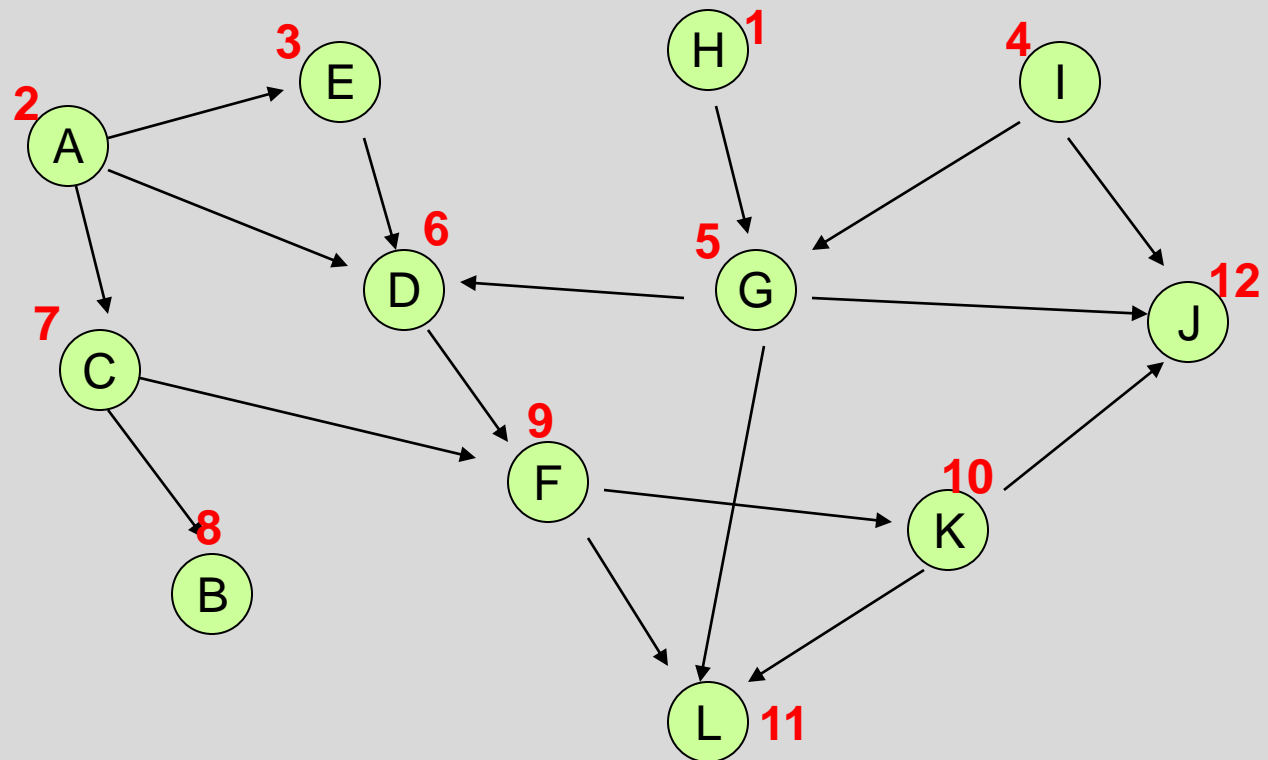## Autumn 2024

Lecture 8

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

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
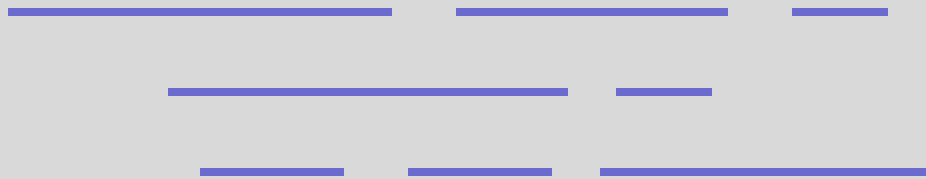
Greedy Algorithms

- 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, . . . 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
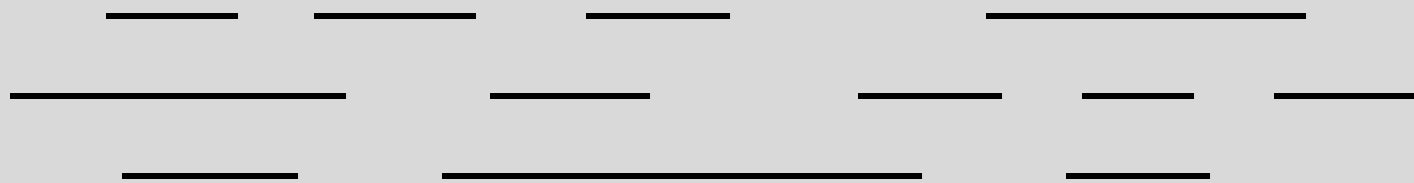
I = { }

While (T is not empty)
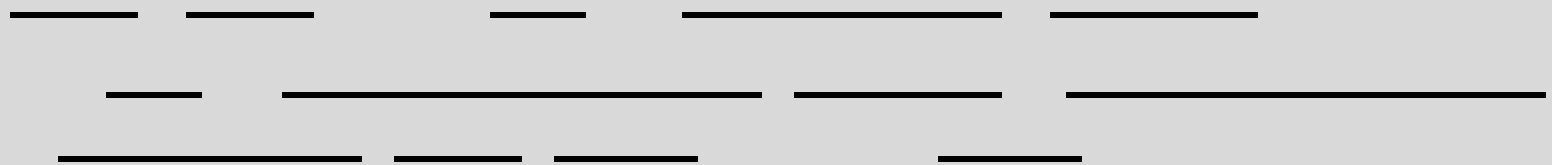
  Select a task t from T by a rule A

  Add t to I

  Remove t and all tasks incompatible with t from T

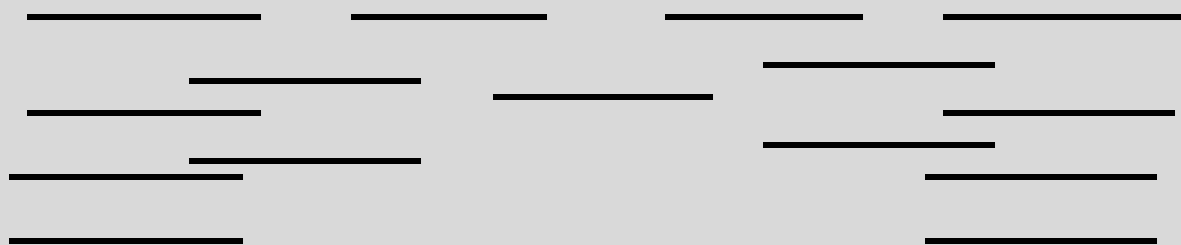# Simulate the greedy algorithm for each of these heuristics
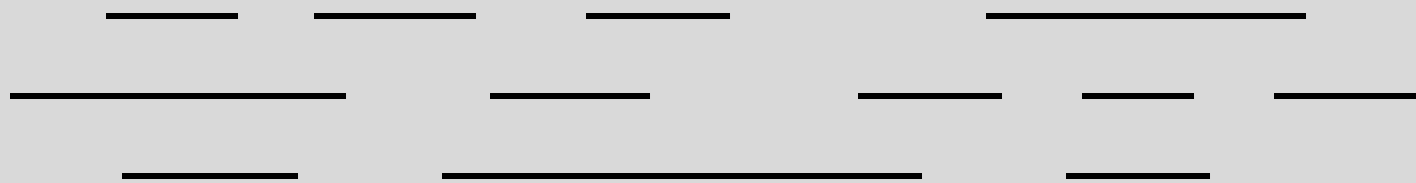
Schedule earliest starting task

___   ____   ____   _____

_____   ____   ___ ___ ____

____   _____   ____

Schedule shortest available task

____ ____   ___ _____ _____

___ _____ ____ _____

_____ ___ ____   ____

Schedule task with fewest conflicting tasks

_____   _____   _____   _____

_____   _____   _____

_____   _____

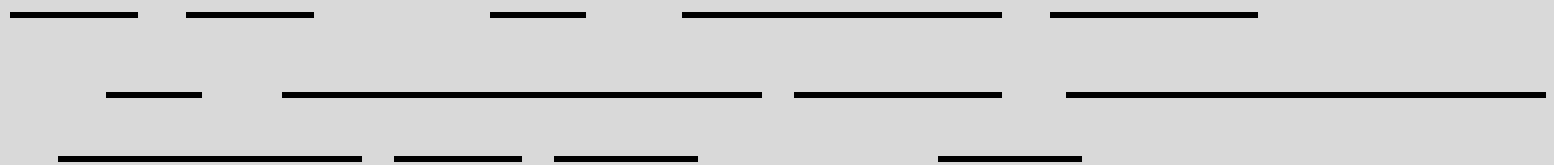_____   _____

_____   _____

Greedy solution based on earliest finishing time
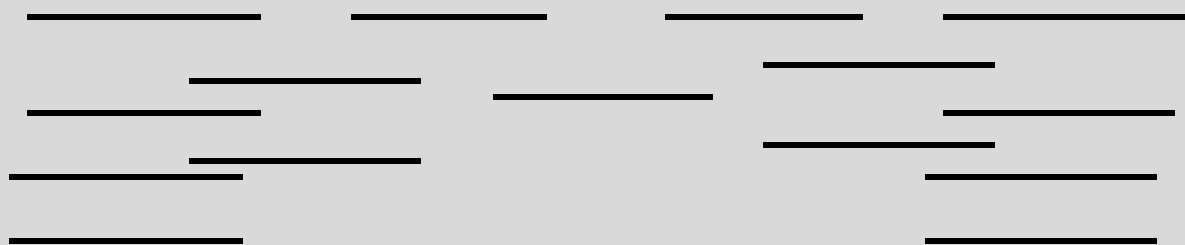
Example 1

Example 2

Example 3

Theorem: Earliest Finish Algorithm is Optimal

- Key idea: Earliest Finish Algorithm stays ahead

- Let A = {i_1, . . ., 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 ≤ min(k, m), f(i_r) ≤ f(j_r)

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 $f(i_r) \leq f(j_r)$

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
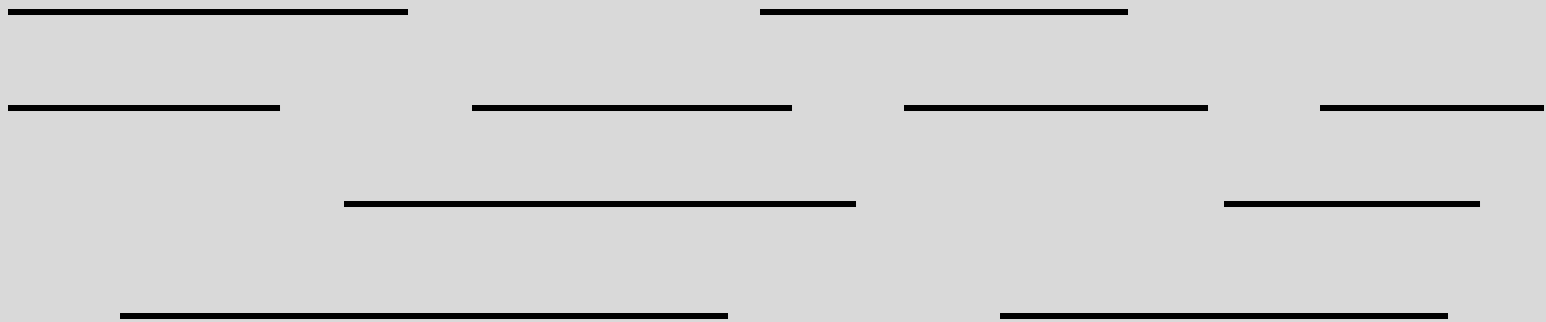
Scheduling all intervals

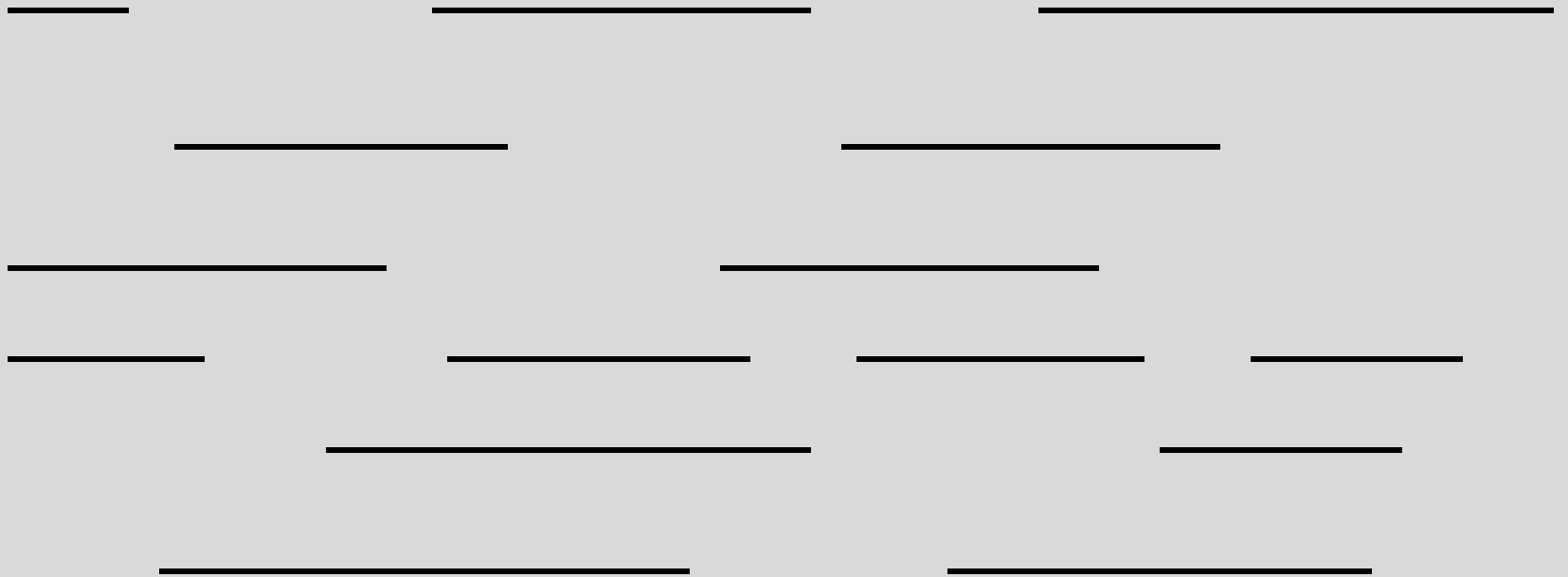- Minimize number of processors to schedule all intervals
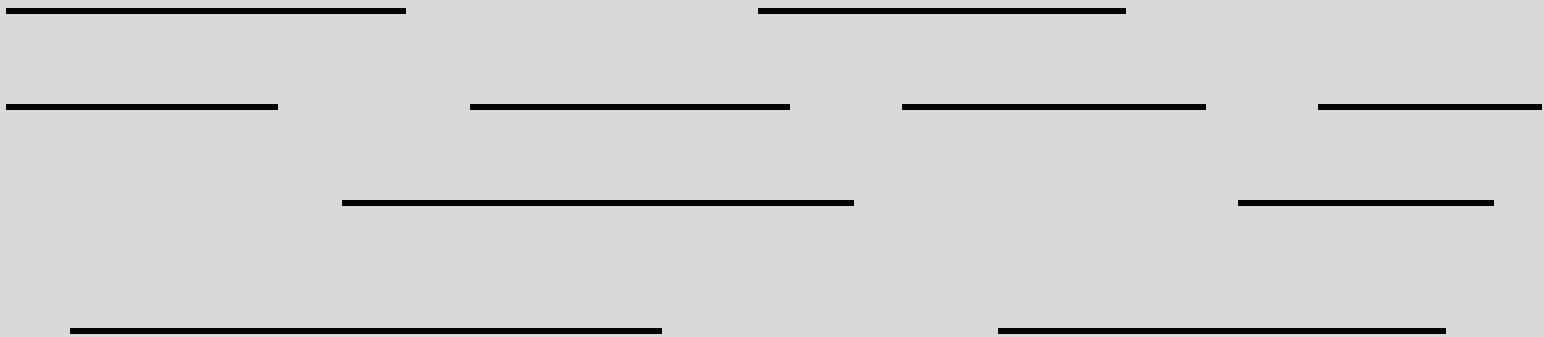
How many processors are needed for this example?
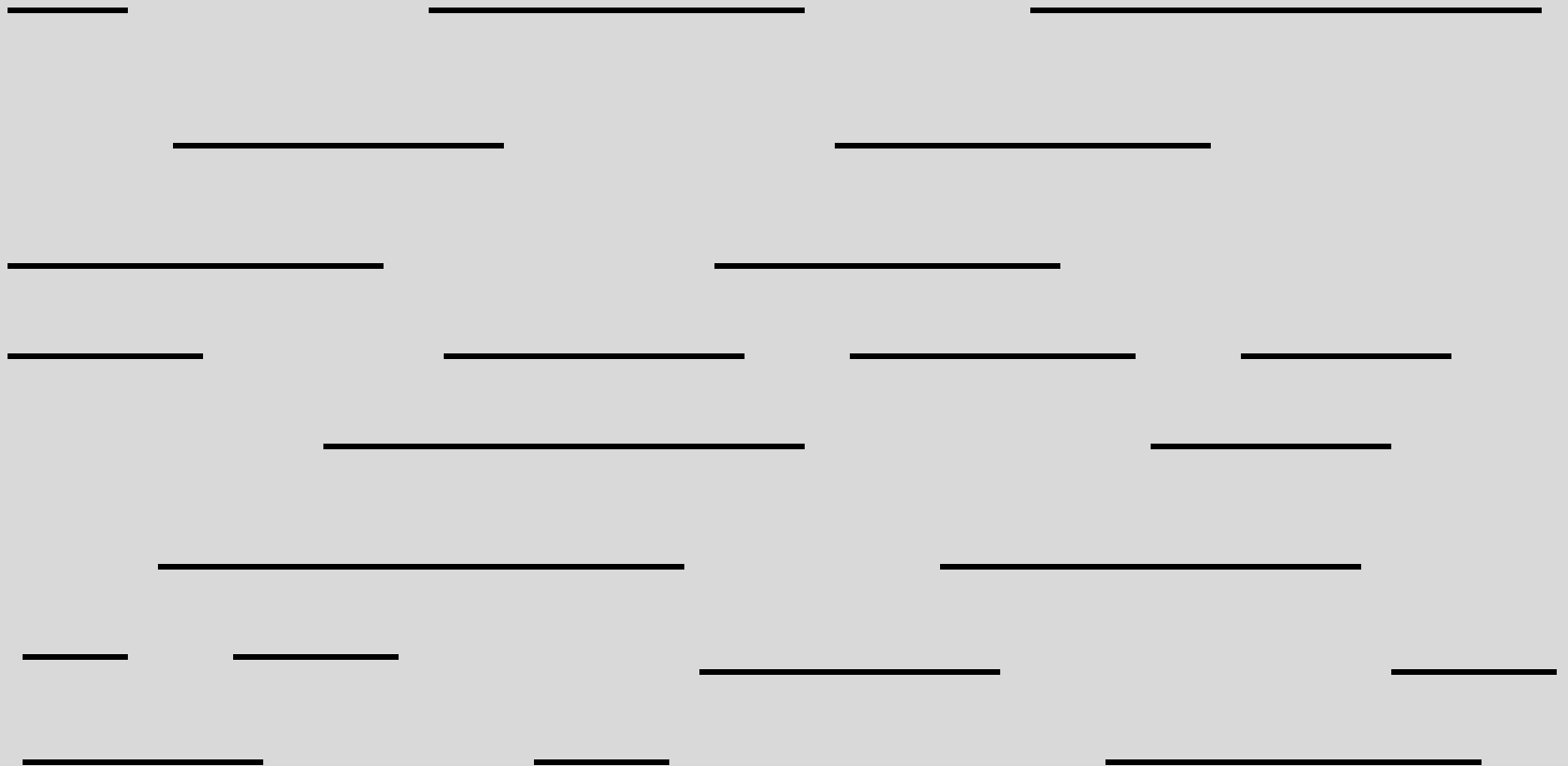
Prove that you cannot schedule this set of intervals with two processors
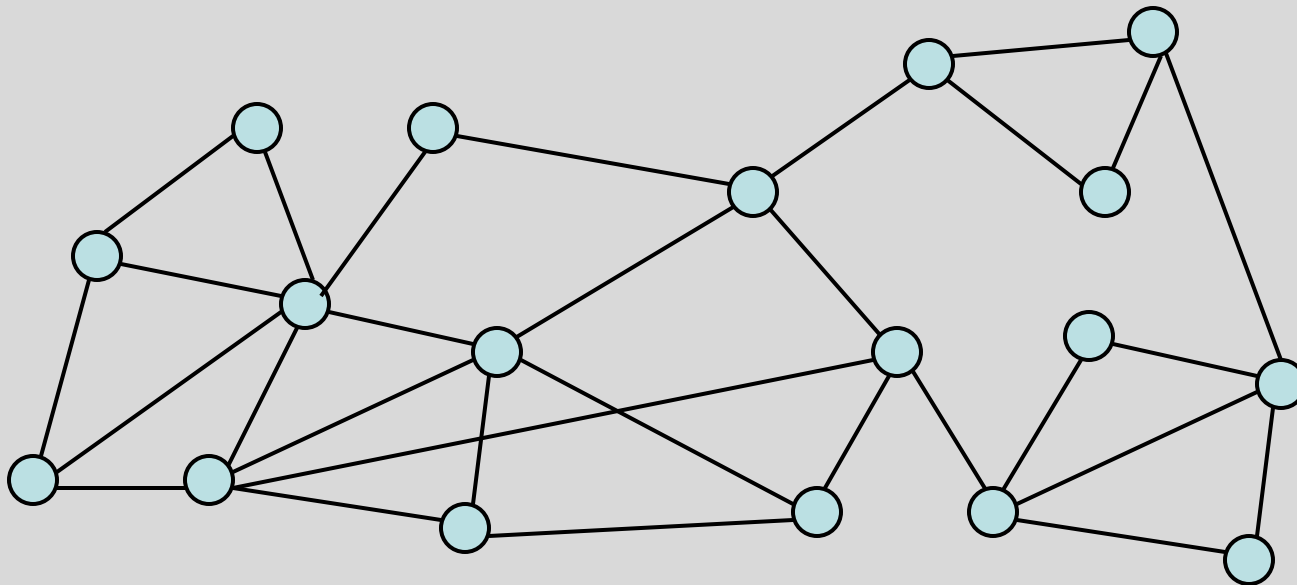
Depth: maximum number of intervals active

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

Greedy Graph Coloring

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

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
```
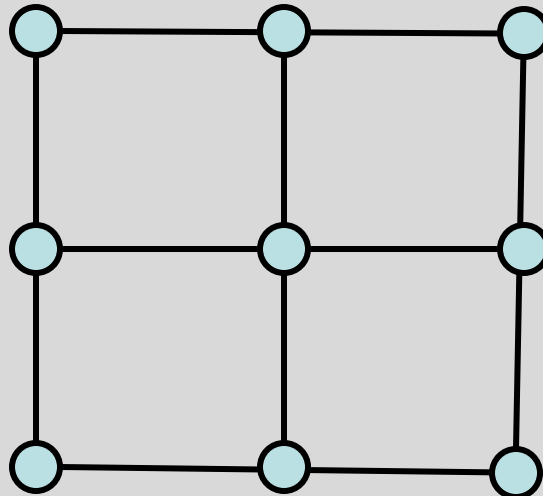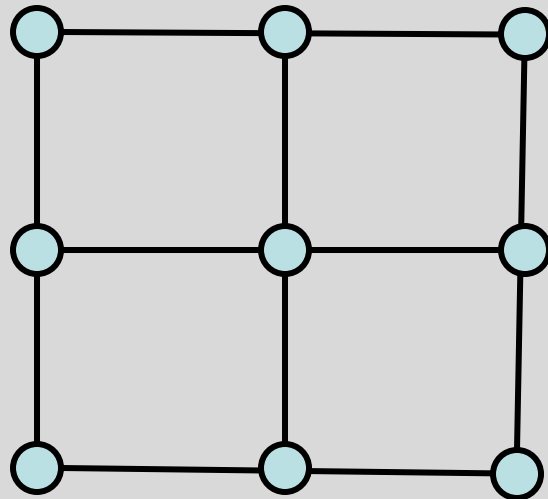
Coloring Algorithm, Version 2

```
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
```
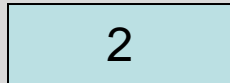
Scheduling tasks

- 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 $f_i \geq d_i$
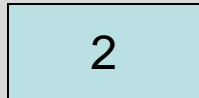
Example

Time   Deadline

2	2
3	4

2	3	Lateness 1

3	2	Lateness 3

Determine the minimum lateness

Time   Deadline

| 2 | 6

| 3 | 4

| 4 | 5

| 5 | 12