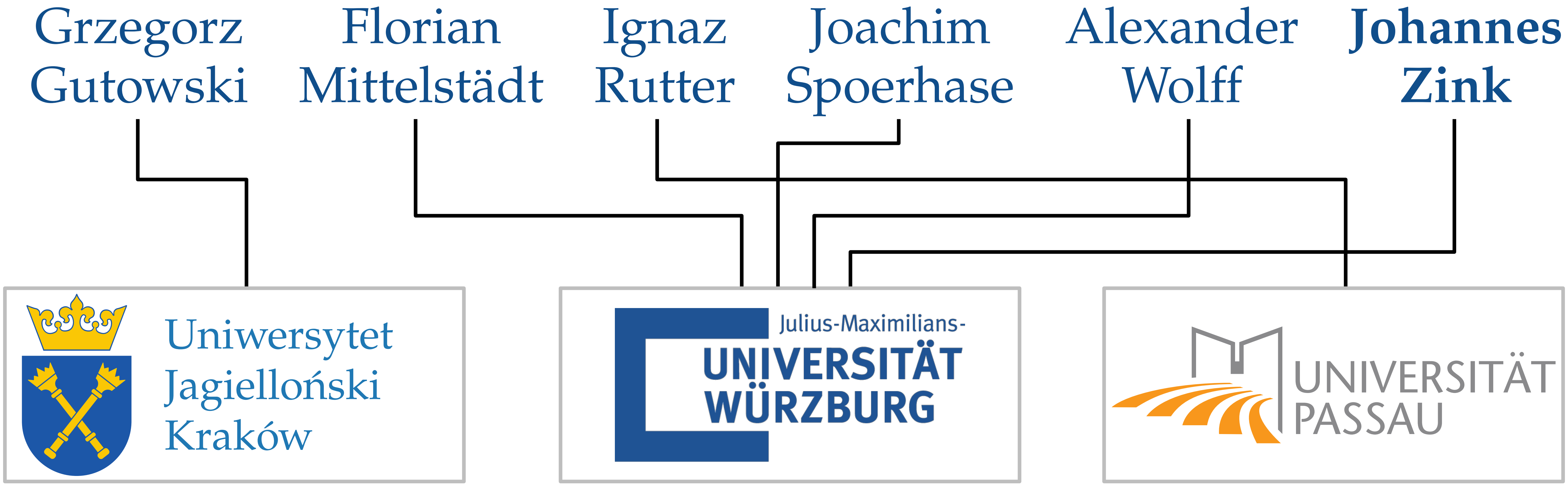
Coloring Mixed and Directional Interval Graphs

GD 2022, Tokyo



Grzegorz Gutowski

Florian Mittelstädt

Ignaz Rutter

Joachim Spoerhase

Alexander Wolff

Johannes Zink

Uniwersytet Jagielloński Kraków

Julius-Maximilians-UNIVERSITÄT WÜRZBURG

UNIVERSITÄT PASSAU

Motivation

Framework for layered graph drawing by Sugiyama, Tagawa, and Toda (1981).

Motivation

Framework for layered graph drawing by Sugiyama, Tagawa, and Toda (1981).

Input: directed graph G **Output:** layered drawing of G

Motivation

Framework for layered graph drawing by Sugiyama, Tagawa, and Toda (1981).

Input: directed graph G **Output:** layered drawing of G

Consists of five phases:

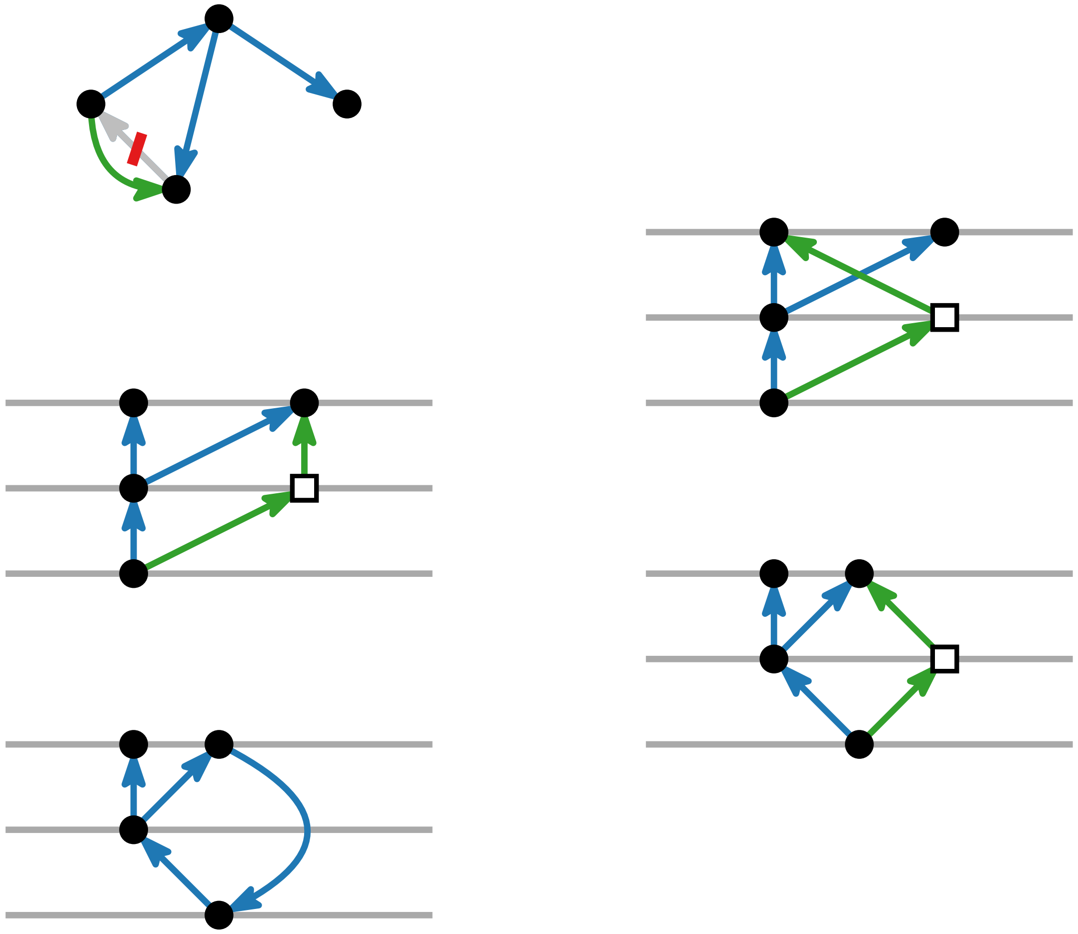
Motivation

Framework for layered graph drawing by Sugiyama, Tagawa, and Toda (1981).

Input: directed graph G **Output:** layered drawing of G

Consists of five phases:

1. cycle elimination
2. layer assignment
3. crossing minimization
4. node placement
5. edge routing



Motivation

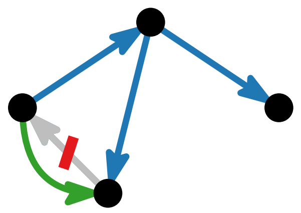
Framework for layered graph drawing by Sugiyama, Tagawa, and Toda (1981).

Input: directed graph *G*

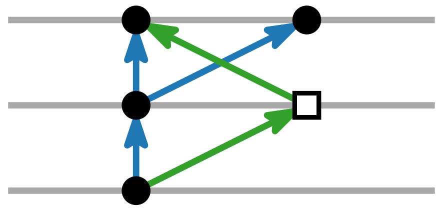
Output: layered drawing of *G*

Consists of five phases:

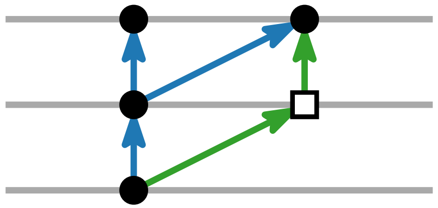
1. cycle elimination



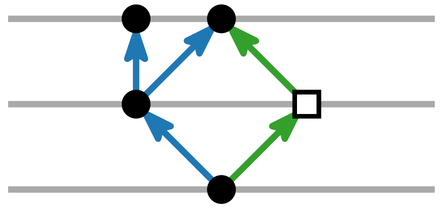
2. layer assignment



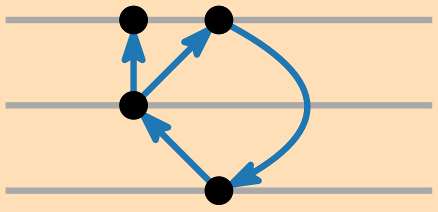
3. crossing minimization



4. node placement



5. edge routing



we want orthogonal edges!

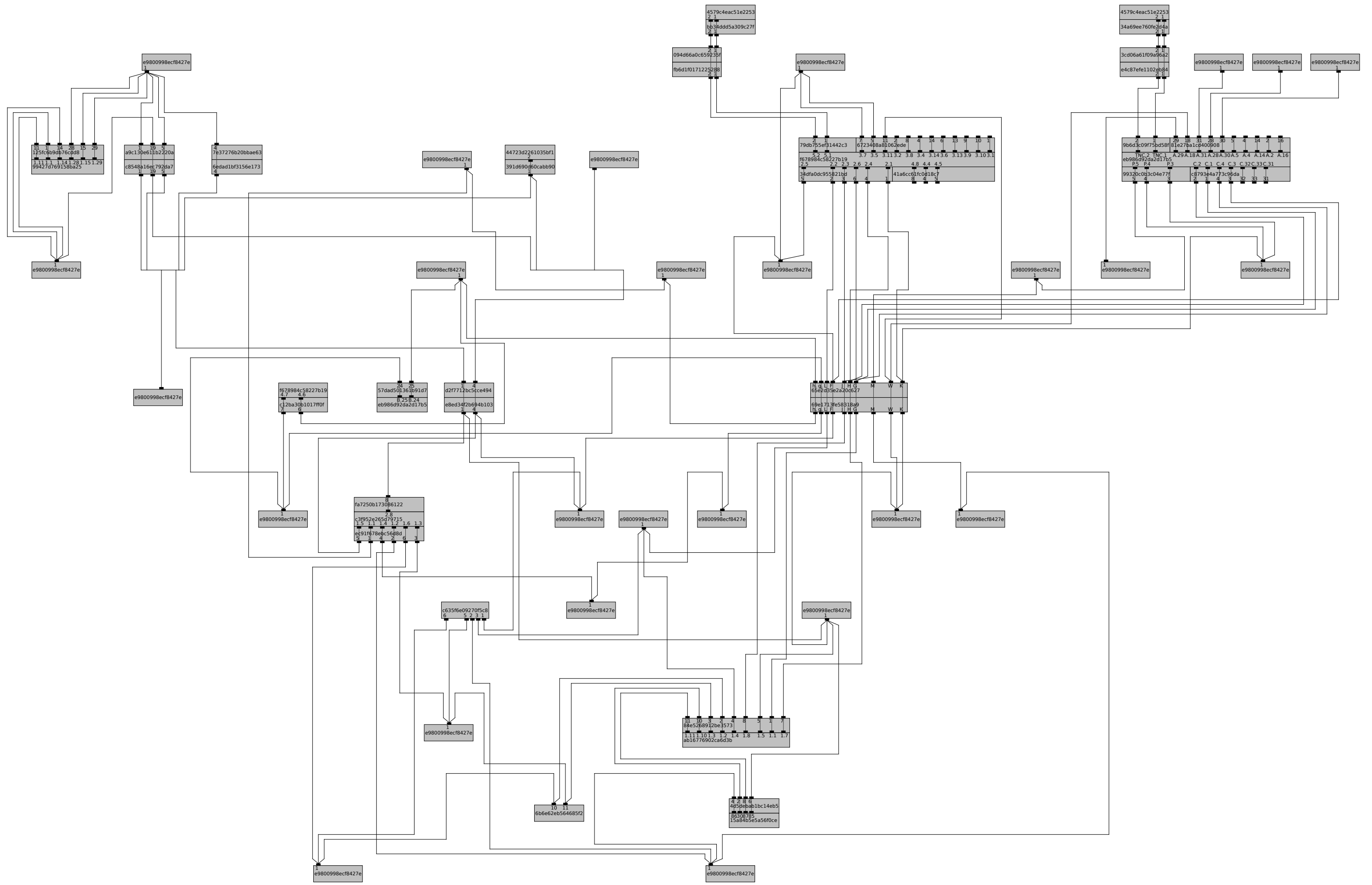
Motivation

Framework for layered g

Input: directed graph G

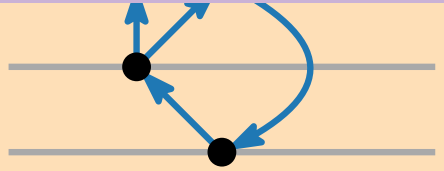
Consists of five phases:

1. cycle elimination
2. layer assignment
3. crossing minimiza
4. node placement
5. edge routing



cable plan

[Zink, Walter, Baumeister, Wolff; CGTA'22]



we want orthogonal edges!

Motivation – Layered Orthogonal Edge Routing

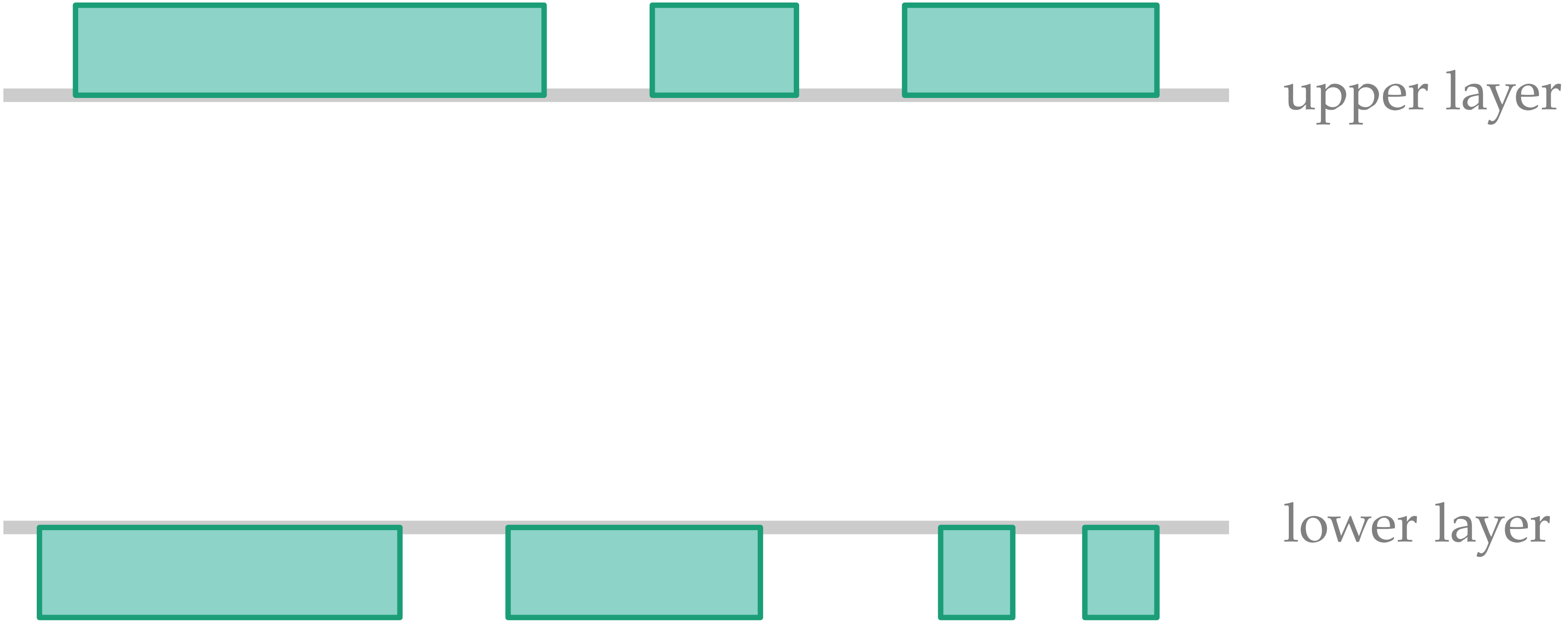
- It suffices to consider each pair of consecutive layers individually.

Motivation – Layered Orthogonal Edge Routing

- It suffices to consider each pair of consecutive layers individually.

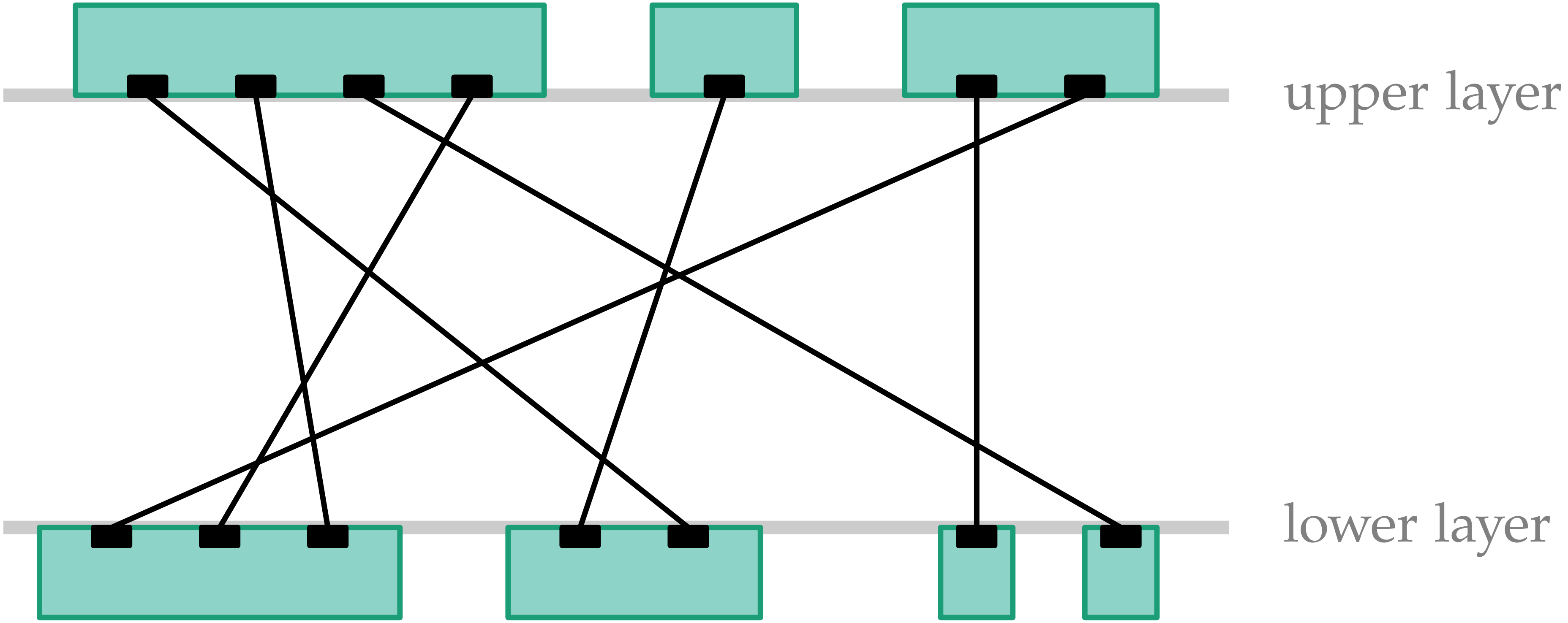
Motivation – Layered Orthogonal Edge Routing

- It suffices to consider each pair of consecutive layers individually.
- Positions of vertices are fixed.



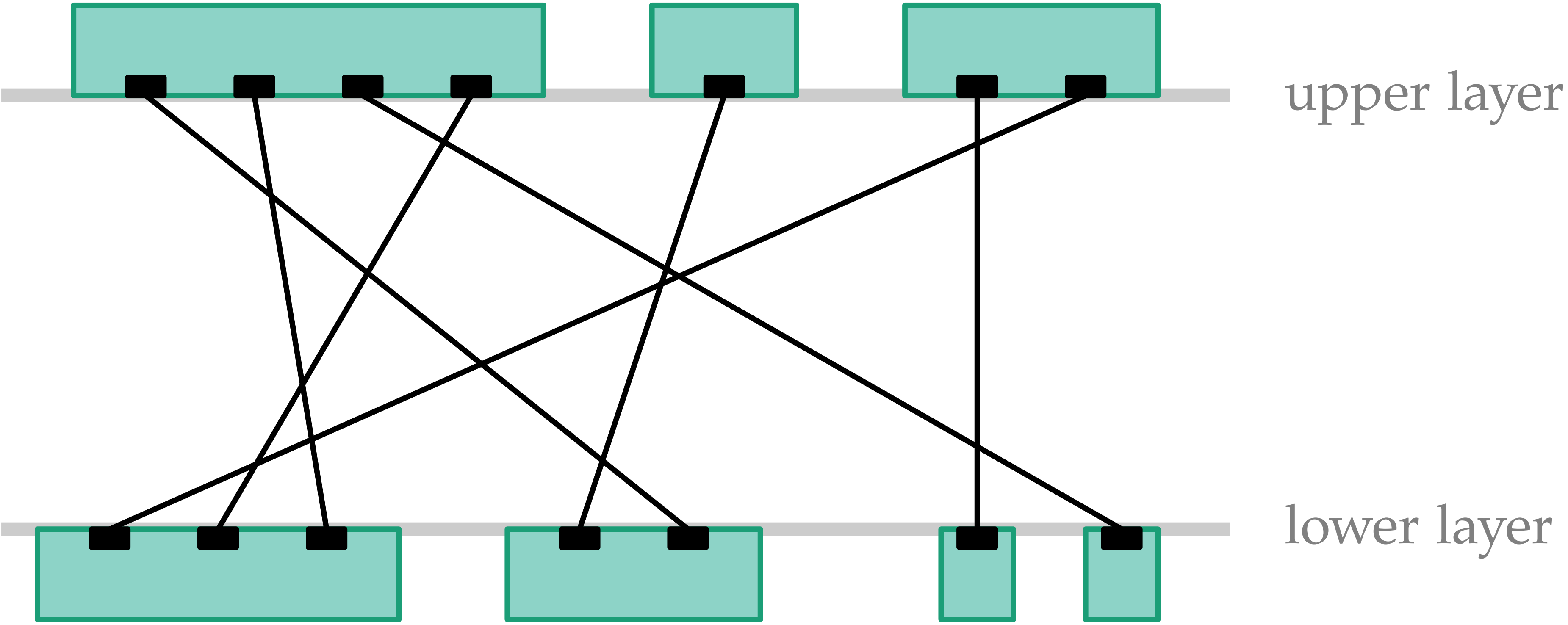
Motivation – Layered Orthogonal Edge Routing

- It suffices to consider each pair of consecutive layers individually.
- Positions of vertices are fixed.
- No two edges share a common end point (vertices have distinct ports).



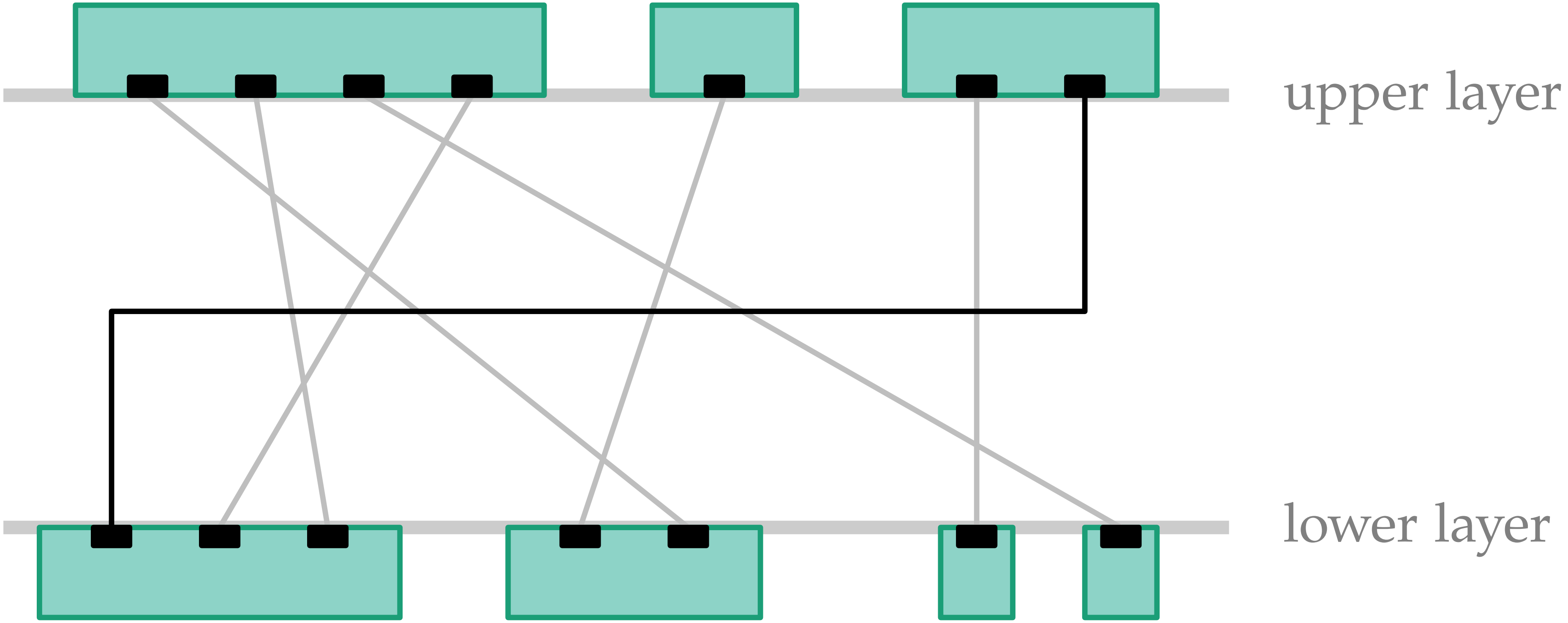
Motivation – Layered Orthogonal Edge Routing

- Draw each edge with at most two vertical and one horizontal line segments.



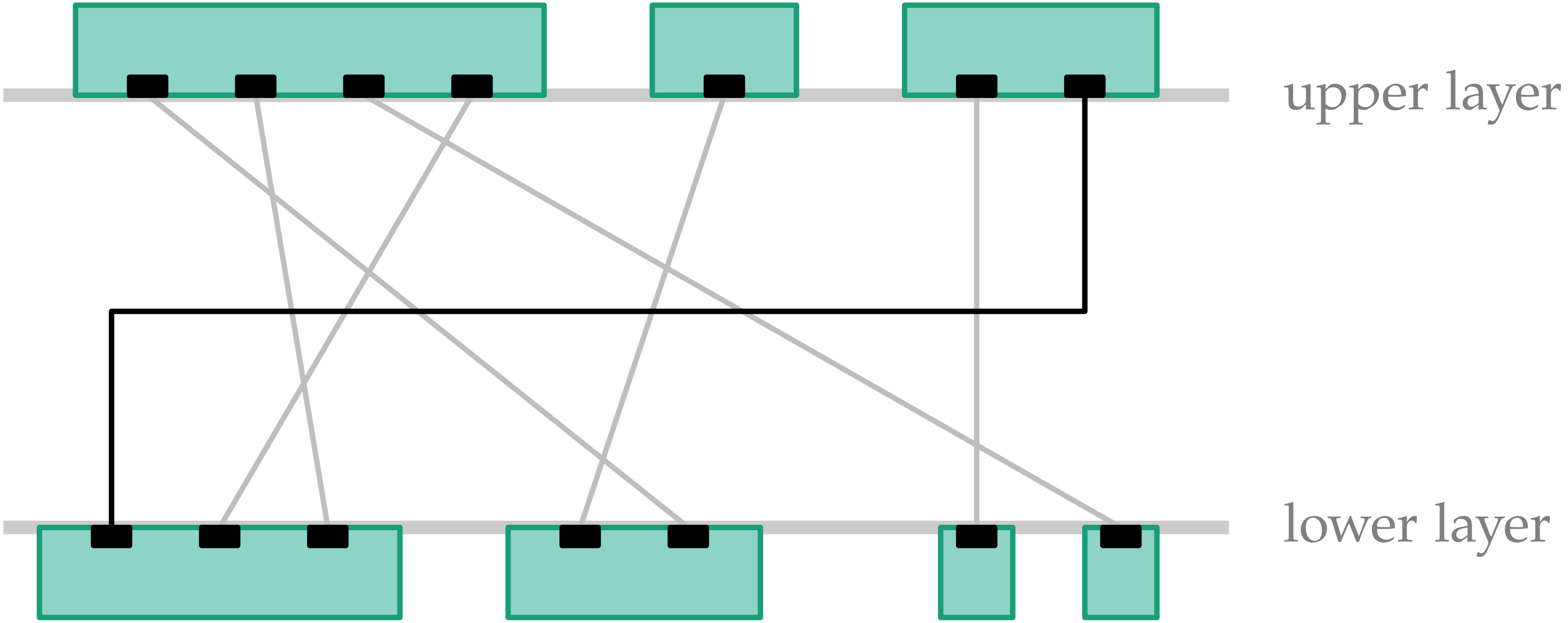
Motivation – Layered Orthogonal Edge Routing

- Draw each edge with at most two vertical and one horizontal line segments.



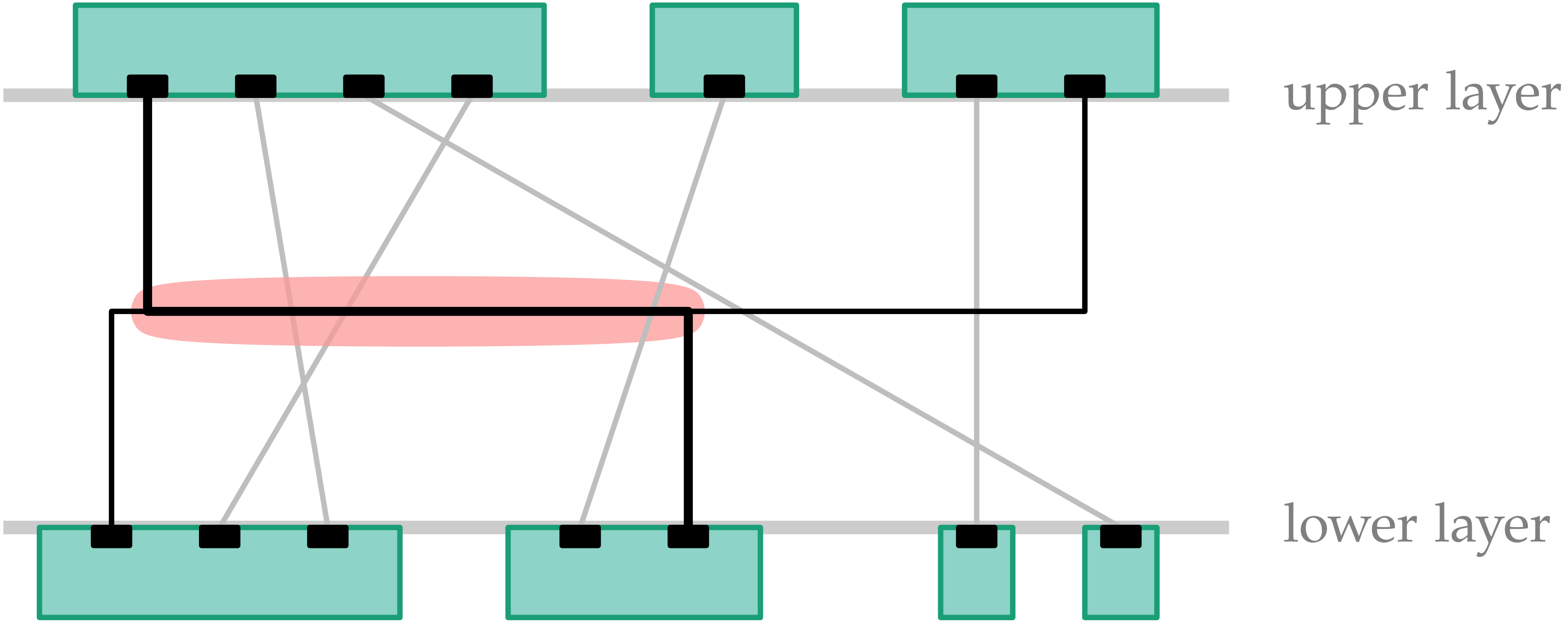
Motivation – Layered Orthogonal Edge Routing

- Draw each edge with at most two vertical and one horizontal line segments.
- Avoid overlaps and double crossings between the same pair of edges.



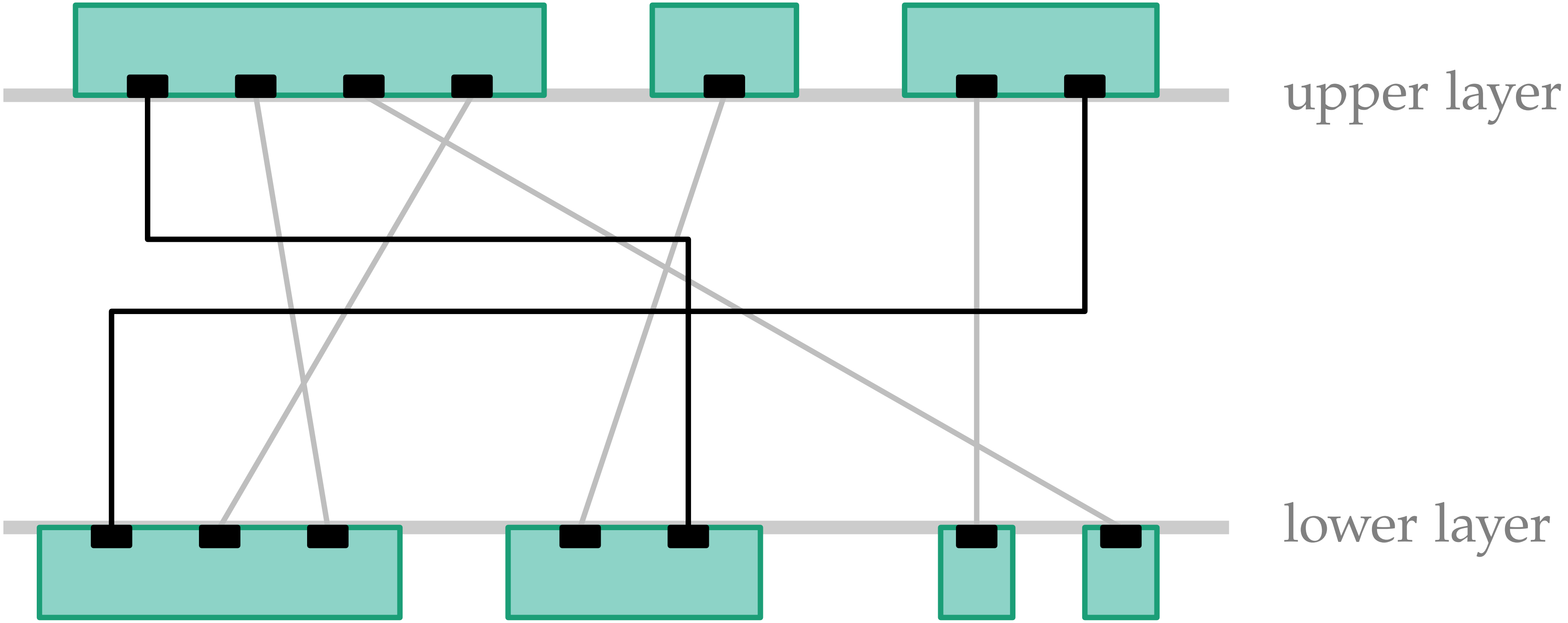
Motivation – Layered Orthogonal Edge Routing

- Draw each edge with at most two vertical and one horizontal line segments.
- Avoid overlaps and double crossings between the same pair of edges.



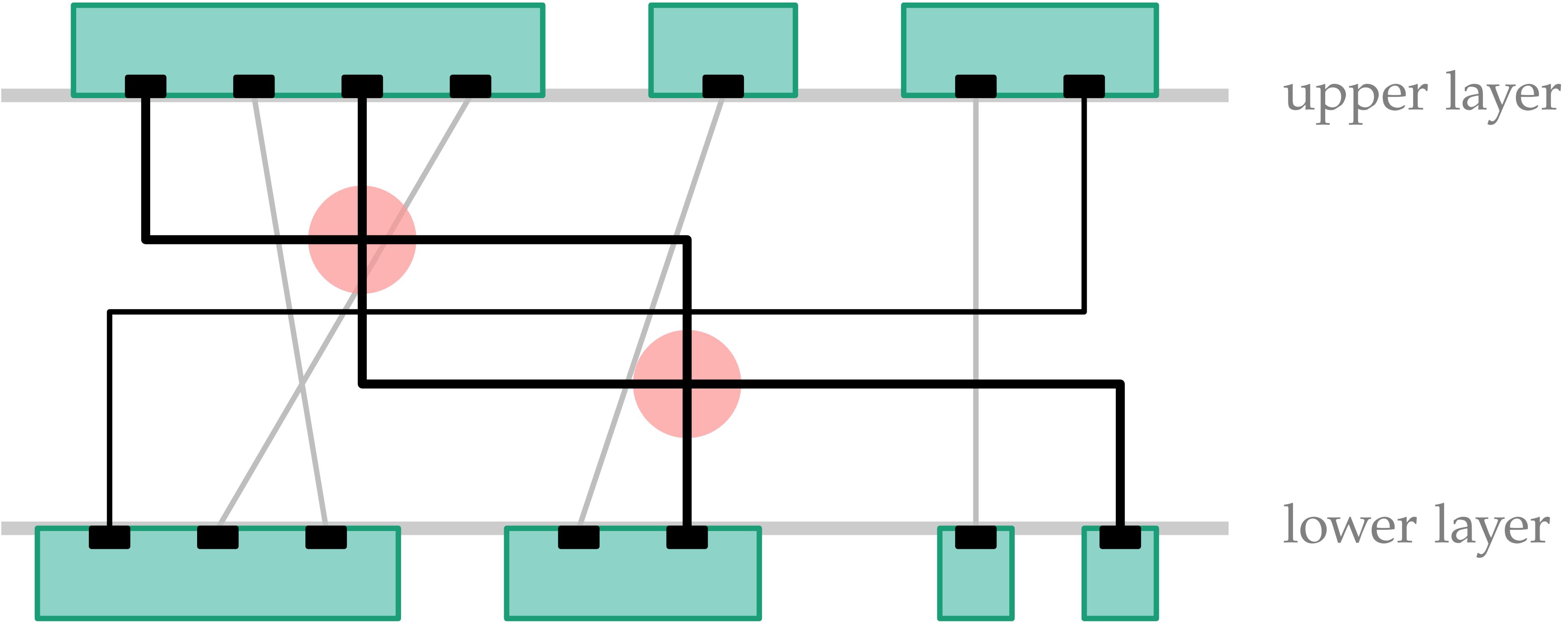
Motivation – Layered Orthogonal Edge Routing

- Draw each edge with at most two vertical and one horizontal line segments.
- Avoid overlaps and double crossings between the same pair of edges.



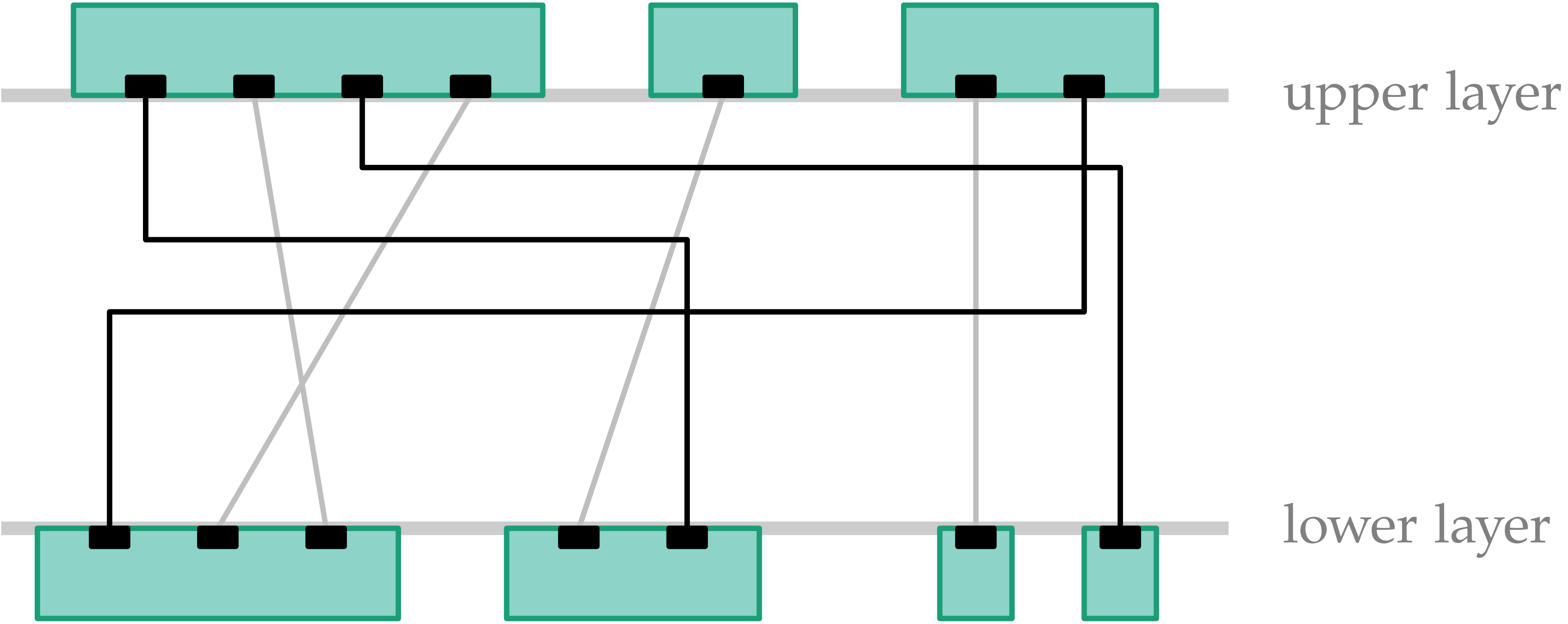
Motivation – Layered Orthogonal Edge Routing

- Draw each edge with at most two vertical and one horizontal line segments.
- Avoid overlaps and double crossings between the same pair of edges.



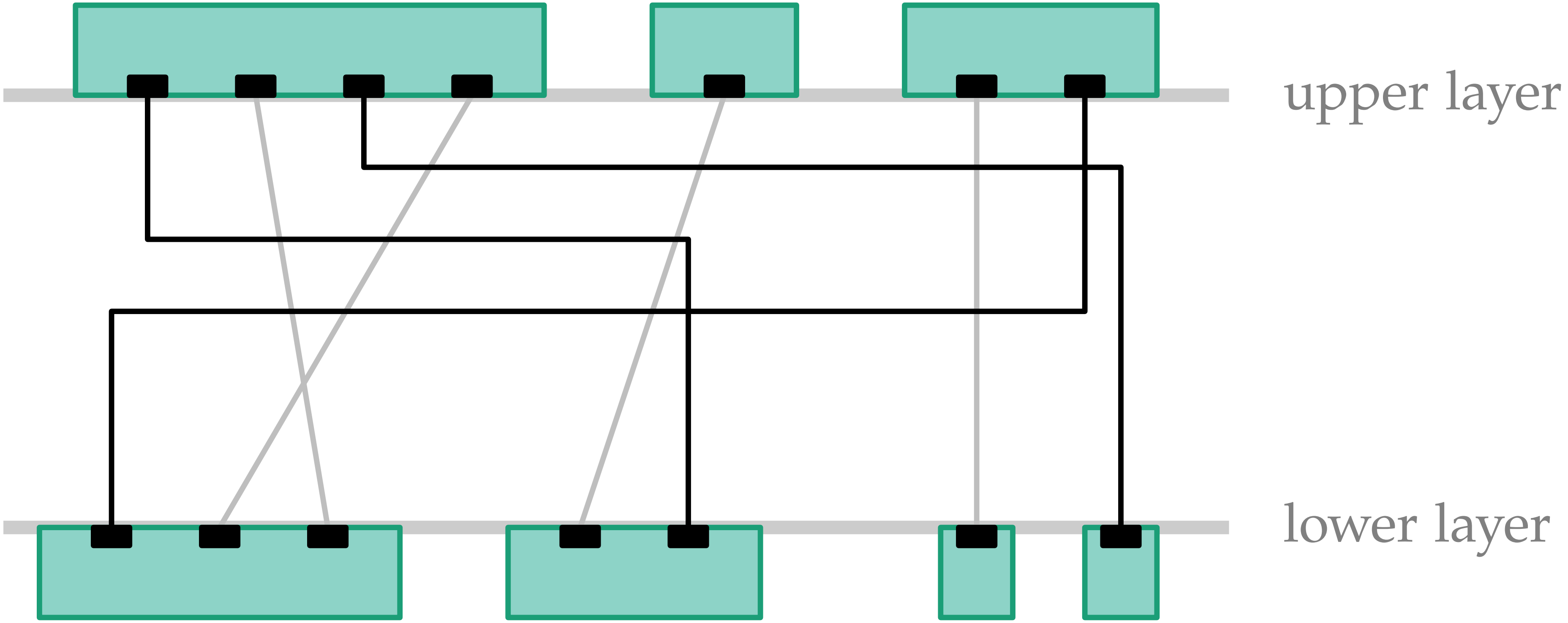
Motivation – Layered Orthogonal Edge Routing

- Draw each edge with at most two vertical and one horizontal line segments.
- Avoid overlaps and double crossings between the same pair of edges.



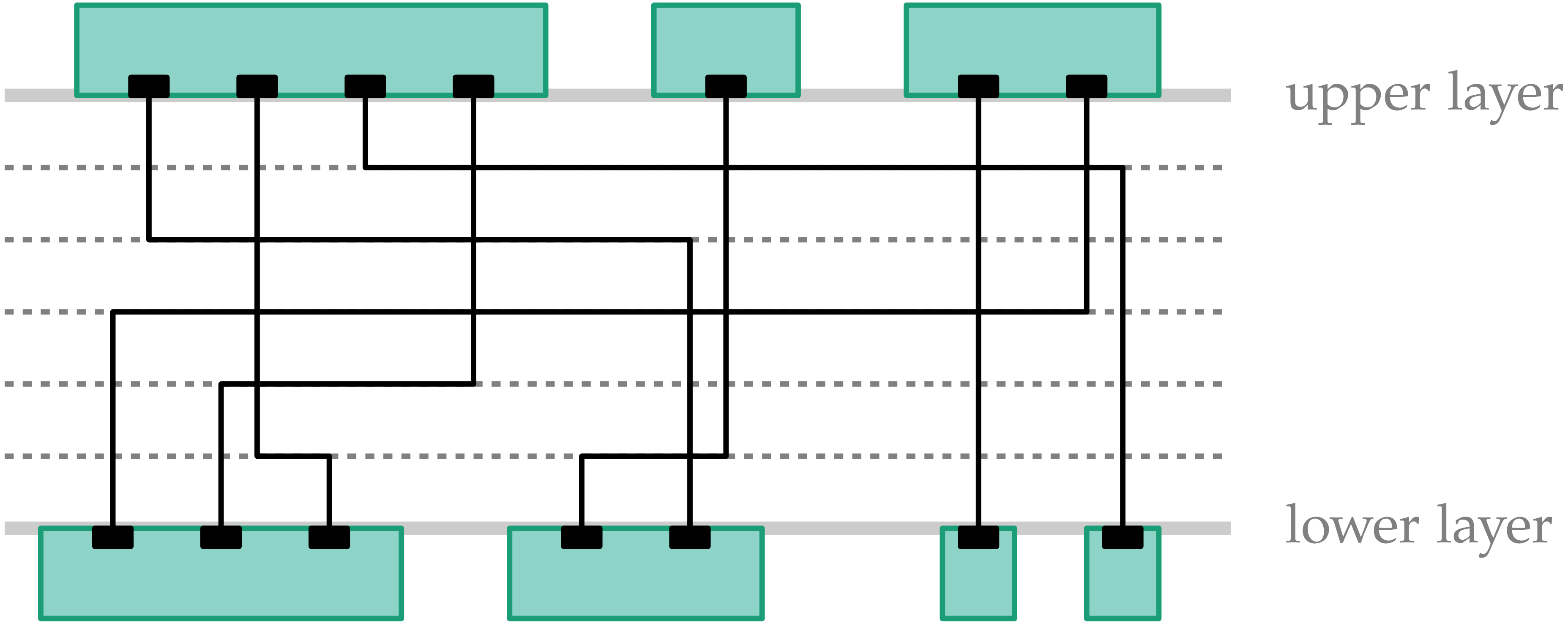
Motivation – Layered Orthogonal Edge Routing

- Draw each edge with at most two vertical and one horizontal line segments.
- Avoid overlaps and double crossings between the same pair of edges.
- Use as few horizontal intermediate layers (tracks) as possible.



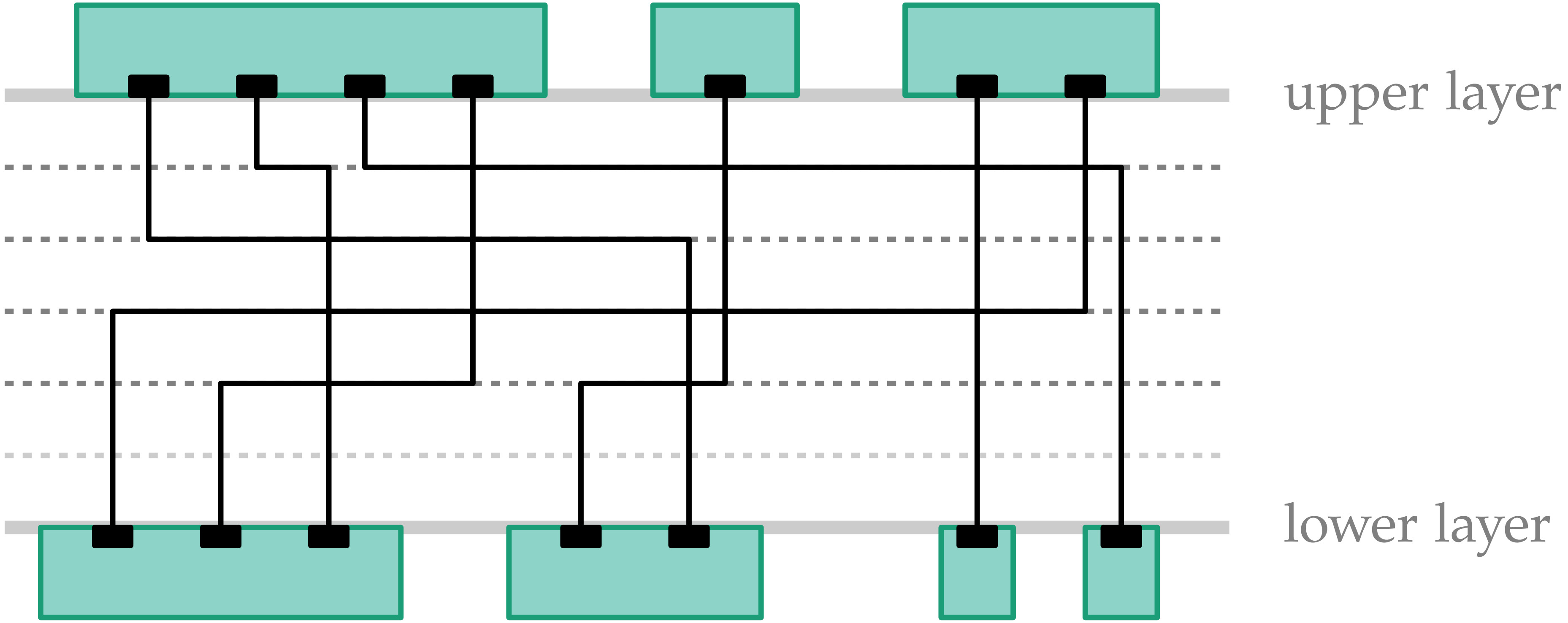
Motivation – Layered Orthogonal Edge Routing

- Draw each edge with at most two vertical and one horizontal line segments.
- Avoid overlaps and double crossings between the same pair of edges.
- Use as few horizontal intermediate layers (tracks) as possible.



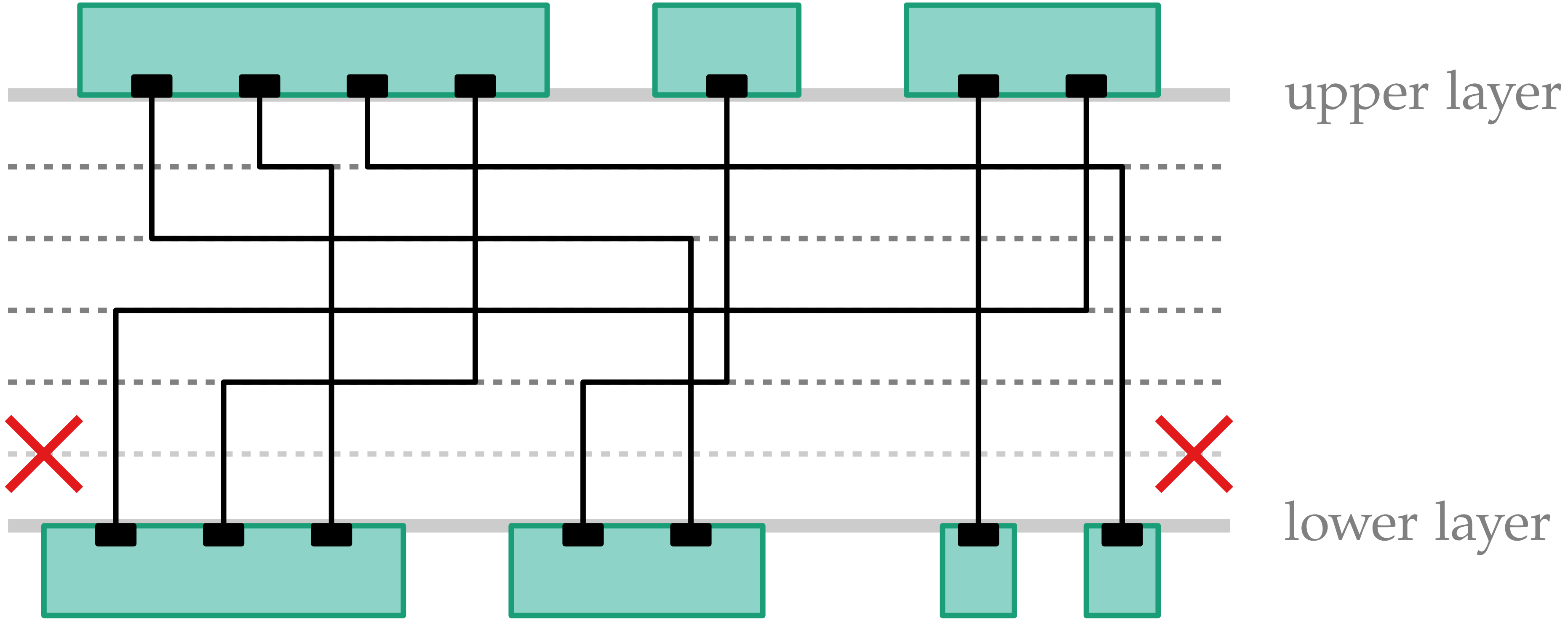
Motivation – Layered Orthogonal Edge Routing

- Draw each edge with at most two vertical and one horizontal line segments.
- Avoid overlaps and double crossings between the same pair of edges.
- Use as few horizontal intermediate layers (tracks) as possible.



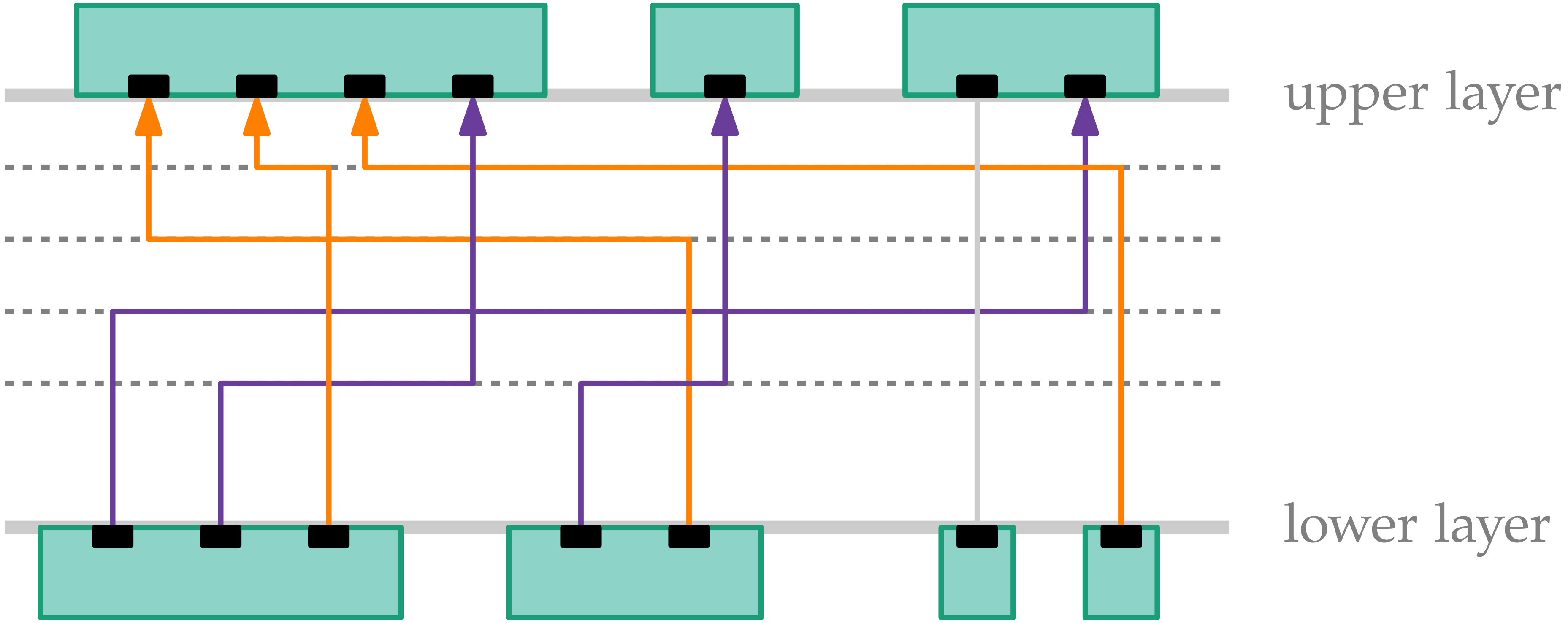
Motivation – Layered Orthogonal Edge Routing

- Draw each edge with at most two vertical and one horizontal line segments.
- Avoid overlaps and double crossings between the same pair of edges.
- Use as few horizontal intermediate layers (tracks) as possible.



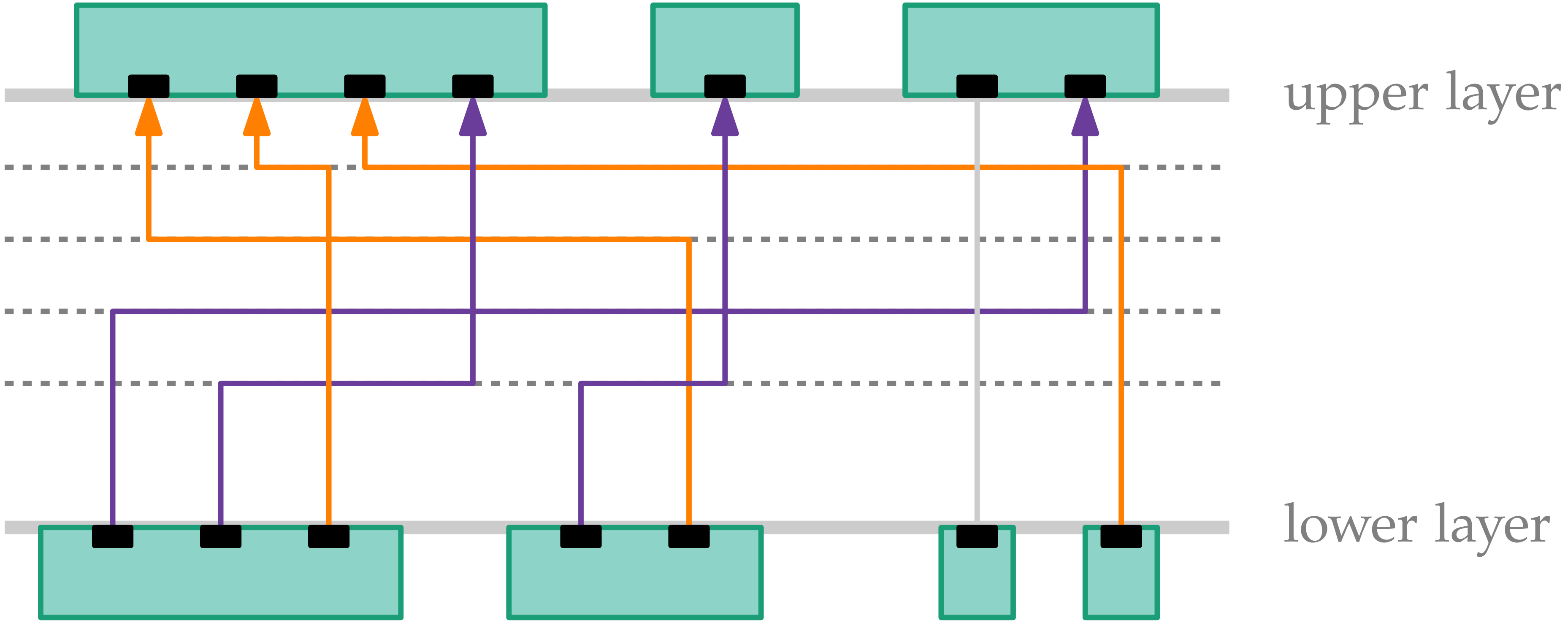
Motivation – Layered Orthogonal Edge Routing

- Distinguish between *left-going* and *right-going* edges.



Motivation – Layered Orthogonal Edge Routing

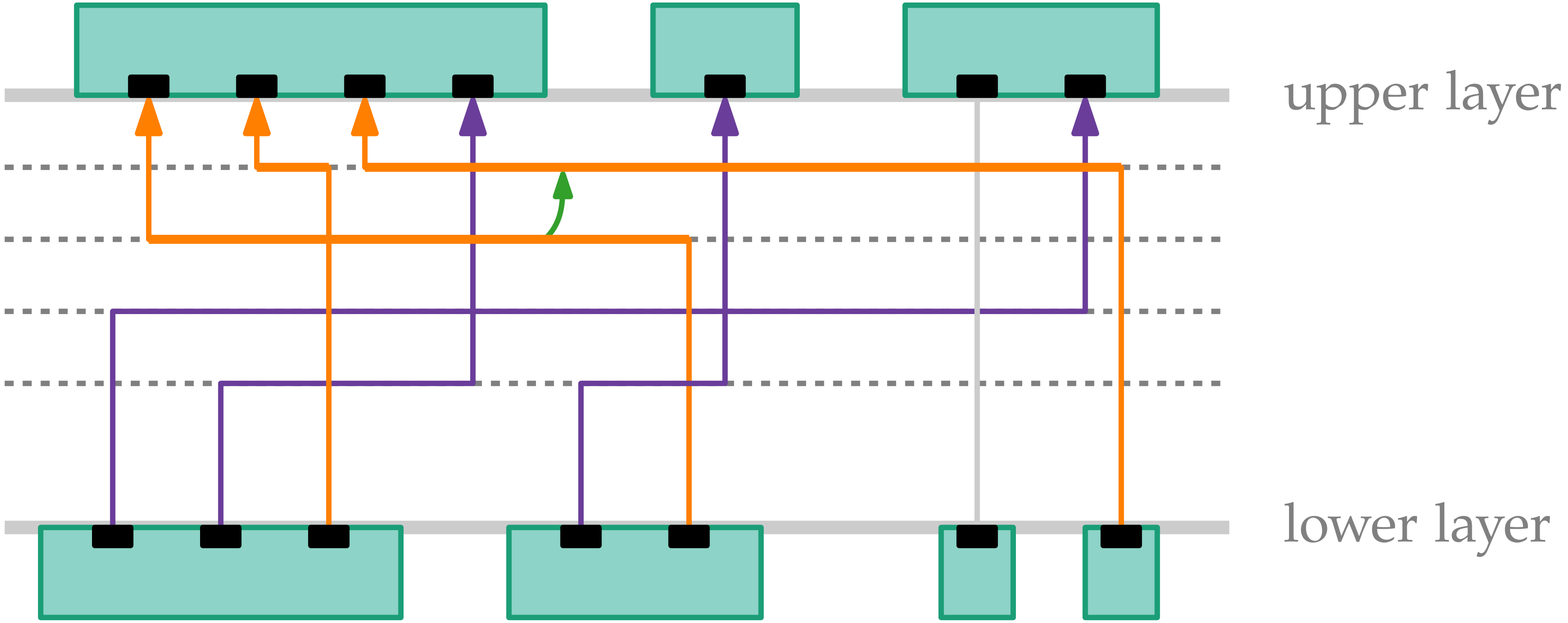
- Distinguish between *left-going* and *right-going* edges.
- Only edges going in the same direction and overlapping partially in x-dimension can cross twice.



Motivation – Layered Orthogonal Edge Routing

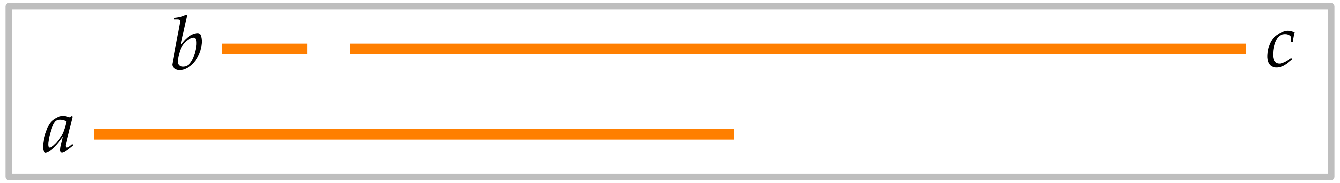
- Distinguish between *left-going* and *right-going* edges.
- Only edges going in the same direction and overlapping partially in x-dimension can cross twice.

  ⇒ They induce a vertical order for the horizontal middle segments.



Definition – Directional Interval Graphs

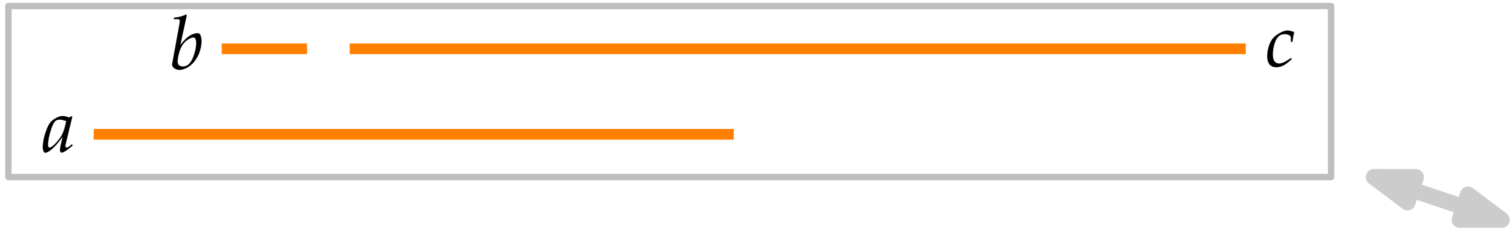
Interval representation: set of intervals



Definition – Directional Interval Graphs

Interval representation: set of intervals

Directional interval graph:

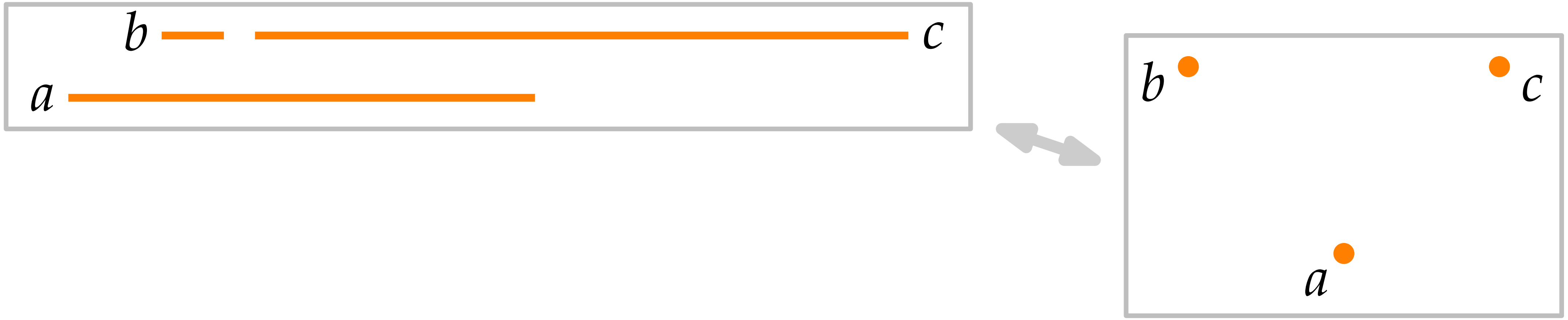


Definition – Directional Interval Graphs

Interval representation: set of intervals

Directional interval graph:

- vertex for each interval

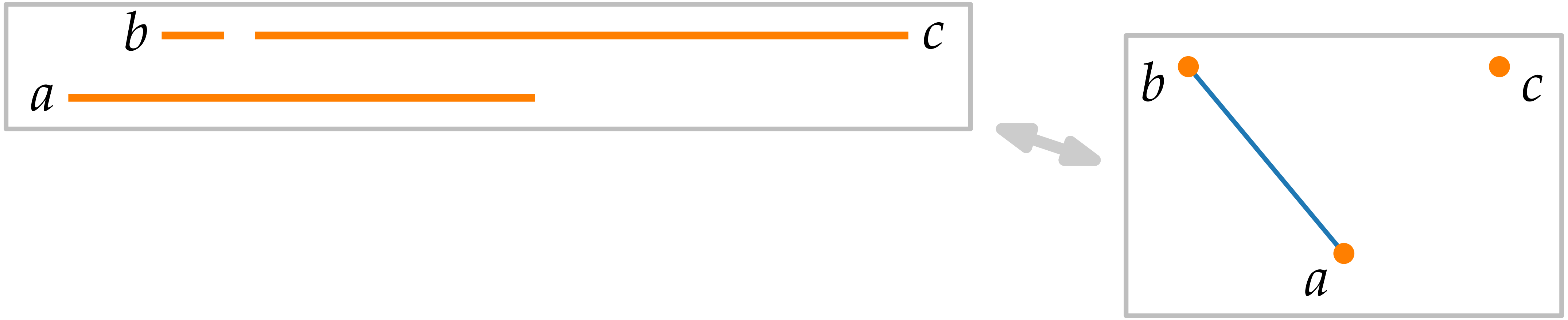


Definition – Directional Interval Graphs

Interval representation: set of intervals

Directional interval graph:

- vertex for each interval
- undirected edge if one interval contains another

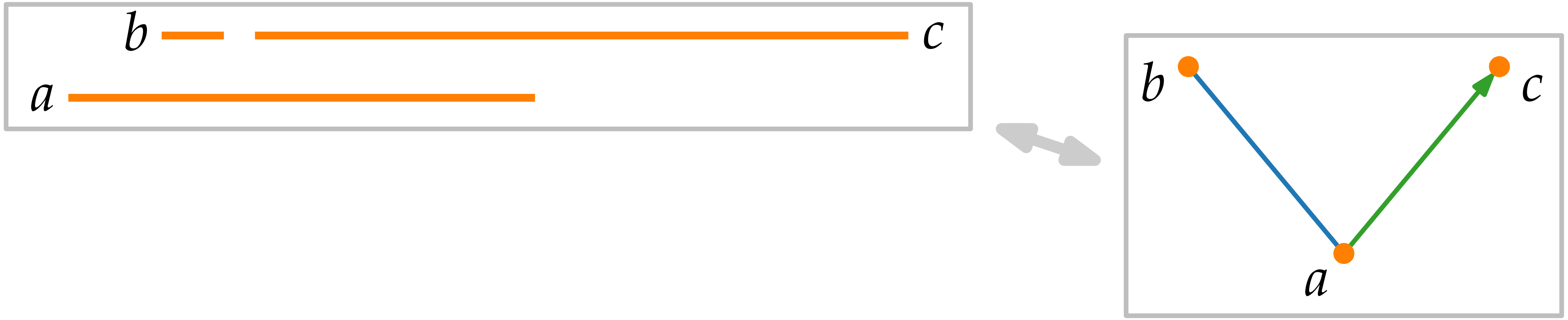


Definition – Directional Interval Graphs

Interval representation: set of intervals

Directional interval graph:

- vertex for each interval
- undirected edge if one interval contains another
- directed edge (towards the right interval) if the intervals overlap partially

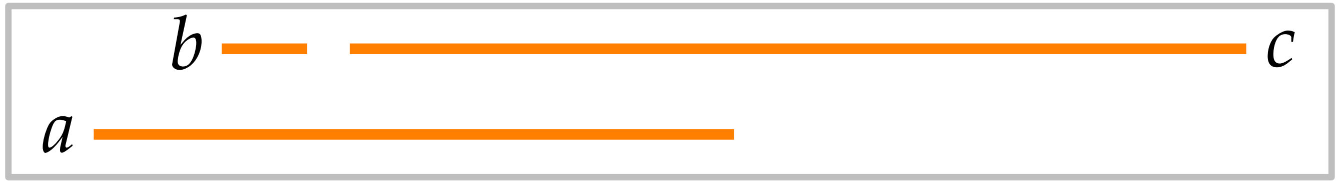


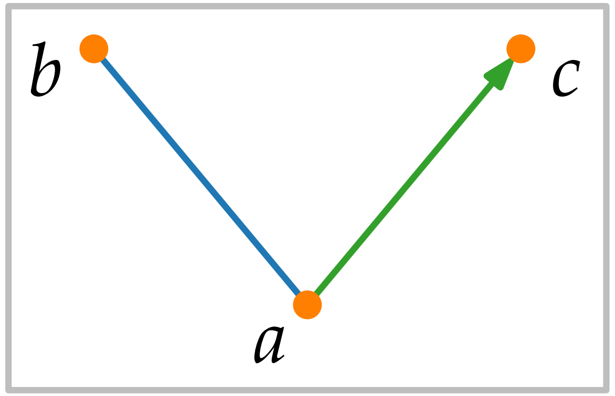
Definition – Directional Interval Graphs

Interval representation: set of intervals

Directional interval graph:

- vertex for each interval
- undirected edge if one interval contains another
- directed edge (towards the right interval) if the intervals overlap partially





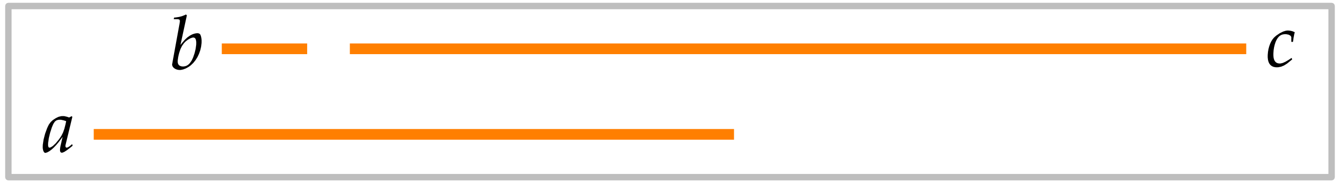
Mixed interval graph:

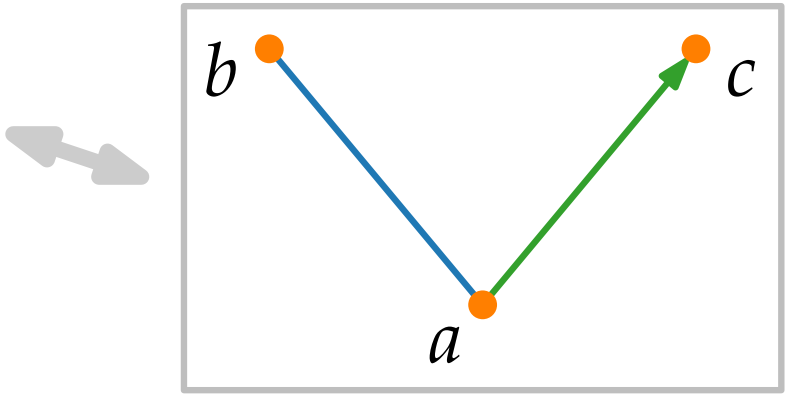
Definition – Directional Interval Graphs

Interval representation: set of intervals

Directional interval graph:

- vertex for each interval
- undirected edge if one interval contains another
- directed edge (towards the right interval) if the intervals overlap partially





Mixed interval graph:

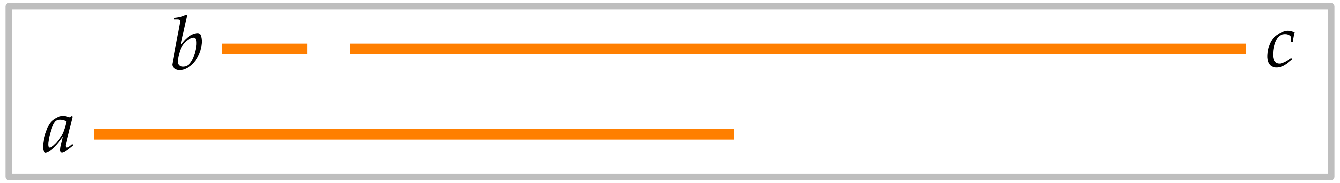
- vertex for each interval

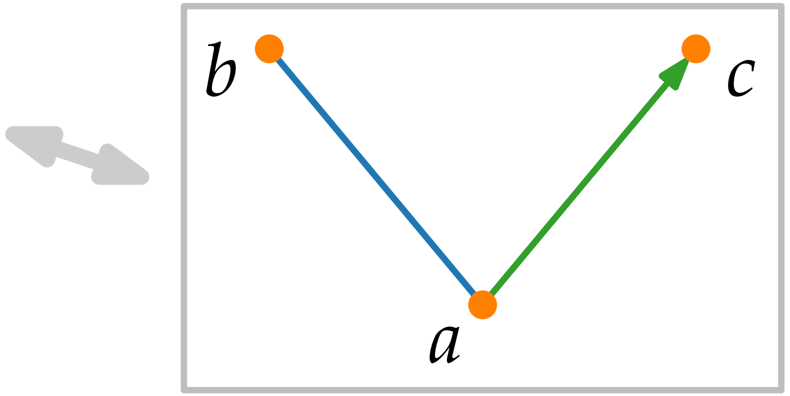
Definition – Directional Interval Graphs

Interval representation: set of intervals

Directional interval graph:

- vertex for each interval
- undirected edge if one interval contains another
- directed edge (towards the right interval) if the intervals overlap partially





Mixed interval graph:

- vertex for each interval
- for each two overlapping intervals: undirected or arbitrarily directed edge

Coloring Mixed Graphs

Given a graph G, find a coloring $c\colon V(G) \to \mathbb{N}$ s.t.

- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V(G)} c(v)$ is minimized.

[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

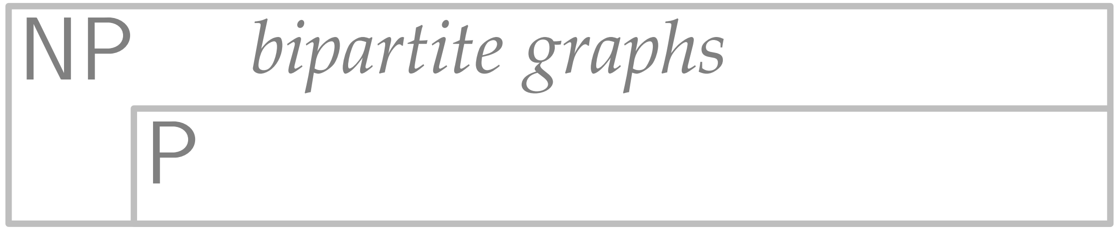
Coloring Mixed Graphs

Given a graph G, find a coloring $c\colon V(G) \to \mathbb{N}$ s.t.

- $\star$ undirected edge uv: $c(u) \neq c(v)$,
- $\star$ directed edge uv: $c(u) < c(v)$,
- $\star$ $\max_{v \in V(G)} c(v)$ is minimized.

[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

Coloring Mixed Graphs



NP

bipartite graphs

P

Given a graph G, find a coloring $c\colon V(G) \to \mathbb{N}$ s.t.

- $\star$ undirected edge uv: $c(u) \neq c(v)$,
- $\star$ directed edge uv: $c(u) < c(v)$,
- $\star$ $\max_{v \in V(G)} c(v)$ is minimized.

[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

Coloring Mixed Graphs

NP *bipartite graphs*

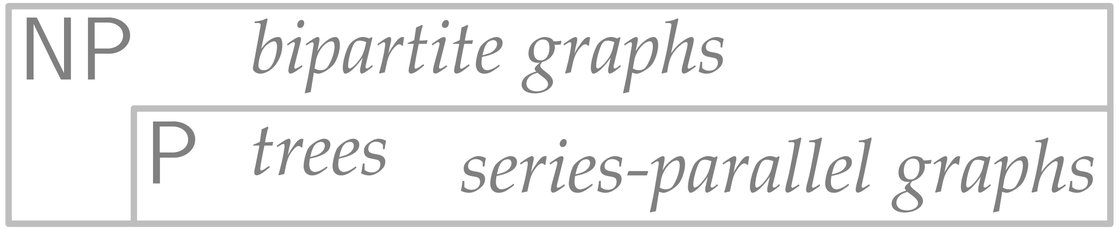
P *trees*

Given a graph G, find a coloring $c\colon V(G) \to \mathbb{N}$ s.t.

[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V(G)} c(v)$ is minimized.

Coloring Mixed Graphs



NP *bipartite graphs*

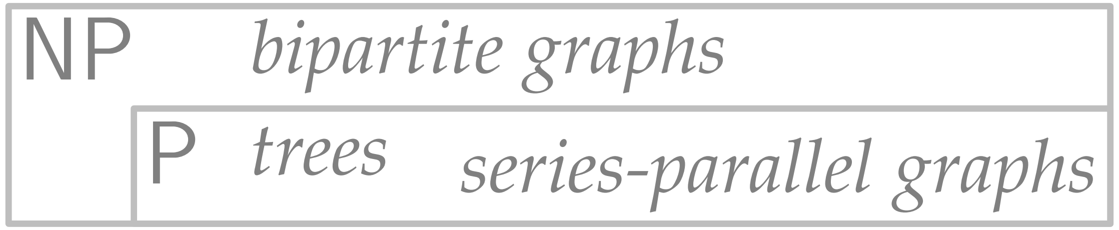
P *trees* *series-parallel graphs*

Given a graph G, find a coloring $c\colon V(G) \to \mathbb{N}$ s.t.

[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

- $\star$ undirected edge uv: $c(u) \neq c(v)$,
- $\star$ directed edge uv: $c(u) < c(v)$,
- $\star$ $\max_{v \in V(G)} c(v)$ is minimized.

Coloring Mixed Graphs



NP *bipartite graphs*

P *trees* *series-parallel graphs*

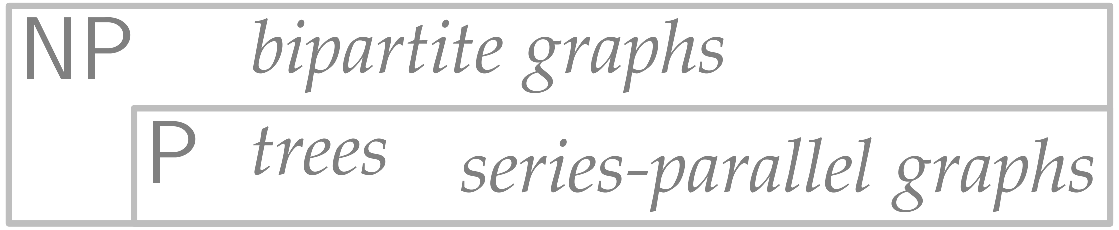
Given a graph G, find a coloring $c\colon V(G) \to \mathbb{N}$ s.t.

- $\star$ undirected edge uv: $c(u) \neq c(v)$,
- $\star$ directed edge uv: $c(u) < c(v)$,
- $\star$ $\max_{v \in V(G)} c(v)$ is minimized.

[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

Interval graphs (no directed edges):

Coloring Mixed Graphs

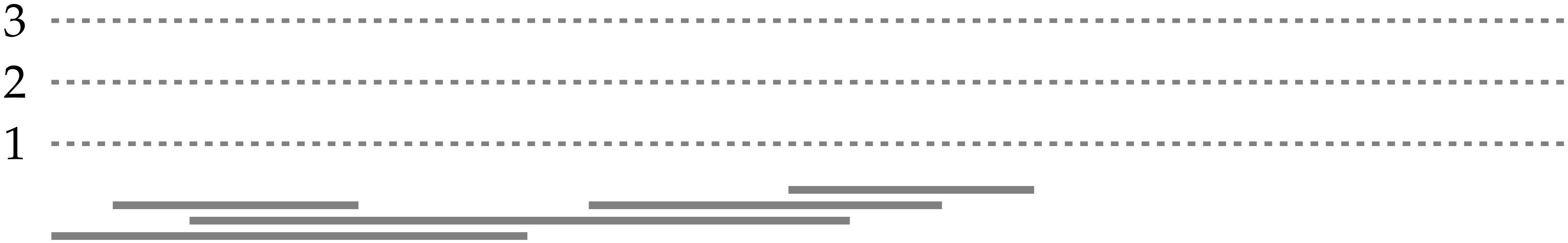


Given a graph G, find a coloring $c\colon V(G) \to \mathbb{N}$ s.t.
[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V(G)} c(v)$ is minimized.

Interval graphs (no directed edges):

- sort by left endpoints, color greedily (in linear time given sorted intervals)



Coloring Mixed Graphs

NP *bipartite graphs*

P *trees* *series-parallel graphs*

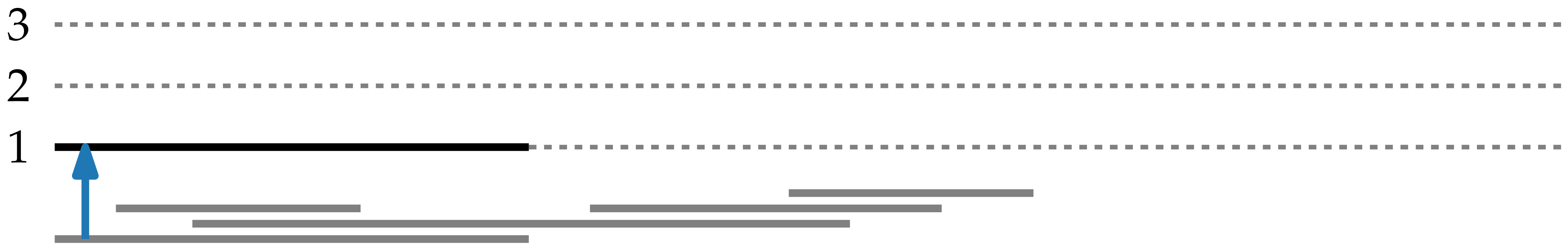
Given a graph G, find a coloring $c\colon V(G) \to \mathbb{N}$ s.t.

[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V(G)} c(v)$ is minimized.

Interval graphs (no directed edges):

- sort by left endpoints, color greedily (in linear time given sorted intervals)



Coloring Mixed Graphs

NP	*bipartite graphs*	
P	*trees*	*series-parallel graphs*

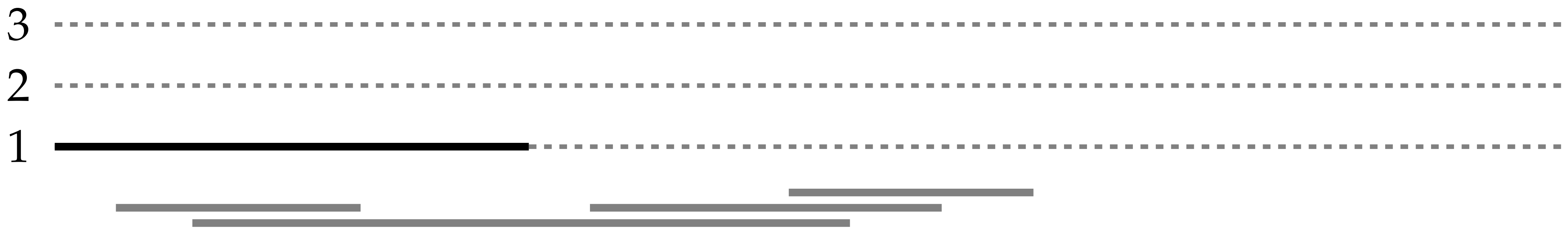
Given a graph G, find a coloring $c\colon V(G) \to \mathbb{N}$ s.t.

[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

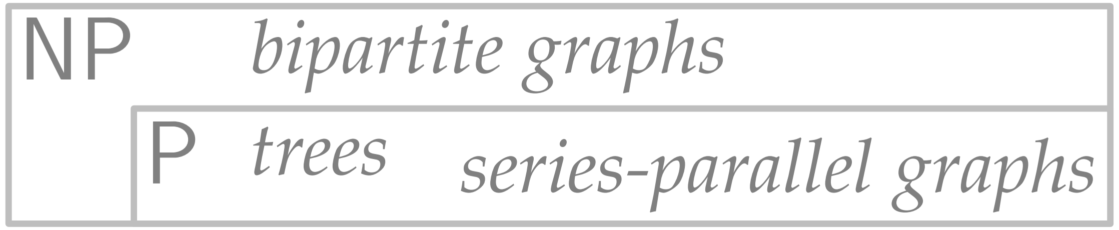
- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V(G)} c(v)$ is minimized.

Interval graphs (no directed edges):

- sort by left endpoints, color greedily (in linear time given sorted intervals)



Coloring Mixed Graphs



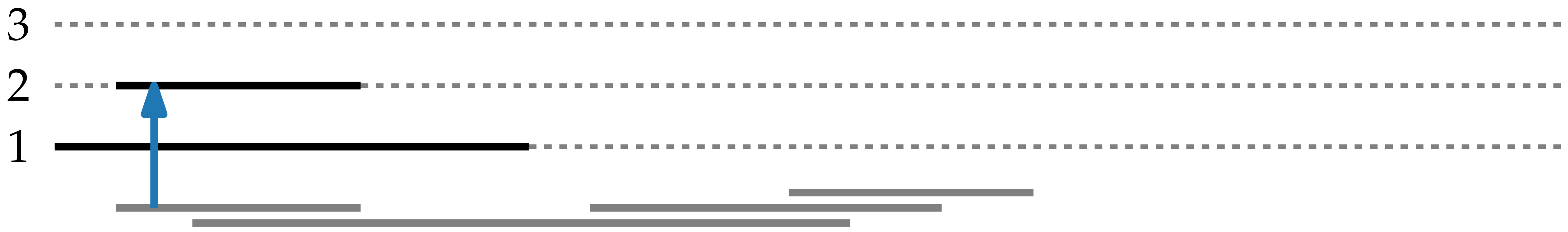
Given a graph G, find a coloring $c\colon V(G) \to \mathbb{N}$ s.t.

[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

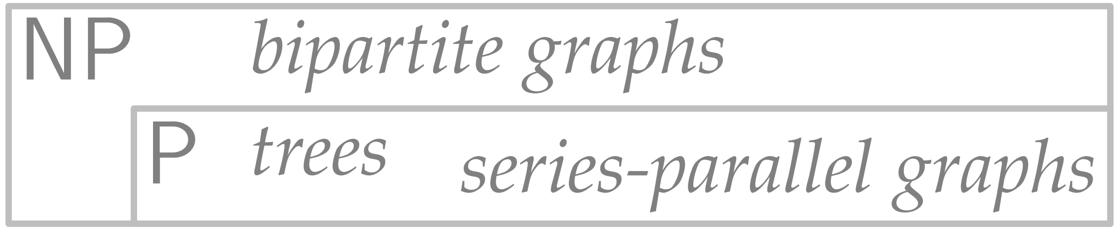
- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V(G)} c(v)$ is minimized.

Interval graphs (no directed edges):

- sort by left endpoints, color greedily (in linear time given sorted intervals)



Coloring Mixed Graphs


NP
bipartite graphs
P
trees
series-parallel graphs

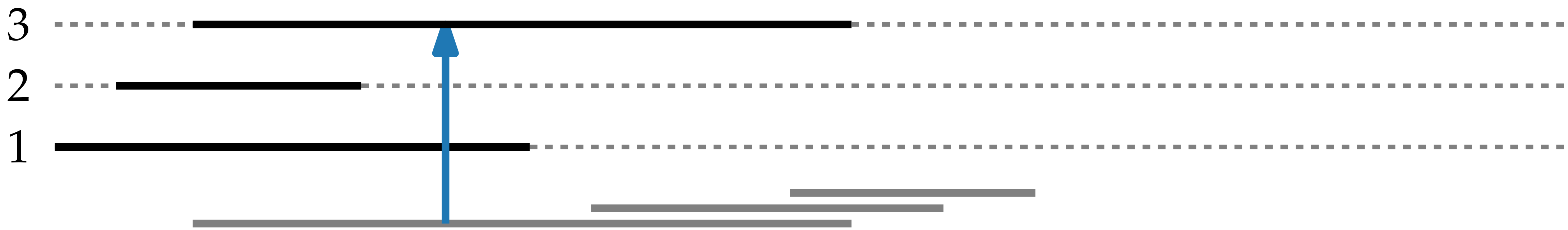
Given a graph G, find a coloring $c\colon V(G) \to \mathbb{N}$ s.t.

[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

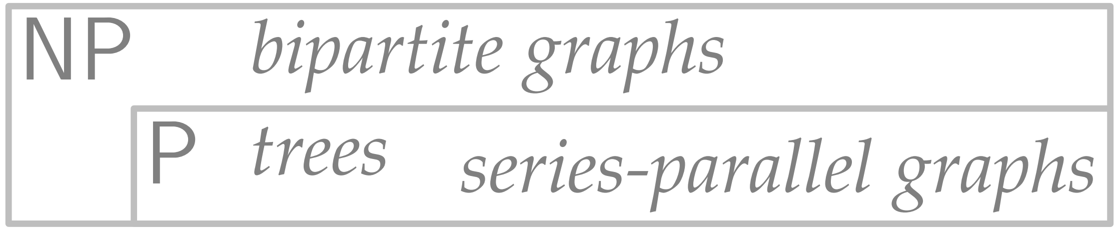
- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V(G)} c(v)$ is minimized.

Interval graphs (no directed edges):

- sort by left endpoints, color greedily (in linear time given sorted intervals)



Coloring Mixed Graphs



NP *bipartite graphs*

P *trees* *series-parallel graphs*

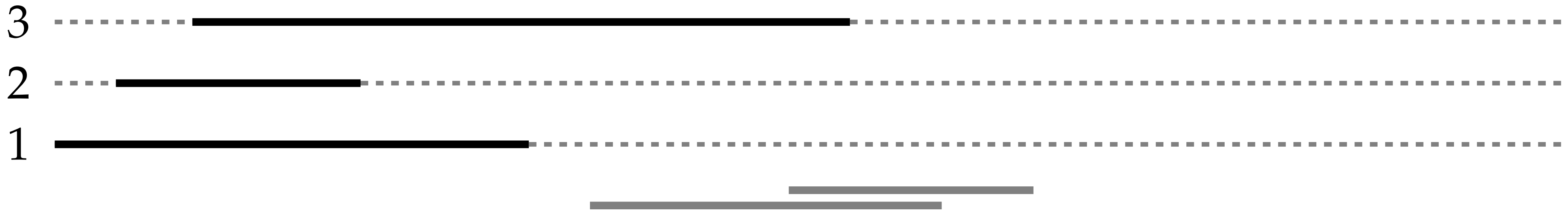
Given a graph G, find a coloring $c\colon V(G) \to \mathbb{N}$ s.t.

[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

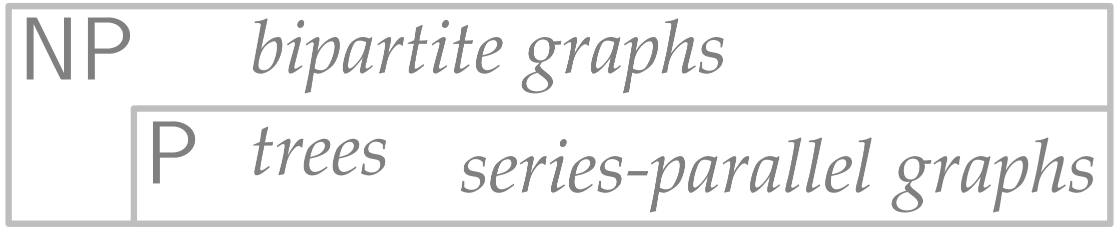
- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V(G)} c(v)$ is minimized.

Interval graphs (no directed edges):

- sort by left endpoints, color greedily (in linear time given sorted intervals)



Coloring Mixed Graphs

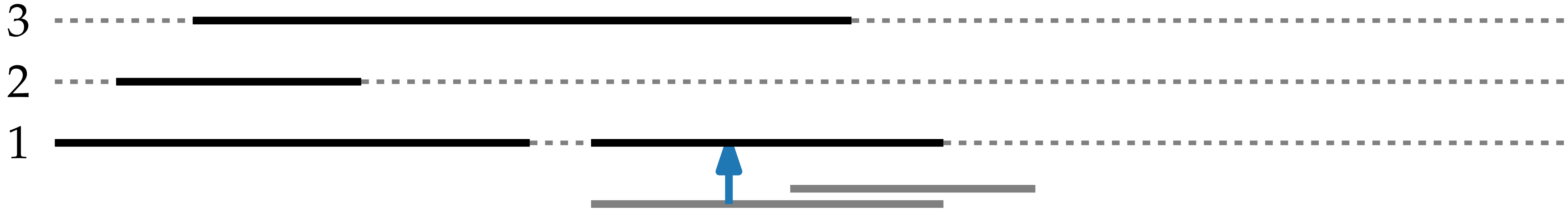


Given a graph G, find a coloring $c\colon V(G) \to \mathbb{N}$ s.t.
- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V(G)} c(v)$ is minimized.

[Sotskov, Tanaev ’76; Hansen, Kuplinsky, de Werra ’97]

Interval graphs (no directed edges):

- sort by left endpoints, color greedily (in linear time given sorted intervals)



Coloring Mixed Graphs

NP *bipartite graphs*

P *trees* *series-parallel graphs*

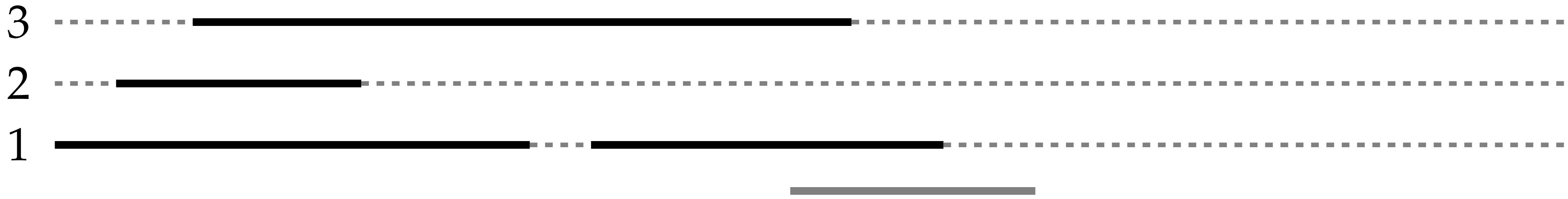
Given a graph G, find a coloring $c\colon V(G) \to \mathbb{N}$ s.t.

[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

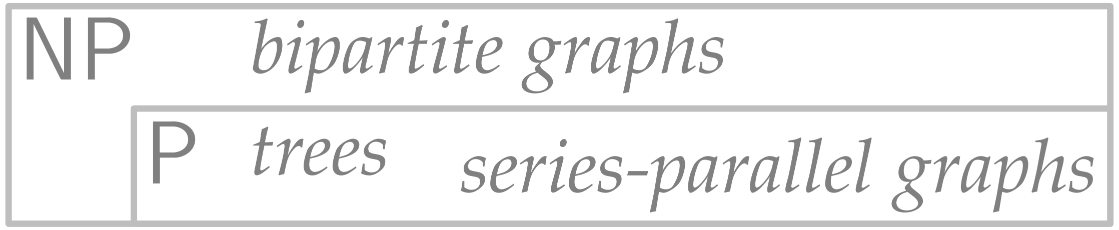
- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V(G)} c(v)$ is minimized.

Interval graphs (no directed edges):

- sort by left endpoints, color greedily (in linear time given sorted intervals)



Coloring Mixed Graphs

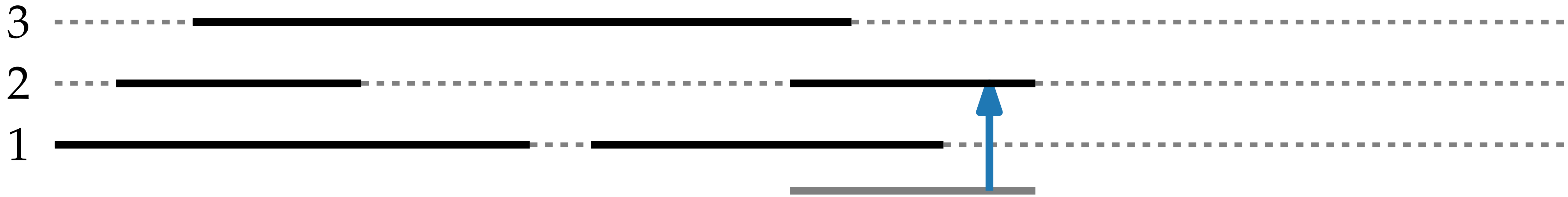


Given a graph G, find a coloring $c\colon V(G) \to \mathbb{N}$ s.t.
[Sotskov, Tanaev ’76; Hansen, Kuplinsky, de Werra ’97]

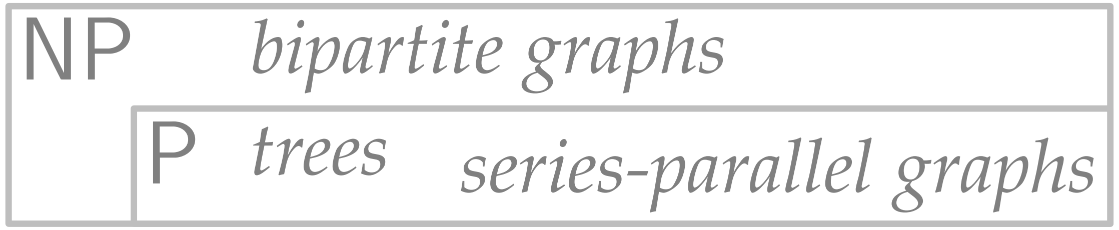
- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V(G)} c(v)$ is minimized.

Interval graphs (no directed edges):

- sort by left endpoints, color greedily (in linear time given sorted intervals)



Coloring Mixed Graphs



NP *bipartite graphs*

P *trees* *series-parallel graphs*

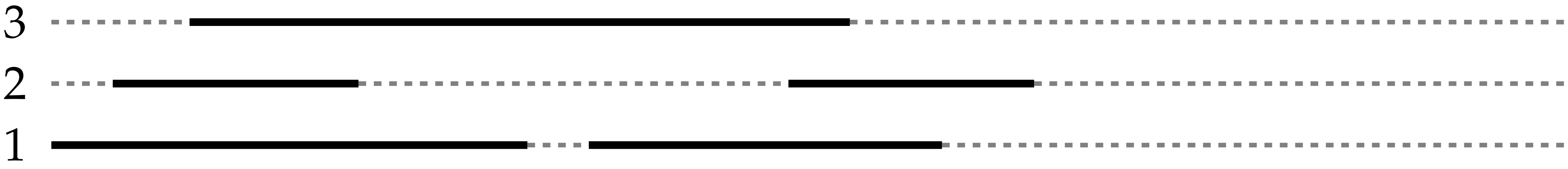
Given a graph G, find a coloring $c\colon V(G) \to \mathbb{N}$ s.t.

[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

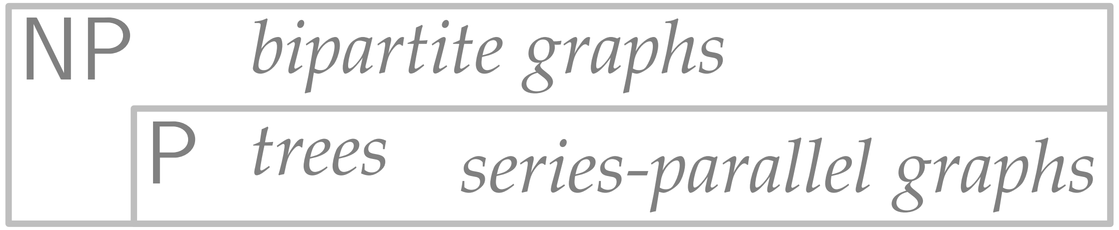
- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V(G)} c(v)$ is minimized.

Interval graphs (no directed edges):

- sort by left endpoints, color greedily (in linear time given sorted intervals)



Coloring Mixed Graphs


NP *bipartite graphs*

P *trees* *series-parallel graphs*

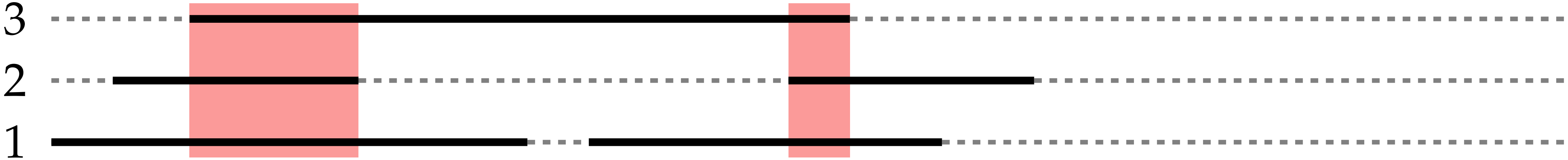
Given a graph G, find a coloring $c\colon V(G) \to \mathbb{N}$ s.t.

- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V(G)} c(v)$ is minimized.

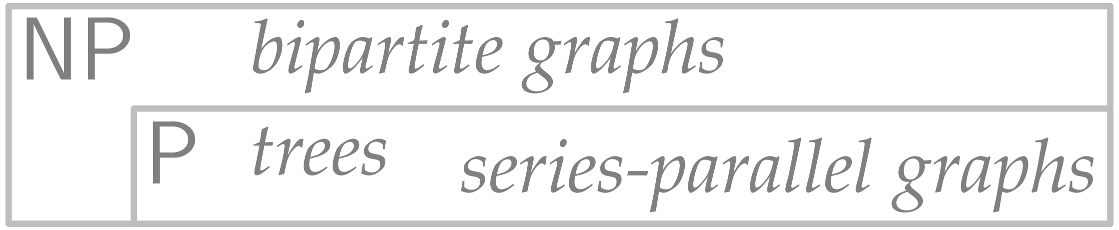
[Sotskov, Tanaev ’76; Hansen, Kuplinsky, de Werra ’97]

Interval graphs (no directed edges):

- sort by left endpoints, color greedily (in linear time given sorted intervals)



Coloring Mixed Graphs


NP *bipartite graphs*

P *trees* *series-parallel graphs*

Given a graph G, find a coloring $c\colon V(G) \to \mathbb{N}$ s.t.
[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

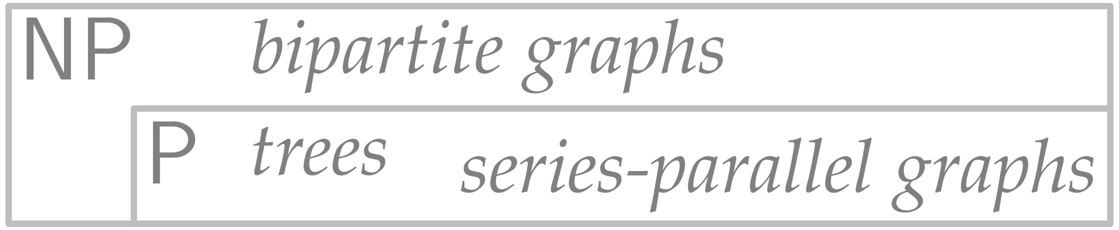
- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V(G)} c(v)$ is minimized.

Interval graphs (no directed edges):

- sort by left endpoints, color greedily (in linear time given sorted intervals)

Directed acyclic graphs (only directed edges):

Coloring Mixed Graphs



Given a graph G, find a coloring $c\colon V(G) \to \mathbb{N}$ s.t.
- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V(G)} c(v)$ is minimized.

[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

Interval graphs (no directed edges):

- sort by left endpoints, color greedily (in linear time given sorted intervals)

Directed acyclic graphs (only directed edges):

- sort topologically, color greedily (in linear time)

Coloring Mixed Graphs

NP *bipartite graphs*

P *trees* *series-parallel graphs*

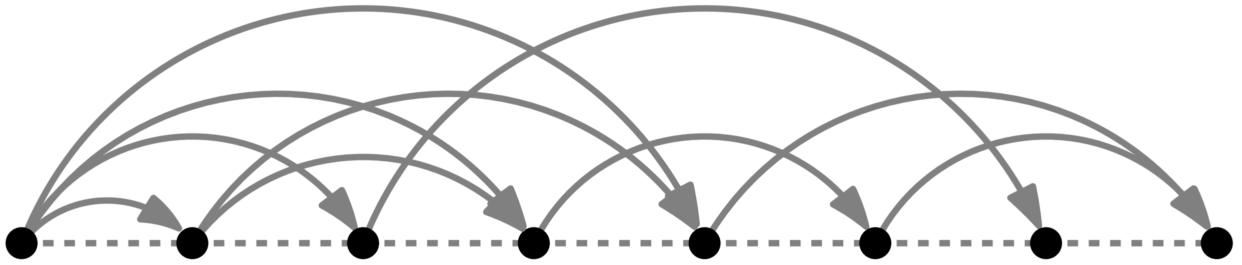
Given a graph G, find a coloring $c\colon V(G) \to \mathbb{N}$ s.t.

[Sotskov, Tanaev ’76; Hansen, Kuplinsky, de Werra ’97]

- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V(G)} c(v)$ is minimized.

Interval graphs (no directed edges):

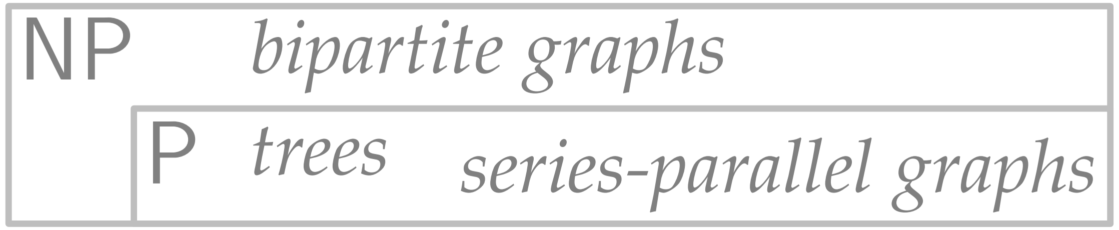
- sort by left endpoints, color greedily (in linear time given sorted intervals)



Directed acyclic graphs (only directed edges):

- sort topologically, color greedily (in linear time)

Coloring Mixed Graphs



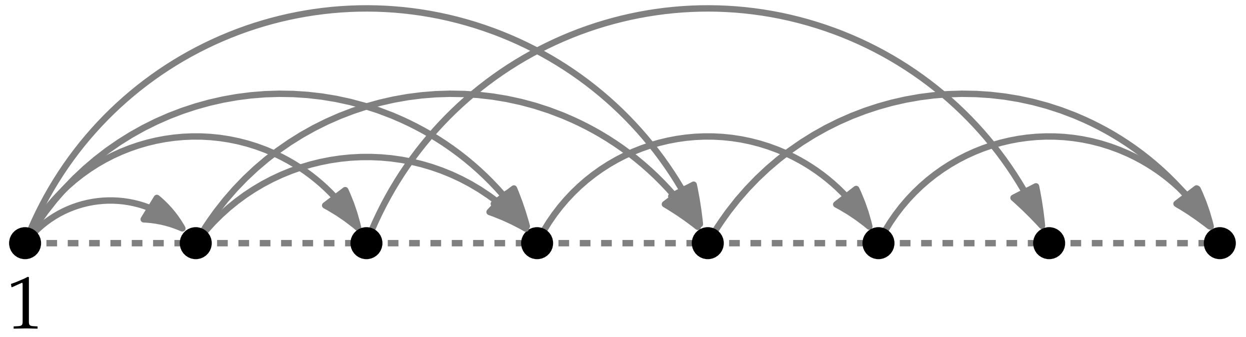
Given a graph G, find a coloring $c\colon V(G) \to \mathbb{N}$ s.t.

[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V(G)} c(v)$ is minimized.

Interval graphs (no directed edges):

- sort by left endpoints, color greedily (in linear time given sorted intervals)



Directed acyclic graphs (only directed edges):

- sort topologically, color greedily (in linear time)

Coloring Mixed Graphs

NP *bipartite graphs*

P *trees* *series-parallel graphs*

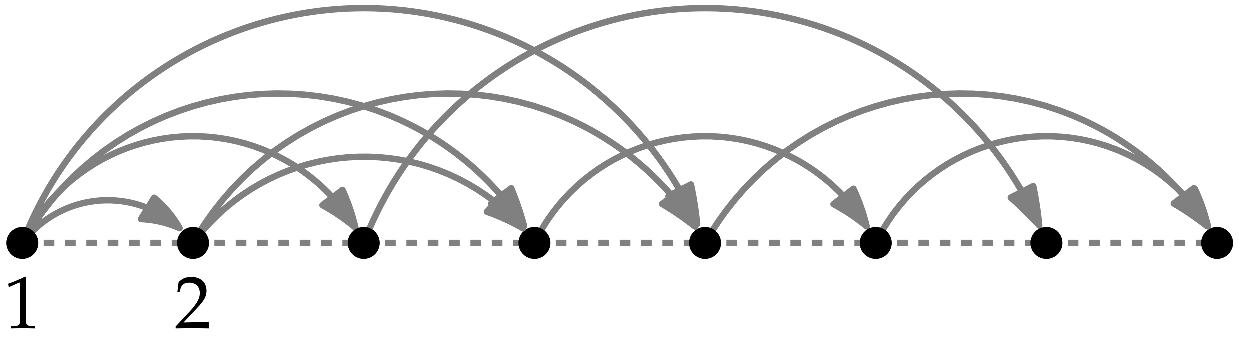
Given a graph G, find a coloring $c\colon V(G) \to \mathbb{N}$ s.t.

[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V(G)} c(v)$ is minimized.

Interval graphs (no directed edges):

- sort by left endpoints, color greedily (in linear time given sorted intervals)



Directed acyclic graphs (only directed edges):

- sort topologically, color greedily (in linear time)

Coloring Mixed Graphs

NP *bipartite graphs*

P *trees* *series-parallel graphs*

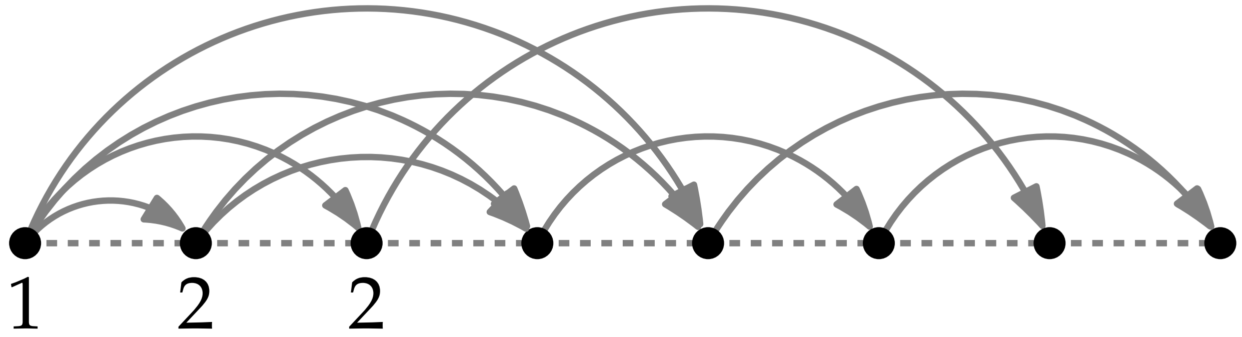
Given a graph G, find a coloring $c\colon V(G) \to \mathbb{N}$ s.t.

- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V(G)} c(v)$ is minimized.

[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

Interval graphs (no directed edges):

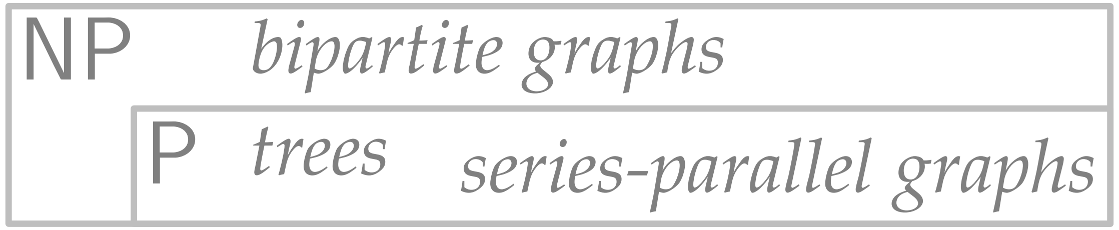
- sort by left endpoints, color greedily (in linear time given sorted intervals)



Directed acyclic graphs (only directed edges):

- sort topologically, color greedily (in linear time)

Coloring Mixed Graphs



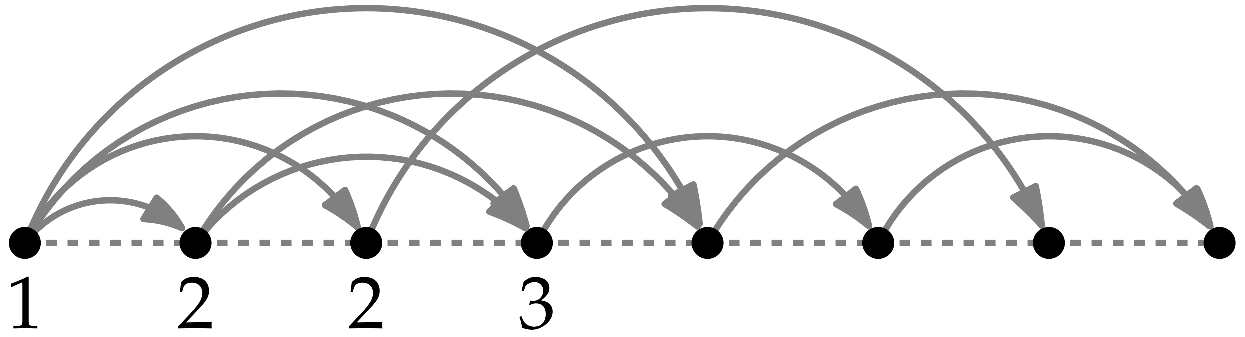
Given a graph G, find a coloring $c\colon V(G) \to \mathbb{N}$ s.t.

[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

- $\star$ undirected edge uv: $c(u) \neq c(v)$,
- $\star$ directed edge uv: $c(u) < c(v)$,
- $\star$ $\max_{v \in V(G)} c(v)$ is minimized.

Interval graphs (no directed edges):

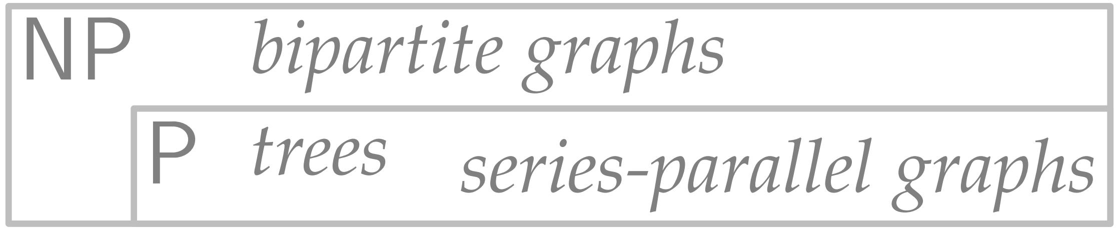
- sort by left endpoints, color greedily (in linear time given sorted intervals)



Directed acyclic graphs (only directed edges):

- sort topologically, color greedily (in linear time)

Coloring Mixed Graphs


NP *bipartite graphs*

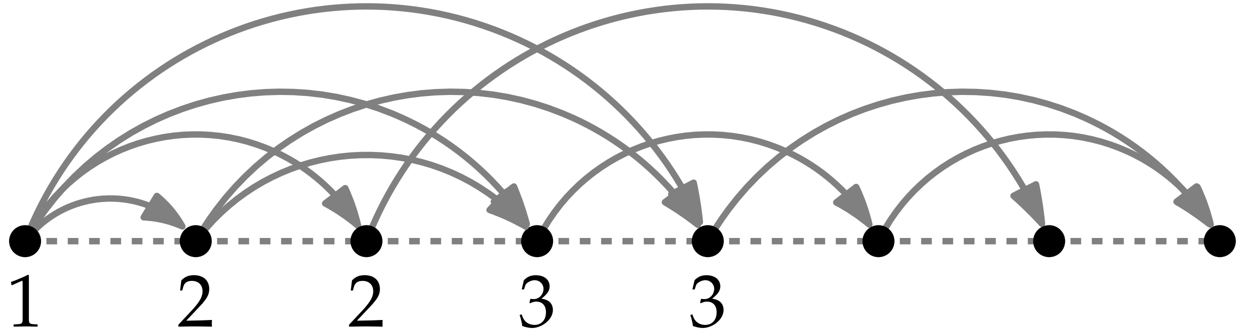
P *trees* *series-parallel graphs*

Given a graph G, find a coloring $c\colon V(G) \to \mathbb{N}$ s.t.
- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V(G)} c(v)$ is minimized.

[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

Interval graphs (no directed edges):

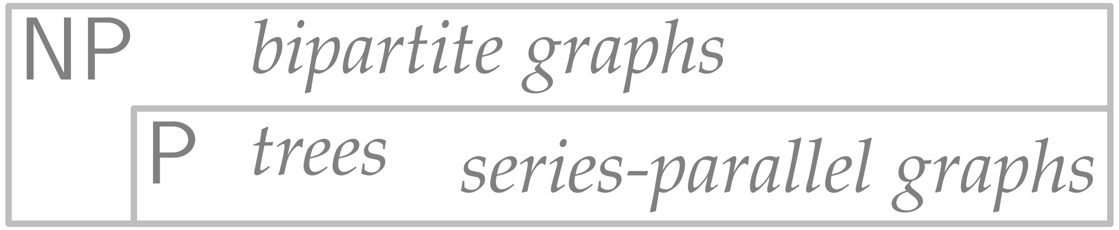
- sort by left endpoints, color greedily (in linear time given sorted intervals)



Directed acyclic graphs (only directed edges):

- sort topologically, color greedily (in linear time)

Coloring Mixed Graphs


NP *bipartite graphs*

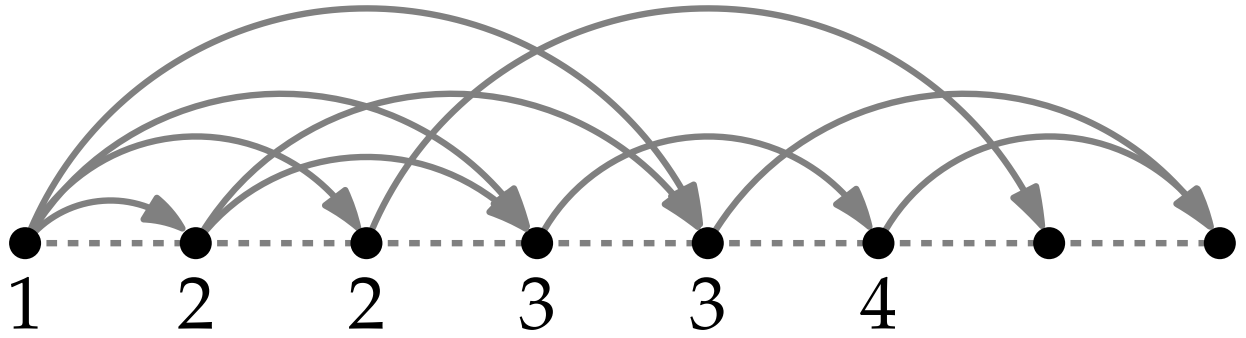
P *trees* *series-parallel graphs*

Given a graph G, find a coloring $c\colon V(G) \to \mathbb{N}$ s.t.
[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V(G)} c(v)$ is minimized.

Interval graphs (no directed edges):

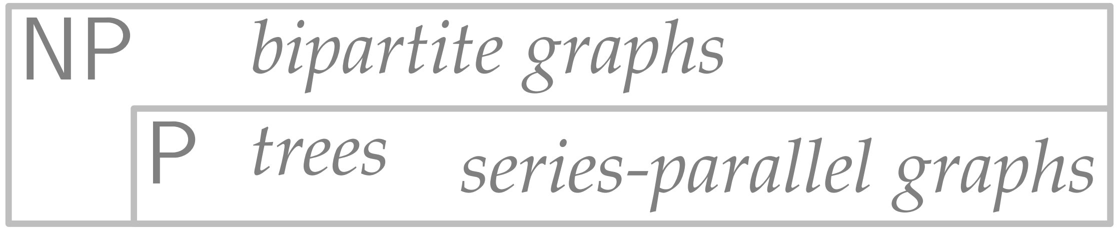
- sort by left endpoints, color greedily (in linear time given sorted intervals)



Directed acyclic graphs (only directed edges):

- sort topologically, color greedily (in linear time)

Coloring Mixed Graphs



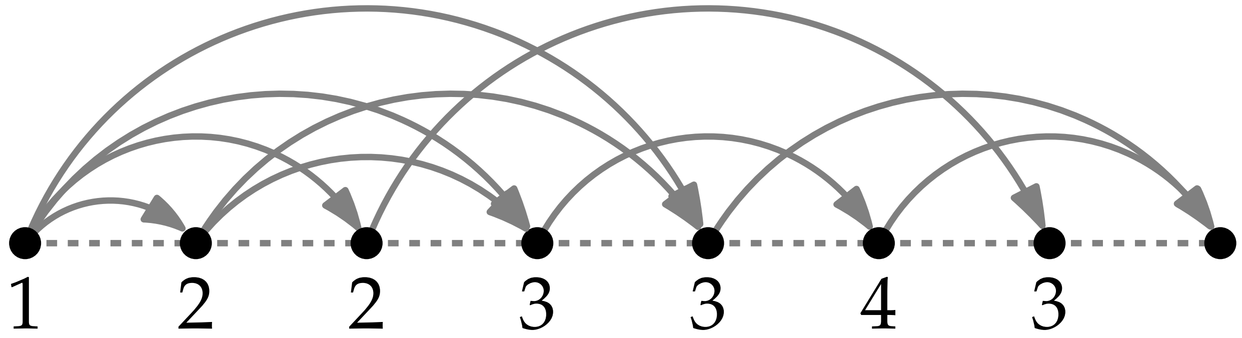
Given a graph G, find a coloring $c\colon V(G) \to \mathbb{N}$ s.t.

[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V(G)} c(v)$ is minimized.

Interval graphs (no directed edges):

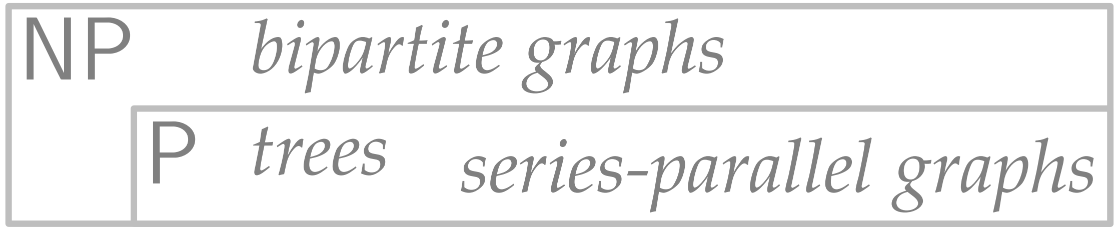
- sort by left endpoints, color greedily (in linear time given sorted intervals)



Directed acyclic graphs (only directed edges):

- sort topologically, color greedily (in linear time)

Coloring Mixed Graphs


NP *bipartite graphs*

P *trees* *series-parallel graphs*

Given a graph G, find a coloring $c\colon V(G) \to \mathbb{N}$ s.t.
[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

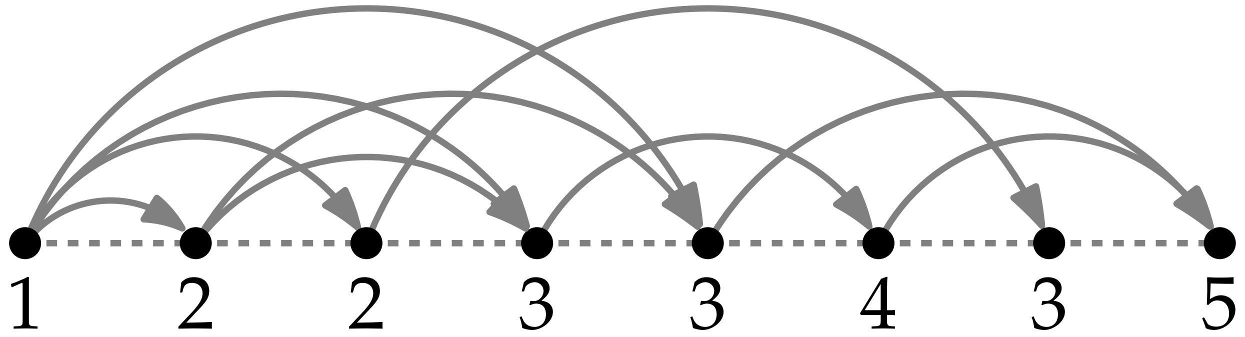
⋆ undirected edge uv: $c(u) \neq c(v)$,

⋆ directed edge uv: $c(u) < c(v)$,

⋆ $\max_{v \in V(G)} c(v)$ is minimized.

Interval graphs (no directed edges):

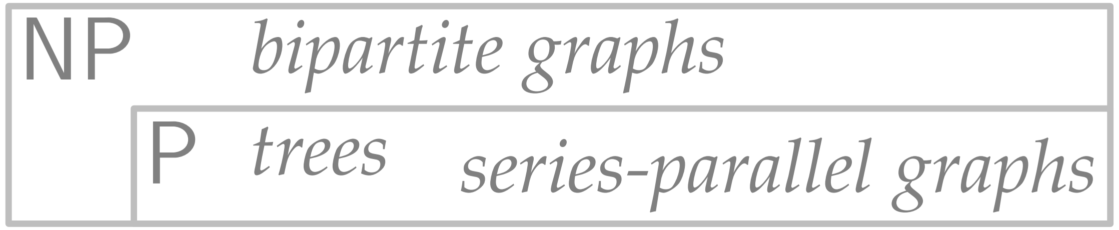
- sort by left endpoints, color greedily (in linear time given sorted intervals)



Directed acyclic graphs (only directed edges):

- sort topologically, color greedily (in linear time)

Coloring Mixed Graphs


NP *bipartite graphs*

P *trees* *series-parallel graphs*

Given a graph G, find a coloring $c\colon V(G) \to \mathbb{N}$ s.t.
[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V(G)} c(v)$ is minimized.

Interval graphs (no directed edges):

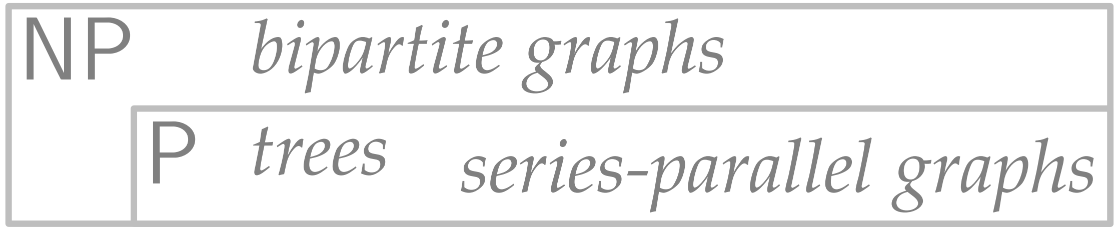
- sort by left endpoints, color greedily (in linear time given sorted intervals)

Directional interval graphs:

Directed acyclic graphs (only directed edges):

- sort topologically, color greedily (in linear time)

Coloring Mixed Graphs


NP *bipartite graphs*

P *trees* *series-parallel graphs*

Given a graph G, find a coloring $c\colon V(G) \to \mathbb{N}$ s.t.

- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V(G)} c(v)$ is minimized.

[Sotskov, Tanaev ’76; Hansen, Kuplinsky, de Werra ’97]

Interval graphs (no directed edges):

- sort by left endpoints, color greedily (in linear time given sorted intervals)

Directional interval graphs:

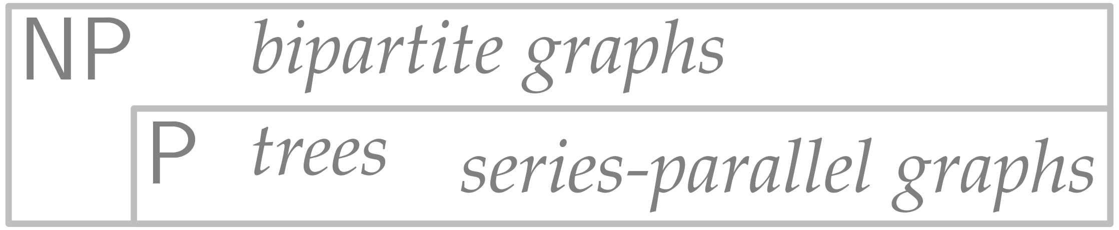
- recognition in $O(n^2)$ time

$n :=$ # intervals

Directed acyclic graphs (only directed edges):

- sort topologically, color greedily (in linear time)

Coloring Mixed Graphs


NP *bipartite graphs*

P *trees* *series-parallel graphs*

Given a graph G, find a coloring $c\colon V(G) \to \mathbb{N}$ s.t.

[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

- $\star$ undirected edge uv: $c(u) \neq c(v)$,
- $\star$ directed edge uv: $c(u) < c(v)$,
- $\star$ $\max_{v \in V(G)} c(v)$ is minimized.

Interval graphs (no directed edges):

- sort by left endpoints, color greedily (in linear time given sorted intervals)

Directional interval graphs:

- recognition in $O(n^2)$ time

$n :=$ # intervals

- coloring in $O(n \log n)$ time by a greedy algorithm

Directed acyclic graphs (only directed edges):

- sort topologically, color greedily (in linear time)

Coloring Mixed Graphs

NP *bipartite graphs*

P *trees* *series-parallel graphs*

Given a graph G, find a coloring $c\colon V(G) \to \mathbb{N}$ s.t.
[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V(G)} c(v)$ is minimized.

Interval graphs (no directed edges):

- sort by left endpoints, color greedily (in linea

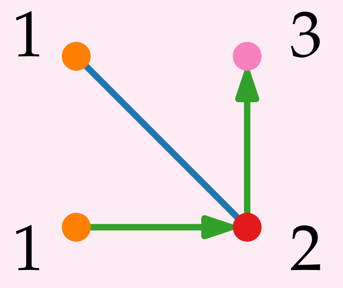
Directional interval graphs:

- recognition in $O(n^2)$ time

$n :=$ # intervals

- coloring in $O(n \log n)$ time by a greedy algorithm

min. coloring rvals)



Directed acyclic graphs (only directed edges):

- sort topologically, color greedily (in linear time)

Coloring Mixed Graphs

NP *bipartite graphs*

P *trees* *series-parallel graphs*

Given a graph G, find a coloring $c\colon V(G) \to \mathbb{N}$ s.t.
[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V(G)} c(v)$ is minimized.

Interval graphs (no directed edges):

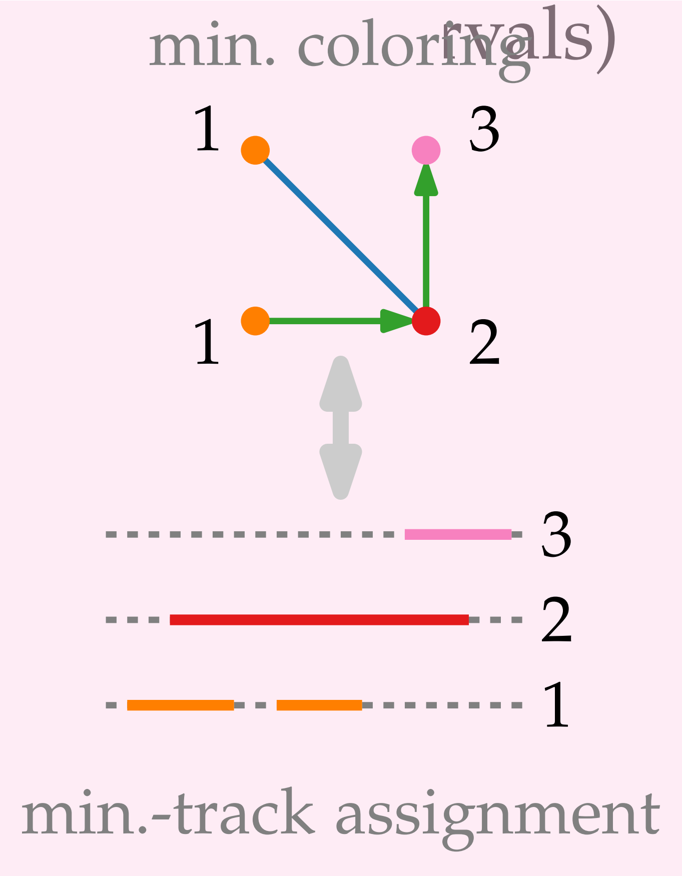
- sort by left endpoints, color greedily (in linea

Directional interval graphs:

- recognition in $O(n^2)$ time

$n :=$ # intervals

- coloring in $O(n \log n)$ time by a greedy algorithm



Directed acyclic graphs (only directed edges):

- sort topologically, color greedily (in linear time)

Coloring Mixed Graphs

NP *bipartite graphs*

P *trees* *series-parallel graphs*

Given a graph G, find a coloring $c\colon V(G) \to \mathbb{N}$ s.t.

[Sotskov, Tanaev ’76; Hansen, Kuplinsky, de Werra ’97]

- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V(G)} c(v)$ is minimized.

Interval graphs (no directed edges):

- sort by left endpoints, color greedily (in linea

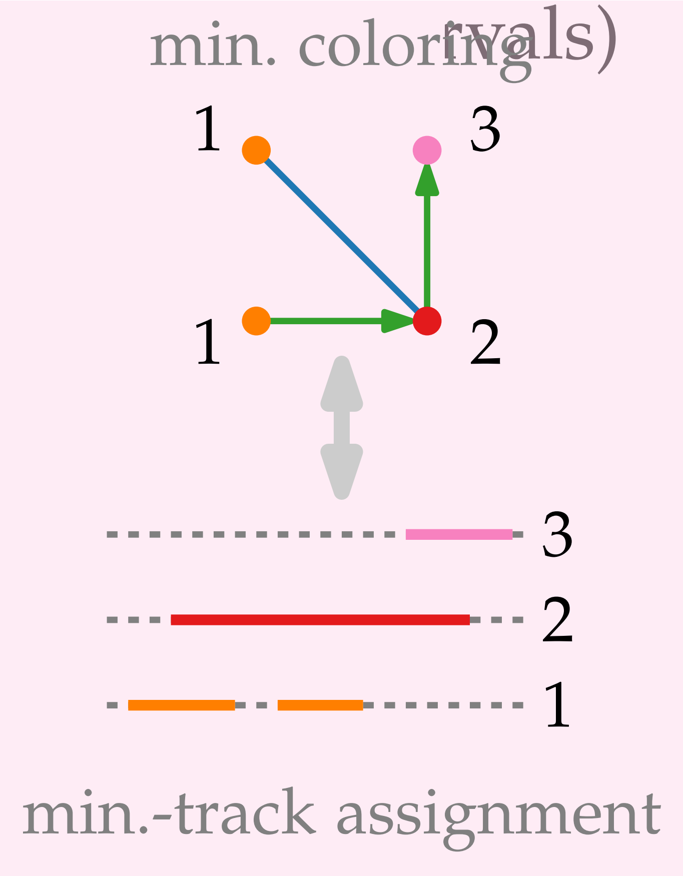
Directional interval graphs:

- recognition in $O(n^2)$ time

$n :=$ # intervals

- coloring in $O(n \log n)$ time by a greedy algorithm

Mixed interval graphs:



Directed acyclic graphs (only directed edges):

- sort topologically, color greedily (in linear time)

Coloring Mixed Graphs

NP *bipartite graphs*

P *trees* *series-parallel graphs*

Given a graph G, find a coloring $c\colon V(G) \to \mathbb{N}$ s.t.

[Sotskov, Tanaev ’76; Hansen, Kuplinsky, de Werra ’97]

- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V(G)} c(v)$ is minimized.

Interval graphs (no directed edges):

- sort by left endpoints, color greedily (in linea

Directional interval graphs:

- recognition in $O(n^2)$ time $n :=$ # intervals
- coloring in $O(n \log n)$ time by a greedy algorithm

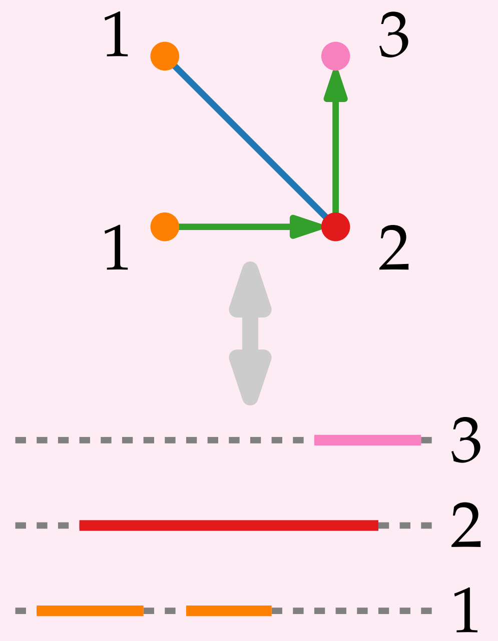
Mixed interval graphs:

- coloring is NP-complete

Directed acyclic graphs (only directed edges):

- sort topologically, color greedily (in linear time)

min. coloring rvals)



min.-track assignment

Coloring Mixed Graphs

NP *bipartite graphs*

P *trees* *series-parallel graphs*

Given a graph G, find a coloring $c\colon V(G) \to \mathbb{N}$ s.t.
[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V(G)} c(v)$ is minimized.

Interval graphs (no directed edges):

- sort by left endpoints, color greedily (in linea

Directional interval graphs: **our contribution**

- recognition in $O(n^2)$ time $n := \#$ intervals
- coloring in $O(n \log n)$ time by a greedy algorithm

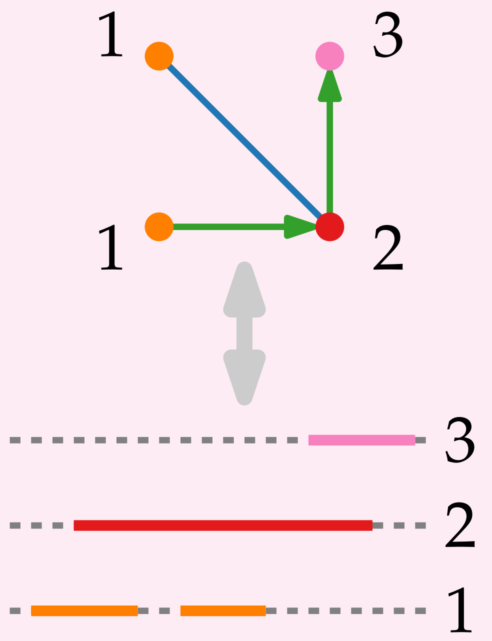
Mixed interval graphs:

- coloring is NP-complete

Directed acyclic graphs (only directed edges):

- sort topologically, color greedily (in linear time)

min. coloring rvals)



min.-track assignment

Coloring Mixed Graphs

NP *bipartite graphs*

P *trees* *series-parallel graphs*

Given a graph G, find a coloring $c \colon V(G) \to \mathbb{N}$ s.t.
[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V(G)} c(v)$ is minimized.

Interval graphs (no directed edges):

- sort by left endpoints, color greedily (in linea

Directional interval graphs: **our contribution**

- recognition in $O(n^2)$ time $n := \#$ intervals
- coloring in $O(n \log n)$ time by a greedy algorithm

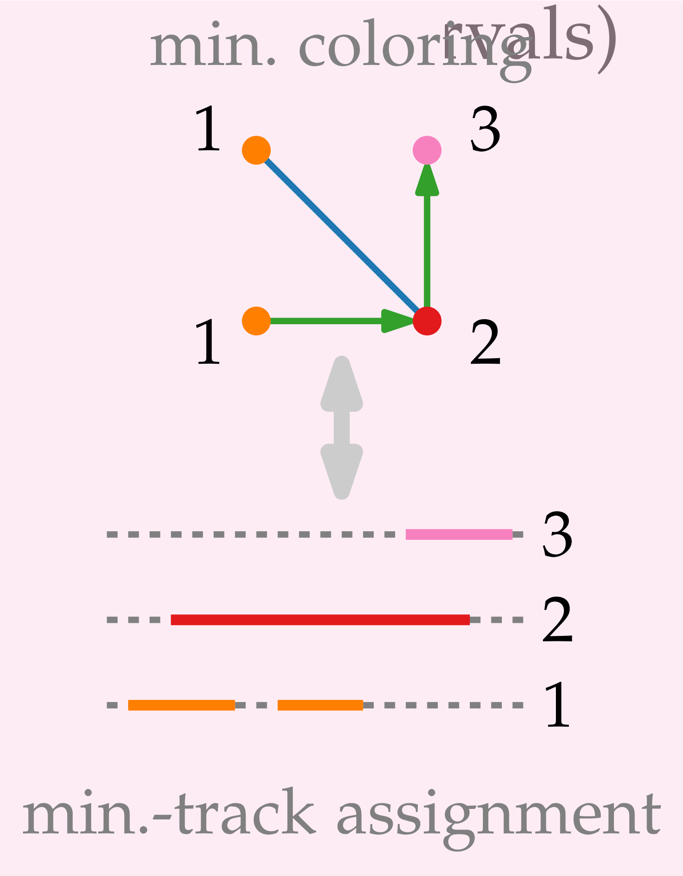
agenda for this talk

Mixed interval graphs:

- coloring is NP-complete

Directed acyclic graphs (only directed edges):

- sort topologically, color greedily (in linear time)



Coloring Directional Interval Graphs

Given: an interval representation of a directional interval graph G

Coloring Directional Interval Graphs

Given: an interval representation of a directional interval graph G

`GreedyColoring:`

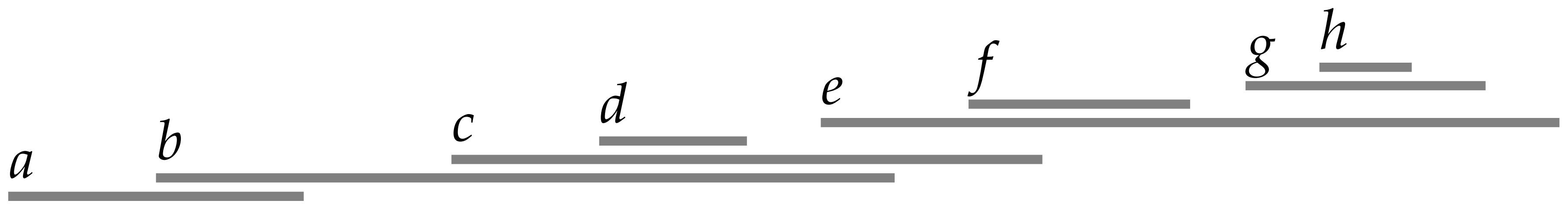
1. sort all intervals by left endpoint
2. for each interval, assign the smallest available color respecting incident edges

Coloring Directional Interval Graphs

Given: an interval representation of a directional interval graph G

`GreedyColoring:`

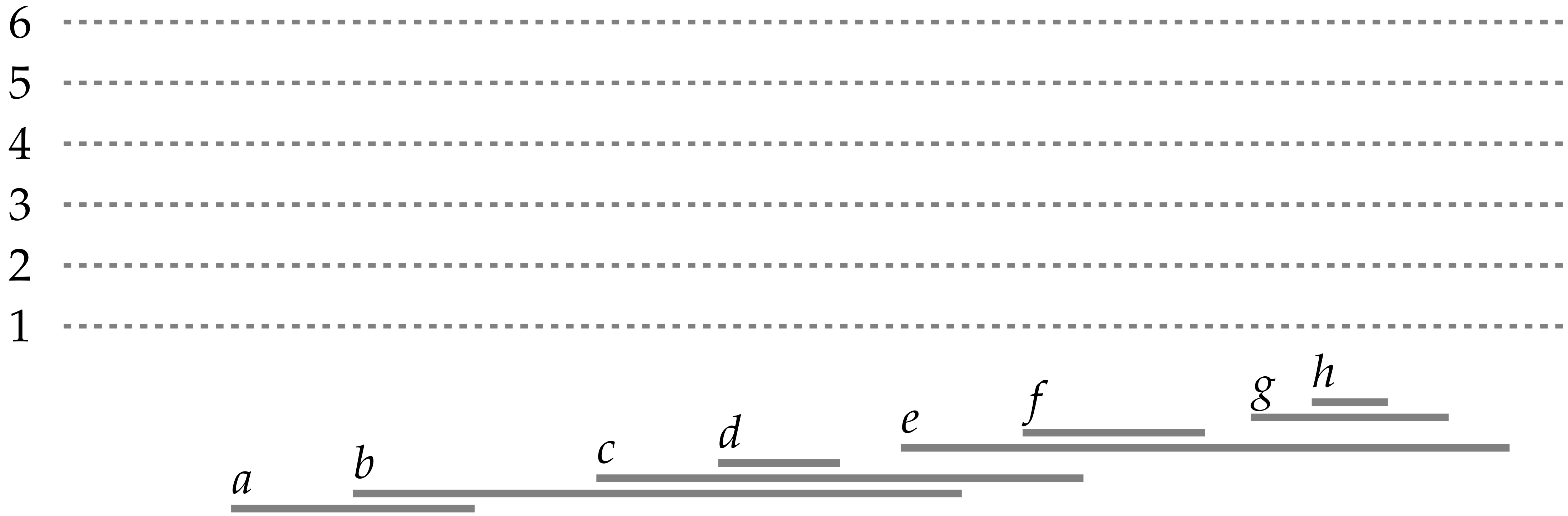
1. sort all intervals by left endpoint
2. for each interval, assign the smallest available color respecting incident edges



Coloring Directional Interval Graphs

Given: an interval representation of a directional interval graph G

```
GreedyColoring:
1. sort all intervals by left endpoint
2. for each interval, assign the smallest available color respecting incident edges
```

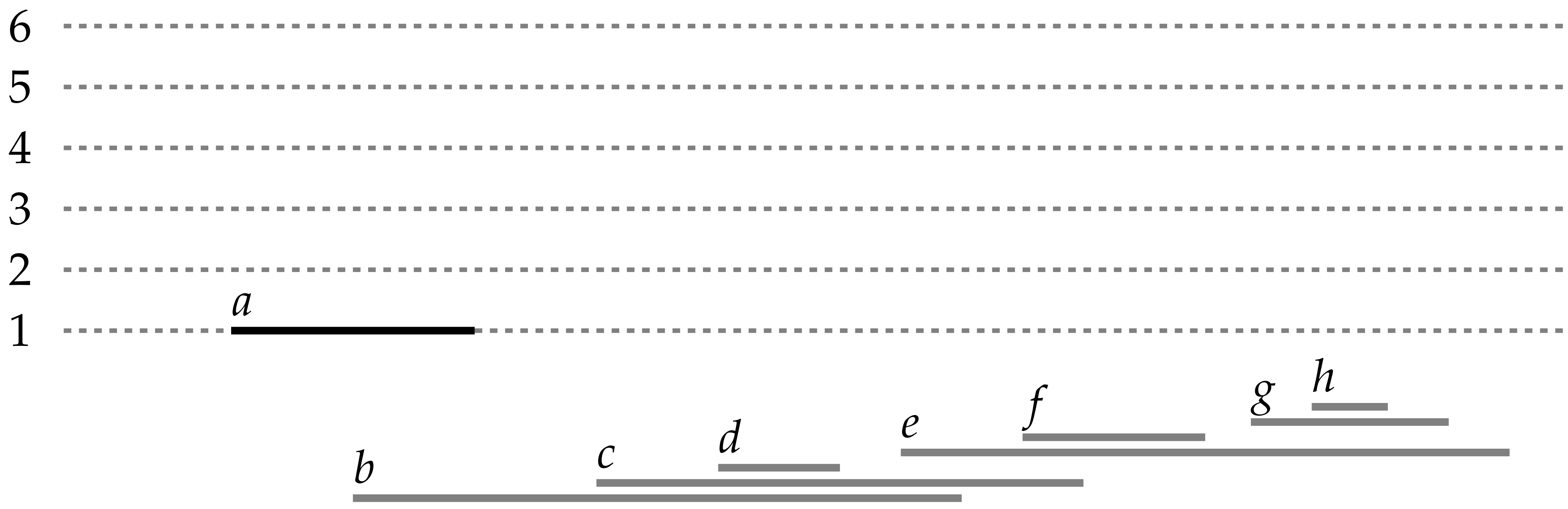


Coloring Directional Interval Graphs

Given: an interval representation of a directional interval graph G

`GreedyColoring:`

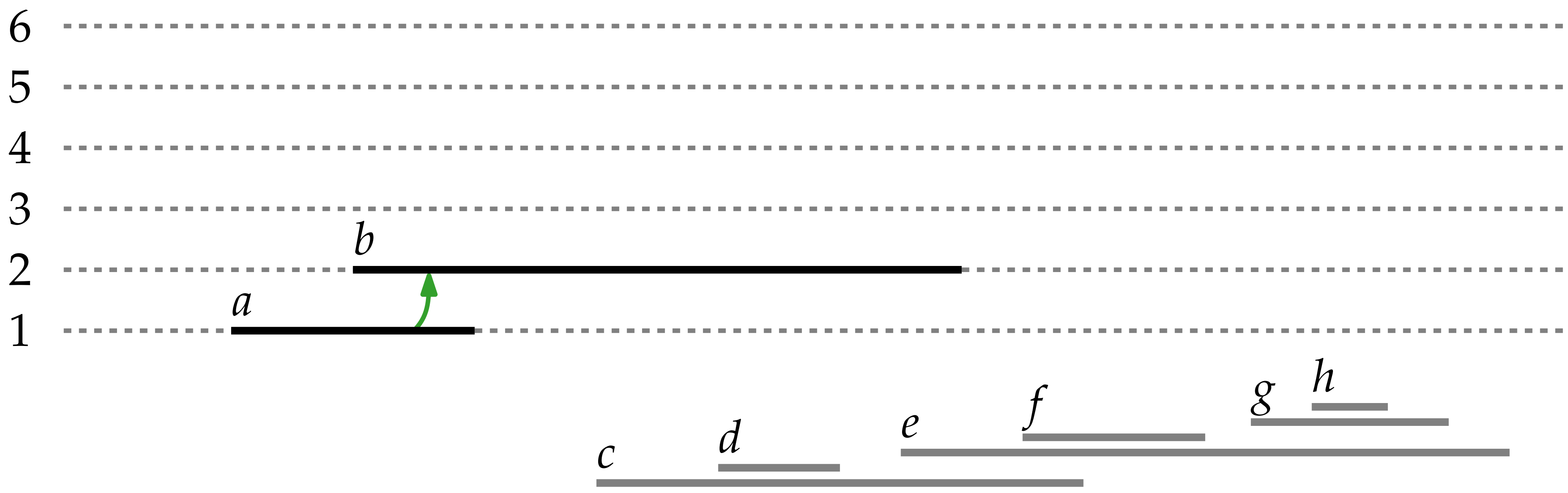
1. sort all intervals by left endpoint
2. for each interval, assign the smallest available color respecting incident edges



Coloring Directional Interval Graphs

Given: an interval representation of a directional interval graph G

```
GreedyColoring:
1. sort all intervals by left endpoint
2. for each interval, assign the smallest available color respecting incident edges
```

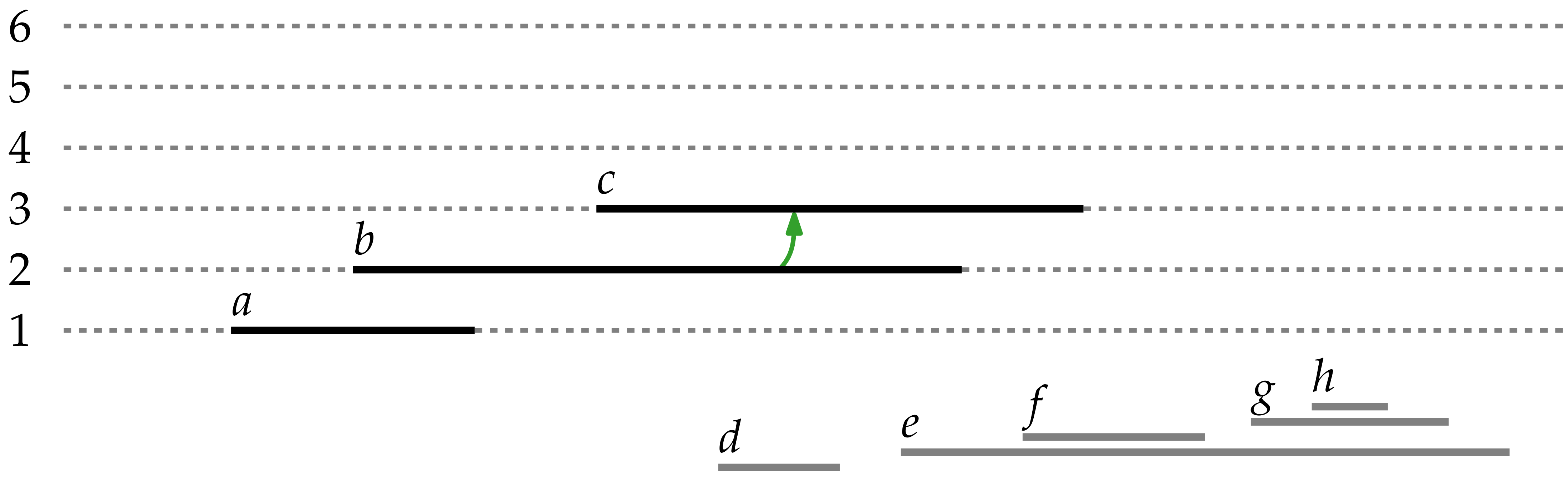


Coloring Directional Interval Graphs

Given: an interval representation of a directional interval graph G

`GreedyColoring:`

1. sort all intervals by left endpoint
2. for each interval, assign the smallest available color respecting incident edges

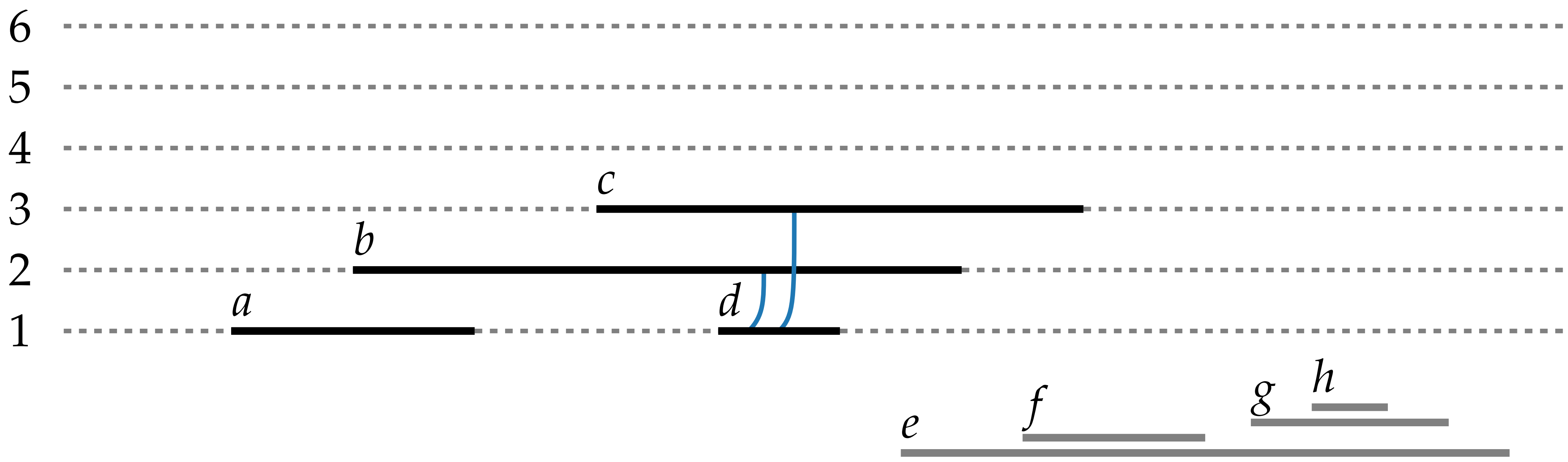


Coloring Directional Interval Graphs

Given: an interval representation of a directional interval graph G

`GreedyColoring:`

1. sort all intervals by left endpoint
2. for each interval, assign the smallest available color respecting incident edges

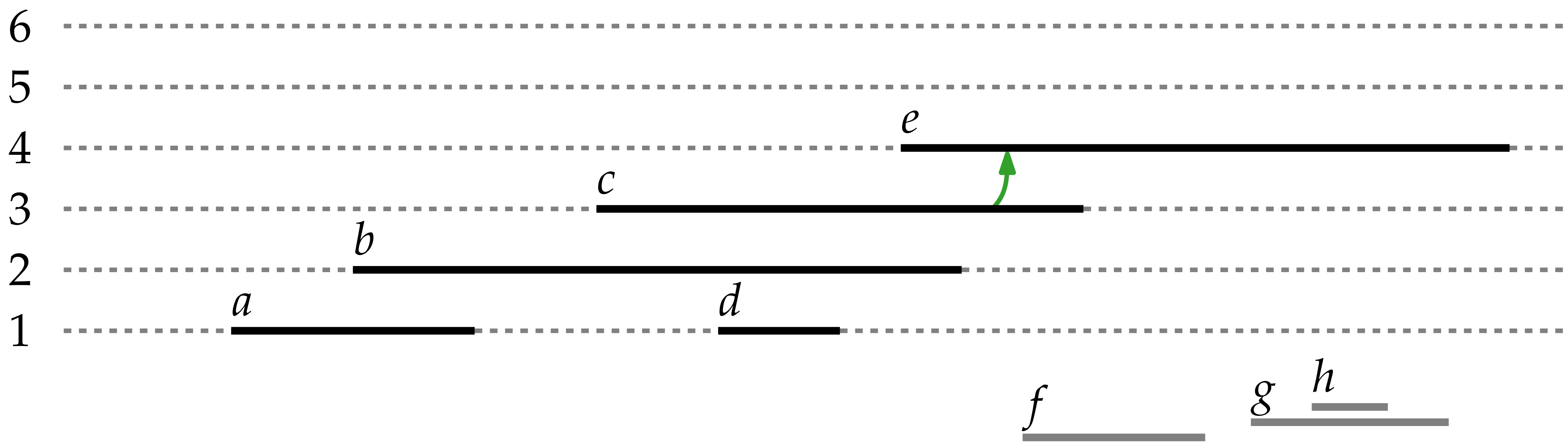


Coloring Directional Interval Graphs

Given: an interval representation of a directional interval graph G

`GreedyColoring:`

1. sort all intervals by left endpoint
2. for each interval, assign the smallest available color respecting incident edges

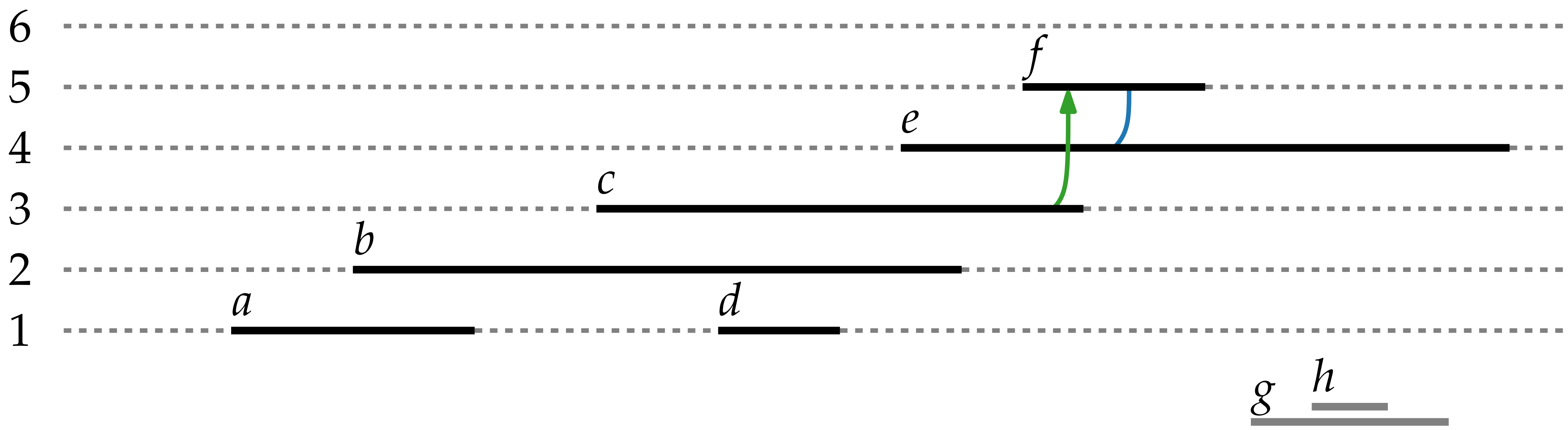


Coloring Directional Interval Graphs

Given: an interval representation of a directional interval graph G

`GreedyColoring:`

1. sort all intervals by left endpoint
2. for each interval, assign the smallest available color respecting incident edges

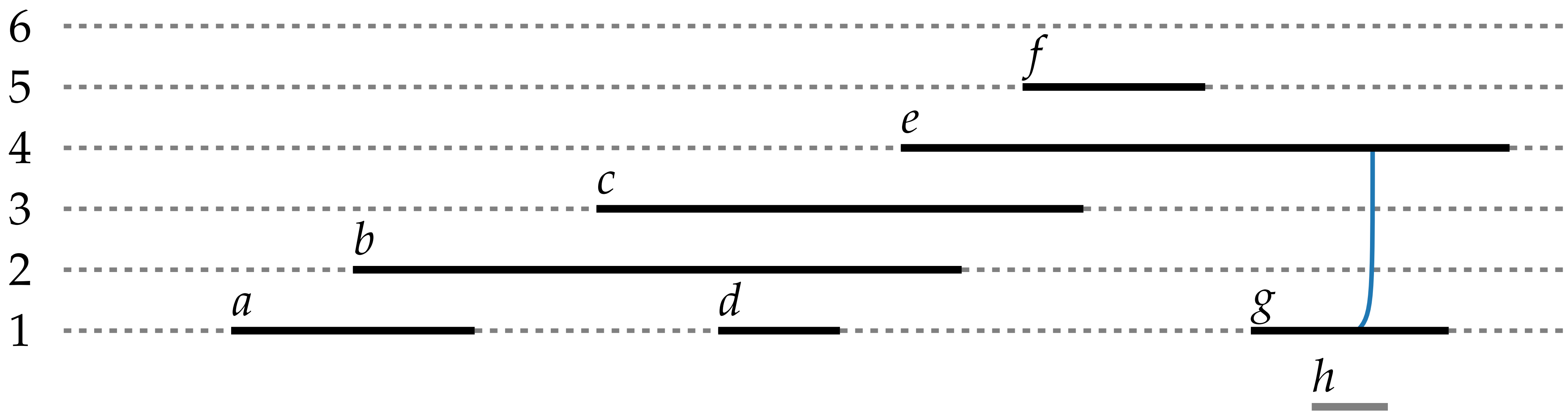


Coloring Directional Interval Graphs

Given: an interval representation of a directional interval graph G

`GreedyColoring:`

1. sort all intervals by left endpoint
2. for each interval, assign the smallest available color respecting incident edges

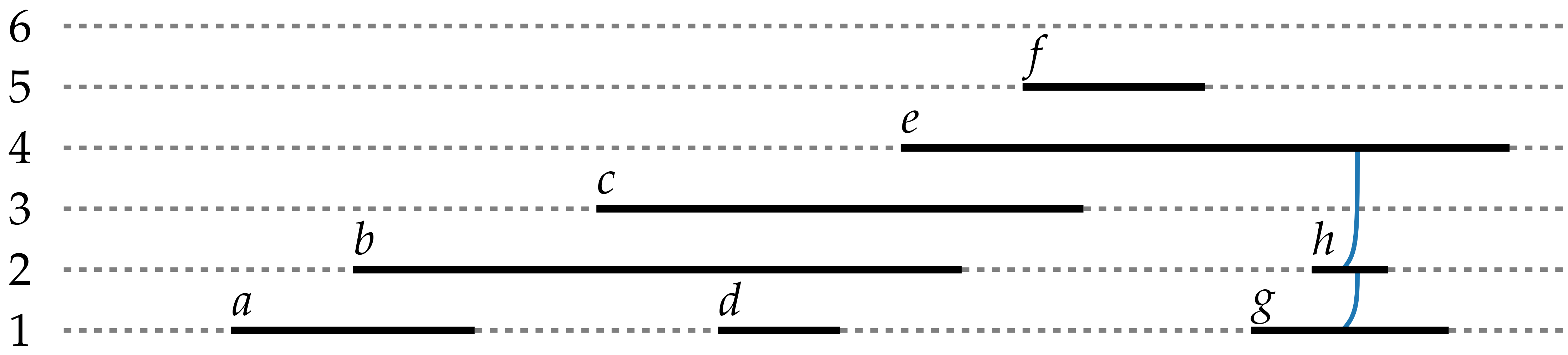


Coloring Directional Interval Graphs

Given: an interval representation of a directional interval graph G

`GreedyColoring:`

1. sort all intervals by left endpoint
2. for each interval, assign the smallest available color respecting incident edges

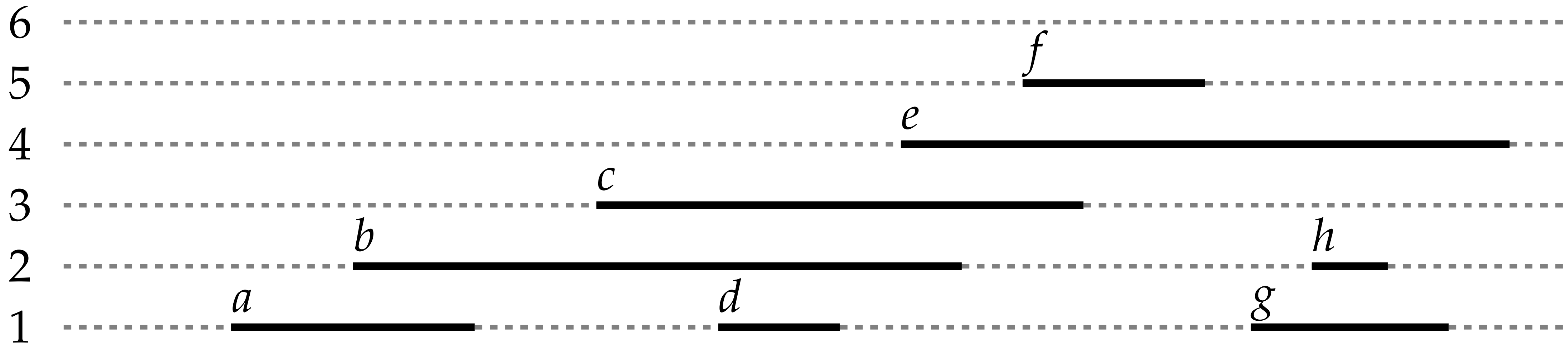


Coloring Directional Interval Graphs

Given: an interval representation of a directional interval graph G

`GreedyColoring:`

1. sort all intervals by left endpoint
2. for each interval, assign the smallest available color respecting incident edges



Coloring Directional Interval Graphs

Theorem 1:

A coloring c computed by `GreedyColoring` has the minimum number of colors.

Coloring Directional Interval Graphs

Theorem 1:

A coloring c computed by `GreedyColoring` has the minimum number of colors.

Proof sketch:

Coloring Directional Interval Graphs

Theorem 1:
A coloring c computed by `GreedyColoring` has the minimum number of colors.

Proof sketch:

- Let G^{+} be the *transitive closure* of G
  (the graph obtained by exhaustively adding transitive directed edges to G).

Coloring Directional Interval Graphs

Theorem 1:
A coloring c computed by `GreedyColoring` has the minimum number of colors.

Proof sketch:

- Let G^+ be the *transitive closure* of G
  (the graph obtained by exhaustively adding transitive directed edges to G).
- Show: the size of a largest clique in G^+ equals the maximum color m in c.

Coloring Directional Interval Graphs

Theorem 1:
A coloring c computed by `GreedyColoring` has the minimum number of colors.

Proof sketch:

- Let G^{+} be the *transitive closure* of G
  (the graph obtained by exhaustively adding transitive directed edges to G).

- Show: the size of a largest clique in G^{+} equals the maximum color m in c.

  $\Rightarrow$ The coloring c uses the minimum number of colors.

Coloring Directional Interval Graphs

Theorem 1:

A coloring c computed by `GreedyColoring` has the minimum number of colors.

Proof sketch:

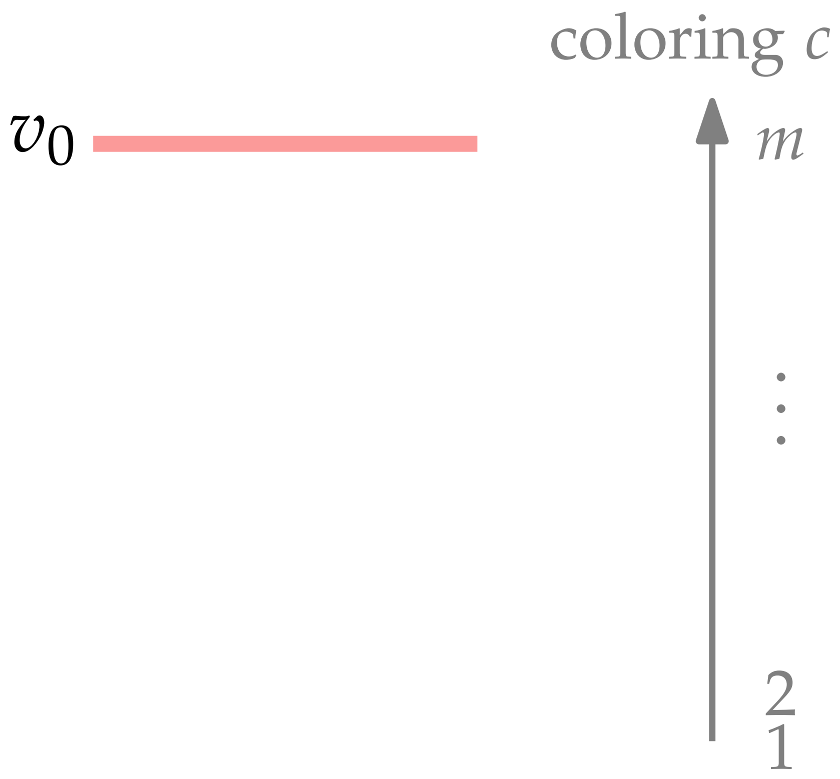
- Let v_0 be an interval of maximum color, i.e., $c(v_0) = m$.

Coloring Directional Interval Graphs

Theorem 1:
A coloring c computed by `GreedyColoring` has the minimum number of colors.

Proof sketch:

- Let v_0 be an interval of maximum color, i.e., $c(v_0) = m$.

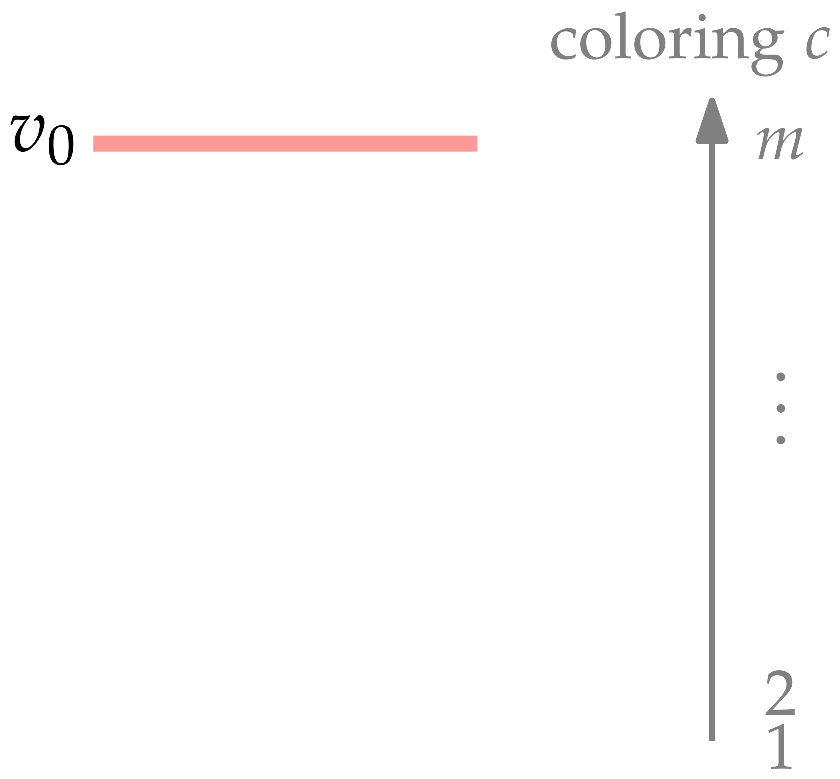


Coloring Directional Interval Graphs

Theorem 1:
A coloring c computed by `GreedyColoring` has the minimum number of colors.

Proof sketch:

- Let v_0 be an interval of maximum color, i.e., $c(v_0) = m$.
- Among all intervals having a directed edge to v_0, let v_1 be the one with the largest color.

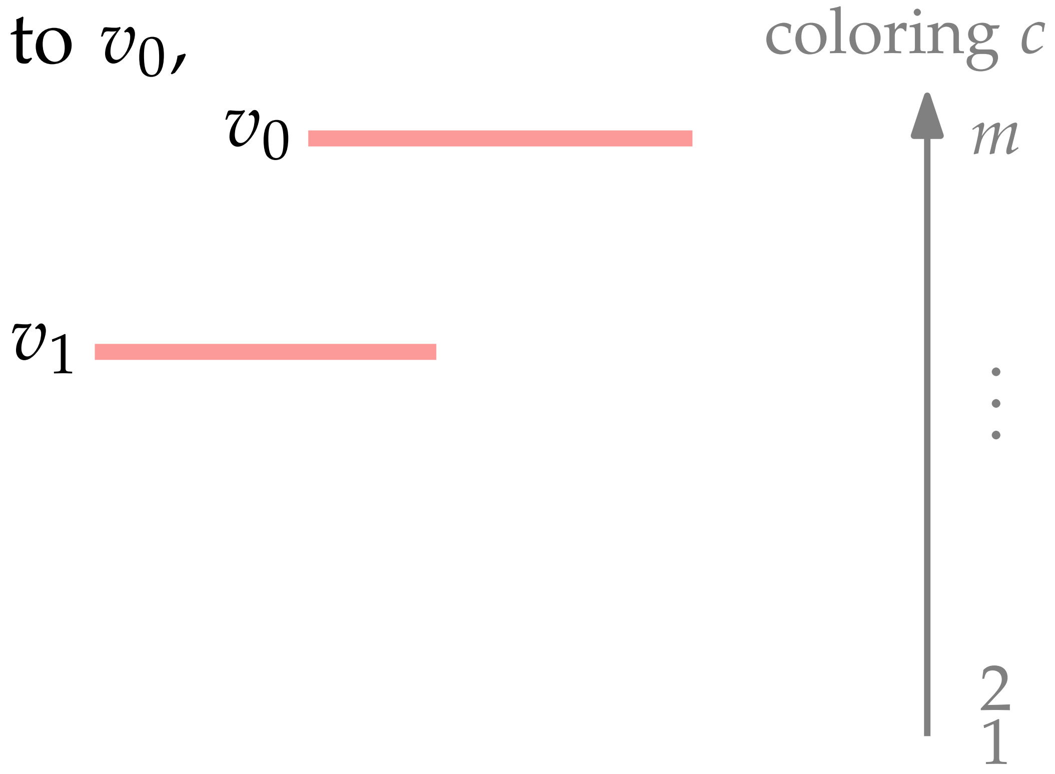


Coloring Directional Interval Graphs

Theorem 1:
A coloring c computed by `GreedyColoring` has the minimum number of colors.

Proof sketch:

- Let v_0 be an interval of maximum color, i.e., $c(v_0) = m$.
- Among all intervals having a directed edge to v_0, let v_1 be the one with the largest color.



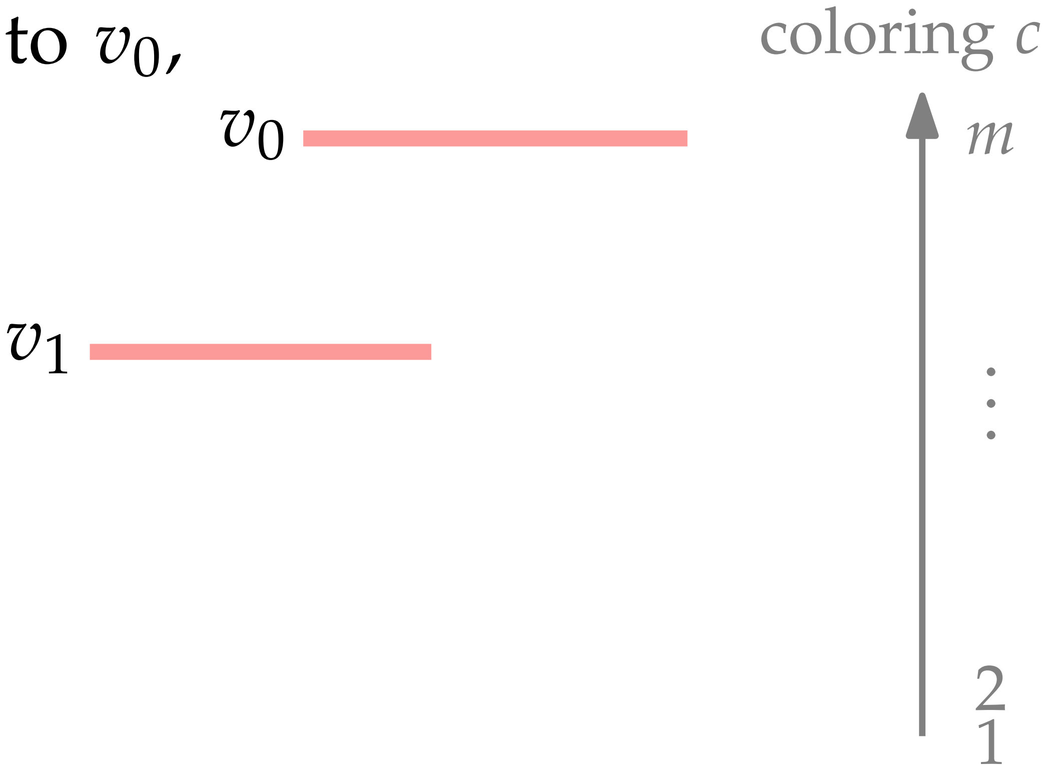
Coloring Directional Interval Graphs

Theorem 1:

A coloring c computed by `GreedyColoring` has the minimum number of colors.

Proof sketch:

- Let v_0 be an interval of maximum color, i.e., $c(v_0) = m$.
- Among all intervals having a directed edge to v_0, let v_1 be the one with the largest color.
- Similarly, define v_2 w.r.t. v_1 and so on.

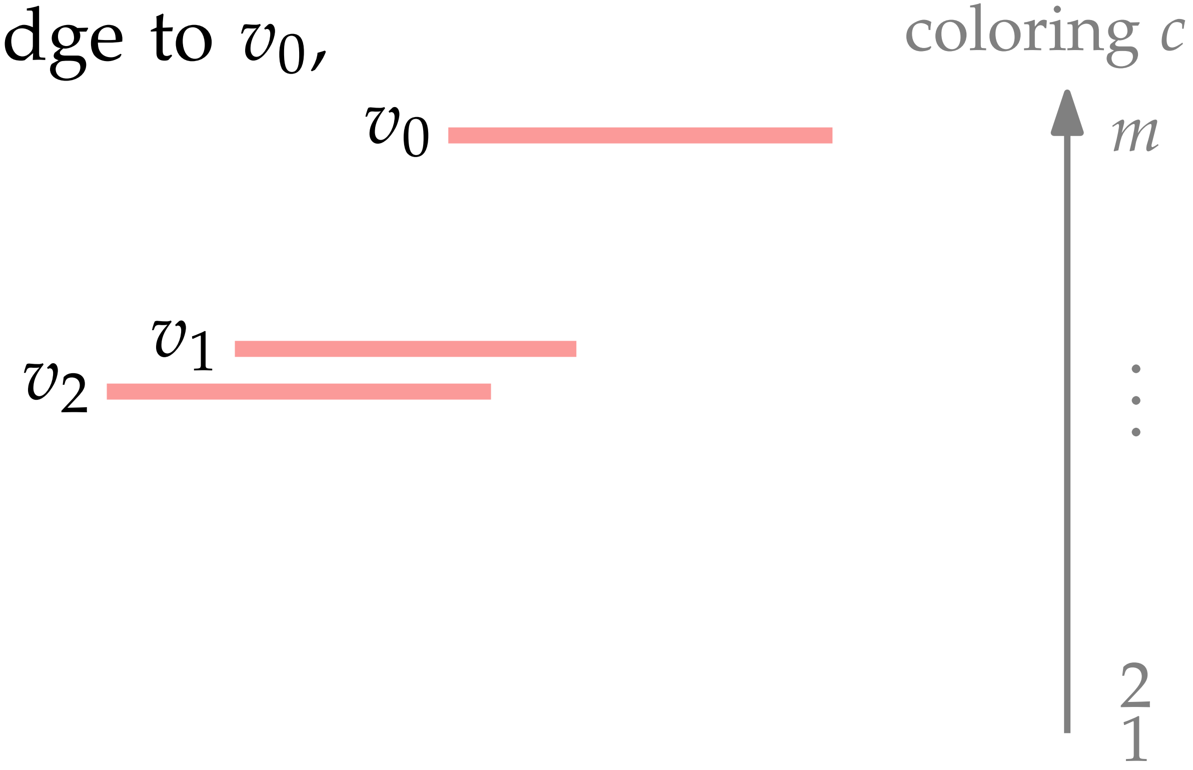


Coloring Directional Interval Graphs

Theorem 1:
A coloring c computed by `GreedyColoring` has the minimum number of colors.

Proof sketch:

- Let v_0 be an interval of maximum color, i.e., $c(v_0) = m$.
- Among all intervals having a directed edge to v_0, let v_1 be the one with the largest color.
- Similarly, define v_2 w.r.t. v_1 and so on.



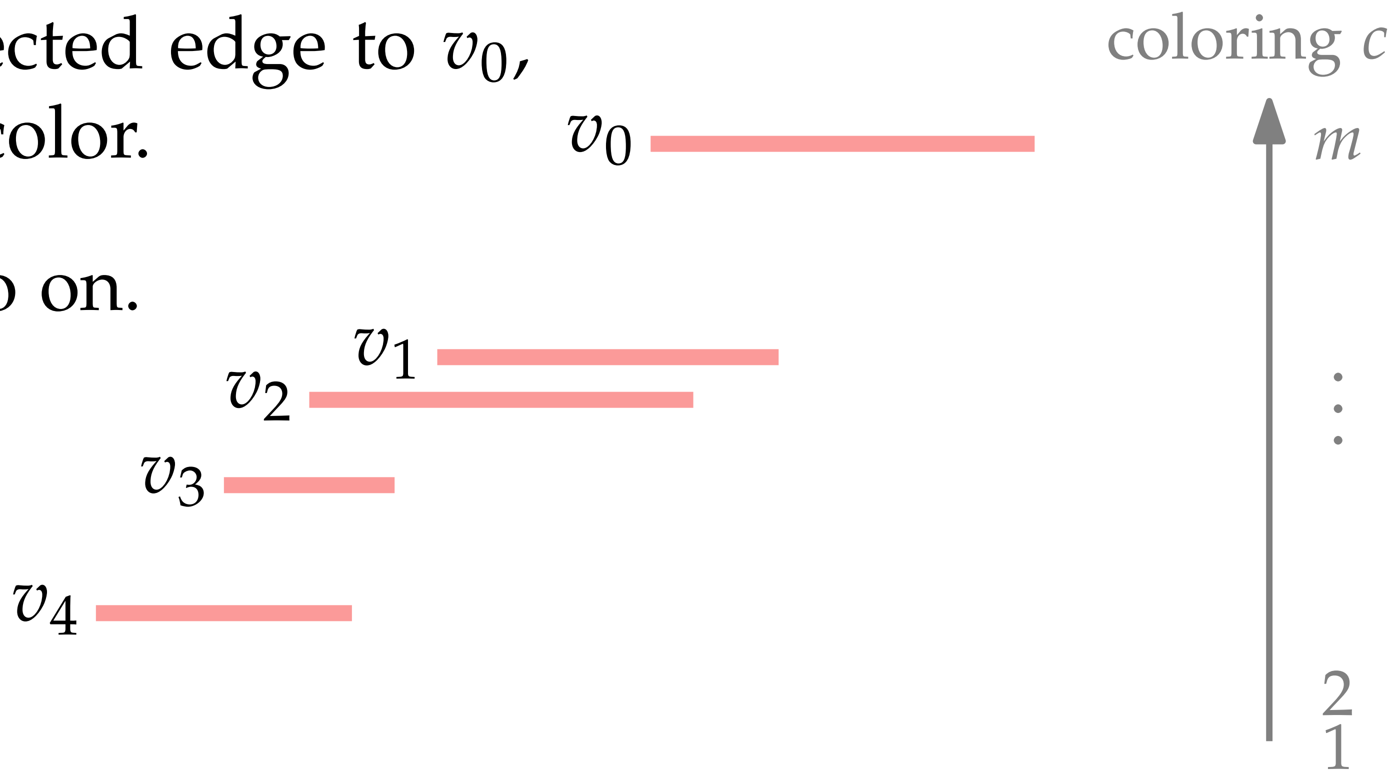
Coloring Directional Interval Graphs

Theorem 1:

A coloring c computed by `GreedyColoring` has the minimum number of colors.

Proof sketch:

- Let v_0 be an interval of maximum color, i.e., $c(v_0) = m$.
- Among all intervals having a directed edge to v_0, let v_1 be the one with the largest color.
- Similarly, define v_2 w.r.t. v_1 and so on.

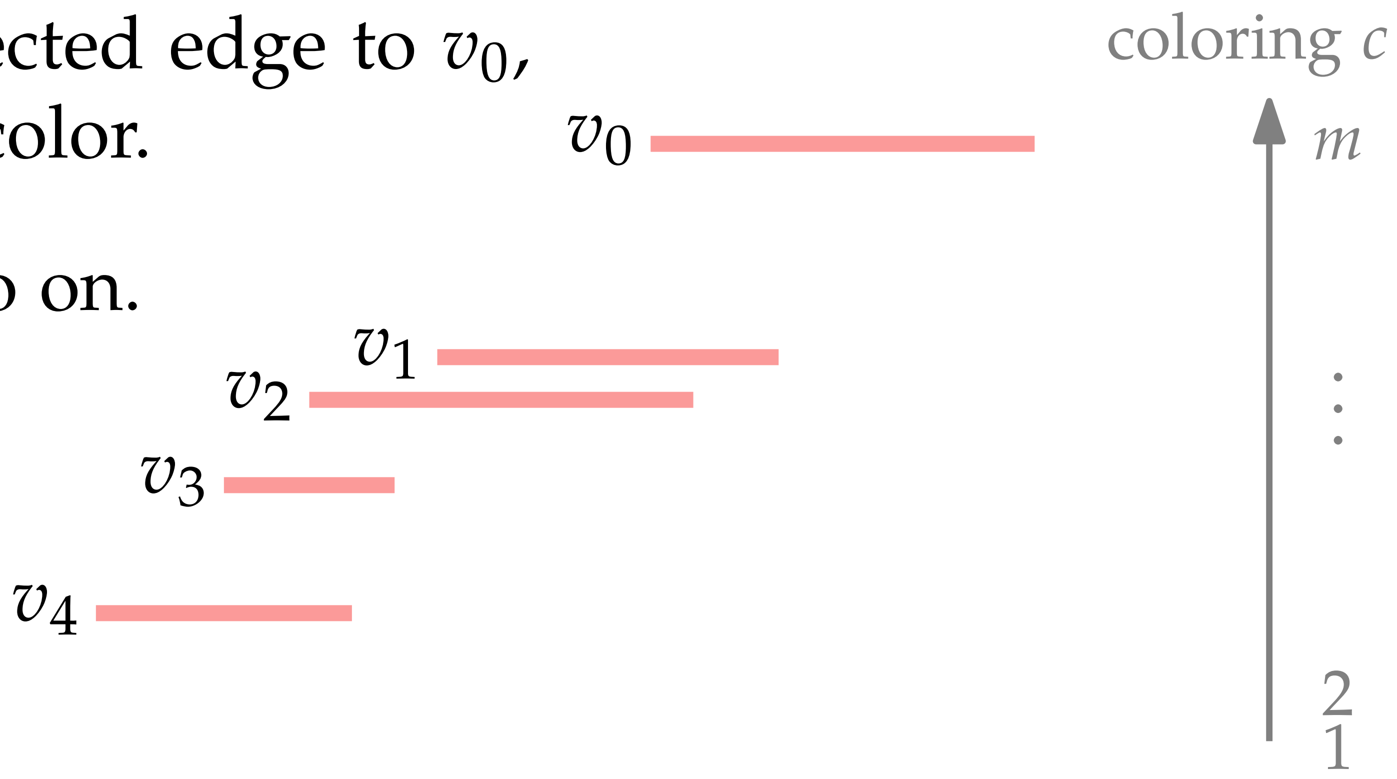


Coloring Directional Interval Graphs

Theorem 1:
A coloring c computed by `GreedyColoring` has the minimum number of colors.

Proof sketch:

- Let v_0 be an interval of maximum color, i.e., $c(v_0) = m$.
- Among all intervals having a directed edge to v_0, let v_1 be the one with the largest color.
- Similarly, define v_2 w.r.t. v_1 and so on.
- By the greedy strategy, the colors between $c(v_i)$ and $c(v_{i+1})$ are occupied by intervals containing the left endpoint of v_i.

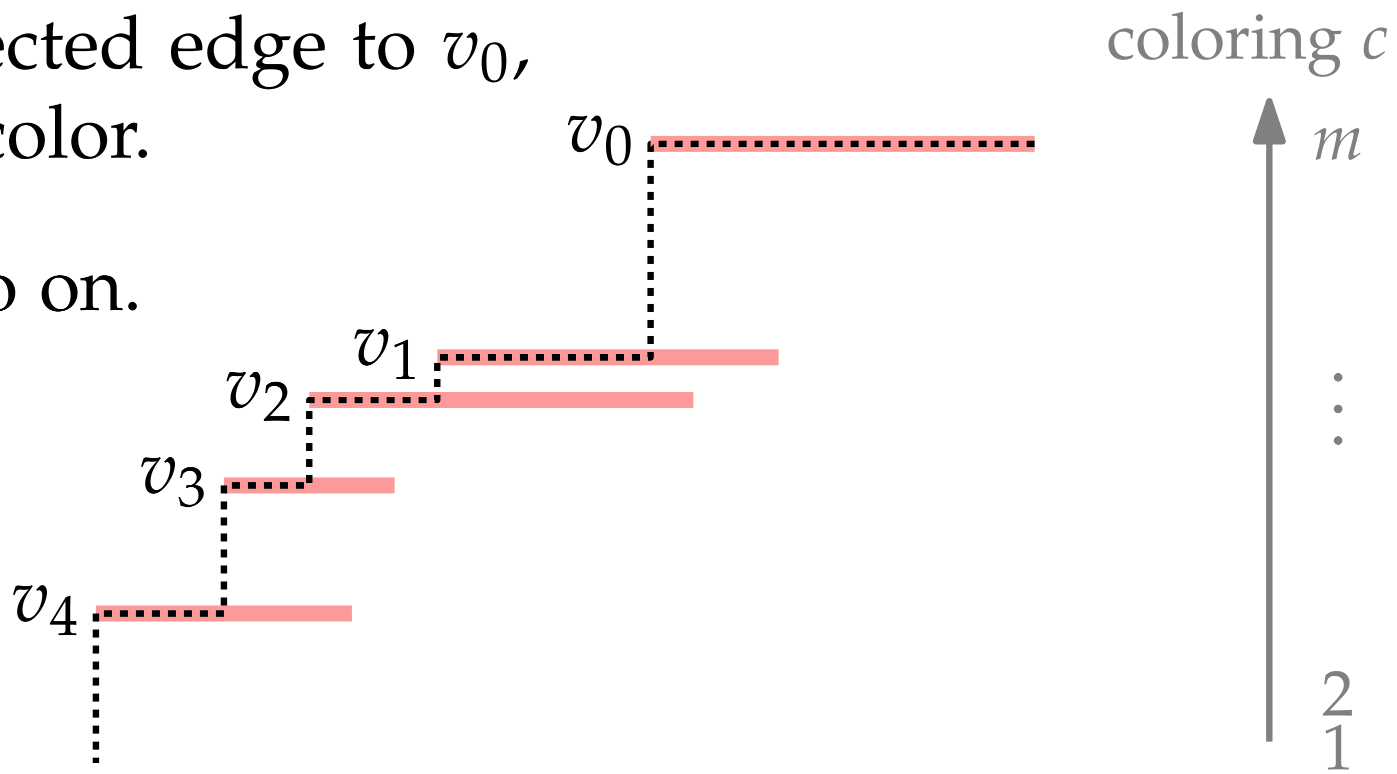


Coloring Directional Interval Graphs

Theorem 1:
A coloring c computed by `GreedyColoring` has the minimum number of colors.

Proof sketch:

- Let v_0 be an interval of maximum color, i.e., $c(v_0) = m$.
- Among all intervals having a directed edge to v_0, let v_1 be the one with the largest color.
- Similarly, define v_2 w.r.t. v_1 and so on.
- By the greedy strategy, the colors between $c(v_i)$ and $c(v_{i+1})$ are occupied by intervals containing the left endpoint of v_i.

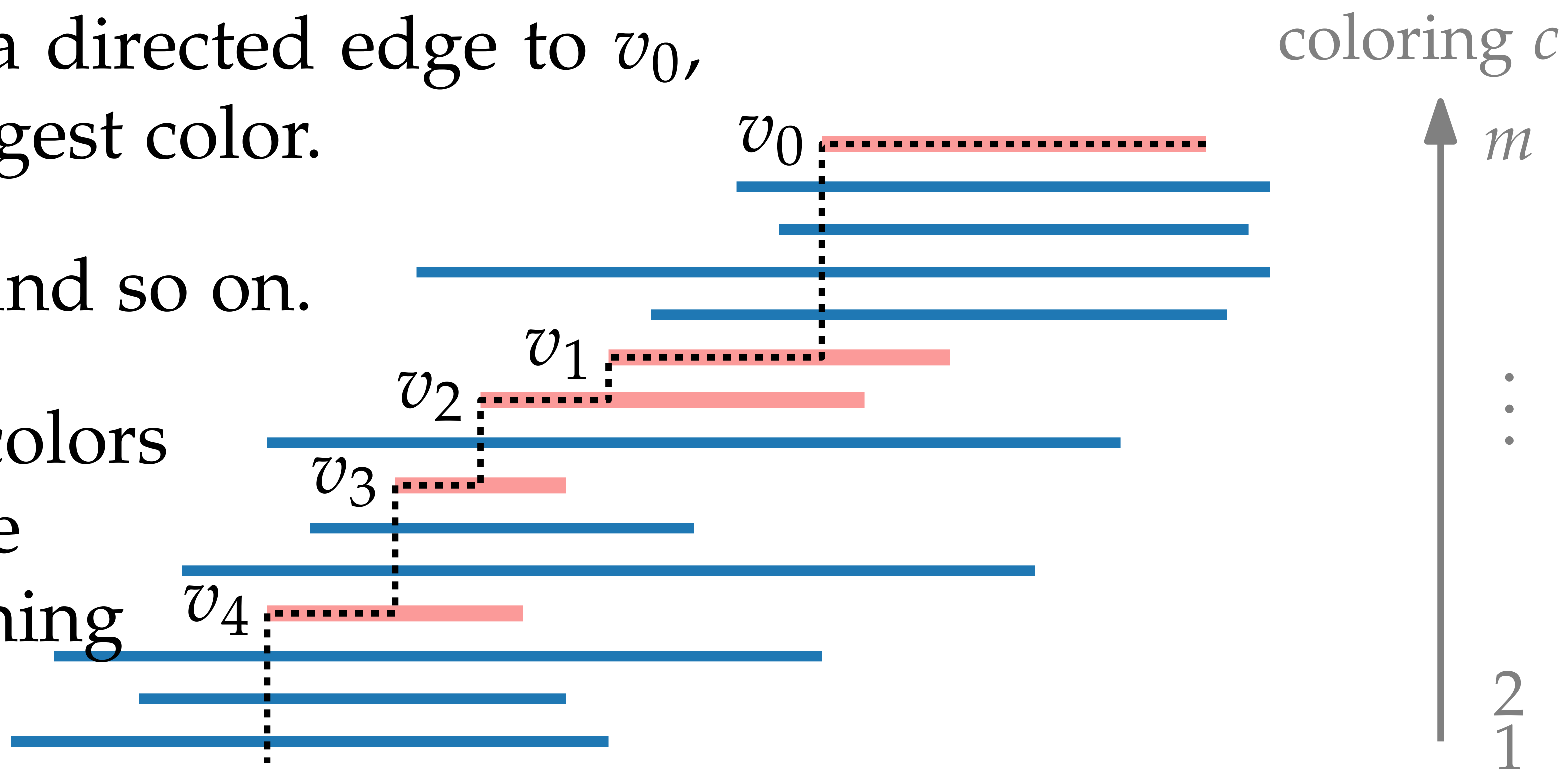


Coloring Directional Interval Graphs

Theorem 1:
A coloring c computed by `GreedyColoring` has the minimum number of colors.

Proof sketch:

- Let v_0 be an interval of maximum color, i.e., $c(v_0) = m$.
- Among all intervals having a directed edge to v_0, let v_1 be the one with the largest color.
- Similarly, define v_2 w.r.t. v_1 and so on.
- By the greedy strategy, the colors between $c(v_i)$ and $c(v_{i+1})$ are occupied by intervals containing the left endpoint of v_i.

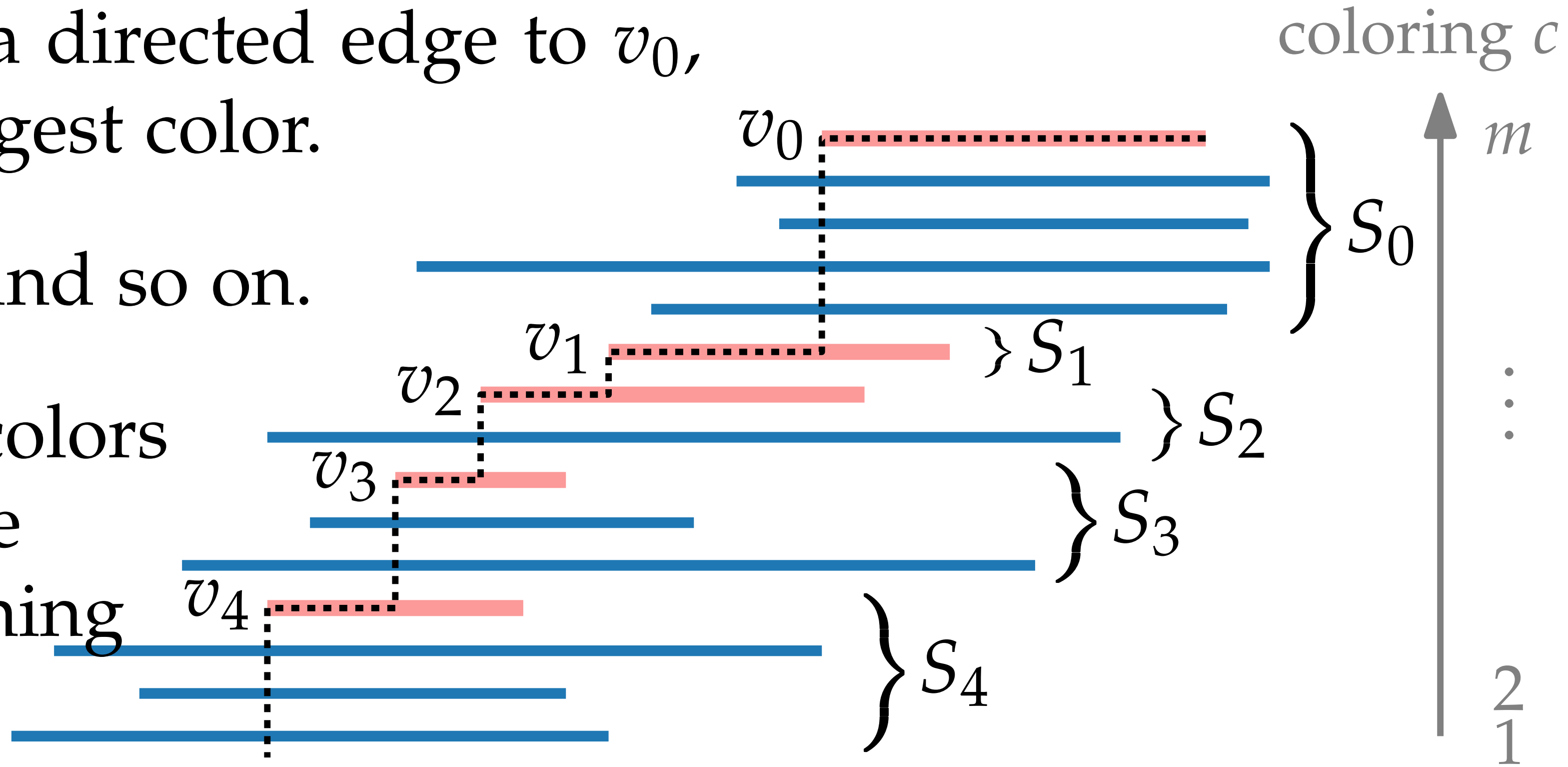


Coloring Directional Interval Graphs

Theorem 1:
A coloring c computed by `GreedyColoring` has the minimum number of colors.

Proof sketch:

- Let v_0 be an interval of maximum color, i.e., $c(v_0) = m$.
- Among all intervals having a directed edge to v_0, let v_1 be the one with the largest color.
- Similarly, define v_2 w.r.t. v_1 and so on.
- By the greedy strategy, the colors between $c(v_i)$ and $c(v_{i+1})$ are occupied by intervals containing the left endpoint of v_i.

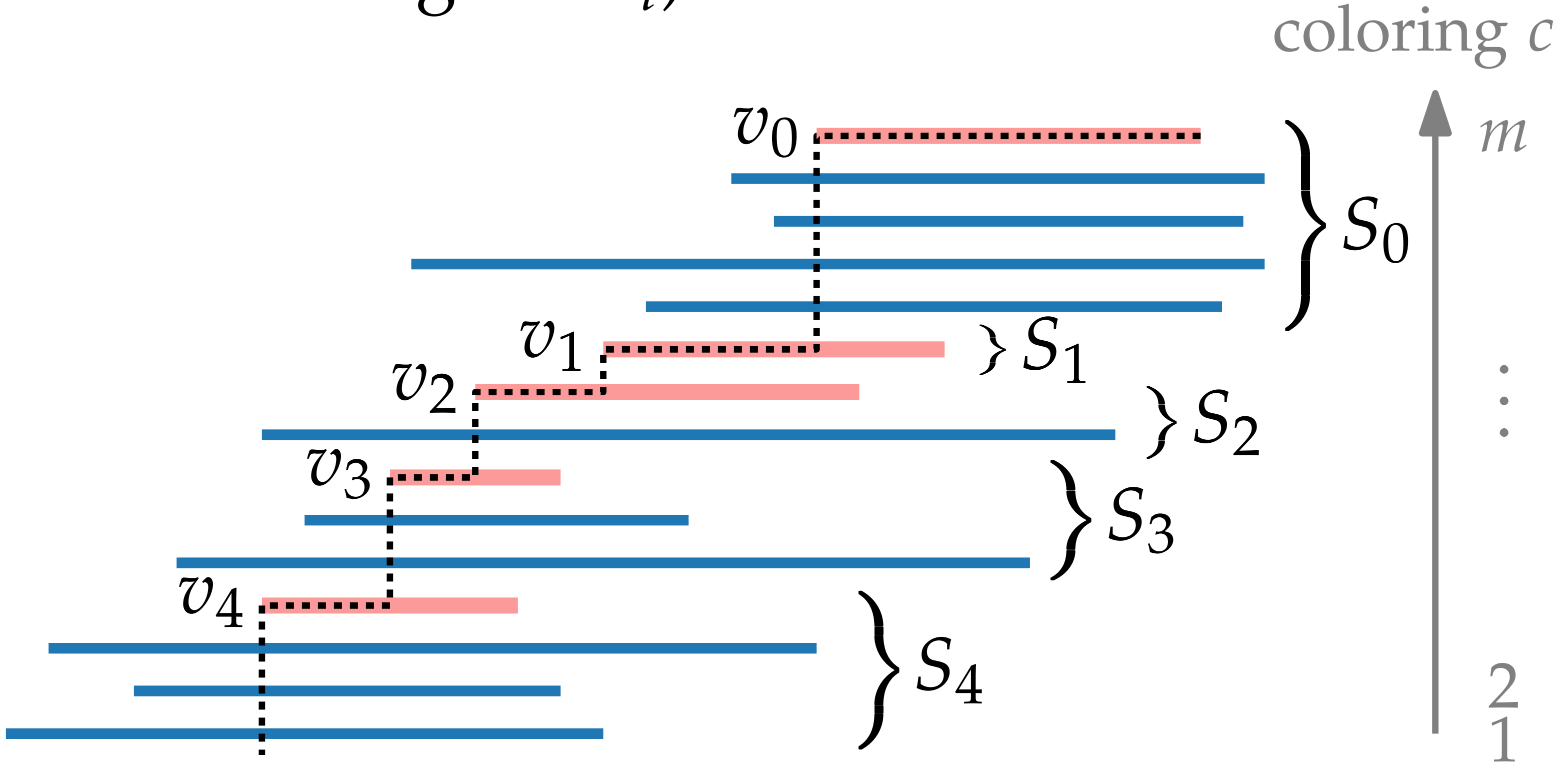


Coloring Directional Interval Graphs

Theorem 1:
A coloring c computed by `GreedyColoring` has the minimum number of colors.

Proof sketch:

- Hence, for every *step* S_i, all intervals contain v_i.
  (otherwise they would have a directed edge to v_i)



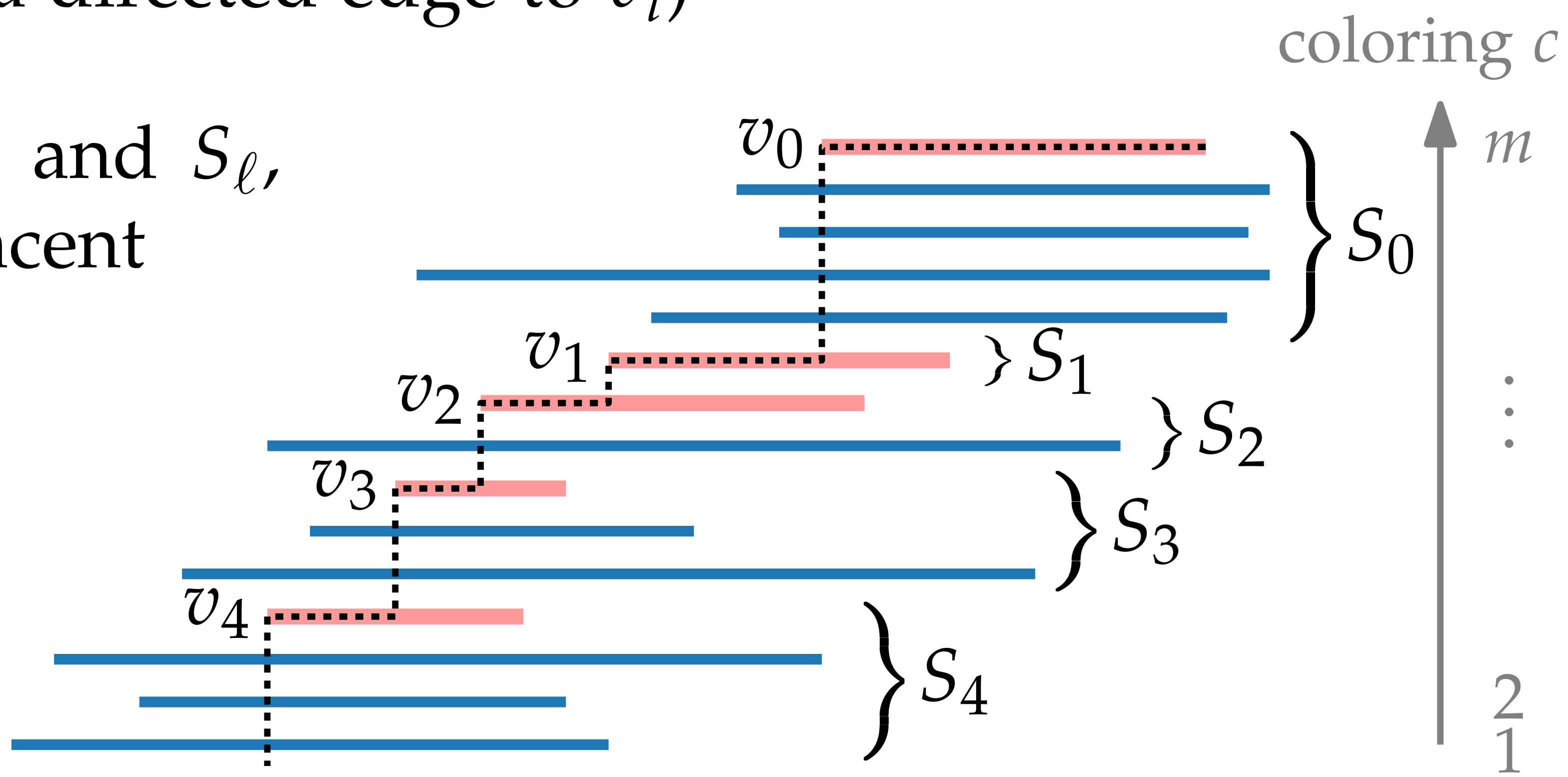
Coloring Directional Interval Graphs

Theorem 1:
A coloring c computed by `GreedyColoring` has the minimum number of colors.

Proof sketch:

- Hence, for every *step* S_i, all intervals contain v_i.
  (otherwise they would have a directed edge to v_i)

- **Claim:** for any two steps S_i and S_ℓ, every pair of intervals is adjacent in the transitive closure G^+.



Coloring Directional Interval Graphs

Theorem 1:

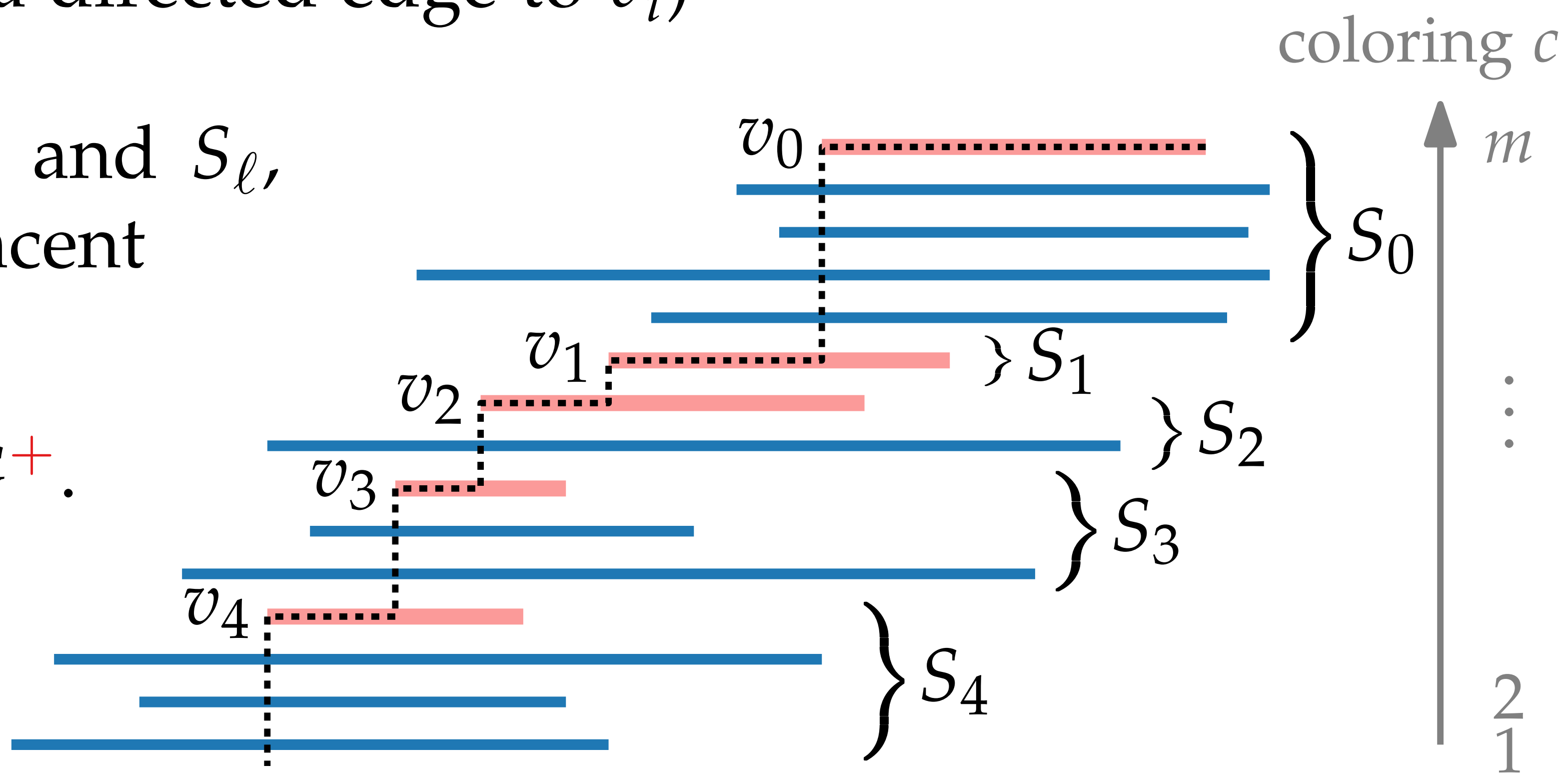
A coloring c computed by `GreedyColoring` has the minimum number of colors.

Proof sketch:

- Hence, for every *step* S_i, all intervals contain v_i.
  (otherwise they would have a directed edge to v_i)

- **Claim:** for any two steps S_i and S_ℓ, every pair of intervals is adjacent in the transitive closure G^+.

  $\Rightarrow$ $S = \bigcup S_i$ is a clique in G^+.



Coloring Directional Interval Graphs

Theorem 1:
A coloring c computed by `GreedyColoring` has the minimum number of colors.

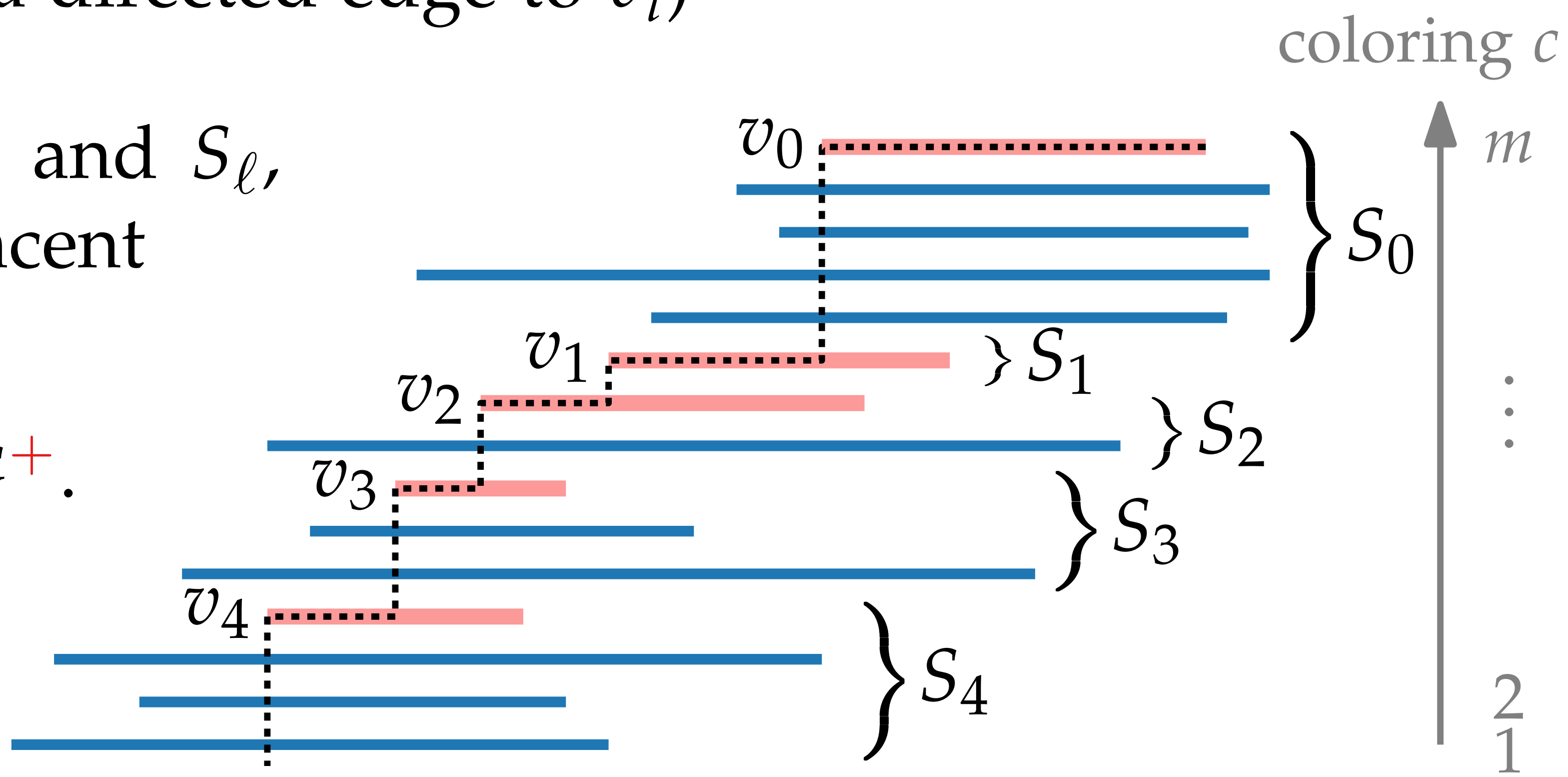
Proof sketch:

- Hence, for every *step* S_i, all intervals contain v_i.
  (otherwise they would have a directed edge to v_i)

- **Claim:** for any two steps S_i and S_ℓ, every pair of intervals is adjacent in the transitive closure G^+.

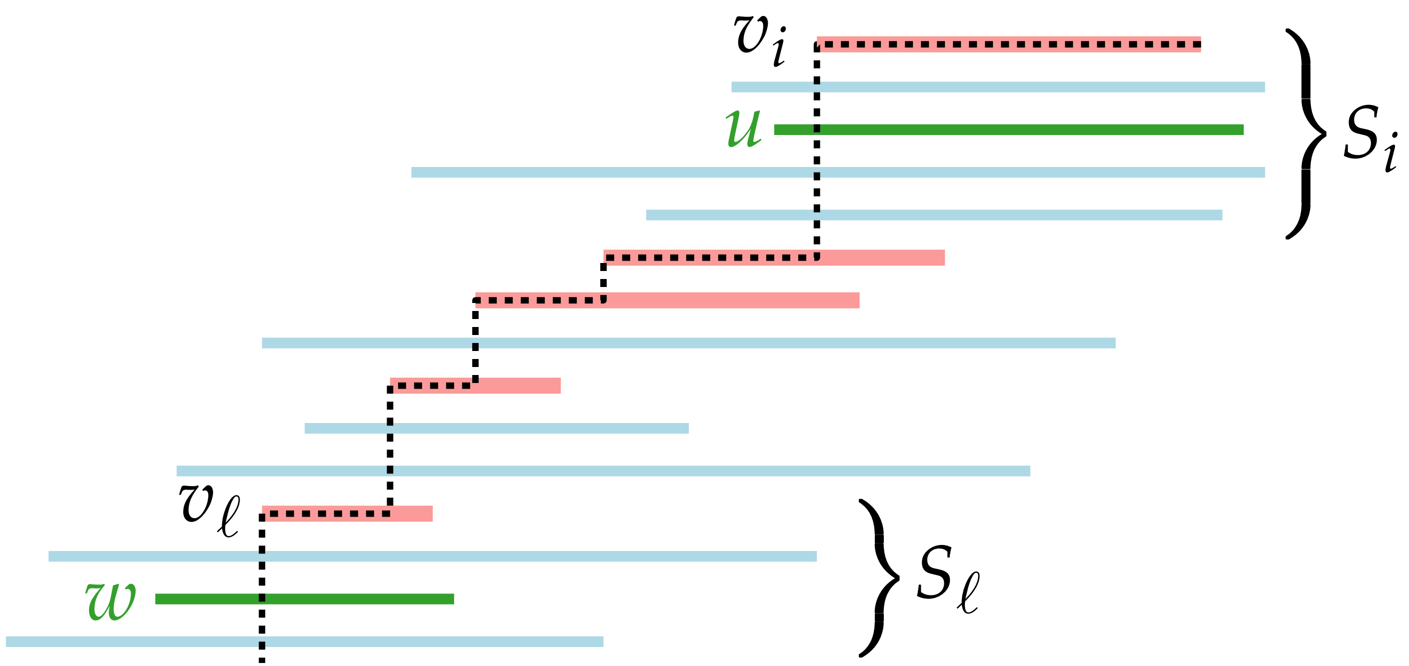
  $\Rightarrow$ $S = \bigcup S_i$ is a clique in G^+.

  $\Rightarrow$ S alone requires m colors in G. □



Proof of the Claim

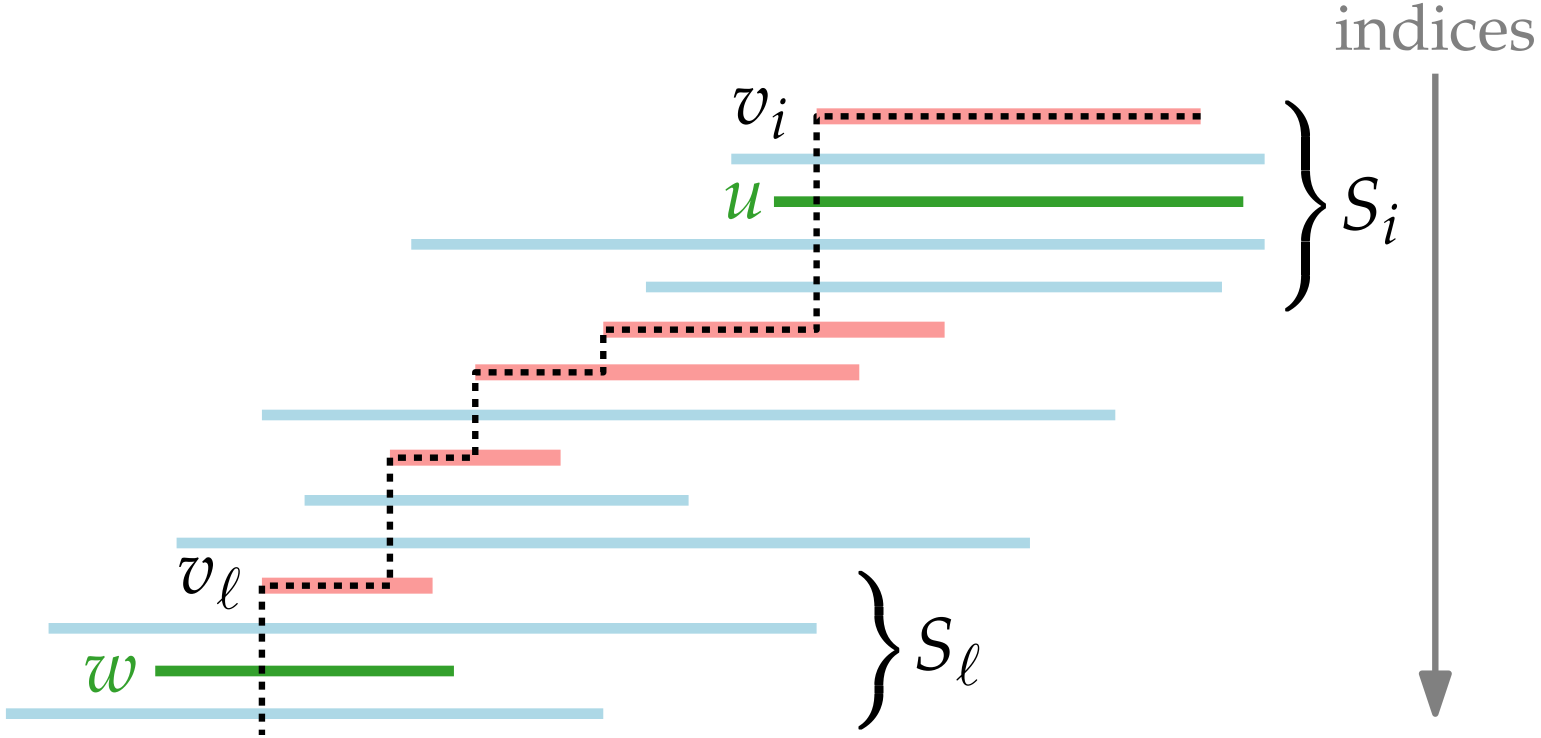
Claim: Any two intervals $u \in S_i$ and $w \in S_\ell$ are adjacent in G^+.



Proof of the Claim

Claim: Any two intervals $u \in S_i$ and $w \in S_\ell$ are adjacent in G^+.

Proof. W.l.o.g., $u \cap w = \emptyset$ and $i < \ell$.

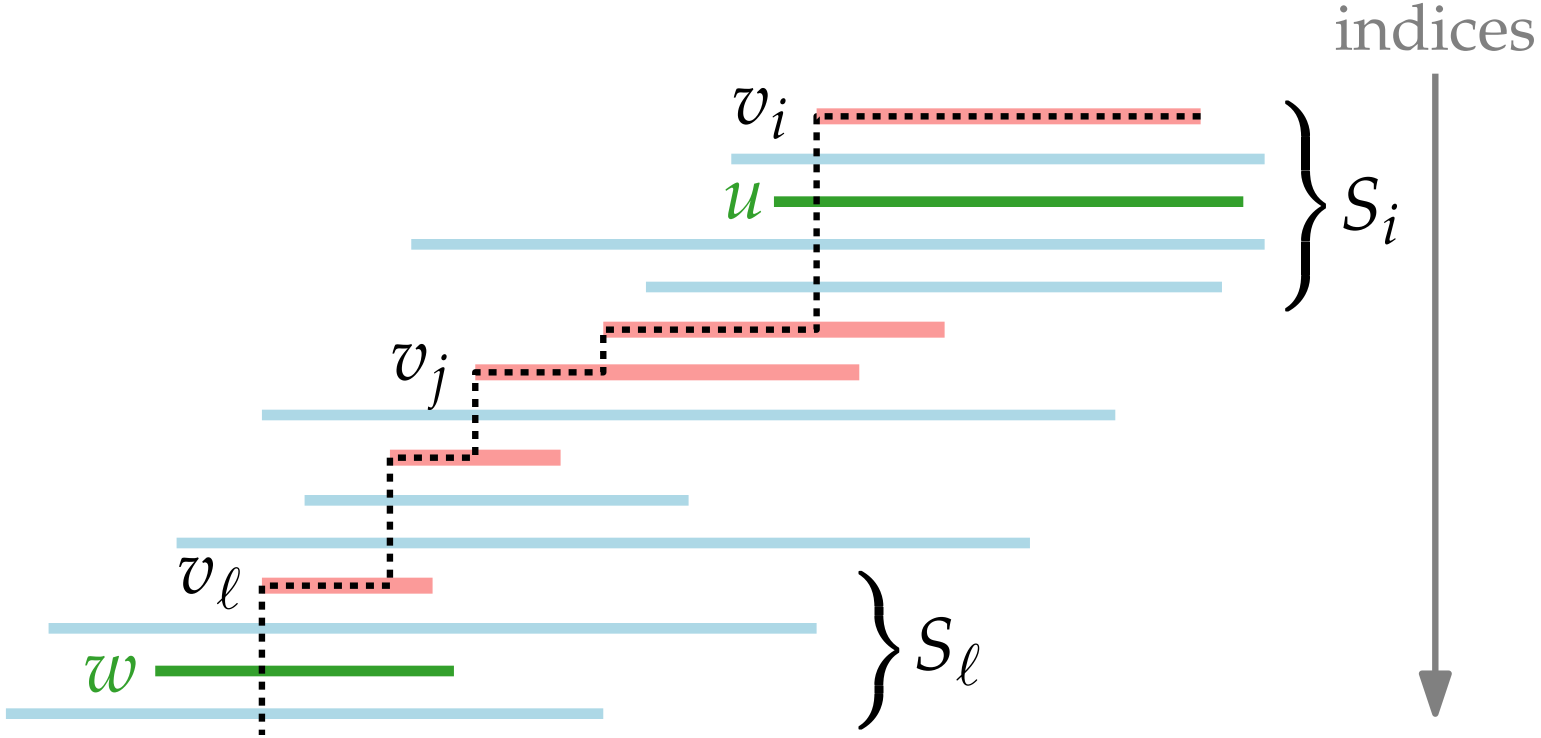


Proof of the Claim

Claim: Any two intervals $u \in S_i$ and $w \in S_\ell$ are adjacent in G^+.

Proof. W.l.o.g., $u \cap w = \emptyset$ and $i < \ell$.

Let j be the largest index s.t. $v_j \cap u \neq \emptyset$.

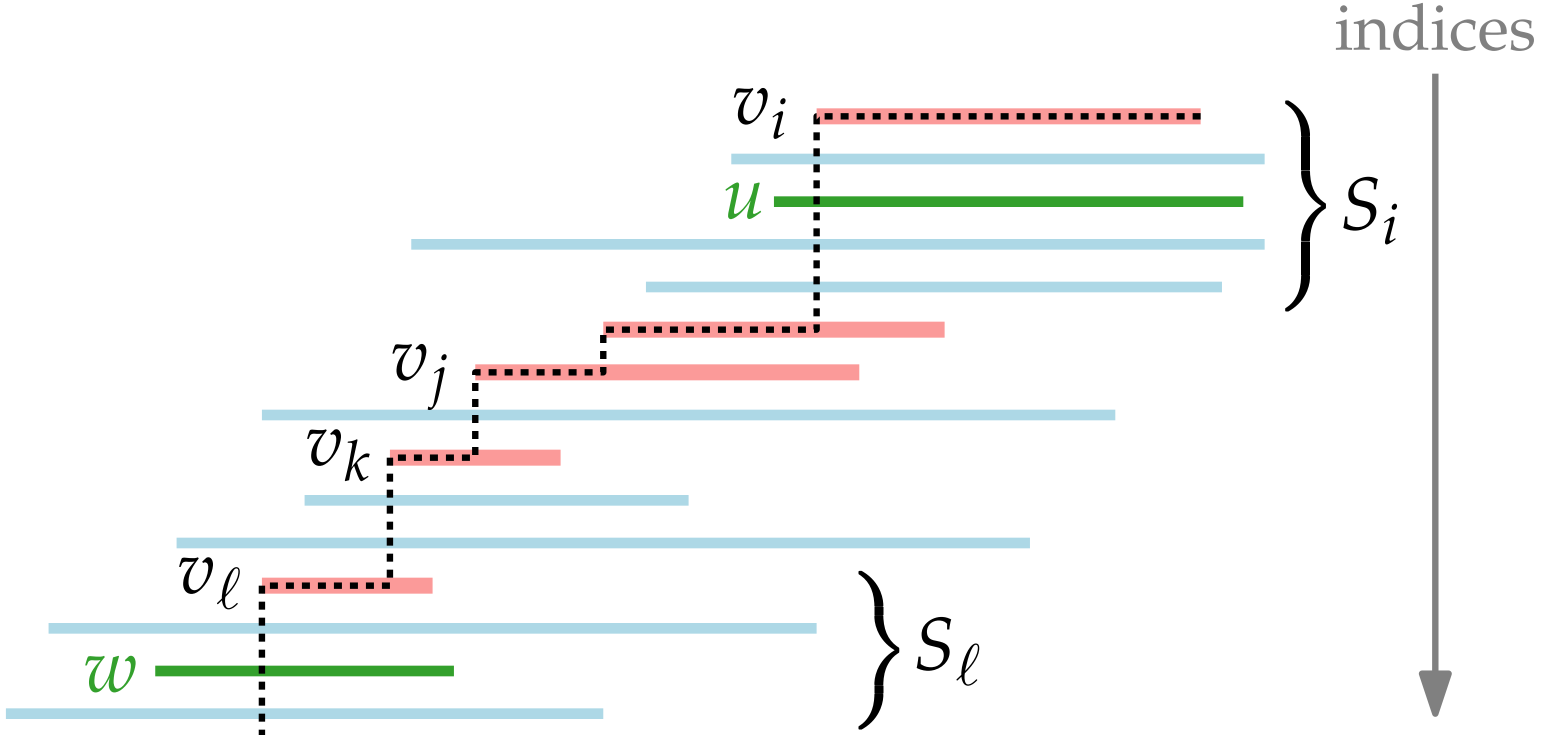


Proof of the Claim

Claim: Any two intervals $u \in S_i$ and $w \in S_\ell$ are adjacent in G^+.

Proof. W.l.o.g., $u \cap w = \emptyset$ and $i < \ell$.

Let j be the largest index s.t. $v_j \cap u \neq \emptyset$.
Let k be the smallest index s.t. $v_k \cap w \neq \emptyset$.



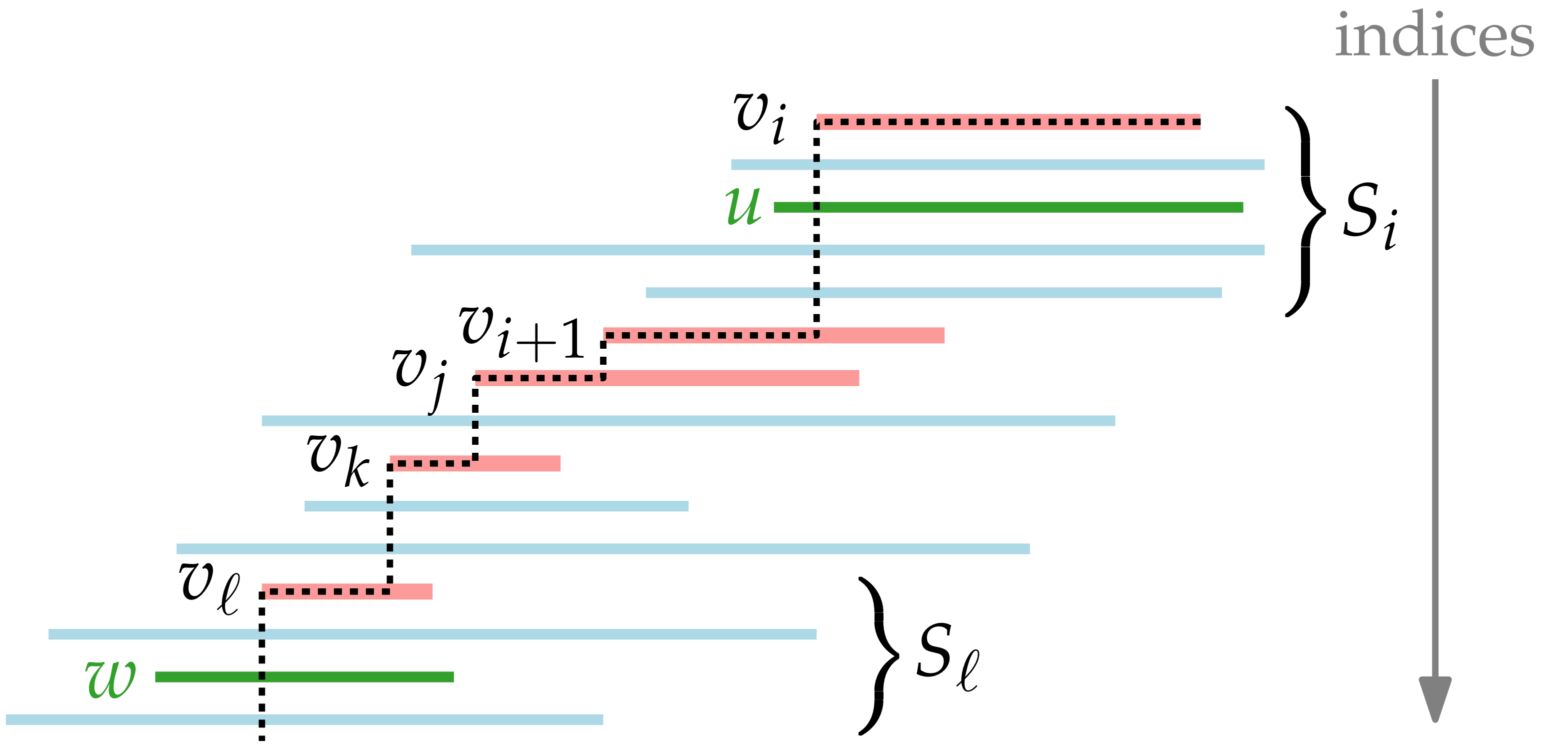
Proof of the Claim

Claim: Any two intervals $u \in S_i$ and $w \in S_\ell$ are adjacent in G^+.

Proof. W.l.o.g., $u \cap w = \emptyset$ and $i < \ell$.

Let j be the largest index s.t. $v_j \cap u \neq \emptyset$.
Let k be the smallest index s.t. $v_k \cap w \neq \emptyset$.

$u \cap v_{i+1} \neq \emptyset$



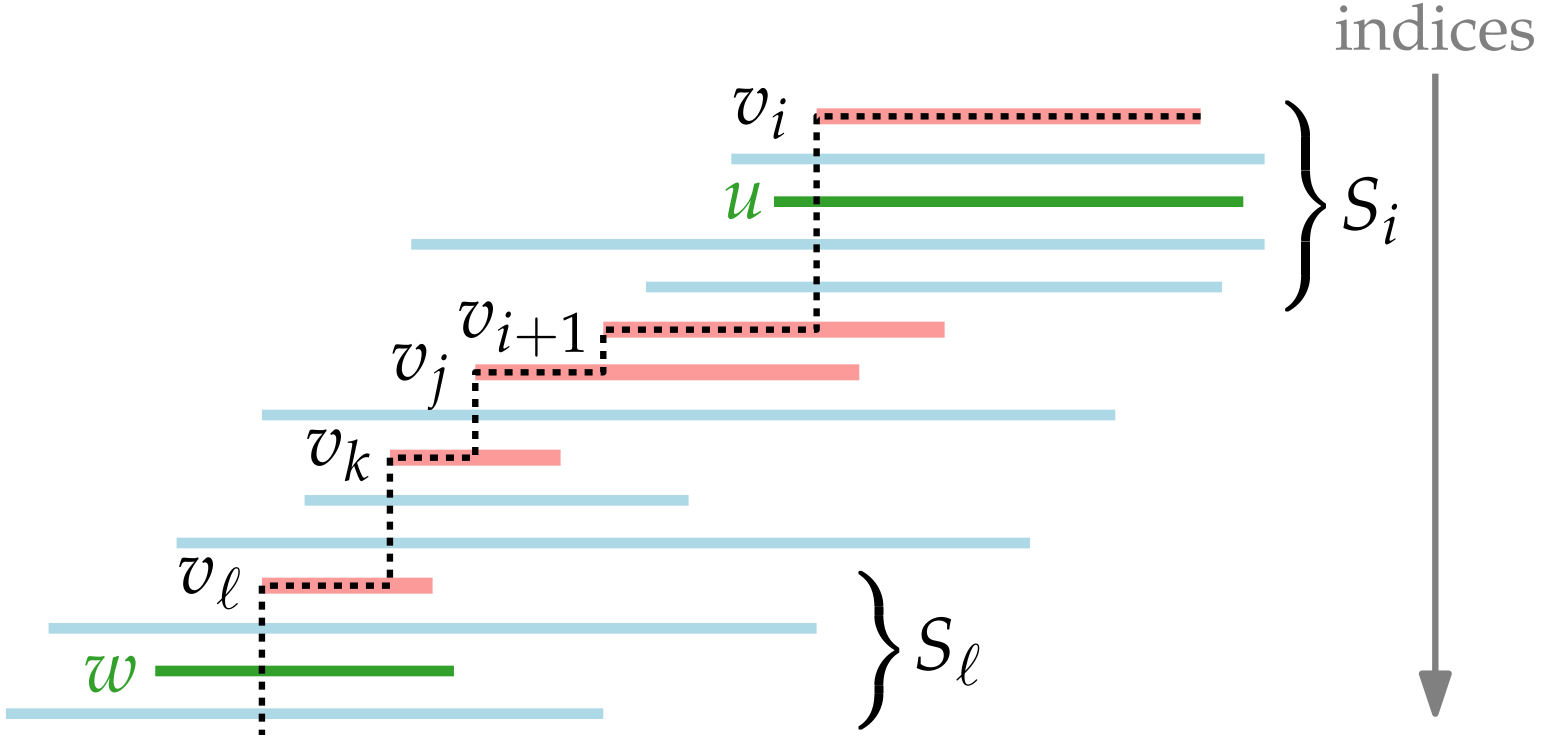
Proof of the Claim

Claim: Any two intervals $u \in S_i$ and $w \in S_\ell$ are adjacent in G^+.

Proof. W.l.o.g., $u \cap w = \emptyset$ and $i < \ell$.

Let j be the largest index s.t. $v_j \cap u \neq \emptyset$.
Let k be the smallest index s.t. $v_k \cap w \neq \emptyset$.

$u \cap v_{i+1} \neq \emptyset$
$w \cap v_{\ell-1} \neq \emptyset$



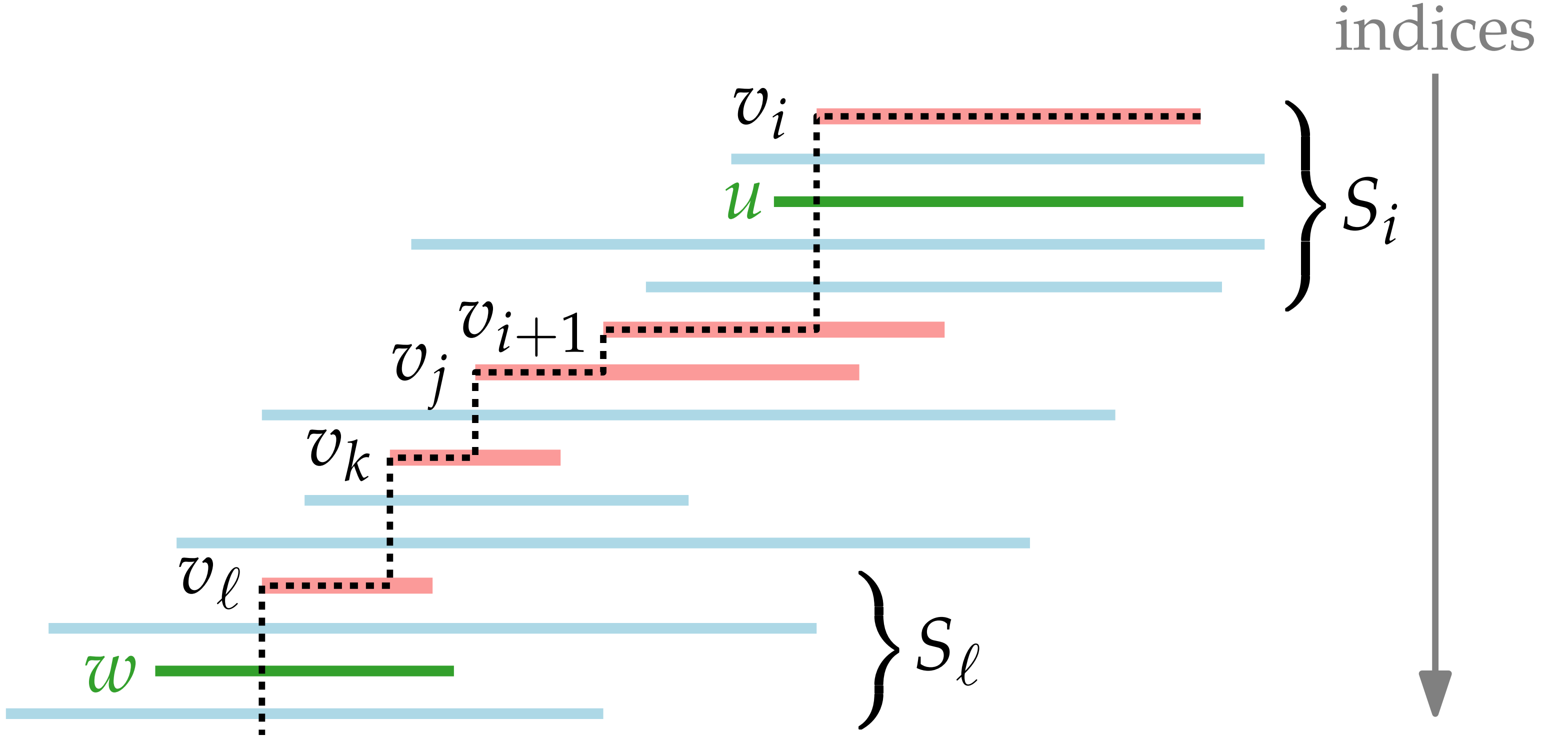
Proof of the Claim

Claim: Any two intervals $u \in S_i$ and $w \in S_\ell$ are adjacent in G^+.

Proof. W.l.o.g., $u \cap w = \emptyset$ and $i < \ell$.

Let j be the largest index s.t. $v_j \cap u \neq \emptyset$.
Let k be the smallest index s.t. $v_k \cap w \neq \emptyset$.

$$\begin{array}{l} u \cap v_{i+1} \neq \emptyset \\ w \cap v_{\ell-1} \neq \emptyset \end{array} \quad \underset{u \cap w = \emptyset}{\Longrightarrow}$$



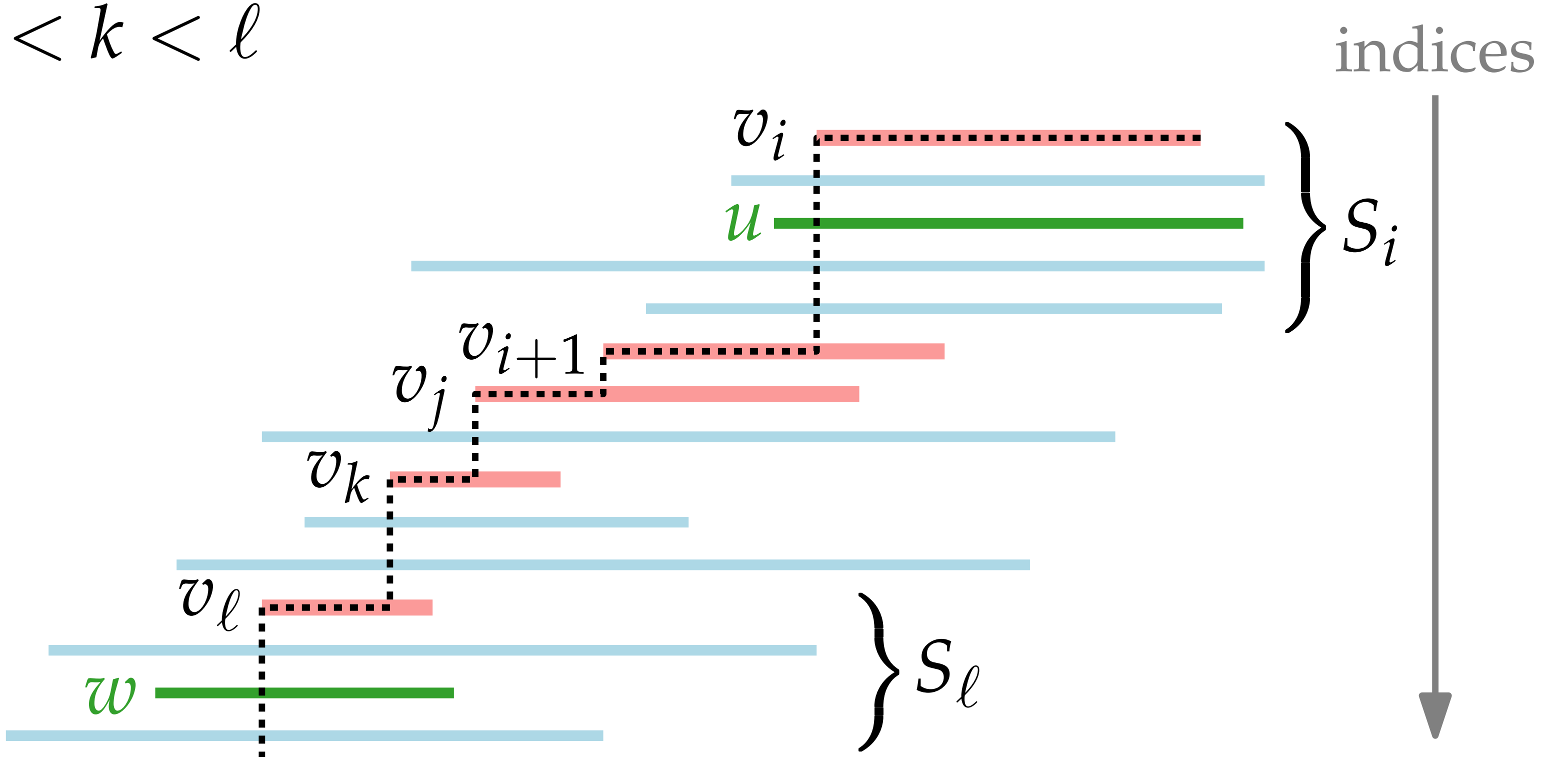
Proof of the Claim

Claim: Any two intervals $u \in S_i$ and $w \in S_\ell$ are adjacent in G^+.

Proof. W.l.o.g., $u \cap w = \emptyset$ and $i < \ell$.

Let j be the largest index s.t. $v_j \cap u \neq \emptyset$.
Let k be the smallest index s.t. $v_k \cap w \neq \emptyset$.

$$\begin{array}{c} u \cap v_{i+1} \neq \emptyset \\ w \cap v_{\ell-1} \neq \emptyset \end{array} \quad \underset{u \cap w = \emptyset}{\Longrightarrow} \quad \begin{array}{c} i < j < \ell \\ i < k < \ell \end{array}$$



Proof of the Claim

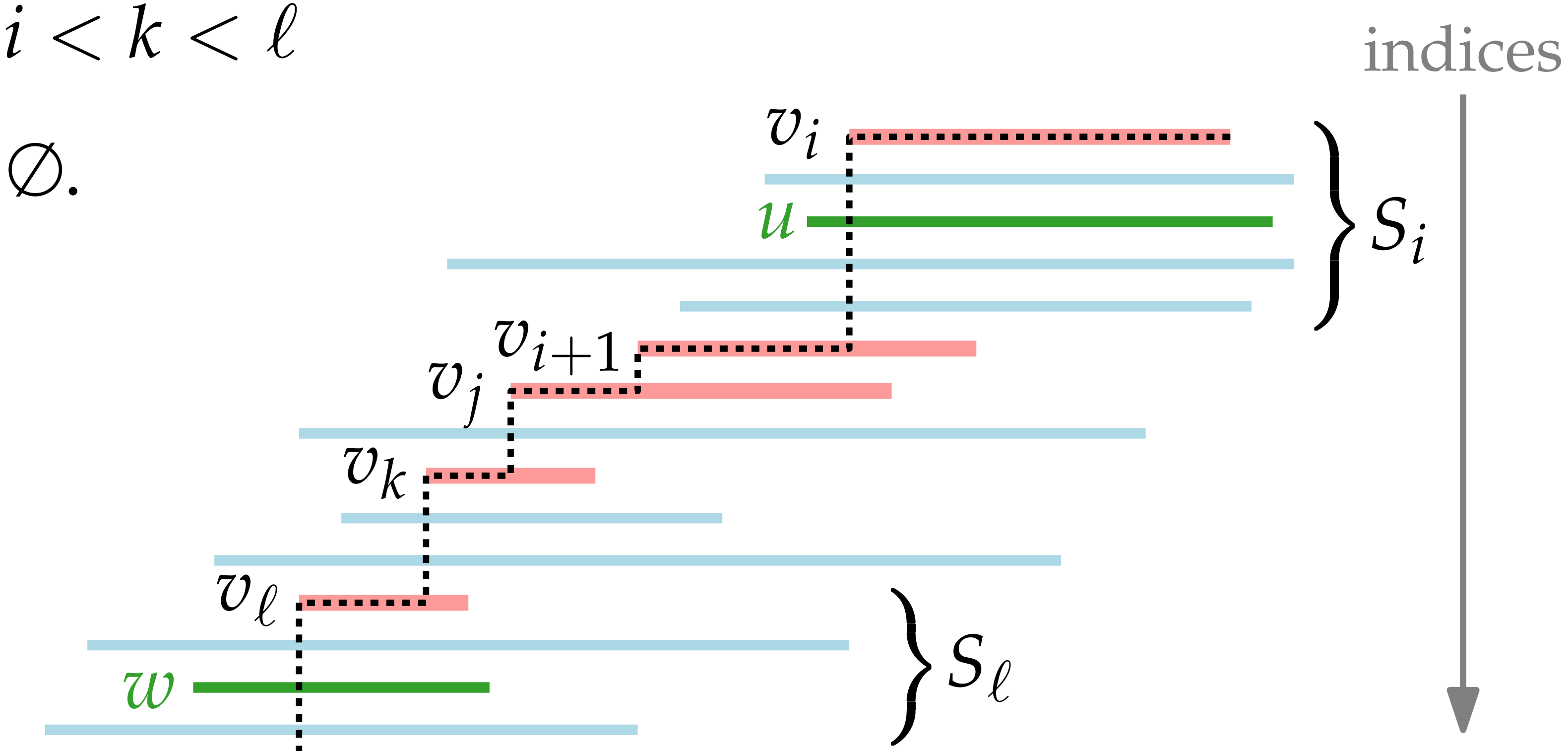
Claim: Any two intervals $u \in S_i$ and $w \in S_\ell$ are adjacent in G^+.

Proof. W.l.o.g., $u \cap w = \emptyset$ and $i < \ell$.

Let j be the largest index s.t. $v_j \cap u \neq \emptyset$.
Let k be the smallest index s.t. $v_k \cap w \neq \emptyset$.

$$\begin{matrix} u \cap v_{i+1} \neq \emptyset \\ w \cap v_{\ell-1} \neq \emptyset \end{matrix} \quad \underset{u \cap w = \emptyset}{\Longrightarrow} \quad \begin{matrix} i < j < \ell \\ i < k < \ell \end{matrix}$$

By definition, $u \cap v_{j+1} = \emptyset$.



Proof of the Claim

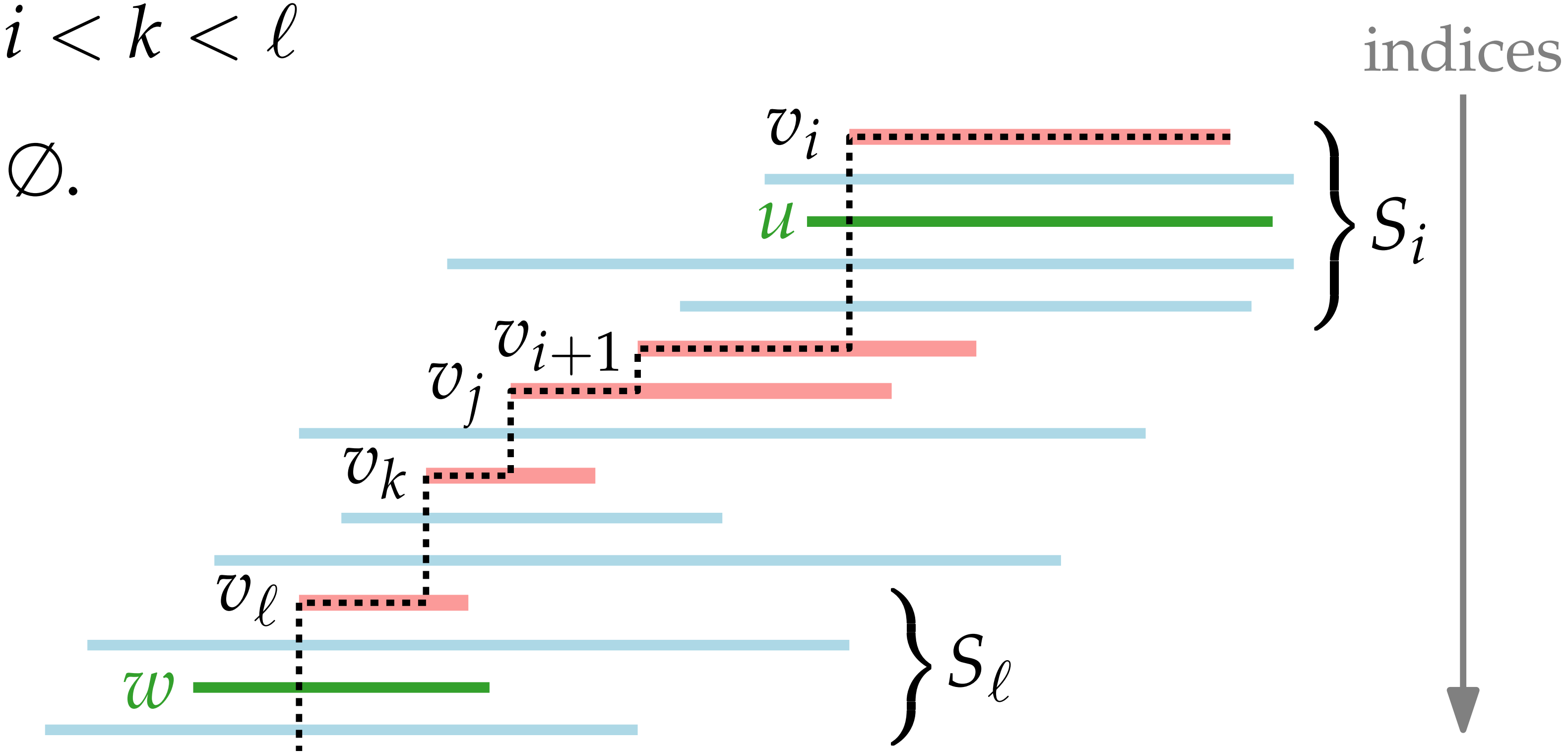
Claim: Any two intervals $u \in S_i$ and $w \in S_\ell$ are adjacent in G^+.

Proof. W.l.o.g., $u \cap w = \emptyset$ and $i < \ell$.

Let j be the largest index s.t. $v_j \cap u \neq \emptyset$.
Let k be the smallest index s.t. $v_k \cap w \neq \emptyset$.

$$\begin{matrix} u \cap v_{i+1} \neq \emptyset \\ w \cap v_{\ell-1} \neq \emptyset \end{matrix} \quad \underset{u \cap w = \emptyset}{\Longrightarrow} \quad \begin{matrix} i < j < \ell \\ i < k < \ell \end{matrix}$$

By definition, $u \cap v_{j+1} = \emptyset$.
$\Rightarrow$ u and v_j overlap



Proof of the Claim

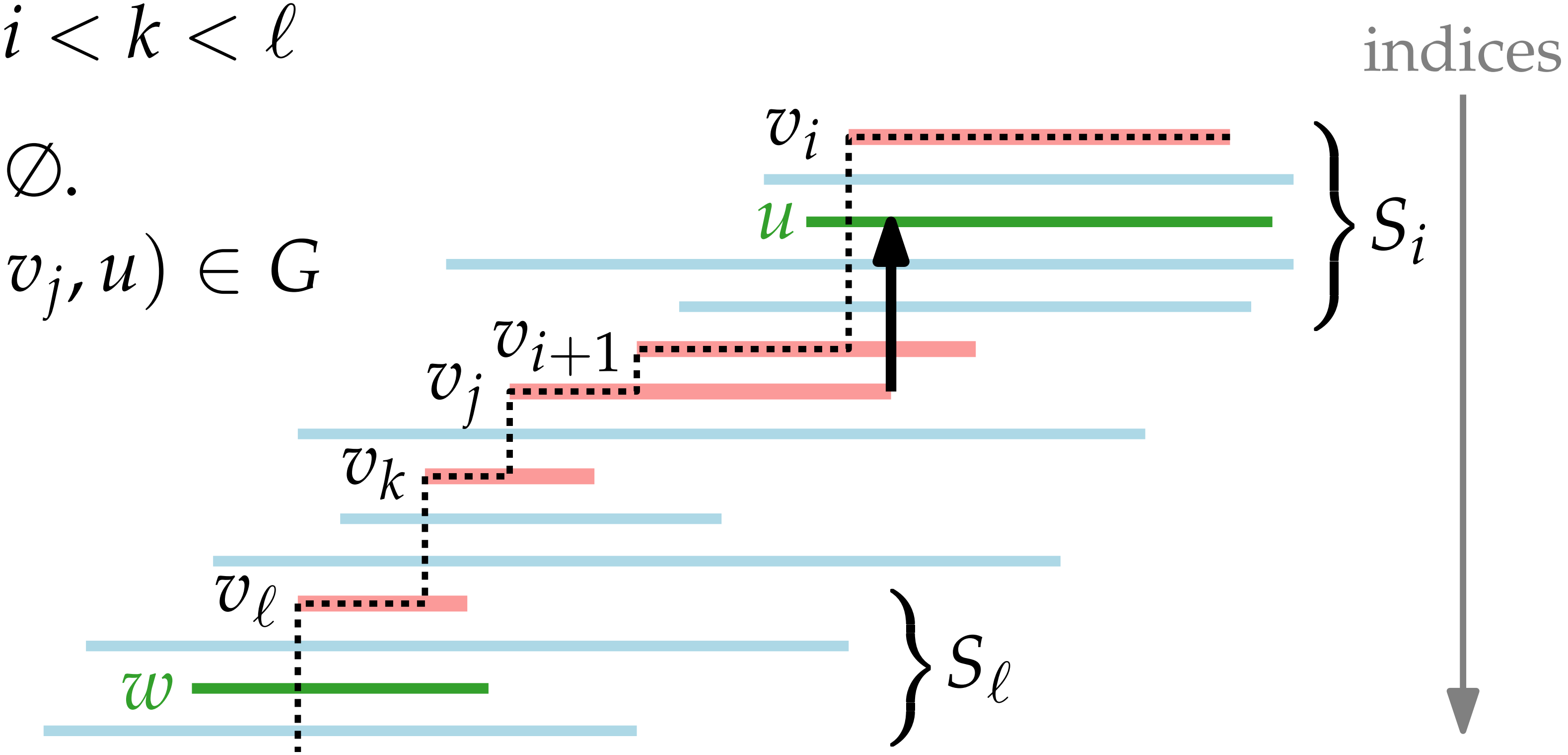
Claim: Any two intervals $u \in S_i$ and $w \in S_\ell$ are adjacent in G^+.

Proof. W.l.o.g., $u \cap w = \emptyset$ and $i < \ell$.

Let j be the largest index s.t. $v_j \cap u \neq \emptyset$.
Let k be the smallest index s.t. $v_k \cap w \neq \emptyset$.

$$\begin{matrix} u \cap v_{i+1} \neq \emptyset \\ w \cap v_{\ell-1} \neq \emptyset \end{matrix} \quad \underset{u \cap w = \emptyset}{\Longrightarrow} \quad \begin{matrix} i < j < \ell \\ i < k < \ell \end{matrix}$$

By definition, $u \cap v_{j+1} = \emptyset$.
$\Rightarrow$ u and v_j overlap $\Rightarrow (v_j, u) \in G$



Proof of the Claim

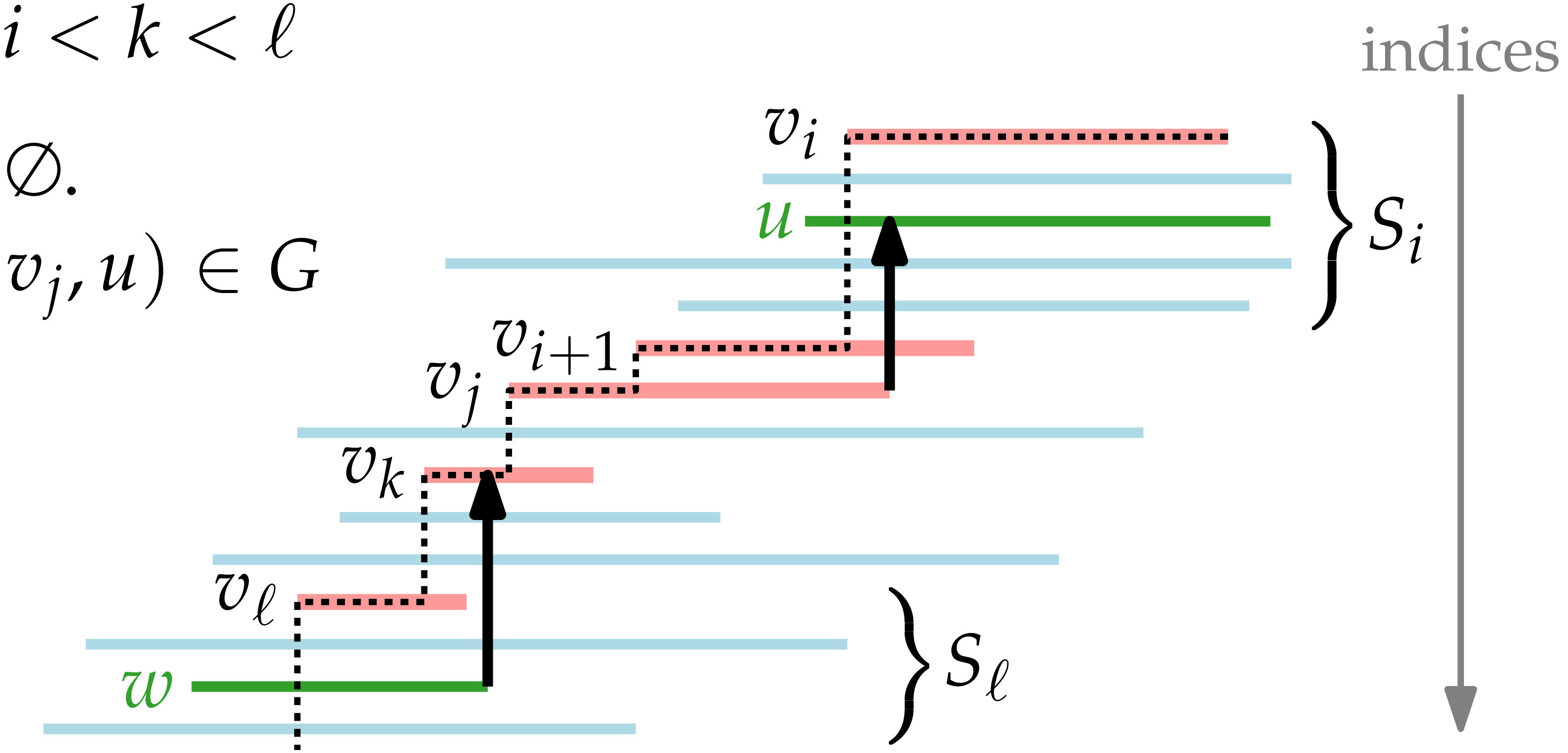
Claim: Any two intervals $u \in S_i$ and $w \in S_\ell$ are adjacent in G^+.

Proof. W.l.o.g., $u \cap w = \emptyset$ and $i < \ell$.

Let j be the largest index s.t. $v_j \cap u \neq \emptyset$.
Let k be the smallest index s.t. $v_k \cap w \neq \emptyset$.

$$\begin{array}{l} u \cap v_{i+1} \neq \emptyset \\ w \cap v_{\ell-1} \neq \emptyset \end{array} \quad \underset{u \cap w = \emptyset}{\Longrightarrow} \quad \begin{array}{l} i < j < \ell \\ i < k < \ell \end{array}$$

By definition, $u \cap v_{j+1} = \emptyset$.
$\Rightarrow$ u and v_j overlap $\Rightarrow (v_j, u) \in G$
Similarly, $(w, v_k) \in G$.



Proof of the Claim

Claim: Any two intervals $u \in S_i$ and $w \in S_\ell$ are adjacent in G^+.

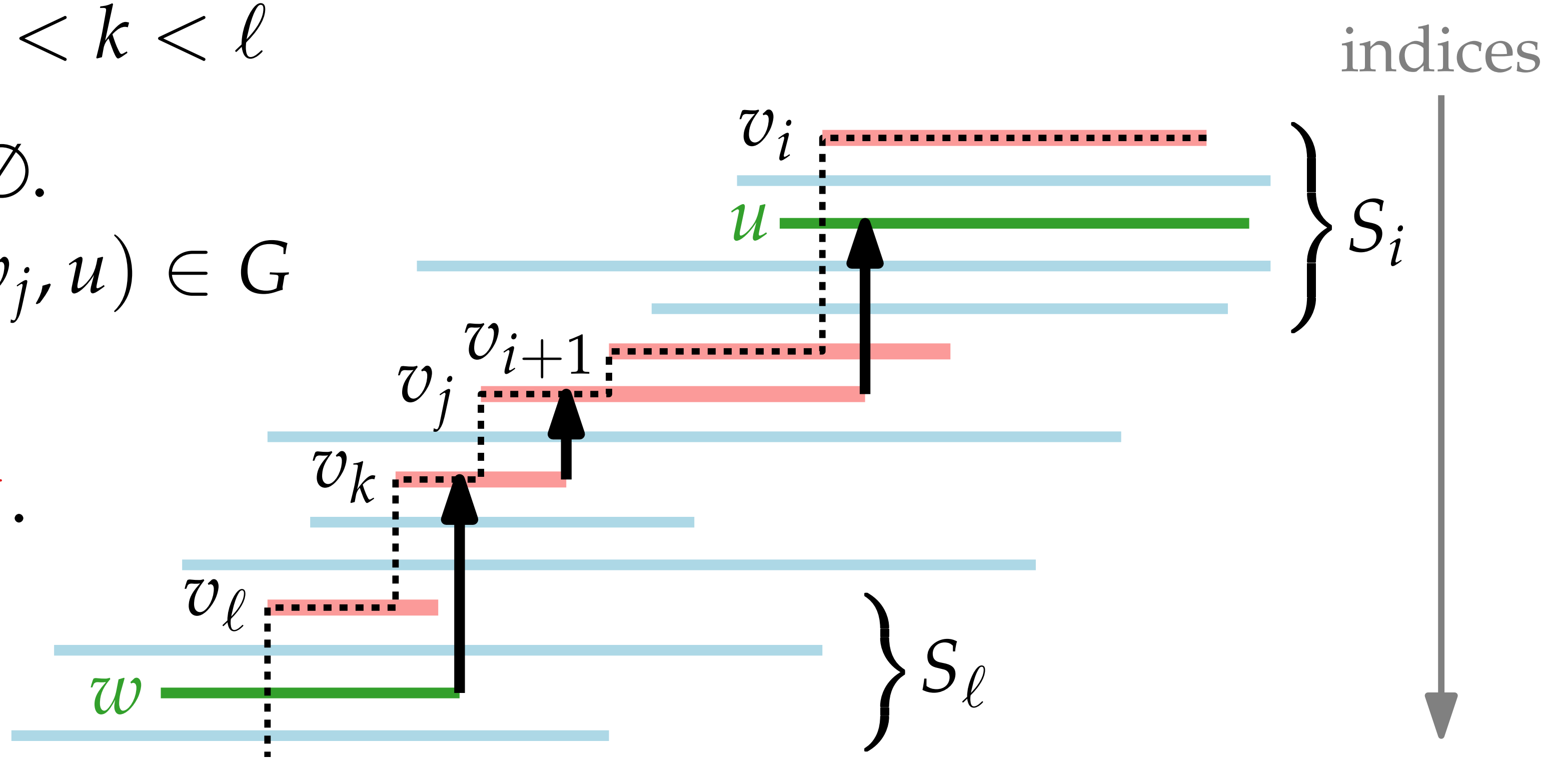
Proof. W.l.o.g., $u \cap w = \emptyset$ and $i < \ell$.

Let j be the largest index s.t. $v_j \cap u \neq \emptyset$.
Let k be the smallest index s.t. $v_k \cap w \neq \emptyset$.

$$\begin{array}{c} u \cap v_{i+1} \neq \emptyset \\ w \cap v_{\ell-1} \neq \emptyset \end{array} \quad \underset{u \cap w = \emptyset}{\Longrightarrow} \quad \begin{array}{c} i < j < \ell \\ i < k < \ell \end{array}$$

By definition, $u \cap v_{j+1} = \emptyset$.
$\Rightarrow$ u and v_j overlap $\Rightarrow (v_j, u) \in G$
Similarly, $(w, v_k) \in G$.

If $j < k$, then $(v_k, v_j) \in G^+$.



Proof of the Claim

Claim: Any two intervals $u \in S_i$ and $w \in S_\ell$ are adjacent in G^+.

Proof. W.l.o.g., $u \cap w = \emptyset$ and $i < \ell$.

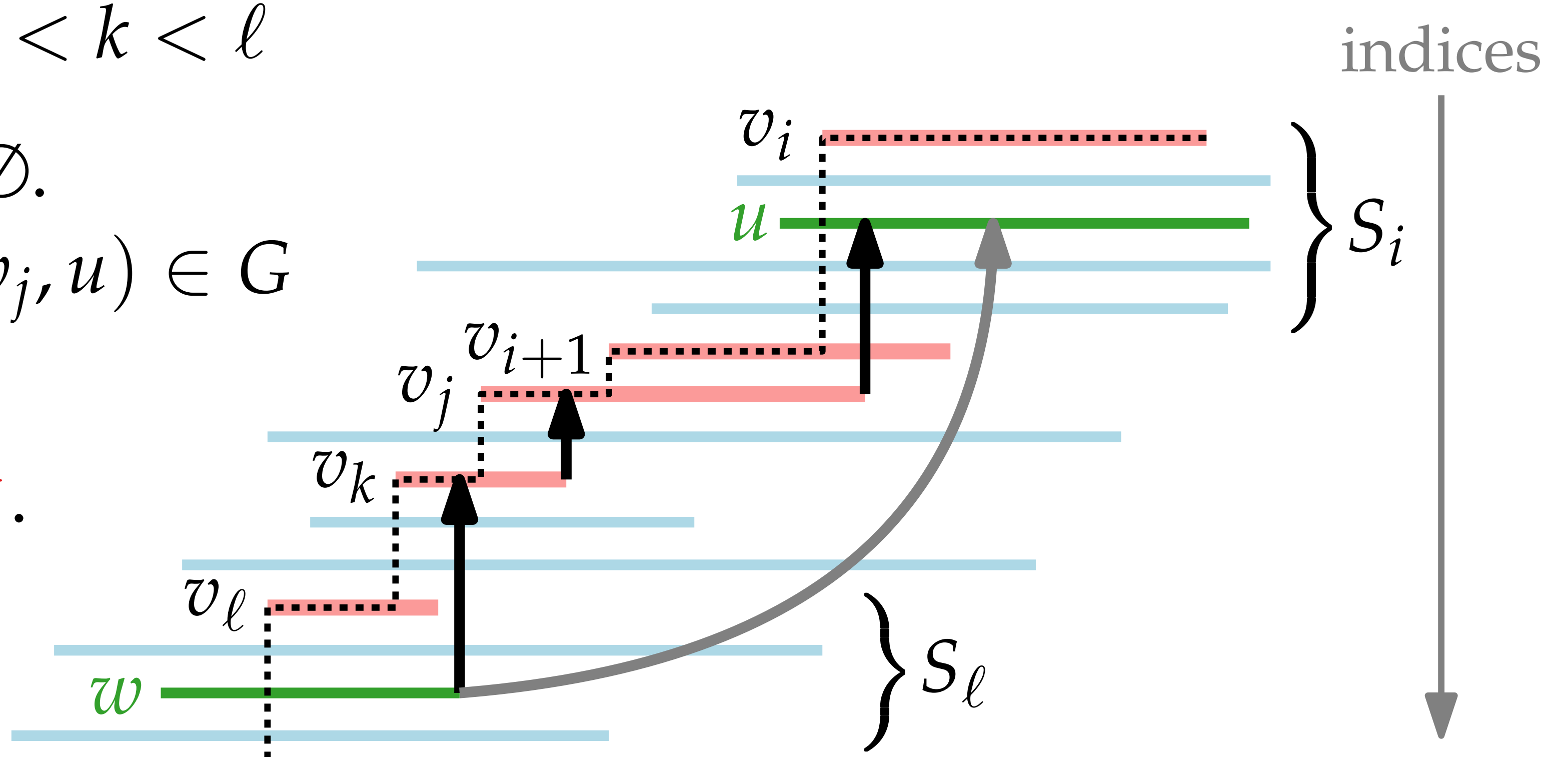
Let j be the largest index s.t. $v_j \cap u \neq \emptyset$.
Let k be the smallest index s.t. $v_k \cap w \neq \emptyset$.

$$\begin{array}{l} u \cap v_{i+1} \neq \emptyset \\ w \cap v_{\ell-1} \neq \emptyset \end{array} \quad \underset{u \cap w = \emptyset}{\Longrightarrow} \quad \begin{array}{l} i < j < \ell \\ i < k < \ell \end{array}$$

By definition, $u \cap v_{j+1} = \emptyset$.
$\Rightarrow$ u and v_j overlap $\Rightarrow (v_j, u) \in G$
Similarly, $(w, v_k) \in G$.

If $j < k$, then $(v_k, v_j) \in G^+$.

Transitivity $\Rightarrow$ claim.



Proof of the Claim

Claim: Any two intervals $u \in S_i$ and $w \in S_\ell$ are adjacent in G^+.

Proof. W.l.o.g., $u \cap w = \emptyset$ and $i < \ell$.

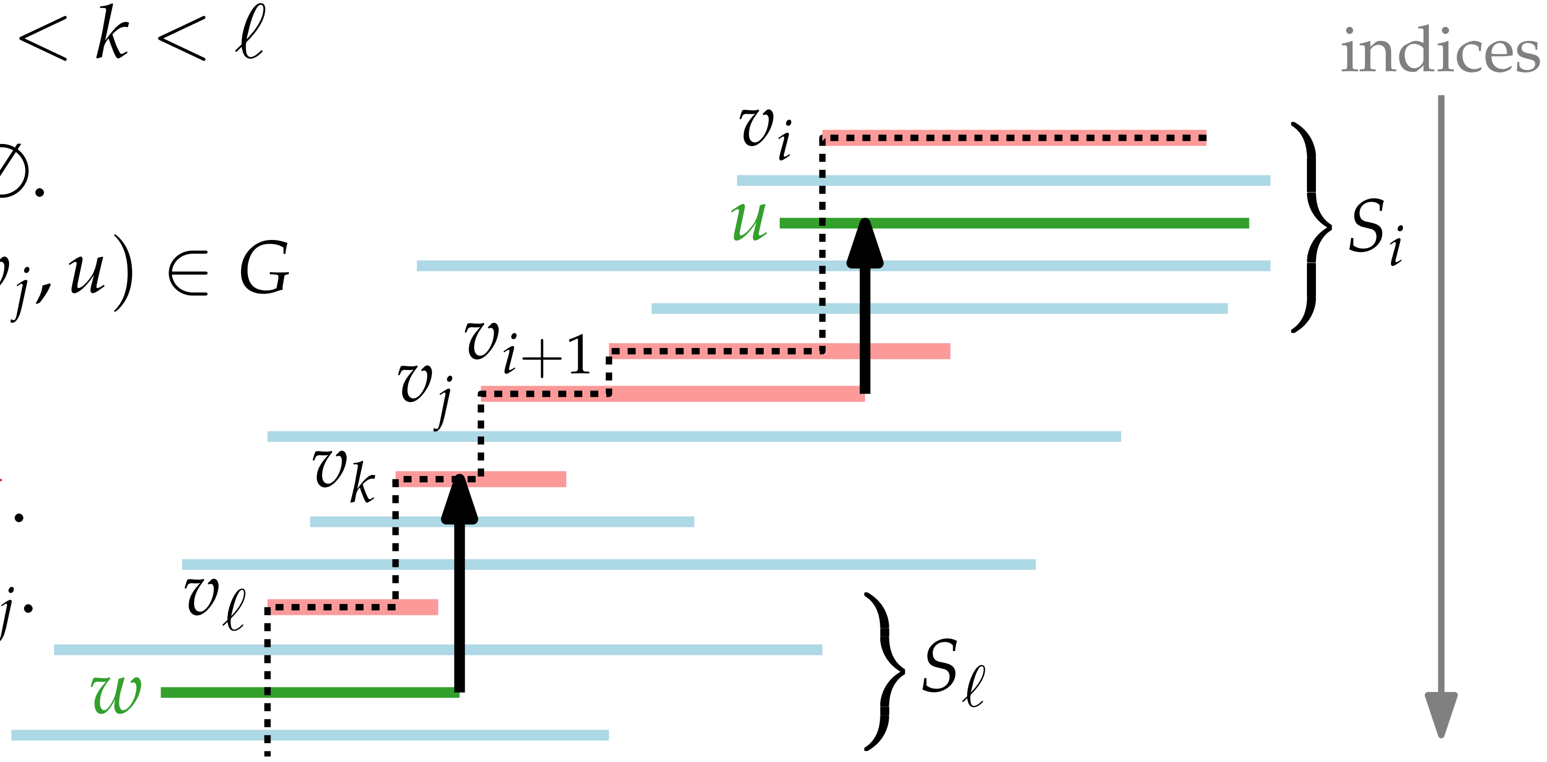
Let j be the largest index s.t. $v_j \cap u \neq \emptyset$.
Let k be the smallest index s.t. $v_k \cap w \neq \emptyset$.

$$\begin{array}{l} u \cap v_{i+1} \neq \emptyset \\ w \cap v_{\ell-1} \neq \emptyset \end{array} \quad \underset{u \cap w = \emptyset}{\Longrightarrow} \quad \begin{array}{l} i < j < \ell \\ i < k < \ell \end{array}$$

By definition, $u \cap v_{j+1} = \emptyset$.
$\Rightarrow$ u and v_j overlap $\Rightarrow (v_j, u) \in G$
Similarly, $(w, v_k) \in G$.

If $j < k$, then $(v_k, v_j) \in G^+$.
If $j \geq k$, then w overlaps v_j.

Transitivity $\Rightarrow$ claim.



Proof of the Claim

Claim: Any two intervals $u \in S_i$ and $w \in S_\ell$ are adjacent in G^+.

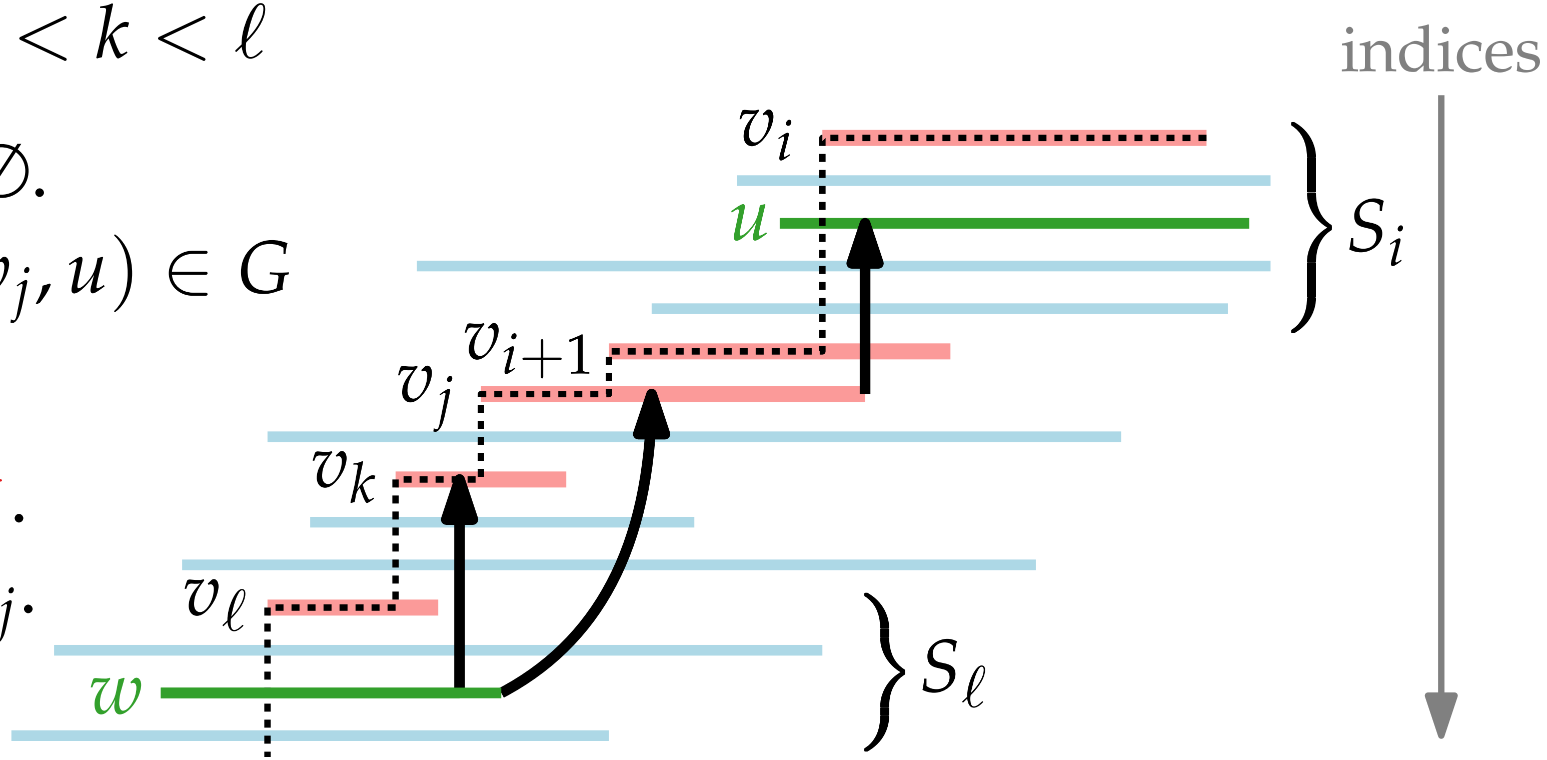
Proof. W.l.o.g., $u \cap w = \emptyset$ and $i < \ell$.
Let j be the largest index s.t. $v_j \cap u \neq \emptyset$.
Let k be the smallest index s.t. $v_k \cap w \neq \emptyset$.

$$\begin{array}{l} u \cap v_{i+1} \neq \emptyset \\ w \cap v_{\ell-1} \neq \emptyset \end{array} \underset{u \cap w = \emptyset}{\Longrightarrow} \begin{array}{l} i < j < \ell \\ i < k < \ell \end{array}$$

By definition, $u \cap v_{j+1} = \emptyset$.
$\Rightarrow$ u and v_j overlap $\Rightarrow$ $(v_j, u) \in G$
Similarly, $(w, v_k) \in G$.

If $j < k$, then $(v_k, v_j) \in G^+$.
If $j \geq k$, then w overlaps v_j.

Transitivity $\Rightarrow$ claim.



Proof of the Claim

Claim: Any two intervals $u \in S_i$ and $w \in S_\ell$ are adjacent in G^+.

Proof. W.l.o.g., $u \cap w = \emptyset$ and $i < \ell$.

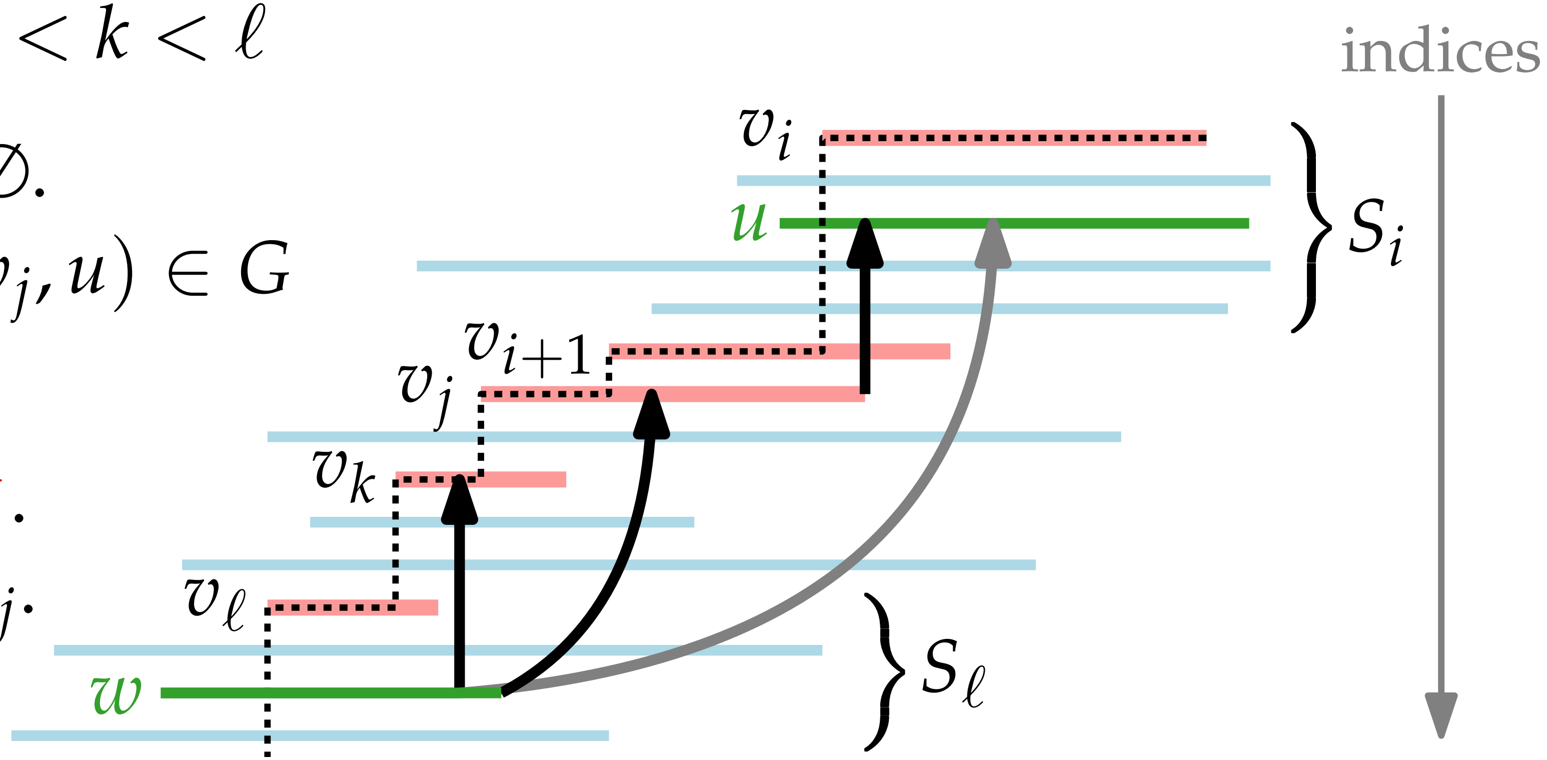
Let j be the largest index s.t. $v_j \cap u \neq \emptyset$.
Let k be the smallest index s.t. $v_k \cap w \neq \emptyset$.

$$\begin{matrix} u \cap v_{i+1} \neq \emptyset \\ w \cap v_{\ell-1} \neq \emptyset \end{matrix} \quad \underset{u \cap w = \emptyset}{\Longrightarrow} \quad \begin{matrix} i < j < \ell \\ i < k < \ell \end{matrix}$$

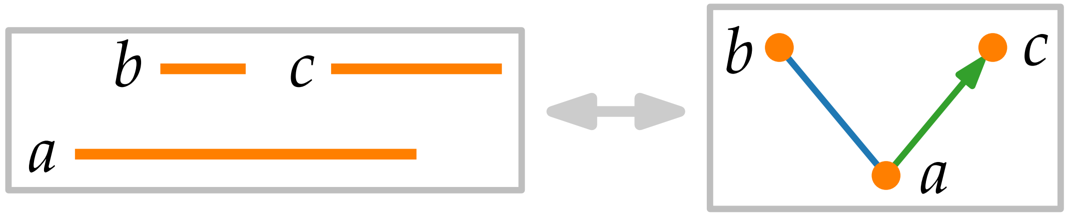
By definition, $u \cap v_{j+1} = \emptyset$.
$\Rightarrow$ u and v_j overlap $\Rightarrow (v_j, u) \in G$
Similarly, $(w, v_k) \in G$.

If $j < k$, then $(v_k, v_j) \in G^+$.
If $j \geq k$, then w overlaps v_j.

Transitivity $\Rightarrow$ claim.

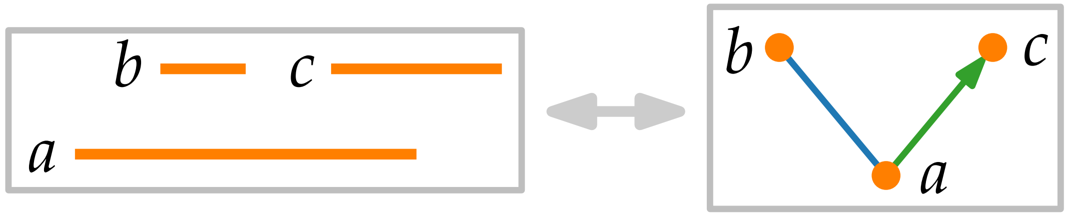


Conclusion and Open Problems



- We have introduced the natural concept of directional interval graphs.

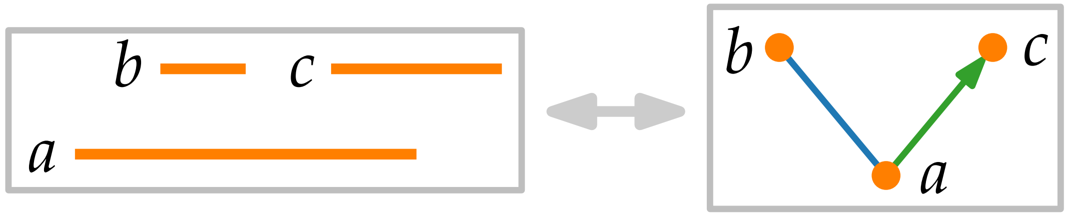
Conclusion and Open Problems



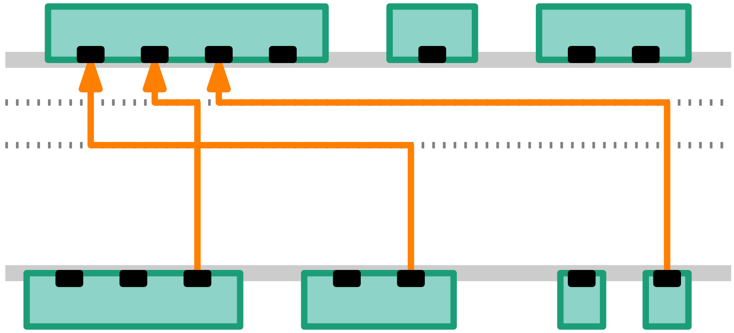
- We have introduced the natural concept of directional interval graphs.
- A simple greedy algorithm colors these graphs optimally in $O(n \log n)$ time.

  $n :=$ # vertices

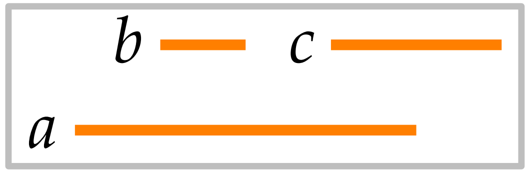
Conclusion and Open Problems

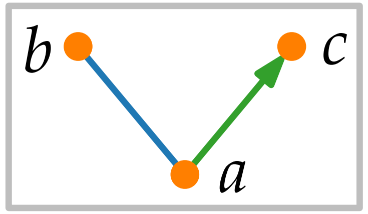


- We have introduced the natural concept of directional interval graphs.
- A simple greedy algorithm colors these graphs optimally in $O(n \log n)$ time.
  $n :=$ # vertices
- In layered graph drawing, this corresponds to routing "left-going" edges orthogonally to the fewest horizontal tracks. (Symmetrically "right-going".)



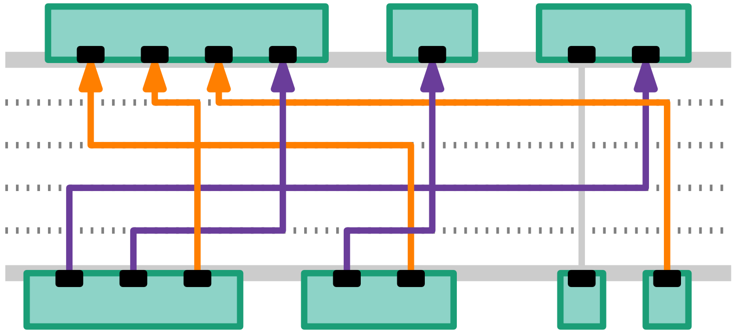
Conclusion and Open Problems



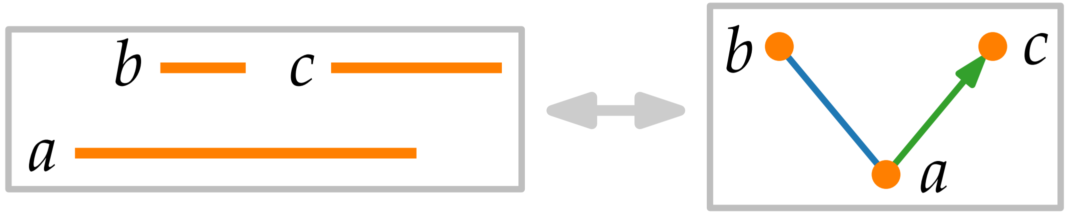


- We have introduced the natural concept of directional interval graphs.
- A simple greedy algorithm colors these graphs optimally in $O(n \log n)$ time.
  $n :=$ # vertices
- In layered graph drawing, this corresponds to routing "left-going" edges orthogonally to the fewest horizontal tracks. (Symmetrically "right-going".)

⇒ Combining the drawings of left-going and right-going edges yields a 2-approximation for the number of tracks. (bidirectional interval graphs)



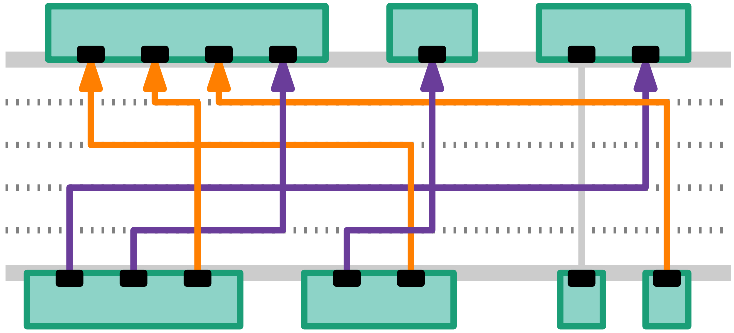
Conclusion and Open Problems



- We have introduced the natural concept of directional interval graphs.
- A simple greedy algorithm colors these graphs optimally in $O(n \log n)$ time.

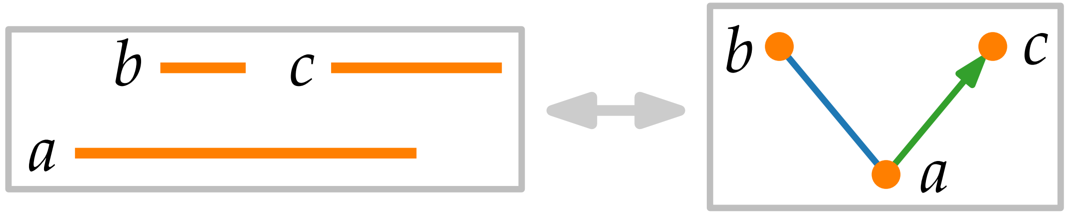
  $n :=$ # vertices
- In layered graph drawing, this corresponds to routing “left-going” edges orthogonally to the fewest horizontal tracks. (Symmetrically “right-going”.)

⇒ Combining the drawings of left-going and right-going edges yields a 2-approximation for the number of tracks. (bidirectional interval graphs)



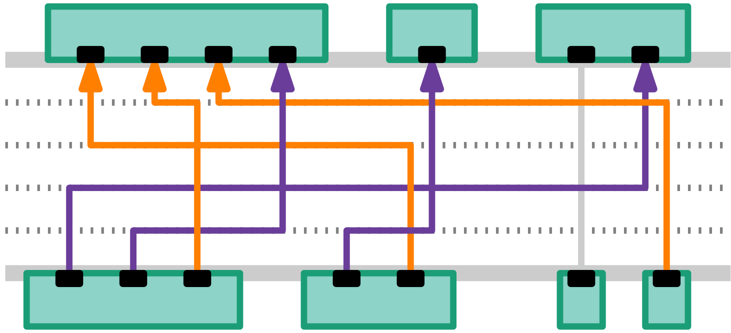
- In our paper, we present a constructive $O(n^2)$-time algorithm for recognizing directional interval graphs, which is based on PQ-trees.

Conclusion and Open Problems



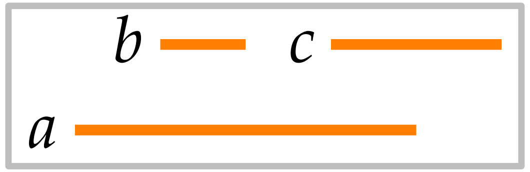
- We have introduced the natural concept of directional interval graphs.
- A simple greedy algorithm colors these graphs optimally in $O(n \log n)$ time.
  $n :=$ # vertices
- In layered graph drawing, this corresponds to routing "left-going" edges orthogonally to the fewest horizontal tracks. (Symmetrically "right-going".)

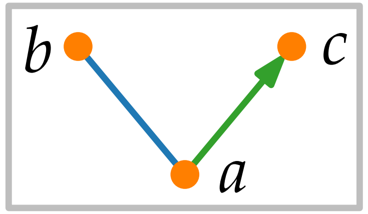
⇒ Combining the drawings of left-going and right-going edges yields a 2-approximation for the number of tracks. (bidirectional interval graphs)



- In our paper, we present a constructive $O(n^2)$-time algorithm for recognizing directional interval graphs, which is based on PQ-trees.
- For the more general case of mixed interval graphs, coloring is NP-hard.
  (Remark: NP-hardness requires both directed and undirected edges.)

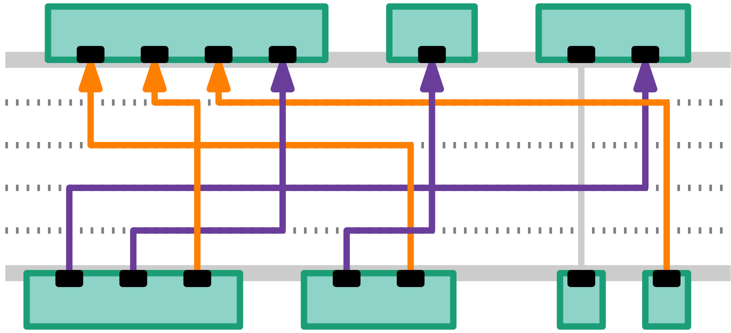
Conclusion and Open Problems





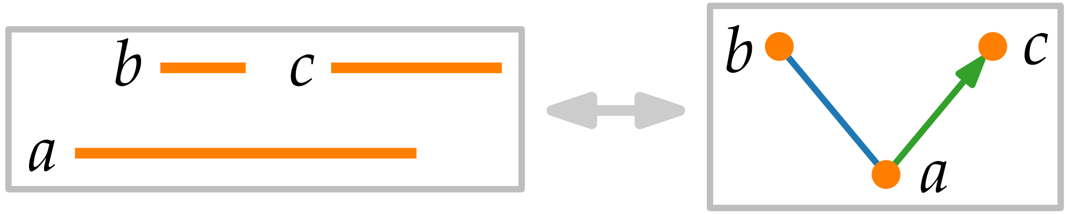
- We have introduced the natural concept of directional interval graphs.
- A simple greedy algorithm colors these graphs optimally in $O(n \log n)$ time. $n :=$ # vertices
- In layered graph drawing, this corresponds to routing "left-going" edges orthogonally to the fewest horizontal tracks. (Symmetrically "right-going".)

⇒ Combining the drawings of left-going and right-going edges yields a 2-approximation for the number of tracks. (bidirectional interval graphs)



- In our paper, we present a constructive $O(n^2)$-time algorithm for recognizing directional interval graphs, which is based on PQ-trees.
- For the more general case of mixed interval graphs, coloring is NP-hard. (Remark: NP-hardness requires both directed and undirected edges.)

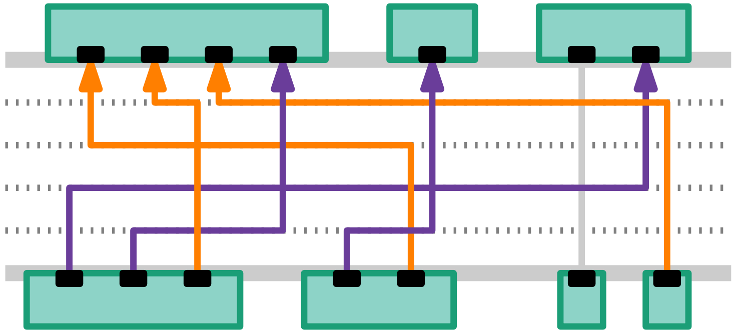
Conclusion and Open Problems



- We have introduced the natural concept of directional interval graphs.
- A simple greedy algorithm colors these graphs optimally in $O(n \log n)$ time.

  $n :=$ # vertices
- In layered graph drawing, this corresponds to routing "left-going" edges orthogonally to the fewest horizontal tracks. (Symmetrically "right-going".)

⇒ Combining the drawings of left-going and right-going edges yields a 2-approximation for the number of tracks. (bidirectional interval graphs)

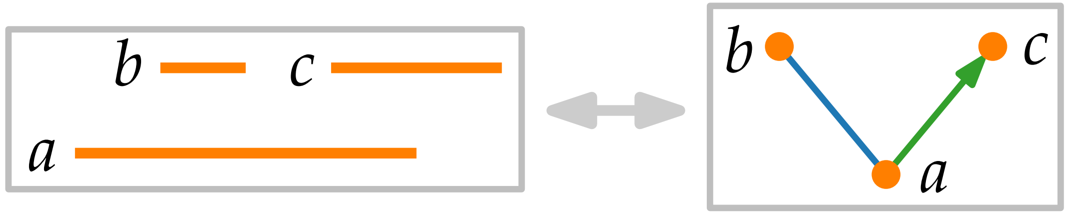


can we do better?

- In our paper, we present a constructive $O(n^2)$-time algorithm for recognizing directional interval graphs, which is based on PQ-trees.
- For the more general case of mixed interval graphs, coloring is NP-hard.

  (Remark: NP-hardness requires both directed and undirected edges.)

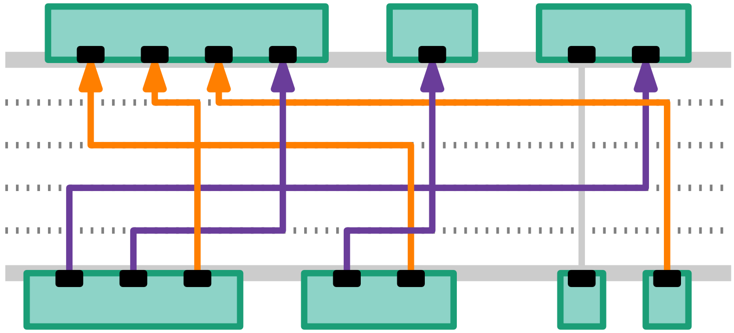
Conclusion and Open Problems



- We have introduced the natural concept of directional interval graphs.
- A simple greedy algorithm colors these graphs optimally in $O(n \log n)$ time.

  $n :=$ # vertices
- In layered graph drawing, this corresponds to routing "left-going" edges orthogonally to the fewest horizontal tracks. (Symmetrically "right-going".)

⇒ Combining the drawings of left-going and right-going edges yields a 2-approximation for the number of tracks. (bidirectional interval graphs)



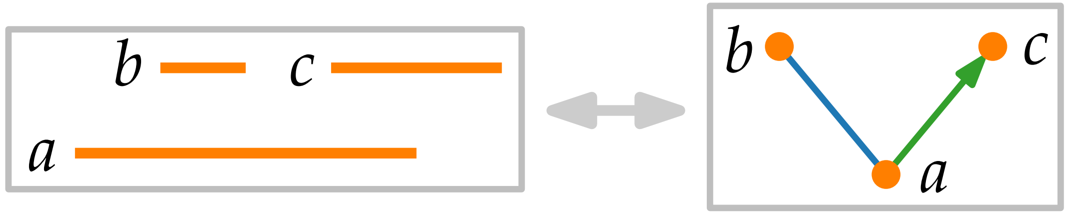
can we do better?

- In our paper, we present a constructive $O(n^2)$-time algorithm for recognizing directional interval graphs, which is based on PQ-trees.

  bidirectional?
- For the more general case of mixed interval graphs, coloring is NP-hard.

  (Remark: NP-hardness requires both directed and undirected edges.)

Conclusion and Open Problems



- We have introduced the natural concept of directional interval graphs.

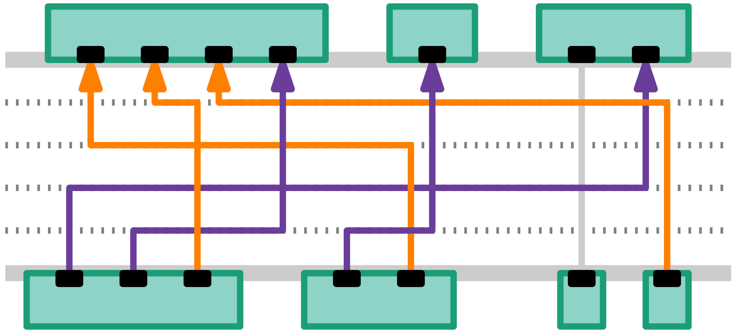
???

- A simple greedy algorithm colors these graphs optimally in $O(n \log n)$ time.

$n :=$ # vertices

- In layered graph drawing, this corresponds to routing "left-going" edges orthogonally to the fewest horizontal tracks. (Symmetrically "right-going".)

⇒ Combining the drawings of left-going and right-going edges yields a 2-approximation for the number of tracks. (bidirectional interval graphs)



can we do better?

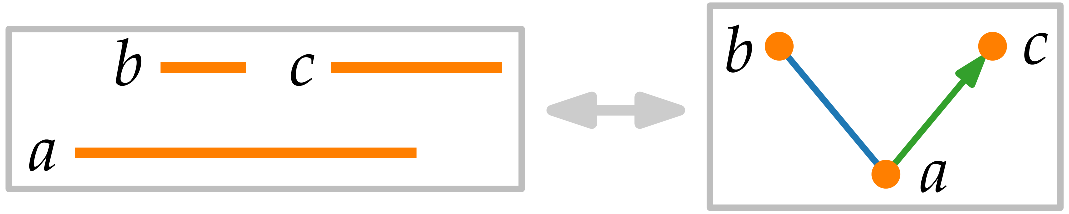
- In our paper, we present a constructive $O(n^2)$-time algorithm for recognizing directional interval graphs, which is based on PQ-trees.

bidirectional?

- For the more general case of mixed interval graphs, coloring is NP-hard.

(Remark: NP-hardness requires both directed and undirected edges.)

Conclusion and Open Problems



- We have introduced the natural concept of directional interval graphs.

???

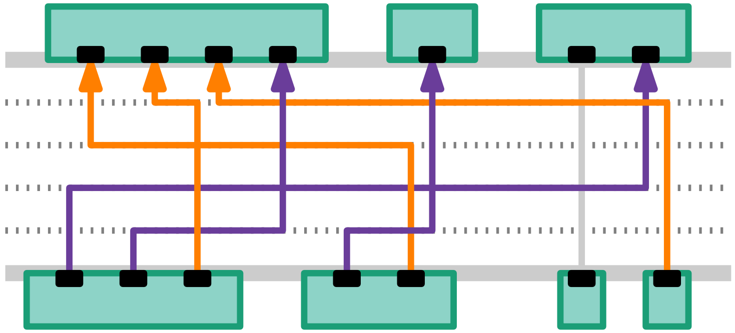
Reviewer: Consider containment interval graphs!

- A simple greedy algorithm colors these graphs optimally in $O(n \log n)$ time.

$n :=$ # vertices

- In layered graph drawing, this corresponds to routing "left-going" edges orthogonally to the fewest horizontal tracks. (Symmetrically "right-going".)

⇒ Combining the drawings of left-going and right-going edges yields a 2-approximation for the number of tracks. (bidirectional interval graphs)



can we do better?

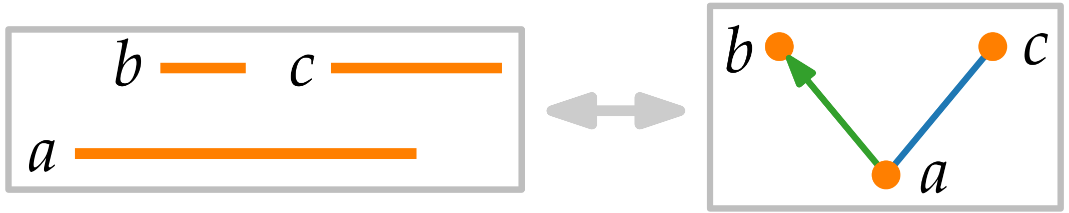
- In our paper, we present a constructive $O(n^2)$-time algorithm for recognizing directional interval graphs, which is based on PQ-trees.

bidirectional?

- For the more general case of mixed interval graphs, coloring is NP-hard.

(Remark: NP-hardness requires both directed and undirected edges.)

Conclusion and Open Problems



- We have introduced the natural concept of directional interval graphs.

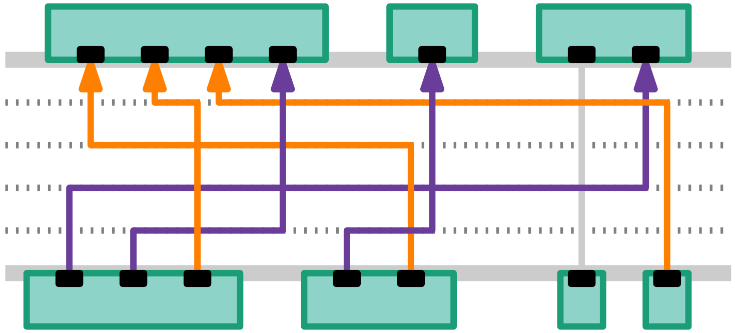
  ??? Reviewer: Consider containment interval graphs!

- A simple greedy algorithm colors these graphs optimally in $O(n \log n)$ time.

  $n :=$ # vertices

- In layered graph drawing, this corresponds to routing “left-going” edges orthogonally to the fewest horizontal tracks. (Symmetrically “right-going”.)

⇒ Combining the drawings of left-going and right-going edges yields a 2-approximation for the number of tracks. (bidirectional interval graphs)



can we do better?

- In our paper, we present a constructive $O(n^2)$-time algorithm for recognizing directional interval graphs, which is based on PQ-trees.

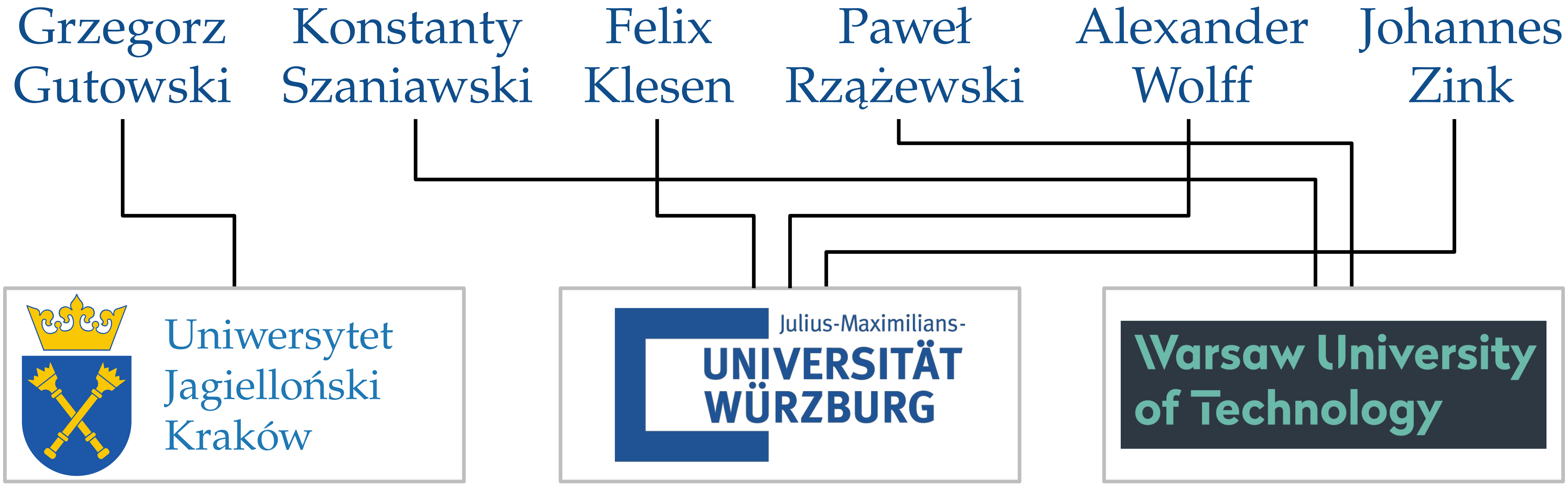
  bidirectional?

- For the more general case of mixed interval graphs, coloring is NP-hard.

  (Remark: NP-hardness requires both directed and undirected edges.)

Coloring and Recognizing Mixed Interval Graphs

ISAAC 2023, Kyoto



Grzegorz Gutowski

Konstanty Szaniawski

Felix Klesen

Paweł Rzążewski

Alexander Wolff

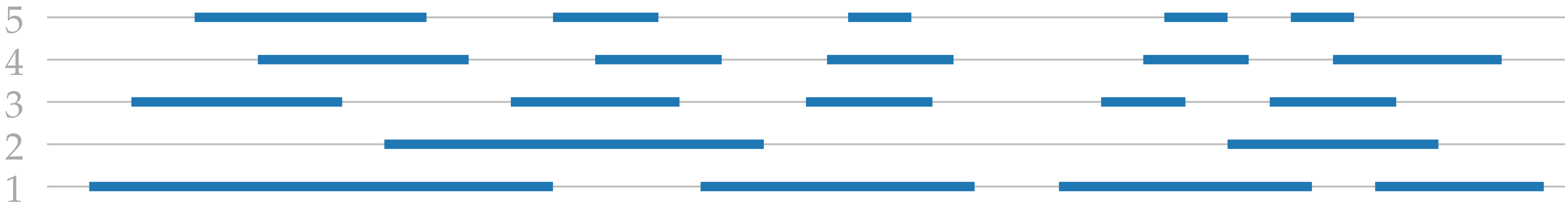
Johannes Zink

Uniwersytet Jagielloński Kraków

Julius-Maximilians-UNIVERSITÄT WÜRZBURG

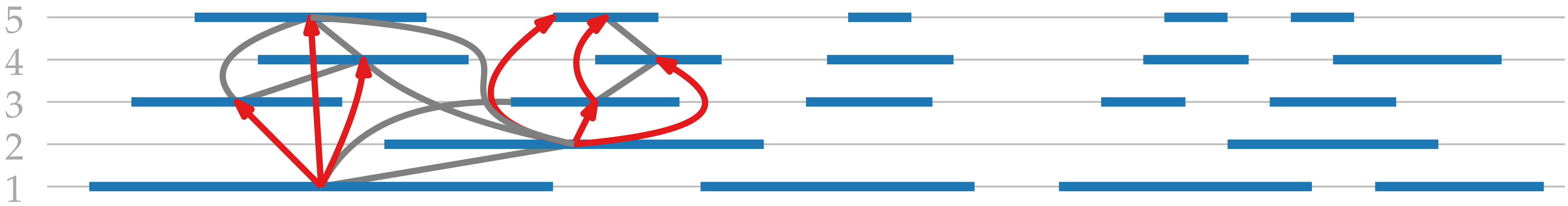
Warsaw University of Technology

Some Observation about Interval Containment Graphs



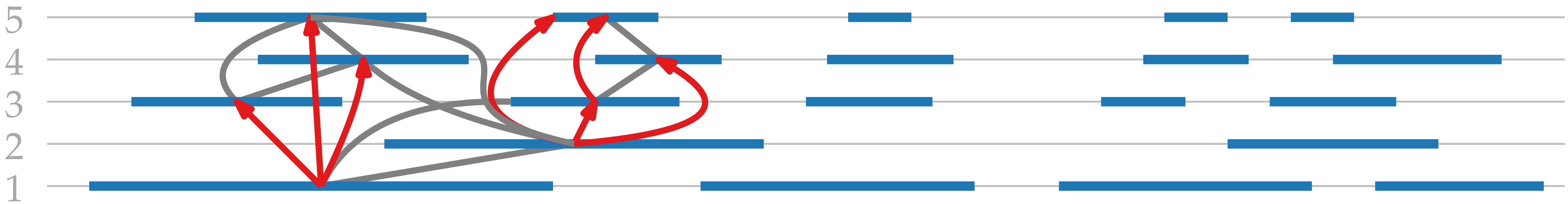
Let $\mathcal{I}$ be a set of intervals.

Some Observation about Interval Containment Graphs



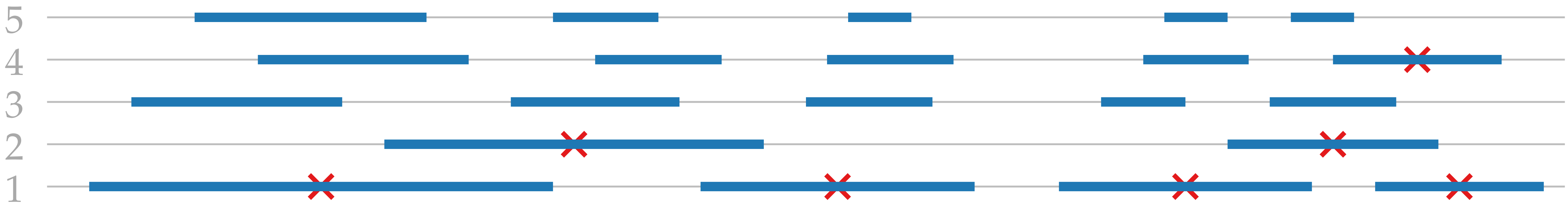
Let $\mathcal{I}$ be a set of intervals. Let $G = \mathcal{C}[\mathcal{I}]$ be the containment graph induced by $\mathcal{I}$.

Some Observation about Interval Containment Graphs



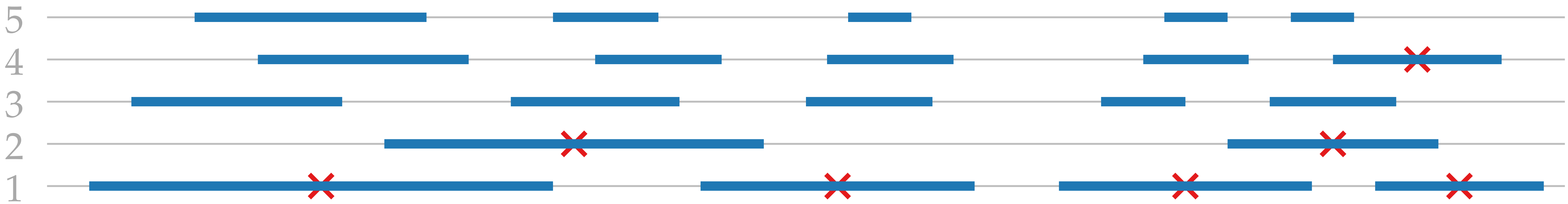
Let $\mathcal{I}$ be a set of intervals. Let $G = \mathcal{C}[\mathcal{I}]$ be the containment graph induced by $\mathcal{I}$.
Let $M(\mathcal{I})$ be the set of inclusion-wise maximum elements in $\mathcal{I}$.

Some Observation about Interval Containment Graphs



Let $\mathcal{I}$ be a set of intervals. Let $G = \mathcal{C}[\mathcal{I}]$ be the containment graph induced by $\mathcal{I}$.
Let $M(\mathcal{I})$ be the set of inclusion-wise maximum elements in $\mathcal{I}$.

Some Observation about Interval Containment Graphs

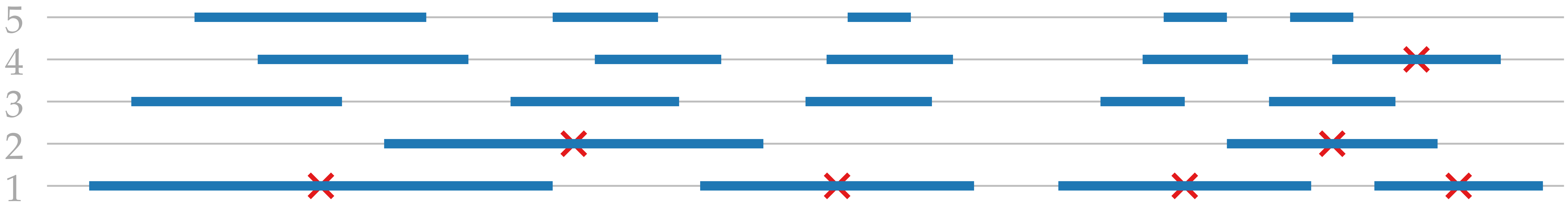


Let $\mathcal{I}$ be a set of intervals. Let $G = \mathcal{C}[\mathcal{I}]$ be the containment graph induced by $\mathcal{I}$.

Let $M(\mathcal{I})$ be the set of inclusion-wise maximum elements in $\mathcal{I}$.

Then $\mathcal{C}[M(\mathcal{I})]$ is a *proper* interval graph

Some Observation about Interval Containment Graphs

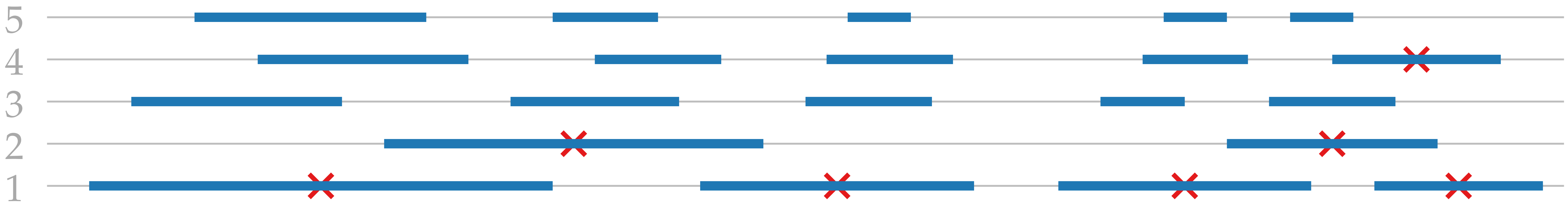


Let $\mathcal{I}$ be a set of intervals. Let $G = \mathcal{C}[\mathcal{I}]$ be the containment graph induced by $\mathcal{I}$.

Let $M(\mathcal{I})$ be the set of inclusion-wise maximum elements in $\mathcal{I}$.

Then $\mathcal{C}[M(\mathcal{I})]$ is a *proper* interval graph – no interval contains another interval.

Some Observation about Interval Containment Graphs



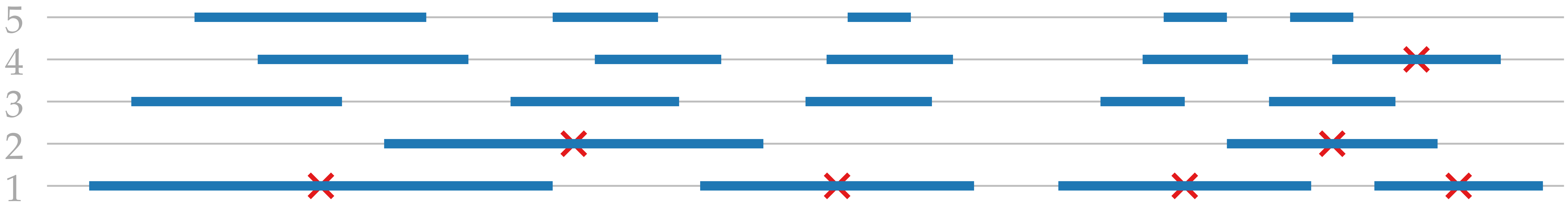
Let $\mathcal{I}$ be a set of intervals. Let $G = \mathcal{C}[\mathcal{I}]$ be the containment graph induced by $\mathcal{I}$.

Let $M(\mathcal{I})$ be the set of inclusion-wise maximum elements in $\mathcal{I}$.

Then $\mathcal{C}[M(\mathcal{I})]$ is a *proper* interval graph – no interval contains another interval.

Also note that $\bigcup M(\mathcal{I}) =$

Some Observation about Interval Containment Graphs



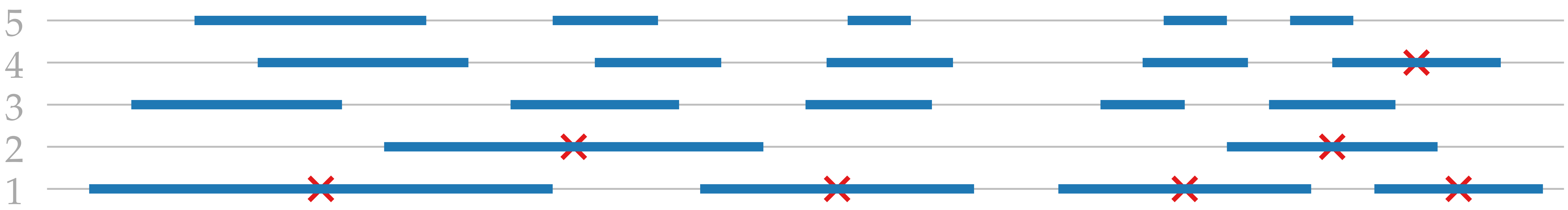
Let $\mathcal{I}$ be a set of intervals. Let $G = \mathcal{C}[\mathcal{I}]$ be the containment graph induced by $\mathcal{I}$.

Let $M(\mathcal{I})$ be the set of inclusion-wise maximum elements in $\mathcal{I}$.

Then $\mathcal{C}[M(\mathcal{I})]$ is a *proper* interval graph – no interval contains another interval.

Also note that $\bigcup M(\mathcal{I}) = \bigcup \mathcal{I}$.

Some Observation about Interval Containment Graphs



Let $\mathcal{I}$ be a set of intervals. Let $G = \mathcal{C}[\mathcal{I}]$ be the containment graph induced by $\mathcal{I}$.

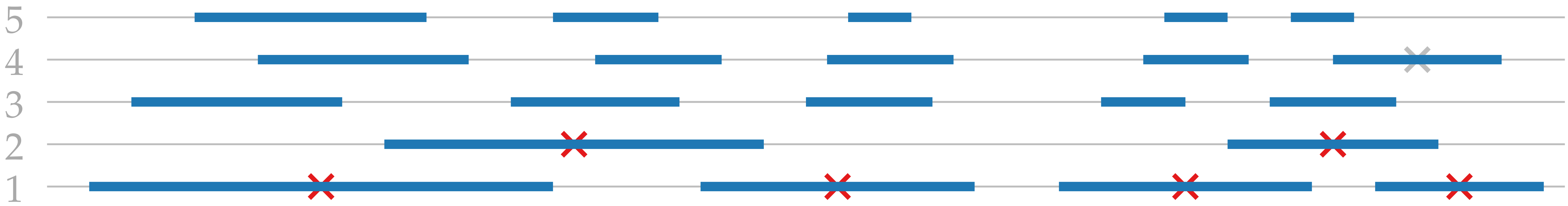
Let $M(\mathcal{I})$ be the set of inclusion-wise maximum elements in $\mathcal{I}$.

Then $\mathcal{C}[M(\mathcal{I})]$ is a *proper* interval graph – no interval contains another interval.

Also note that $\bigcup M(\mathcal{I}) = \bigcup \mathcal{I}$.

Let R be an inclusion-wise minimal subset of $M(\mathcal{I})$ such that $\bigcup R = \bigcup \mathcal{I}$.

Some Observation about Interval Containment Graphs



Let $\mathcal{I}$ be a set of intervals. Let $G = \mathcal{C}[\mathcal{I}]$ be the containment graph induced by $\mathcal{I}$.

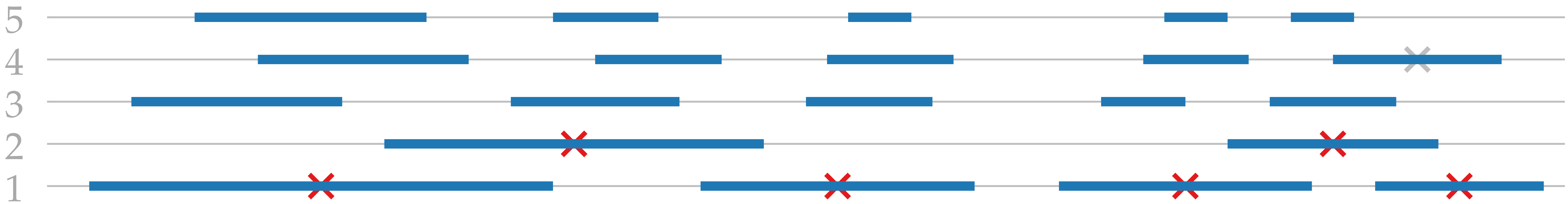
Let $M(\mathcal{I})$ be the set of inclusion-wise maximum elements in $\mathcal{I}$.

Then $\mathcal{C}[M(\mathcal{I})]$ is a *proper* interval graph – no interval contains another interval.

Also note that $\bigcup M(\mathcal{I}) = \bigcup \mathcal{I}$.

Let R be an inclusion-wise minimal subset of $M(\mathcal{I})$ such that $\bigcup R = \bigcup \mathcal{I}$.

Some Observation about Interval Containment Graphs



Let $\mathcal{I}$ be a set of intervals. Let $G = \mathcal{C}[\mathcal{I}]$ be the containment graph induced by $\mathcal{I}$.

Let $M(\mathcal{I})$ be the set of inclusion-wise maximum elements in $\mathcal{I}$.

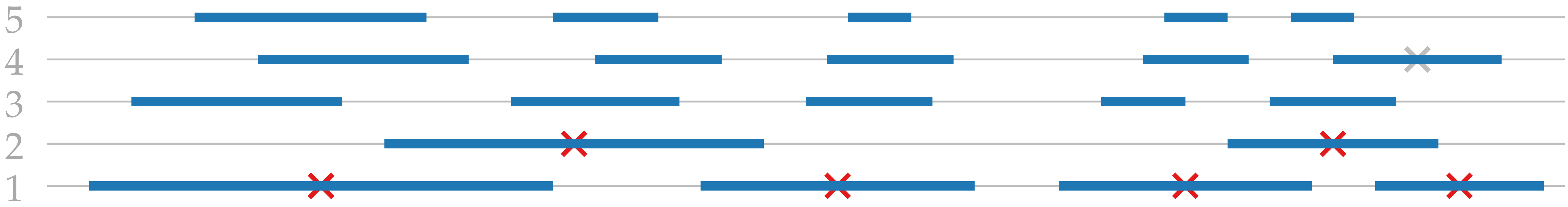
Then $\mathcal{C}[M(\mathcal{I})]$ is a *proper* interval graph – no interval contains another interval.

Also note that $\bigcup M(\mathcal{I}) = \bigcup \mathcal{I}$.

Let R be an inclusion-wise minimal subset of $M(\mathcal{I})$ such that $\bigcup R = \bigcup \mathcal{I}$.

Claim. $\mathcal{C}[R]$ is

Some Observation about Interval Containment Graphs



Let $\mathcal{I}$ be a set of intervals. Let $G = \mathcal{C}[\mathcal{I}]$ be the containment graph induced by $\mathcal{I}$.

Let $M(\mathcal{I})$ be the set of inclusion-wise maximum elements in $\mathcal{I}$.

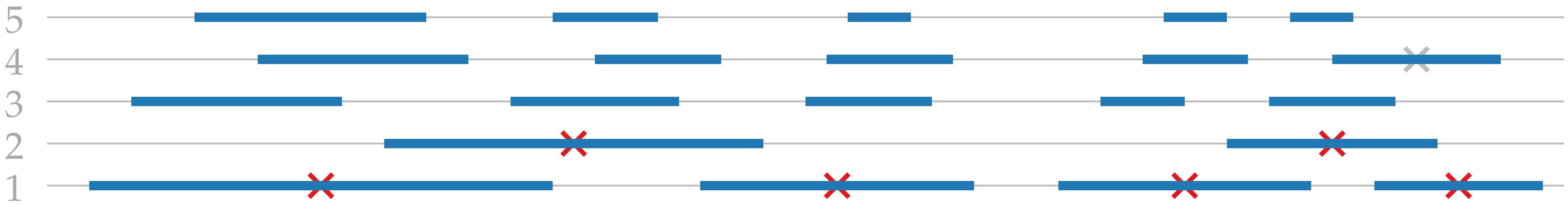
Then $\mathcal{C}[M(\mathcal{I})]$ is a *proper* interval graph – no interval contains another interval.

Also note that $\bigcup M(\mathcal{I}) = \bigcup \mathcal{I}$.

Let R be an inclusion-wise minimal subset of $M(\mathcal{I})$ such that $\bigcup R = \bigcup \mathcal{I}$.

Claim. $\mathcal{C}[R]$ is an undirected linear forest.

Some Observation about Interval Containment Graphs



Let $\mathcal{I}$ be a set of intervals. Let $G = \mathcal{C}[\mathcal{I}]$ be the containment graph induced by $\mathcal{I}$.

Let $M(\mathcal{I})$ be the set of inclusion-wise maximum elements in $\mathcal{I}$.

Then $\mathcal{C}[M(\mathcal{I})]$ is a *proper* interval graph – no interval contains another interval.

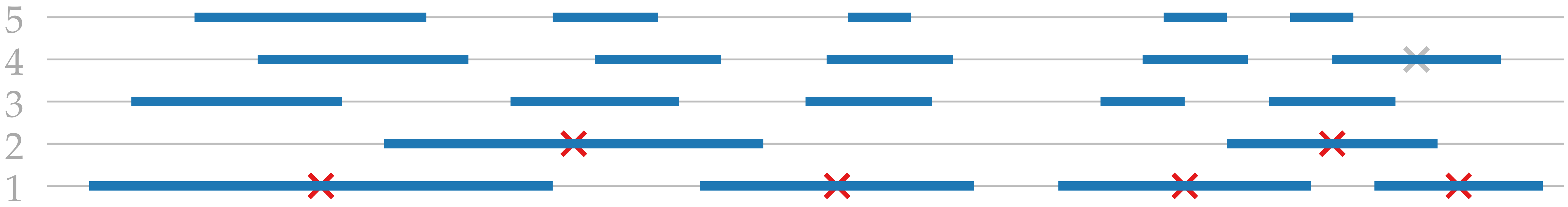
Also note that $\bigcup M(\mathcal{I}) = \bigcup \mathcal{I}$.

Let R be an inclusion-wise minimal subset of $M(\mathcal{I})$ such that $\bigcup R = \bigcup \mathcal{I}$.

Claim. $\mathcal{C}[R]$ is an undirected linear forest.

Proof. $\mathcal{C}[R]$ is proper $\Rightarrow$

Some Observation about Interval Containment Graphs



Let $\mathcal{I}$ be a set of intervals. Let $G = \mathcal{C}[\mathcal{I}]$ be the containment graph induced by $\mathcal{I}$.

Let $M(\mathcal{I})$ be the set of inclusion-wise maximum elements in $\mathcal{I}$.

Then $\mathcal{C}[M(\mathcal{I})]$ is a *proper* interval graph – no interval contains another interval.

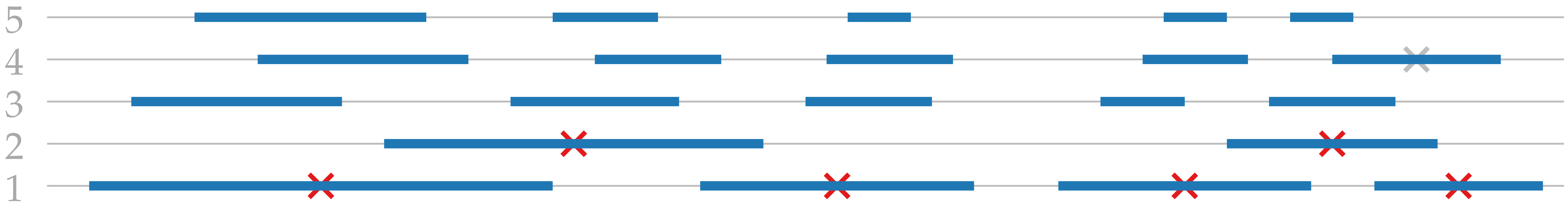
Also note that $\bigcup M(\mathcal{I}) = \bigcup \mathcal{I}$.

Let R be an inclusion-wise minimal subset of $M(\mathcal{I})$ such that $\bigcup R = \bigcup \mathcal{I}$.

Claim. $\mathcal{C}[R]$ is an undirected linear forest.

Proof. $\mathcal{C}[R]$ is proper $\Rightarrow$ contains no induced

Some Observation about Interval Containment Graphs



Let $\mathcal{I}$ be a set of intervals. Let $G = \mathcal{C}[\mathcal{I}]$ be the containment graph induced by $\mathcal{I}$.

Let $M(\mathcal{I})$ be the set of inclusion-wise maximum elements in $\mathcal{I}$.

Then $\mathcal{C}[M(\mathcal{I})]$ is a *proper* interval graph – no interval contains another interval.

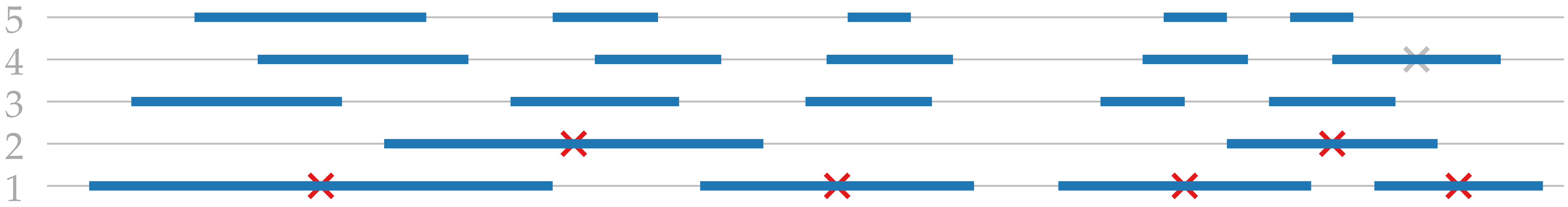
Also note that $\bigcup M(\mathcal{I}) = \bigcup \mathcal{I}$.

Let R be an inclusion-wise minimal subset of $M(\mathcal{I})$ such that $\bigcup R = \bigcup \mathcal{I}$.

Claim. $\mathcal{C}[R]$ is an undirected linear forest.

Proof. $\mathcal{C}[R]$ is proper $\Rightarrow$ contains no induced $K_{1,3}$ and no induced C_ℓ for $\ell \geq 4$.

Some Observation about Interval Containment Graphs



Let $\mathcal{I}$ be a set of intervals. Let $G = \mathcal{C}[\mathcal{I}]$ be the containment graph induced by $\mathcal{I}$.

Let $M(\mathcal{I})$ be the set of inclusion-wise maximum elements in $\mathcal{I}$.

Then $\mathcal{C}[M(\mathcal{I})]$ is a *proper* interval graph – no interval contains another interval.

Also note that $\bigcup M(\mathcal{I}) = \bigcup \mathcal{I}$.

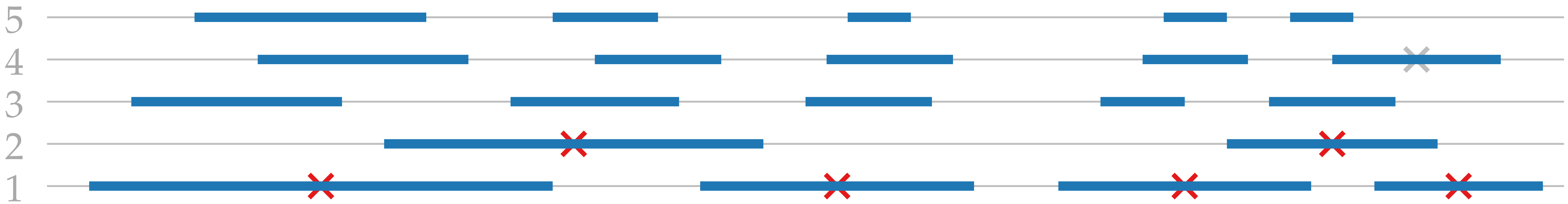
Let R be an inclusion-wise minimal subset of $M(\mathcal{I})$ such that $\bigcup R = \bigcup \mathcal{I}$.

Claim. $\mathcal{C}[R]$ is an undirected linear forest.

Proof. $\mathcal{C}[R]$ is proper $\Rightarrow$ contains no induced $K_{1,3}$ and no induced C_ℓ for $\ell \geq 4$.

It remains to show that $\mathcal{C}[R]$ contains no triangle.

Some Observation about Interval Containment Graphs



Let $\mathcal{I}$ be a set of intervals. Let $G = \mathcal{C}[\mathcal{I}]$ be the containment graph induced by $\mathcal{I}$.

Let $M(\mathcal{I})$ be the set of inclusion-wise maximum elements in $\mathcal{I}$.

Then $\mathcal{C}[M(\mathcal{I})]$ is a *proper* interval graph – no interval contains another interval.

Also note that $\bigcup M(\mathcal{I}) = \bigcup \mathcal{I}$.

Let R be an inclusion-wise minimal subset of $M(\mathcal{I})$ such that $\bigcup R = \bigcup \mathcal{I}$.

Claim. $\mathcal{C}[R]$ is an undirected linear forest.

Proof. $\mathcal{C}[R]$ is proper $\Rightarrow$ contains no induced $K_{1,3}$ and no induced C_ℓ for $\ell \geq 4$.

It remains to show that $\mathcal{C}[R]$ contains no triangle. ✓

A 2-Approximation Algorithm for Coloring

Theorem. For any set $\mathcal{I}$ of intervals,
the graph $\mathcal{C}[\mathcal{I}]$ admits a coloring with at most $2 \cdot \omega(\mathcal{C}[\mathcal{I}]) - 1$ colors.

A 2-Approximation Algorithm for Coloring

Theorem. For any set $\mathcal{I}$ of intervals, the graph $\mathcal{C}[\mathcal{I}]$ admits a coloring with at most $2 \cdot \overbrace{\omega(\mathcal{C}[\mathcal{I}])}^{\omega\,=\,\text{clique number}} - 1$ colors.

A 2-Approximation Algorithm for Coloring

Theorem. For any set $\mathcal{I}$ of intervals,
the graph $\mathcal{C}[\mathcal{I}]$ admits a coloring with at most $2 \cdot \overbrace{\omega(\mathcal{C}[\mathcal{I}])}^{\omega\,=\,\text{clique number}} - 1$ colors.

Since $\mathcal{C}[R]$ is a linear forest, it admits a coloring $f_1\colon R \to \{1, 2\}$.

A 2-Approximation Algorithm for Coloring

Theorem. For any set $\mathcal{I}$ of intervals, the graph $\mathcal{C}[\mathcal{I}]$ admits a coloring with at most $2 \cdot \overbrace{\omega(\mathcal{C}[\mathcal{I}])}^{\omega\,=\,\text{clique number}} - 1$ colors.

Since $\mathcal{C}[R]$ is a linear forest, it admits a coloring $f_1 \colon R \to \{1, 2\}$.

If $R = \mathcal{I}$, we are done (using only ω many colors), so we assume $\mathcal{I} \setminus R \neq \emptyset$.

A 2-Approximation Algorithm for Coloring

Theorem. For any set $\mathcal{I}$ of intervals, the graph $\mathcal{C}[\mathcal{I}]$ admits a coloring with at most $2 \cdot \overbrace{\omega(\mathcal{C}[\mathcal{I}])}^{\omega\,=\,\text{clique number}} - 1$ colors.

Since $\mathcal{C}[R]$ is a linear forest, it admits a coloring $f_1 \colon R \to \{1, 2\}$.

If $R = \mathcal{I}$, we are done (using only ω many colors), so we assume $\mathcal{I} \setminus R \neq \emptyset$.

Let $G' := \mathcal{C}[\mathcal{I} \setminus R]$.

A 2-Approximation Algorithm for Coloring

Theorem. For any set $\mathcal{I}$ of intervals, the graph $\mathcal{C}[\mathcal{I}]$ admits a coloring with at most $2 \cdot \overbrace{\omega(\mathcal{C}[\mathcal{I}])}^{\omega\,=\,\text{clique number}} - 1$ colors.

Since $\mathcal{C}[R]$ is a linear forest, it admits a coloring $f_1 \colon R \to \{1, 2\}$.

If $R = \mathcal{I}$, we are done (using only ω many colors), so we assume $\mathcal{I} \setminus R \neq \emptyset$.

Let $G' := \mathcal{C}[\mathcal{I} \setminus R]$.

Claim. $\omega(G') \leq$

A 2-Approximation Algorithm for Coloring

Theorem. For any set $\mathcal{I}$ of intervals,
the graph $\mathcal{C}[\mathcal{I}]$ admits a coloring with at most $2 \cdot \overbrace{\omega(\mathcal{C}[\mathcal{I}])}^{\omega\, =\, \text{clique number}} - 1$ colors.

Since $\mathcal{C}[R]$ is a linear forest, it admits a coloring $f_1 \colon R \to \{1,2\}$.

If $R = \mathcal{I}$, we are done (using only ω many colors), so we assume $\mathcal{I} \setminus R \neq \emptyset$.

Let $G' := \mathcal{C}[\mathcal{I} \setminus R]$.

Claim. $\omega(G') \leq \omega - 1$.

A 2-Approximation Algorithm for Coloring

Theorem. For any set $\mathcal{I}$ of intervals, the graph $\mathcal{C}[\mathcal{I}]$ admits a coloring with at most $2 \cdot \overbrace{\omega(\mathcal{C}[\mathcal{I}])}^{\omega \,=\, \text{clique number}} - 1$ colors.

Since $\mathcal{C}[R]$ is a linear forest, it admits a coloring $f_1 \colon R \to \{1, 2\}$.

If $R = \mathcal{I}$, we are done (using only ω many colors), so we assume $\mathcal{I} \setminus R \neq \emptyset$.

Let $G' := \mathcal{C}[\mathcal{I} \setminus R]$.

Claim. $\omega(G') \leq \omega - 1$.

Proof. Suppose that there is a clique S in G' of size ω.

A 2-Approximation Algorithm for Coloring

Theorem. For any set $\mathcal{I}$ of intervals, the graph $\mathcal{C}[\mathcal{I}]$ admits a coloring with at most $2 \cdot \overbrace{\omega(\mathcal{C}[\mathcal{I}])}^{\omega\,=\,\text{clique number}} - 1$ colors.

Since $\mathcal{C}[R]$ is a linear forest, it admits a coloring $f_1 \colon R \to \{1,2\}$.

If $R = \mathcal{I}$, we are done (using only ω many colors), so we assume $\mathcal{I} \setminus R \neq \emptyset$.

Let $G' := \mathcal{C}[\mathcal{I} \setminus R]$.

Claim. $\omega(G') \leq \omega - 1$.

Proof. Suppose that there is a clique S in G' of size ω.

Helly property of intervals $\Rightarrow \bigcap S \neq \emptyset$.

A 2-Approximation Algorithm for Coloring

Theorem. For any set $\mathcal{I}$ of intervals, the graph $\mathcal{C}[\mathcal{I}]$ admits a coloring with at most $2 \cdot \overbrace{\omega(\mathcal{C}[\mathcal{I}])}^{\omega\,=\,\text{clique number}} - 1$ colors.

Since $\mathcal{C}[R]$ is a linear forest, it admits a coloring $f_1 \colon R \to \{1, 2\}$.

If $R = \mathcal{I}$, we are done (using only ω many colors), so we assume $\mathcal{I} \setminus R \neq \emptyset$.

Let $G' := \mathcal{C}[\mathcal{I} \setminus R]$.

Claim. $\omega(G') \leq \omega - 1$.

Proof. Suppose that there is a clique S in G' of size ω.

Helly property of intervals $\Rightarrow \bigcap S \neq \emptyset$. Let $p \in \bigcap S$.

A 2-Approximation Algorithm for Coloring

Theorem. For any set $\mathcal{I}$ of intervals,
the graph $\mathcal{C}[\mathcal{I}]$ admits a coloring with at most $2 \cdot \overbrace{\omega(\mathcal{C}[\mathcal{I}])}^{\omega\,=\,\text{clique number}} - 1$ colors.

Since $\mathcal{C}[R]$ is a linear forest, it admits a coloring $f_1 \colon R \to \{1, 2\}$.

If $R = \mathcal{I}$, we are done (using only ω many colors), so we assume $\mathcal{I} \setminus R \neq \emptyset$.

Let $G' := \mathcal{C}[\mathcal{I} \setminus R]$.

Claim. $\omega(G') \leq \omega - 1$.

Proof. Suppose that there is a clique S in G' of size ω.

Helly property of intervals $\Rightarrow \bigcap S \neq \emptyset$. Let $p \in \bigcap S$.

Pick an $r \in R$ that contains p.

A 2-Approximation Algorithm for Coloring

Theorem. For any set $\mathcal{I}$ of intervals, the graph $\mathcal{C}[\mathcal{I}]$ admits a coloring with at most $2 \cdot \overbrace{\omega(\mathcal{C}[\mathcal{I}])}^{\omega\, =\, \text{clique number}} - 1$ colors.

Since $\mathcal{C}[R]$ is a linear forest, it admits a coloring $f_1 \colon R \to \{1,2\}$.

If $R = \mathcal{I}$, we are done (using only ω many colors), so we assume $\mathcal{I} \setminus R \neq \emptyset$.

Let $G' := \mathcal{C}[\mathcal{I} \setminus R]$.

Claim. $\omega(G') \leq \omega - 1$.

Proof. Suppose that there is a clique S in G' of size ω.

Helly property of intervals $\Rightarrow \bigcap S \neq \emptyset$. Let $p \in \bigcap S$.

Pick an $r \in R$ that contains p. $\Rightarrow S \cup \{r\}$ is a clique of size $\omega + 1$ in G.

A 2-Approximation Algorithm for Coloring

Theorem. For any set $\mathcal{I}$ of intervals, the graph $\mathcal{C}[\mathcal{I}]$ admits a coloring with at most $2 \cdot \overbrace{\omega(\mathcal{C}[\mathcal{I}])}^{\omega\,=\,\text{clique number}} - 1$ colors.

Since $\mathcal{C}[R]$ is a linear forest, it admits a coloring $f_1 \colon R \to \{1, 2\}$.

If $R = \mathcal{I}$, we are done (using only ω many colors), so we assume $\mathcal{I} \setminus R \neq \emptyset$.

Let $G' := \mathcal{C}[\mathcal{I} \setminus R]$.

Claim. $\omega(G') \leq \omega - 1$.

Proof. Suppose that there is a clique S in G' of size ω.

Helly property of intervals $\Rightarrow \bigcap S \neq \emptyset$. Let $p \in \bigcap S$.

Pick an $r \in R$ that contains p. $\Rightarrow S \cup \{r\}$ is a clique of size $\omega + 1$ in G. ↯

A 2-Approximation Algorithm for Coloring

Theorem. For any set $\mathcal{I}$ of intervals,
the graph $\mathcal{C}[\mathcal{I}]$ admits a coloring with at most $2 \cdot \overbrace{\omega(\mathcal{C}[\mathcal{I}])}^{\omega\,=\,\text{clique number}} - 1$ colors.

Since $\mathcal{C}[R]$ is a linear forest, it admits a coloring $f_1 \colon R \to \{1, 2\}$.

If $R = \mathcal{I}$, we are done (using only ω many colors), so we assume $\mathcal{I} \setminus R \neq \emptyset$.

Let $G' := \mathcal{C}[\mathcal{I} \setminus R]$.

Claim. $\omega(G') \leq \omega - 1$.

Proof. Suppose that there is a clique S in G' of size ω.
Helly property of intervals $\Rightarrow \bigcap S \neq \emptyset$. Let $p \in \bigcap S$.
Pick an $r \in R$ that contains p. $\Rightarrow S \cup \{r\}$ is a clique of size $\omega + 1$ in G. ↯

Induction $\Rightarrow G'$ admits a coloring f_2 using at most $2 \cdot \omega(G') - 1$ colors.

A 2-Approximation Algorithm for Coloring

Theorem. For any set $\mathcal{I}$ of intervals,
the graph $\mathcal{C}[\mathcal{I}]$ admits a coloring with at most $2 \cdot \overbrace{\omega(\mathcal{C}[\mathcal{I}])}^{\omega\, =\, \text{clique number}} - 1$ colors.

Since $\mathcal{C}[R]$ is a linear forest, it admits a coloring $f_1 \colon R \to \{1,2\}$.

If $R = \mathcal{I}$, we are done (using only ω many colors), so we assume $\mathcal{I} \setminus R \neq \emptyset$.

Let $G' := \mathcal{C}[\mathcal{I} \setminus R]$.

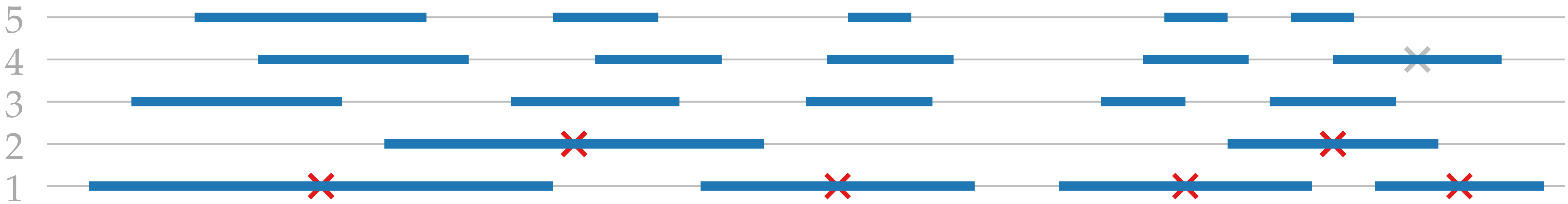
Claim. $\omega(G') \leq \omega - 1$.

Proof. Suppose that there is a clique S in G' of size ω.
Helly property of intervals $\Rightarrow \bigcap S \neq \emptyset$. Let $p \in \bigcap S$.
Pick an $r \in R$ that contains p. $\Rightarrow S \cup \{r\}$ is a clique of size $\omega + 1$ in G. ↯

Induction $\Rightarrow$ G' admits a coloring f_2 using at most $2 \cdot \omega(G') - 1$ colors.

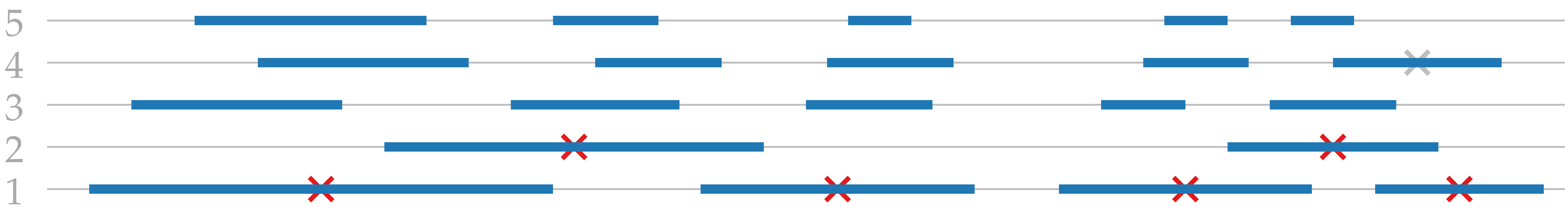
With f_1 and f_2, we construct a coloring f of G using colors $\{1, \ldots, 2\omega - 1\}$.

An Inductive Coloring



$$\text{Let } f(x) = \begin{cases} f_1(x) & \text{if } x \in R, \\ f_2(x) + 2 & \text{else.} \end{cases}$$

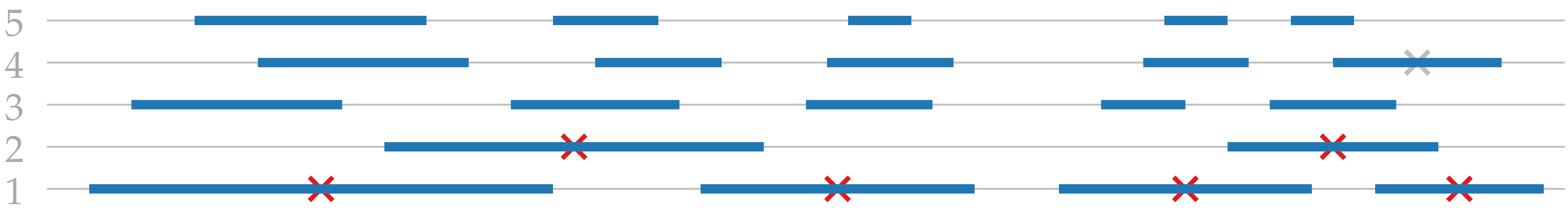
An Inductive Coloring



Let $f(x) = \begin{cases} f_1(x) & \text{if } x \in R, \\ f_2(x) + 2 & \text{else.} \end{cases}$

This defines a coloring of G:

An Inductive Coloring



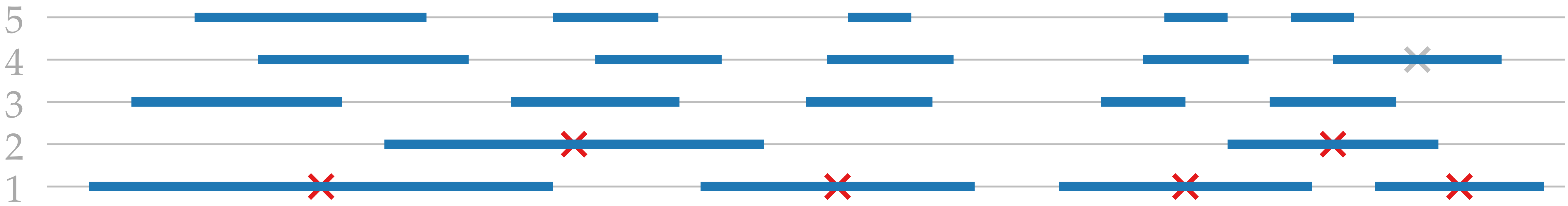
$$\text{Let } f(x) = \begin{cases} f_1(x) & \text{if } x \in R, \\ f_2(x) + 2 & \text{else.} \end{cases}$$

This defines a coloring of G:

1. If $x \cap y \neq \emptyset$, then $f(x) \neq f(y)$.

2. If $x \subseteq y$, then $f(x) > f(y)$.

An Inductive Coloring

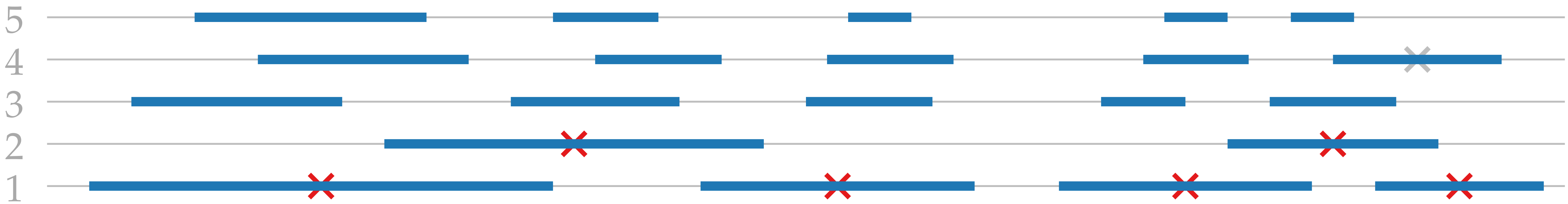


Let $f(x) = \begin{cases} f_1(x) & \text{if } x \in R, \\ f_2(x) + 2 & \text{else.} \end{cases}$

This defines a coloring of G:

1. If $x \cap y \neq \emptyset$, then $f(x) \neq f(y)$. Check: $x, y \in R$; $x, y \neq R$; $x \in R$ and $y \neq R$.
2. If $x \subseteq y$, then $f(x) > f(y)$.

An Inductive Coloring



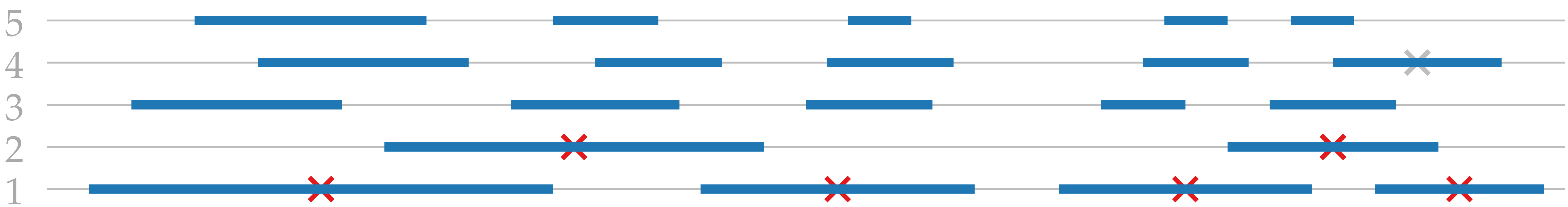
Let $f(x) = \begin{cases} f_1(x) & \text{if } x \in R, \\ f_2(x) + 2 & \text{else.} \end{cases}$

This defines a coloring of G:

1. If $x \cap y \neq \emptyset$, then $f(x) \neq f(y)$. Check: $x, y \in R$; $x, y \neq R$; $x \in R$ and $y \neq R$.

2. If $x \subseteq y$, then $f(x) > f(y)$. Observe that $x \neq R$

An Inductive Coloring



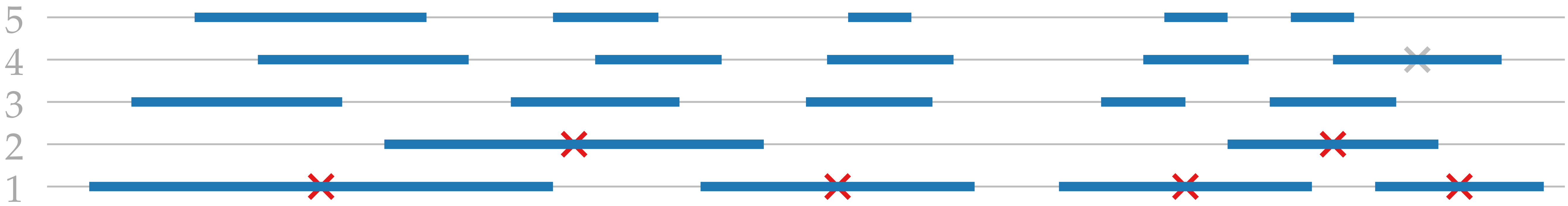
$$\text{Let } f(x) = \begin{cases} f_1(x) & \text{if } x \in R, \\ f_2(x) + 2 & \text{else.} \end{cases}$$

This defines a coloring of G:

1. If $x \cap y \neq \emptyset$, then $f(x) \neq f(y)$. Check: $x, y \in R$; $x, y \neq R$; $x \in R$ and $y \neq R$.

2. If $x \subseteq y$, then $f(x) > f(y)$. Observe that $x \neq R \Rightarrow f(x) \geq 3$

An Inductive Coloring



Let $f(x) = \begin{cases} f_1(x) & \text{if } x \in R, \\ f_2(x) + 2 & \text{else.} \end{cases}$

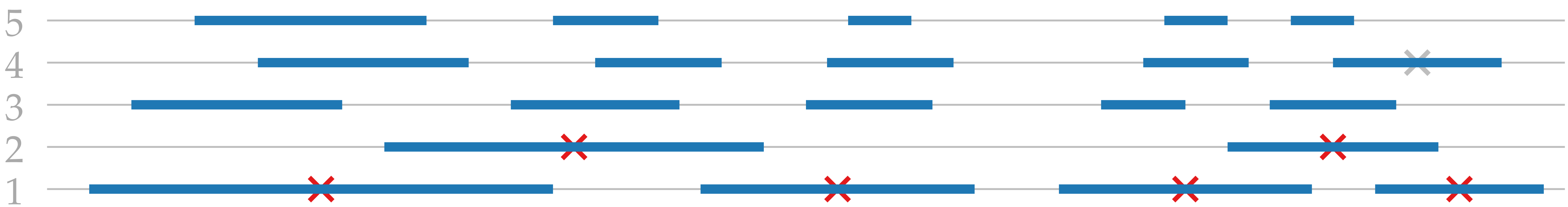
This defines a coloring of G:

1. If $x \cap y \neq \emptyset$, then $f(x) \neq f(y)$. Check: $x, y \in R$; $x, y \neq R$; $x \in R$ and $y \neq R$.

2. If $x \subseteq y$, then $f(x) > f(y)$. Observe that $x \neq R \Rightarrow f(x) \geq 3$

   Suppose $f(y) > f(x)$

An Inductive Coloring



Let $f(x) = \begin{cases} f_1(x) & \text{if } x \in R, \\ f_2(x) + 2 & \text{else.} \end{cases}$

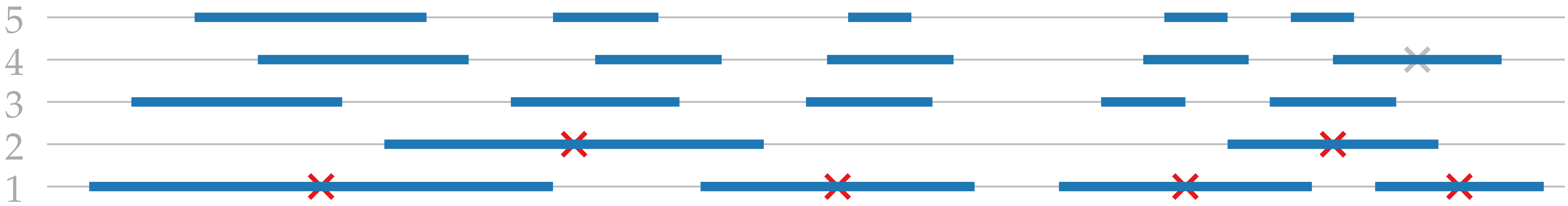
This defines a coloring of G:

1. If $x \cap y \neq \emptyset$, then $f(x) \neq f(y)$. Check: $x, y \in R$; $x, y \neq R$; $x \in R$ and $y \neq R$.

2. If $x \subseteq y$, then $f(x) > f(y)$. Observe that $x \neq R \ \Rightarrow f(x) \geq 3$

   Suppose $f(y) > f(x) \ \Rightarrow y \neq R$

An Inductive Coloring



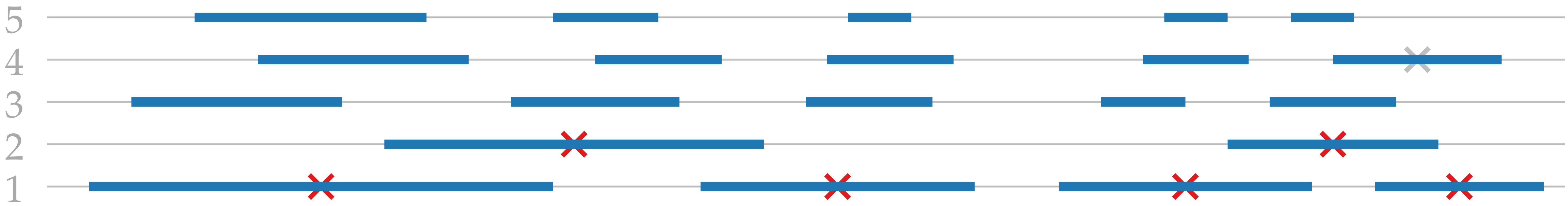
$$\text{Let } f(x) = \begin{cases} f_1(x) & \text{if } x \in R, \\ f_2(x) + 2 & \text{else.} \end{cases}$$

This defines a coloring of G:

1. If $x \cap y \neq \emptyset$, then $f(x) \neq f(y)$. Check: $x, y \in R$; $x, y \neq R$; $x \in R$ and $y \neq R$.
2. If $x \subseteq y$, then $f(x) > f(y)$. Observe that $x \neq R \Rightarrow f(x) \geq 3$

   Suppose $f(y) > f(x) \Rightarrow y \neq R$, but $f_2(x) > f_2(y)$.

An Inductive Coloring



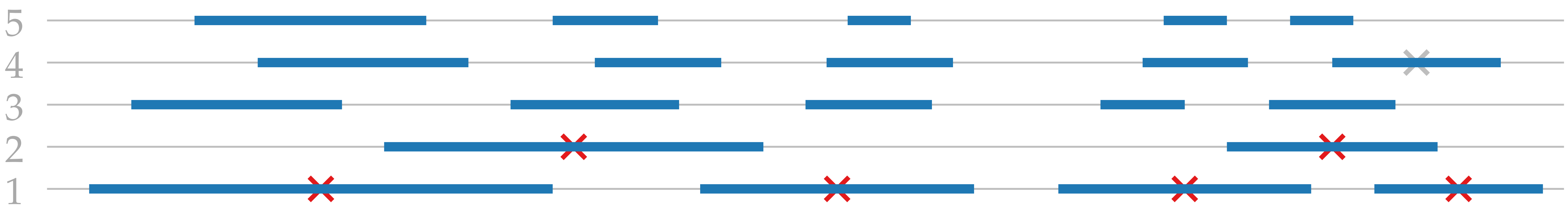
Let $f(x) = \begin{cases} f_1(x) & \text{if } x \in R, \\ f_2(x) + 2 & \text{else.} \end{cases}$

This defines a coloring of G:

1. If $x \cap y \neq \emptyset$, then $f(x) \neq f(y)$. Check: $x, y \in R$; $x, y \neq R$; $x \in R$ and $y \neq R$.
2. If $x \subseteq y$, then $f(x) > f(y)$. Observe that $x \neq R \;\Rightarrow f(x) \geq 3$

   Suppose $f(y) > f(x) \;\Rightarrow y \neq R$, but $f_2(x) > f_2(y)$. ↯

An Inductive Coloring



Let $f(x) = \begin{cases} f_1(x) & \text{if } x \in R, \\ f_2(x) + 2 & \text{else.} \end{cases}$

This defines a coloring of G:

1. If $x \cap y \neq \emptyset$, then $f(x) \neq f(y)$. Check: $x, y \in R$; $x, y \neq R$; $x \in R$ and $y \neq R$.
2. If $x \subseteq y$, then $f(x) > f(y)$. Observe that $x \neq R \Rightarrow f(x) \geq 3$

   Suppose $f(y) > f(x) \Rightarrow y \neq R$, but $f_2(x) > f_2(y)$. ↯

Corollary. There is a 2-approximation for coloring interval containment graphs. Given n intervals, the algorithm runs in $O(n \log n)$ time.

A Lower Bound Example

Proposition. There is an infinite family $(\mathcal{I}_n)_{n\geq 1}$ of sets of intervals with $|\mathcal{I}_n| = 3 \cdot 2^{n-1} - 2$, $\chi(\mathcal{C}[\mathcal{I}_n]) = 2n - 1$, and $\omega(\mathcal{C}[\mathcal{I}_n]) = n$.

A Lower Bound Example

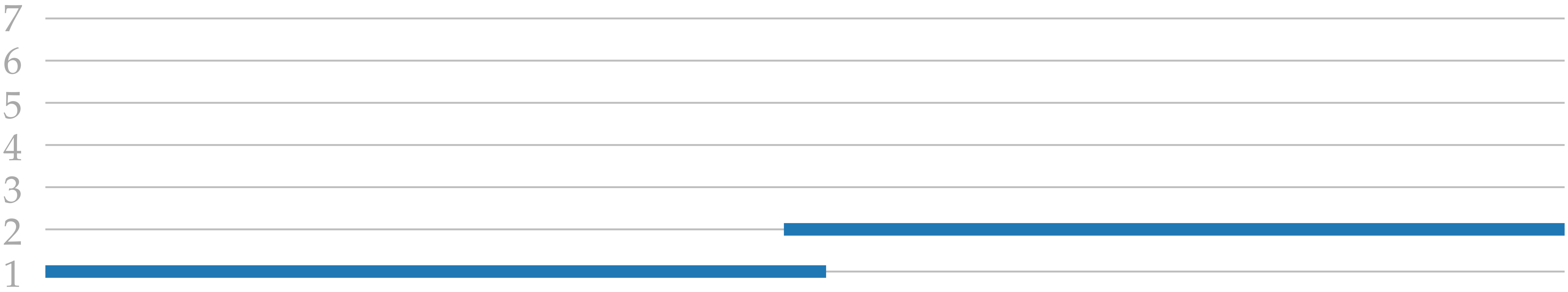
Proposition. There is an infinite family $(\mathcal{I}_n)_{n\geq 1}$ of sets of intervals with $|\mathcal{I}_n| = 3 \cdot 2^{n-1} - 2$, $\chi(\mathcal{C}[\mathcal{I}_n]) = 2n - 1$, and $\omega(\mathcal{C}[\mathcal{I}_n]) = n$.

This yields $\lim\limits_{n\to\infty} \chi(\mathcal{I}_n)/\omega(\mathcal{I}_n) = 2$.

A Lower Bound Example

Proposition. There is an infinite family $(\mathcal{I}_n)_{n\geq 1}$ of sets of intervals with $|\mathcal{I}_n| = 3 \cdot 2^{n-1} - 2$, $\chi(\mathcal{C}[\mathcal{I}_n]) = 2n - 1$, and $\omega(\mathcal{C}[\mathcal{I}_n]) = n$.

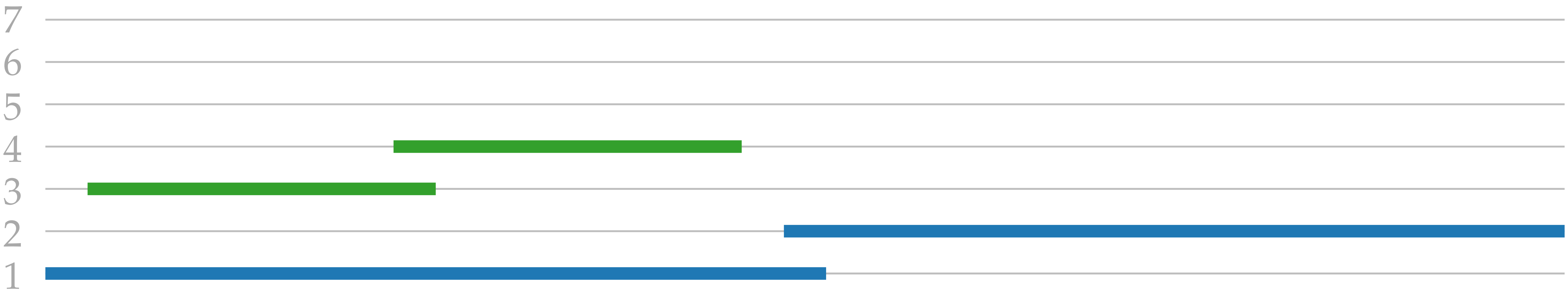
This yields $\lim\limits_{n\to\infty} \chi(\mathcal{I}_n)/\omega(\mathcal{I}_n) = 2$.



A Lower Bound Example

Proposition. There is an infinite family $(\mathcal{I}_n)_{n\geq 1}$ of sets of intervals with $|\mathcal{I}_n| = 3\cdot 2^{n-1} - 2$, $\chi(\mathcal{C}[\mathcal{I}_n]) = 2n-1$, and $\omega(\mathcal{C}[\mathcal{I}_n]) = n$.

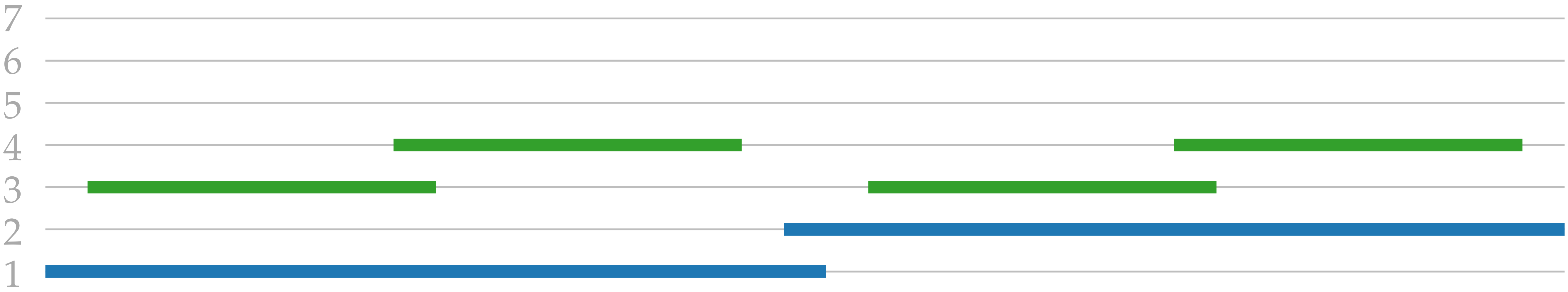
This yields $\lim_{n\to\infty} \chi(\mathcal{I}_n)/\omega(\mathcal{I}_n) = 2$.



A Lower Bound Example

Proposition. There is an infinite family $(\mathcal{I}_n)_{n \geq 1}$ of sets of intervals with $|\mathcal{I}_n| = 3 \cdot 2^{n-1} - 2$, $\chi(\mathcal{C}[\mathcal{I}_n]) = 2n - 1$, and $\omega(\mathcal{C}[\mathcal{I}_n]) = n$.

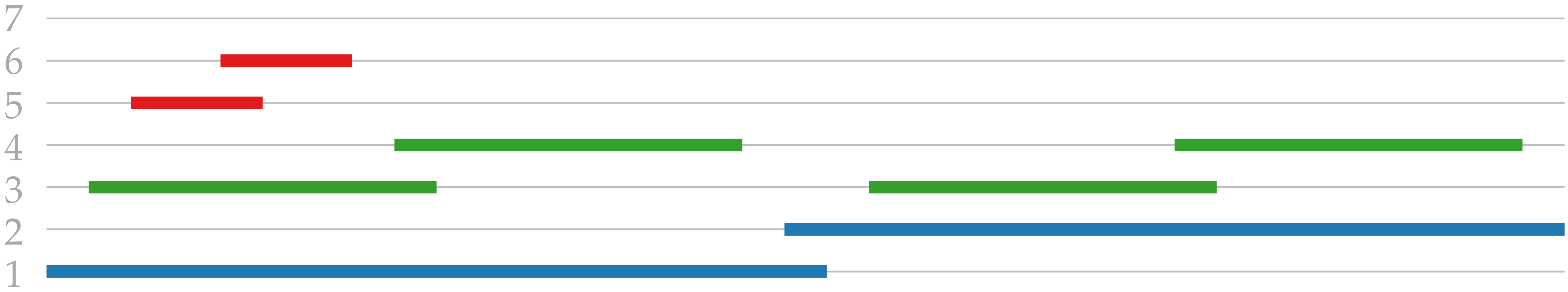
This yields $\lim\limits_{n \to \infty} \chi(\mathcal{I}_n)/\omega(\mathcal{I}_n) = 2$.



A Lower Bound Example

Proposition. There is an infinite family $(\mathcal{I}_n)_{n\geq 1}$ of sets of intervals with $|\mathcal{I}_n| = 3 \cdot 2^{n-1} - 2$, $\chi(\mathcal{C}[\mathcal{I}_n]) = 2n - 1$, and $\omega(\mathcal{C}[\mathcal{I}_n]) = n$.

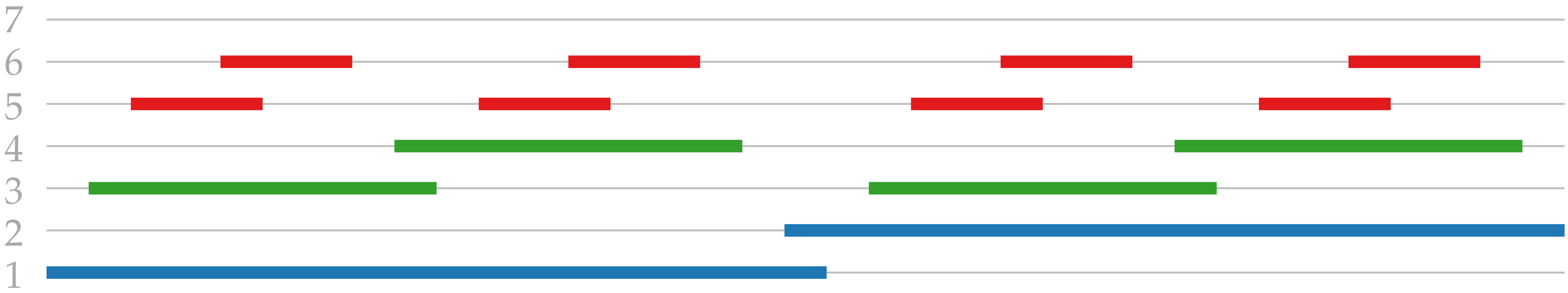
This yields $\lim_{n\to\infty} \chi(\mathcal{I}_n)/\omega(\mathcal{I}_n) = 2$.



A Lower Bound Example

Proposition. There is an infinite family $(\mathcal{I}_n)_{n\geq 1}$ of sets of intervals with $|\mathcal{I}_n| = 3 \cdot 2^{n-1} - 2$, $\chi(\mathcal{C}[\mathcal{I}_n]) = 2n-1$, and $\omega(\mathcal{C}[\mathcal{I}_n]) = n$.

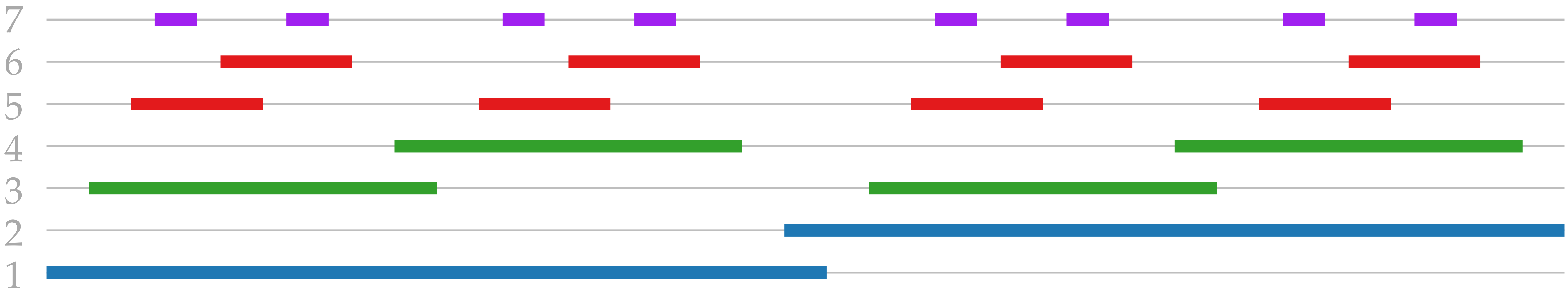
This yields $\lim_{n\to\infty} \chi(\mathcal{I}_n)/\omega(\mathcal{I}_n) = 2$.



A Lower Bound Example

Proposition. There is an infinite family $(\mathcal{I}_n)_{n\geq 1}$ of sets of intervals with $|\mathcal{I}_n| = 3 \cdot 2^{n-1} - 2$, $\chi(\mathcal{C}[\mathcal{I}_n]) = 2n - 1$, and $\omega(\mathcal{C}[\mathcal{I}_n]) = n$.

This yields $\lim\limits_{n\to\infty} \chi(\mathcal{I}_n)/\omega(\mathcal{I}_n) = 2$.



Computational Complexity

Theorem. Given a set $\mathcal{I}$ of intervals and a positive integer k, it is NP-hard to decide whether $\chi(\mathcal{C}[\mathcal{I}]) \leq k$.

Computational Complexity

Theorem. Given a set $\mathcal{I}$ of intervals and a positive integer k, it is NP-hard to decide whether $\chi(\mathcal{C}[\mathcal{I}]) \leq k$.

Proof. By reduction from (exact) 3-Sat, where each clause has exactly 3 literals.

Computational Complexity

Theorem. Given a set $\mathcal{I}$ of intervals and a positive integer k, it is NP-hard to decide whether $\chi(\mathcal{C}[\mathcal{I}]) \leq k$.

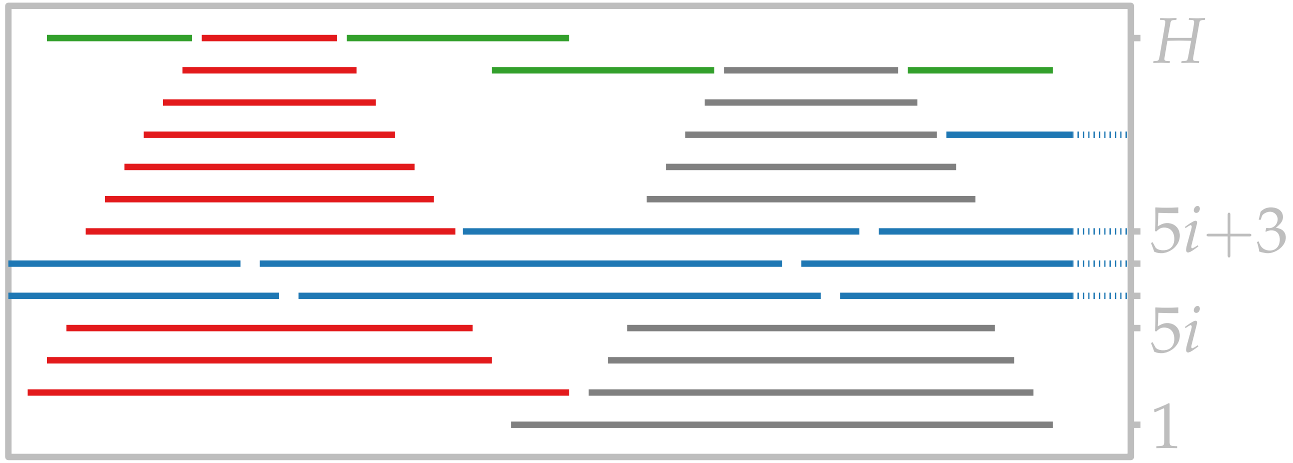
Proof. By reduction from (exact) 3-Sat, where each clause has exactly 3 literals.

Let $\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ be an instance of 3-Sat with variables $\{x_1, x_2, \ldots, x_n\}$, and let $H = 5m + 1$.

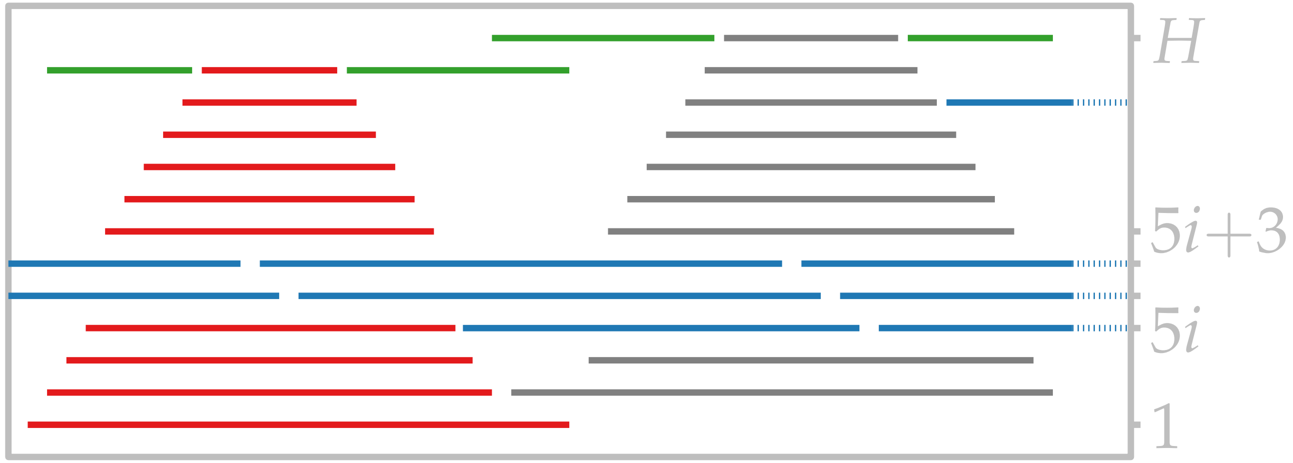
Computational Complexity

Theorem. Given a set $\mathcal{I}$ of intervals and a positive integer k, it is NP-hard to decide whether $\chi(\mathcal{C}[\mathcal{I}]) \leq k$.

Proof. By reduction from (exact) 3-SAT, where each clause has exactly 3 literals.



x true



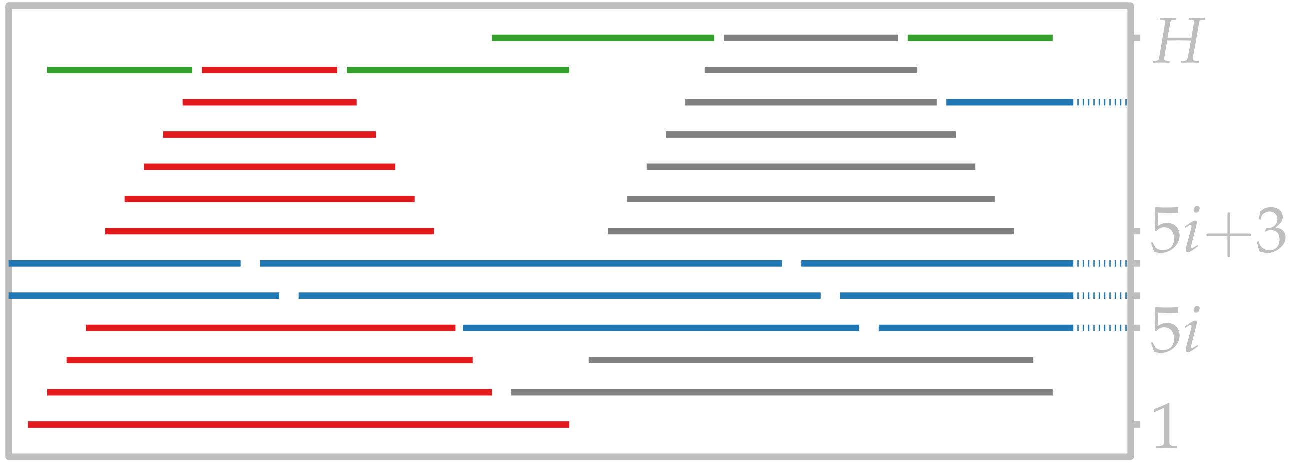
x false

Let $\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ be an instance of 3-SAT with variables $\{x_1, x_2, \ldots, x_n\}$, and let $H = 5m + 1$.

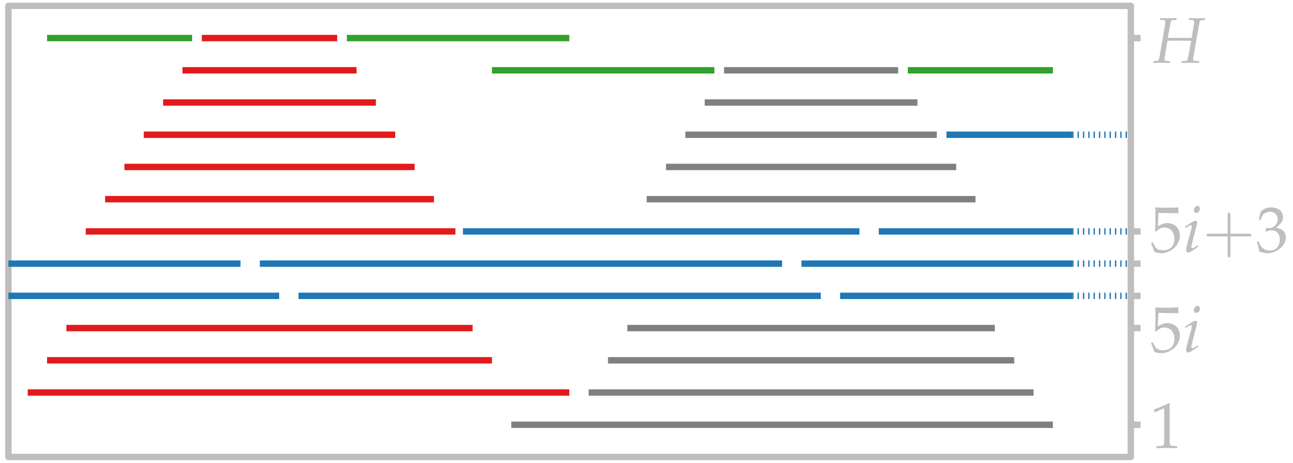
Computational Complexity

Theorem. Given a set $\mathcal{I}$ of intervals and a positive integer k, it is NP-hard to decide whether $\chi(\mathcal{C}[\mathcal{I}]) \le k$.

Proof. By reduction from (exact) 3-Sat, where each clause has exactly 3 literals.



x false



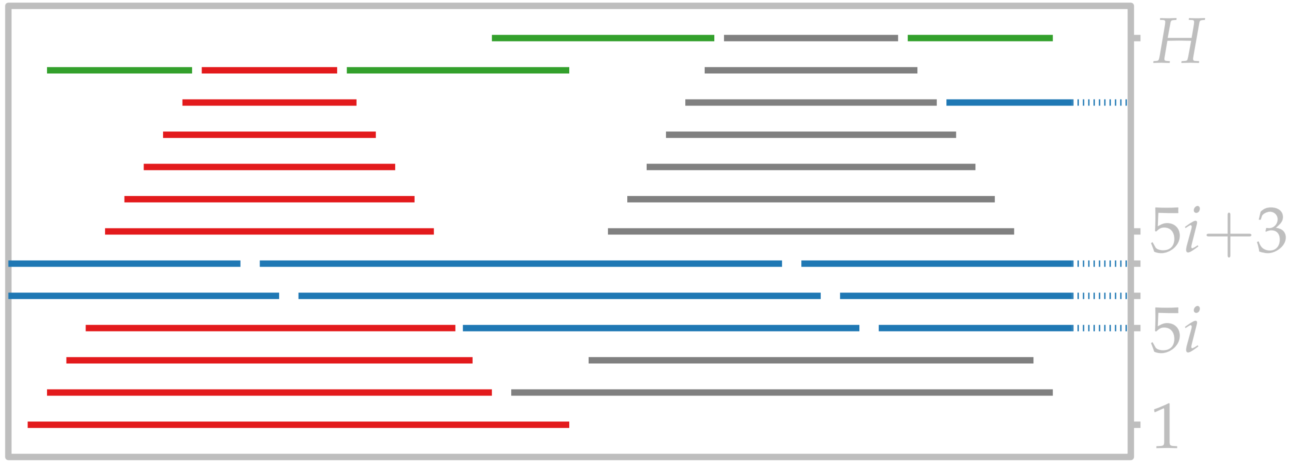
x true

Let $\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ be an instance of 3-Sat with variables $\{x_1, x_2, \ldots, x_n\}$, and let $H = 5m + 1$.

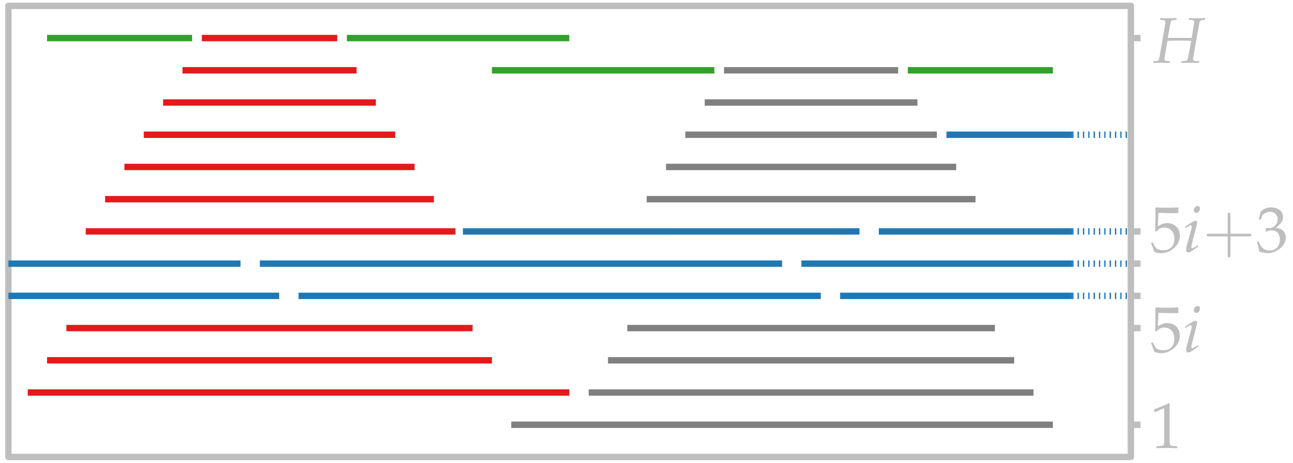
Computational Complexity

Theorem. Given a set $\mathcal{I}$ of intervals and a positive integer k, it is NP-hard to decide whether $\chi(\mathcal{C}[\mathcal{I}]) \leq k$.

Proof. By reduction from (exact) 3-SAT, where each clause has exactly 3 literals.



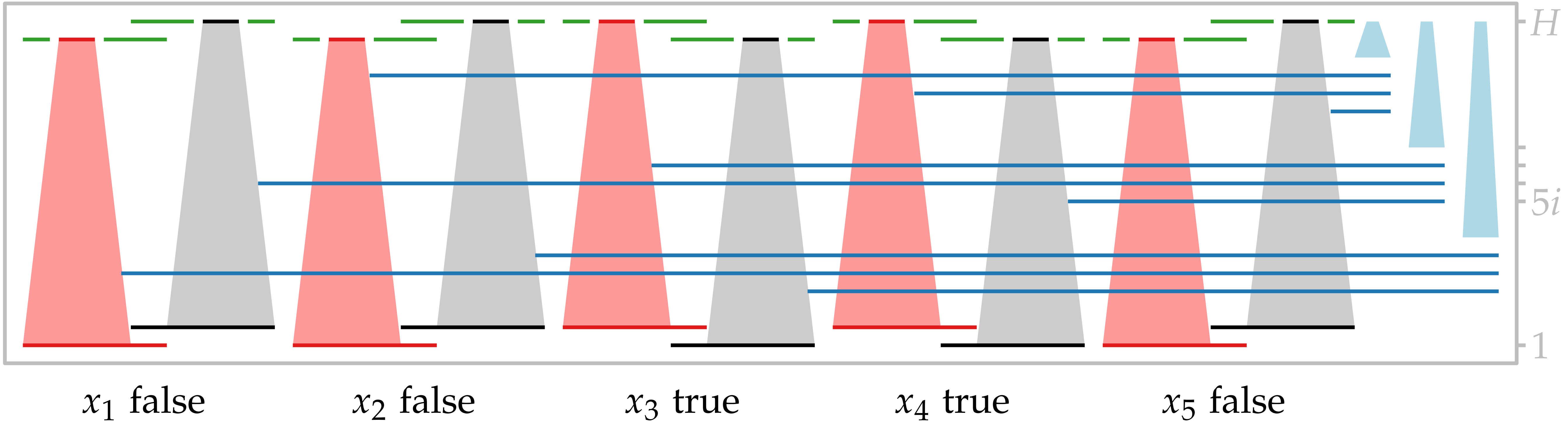
x false



x true

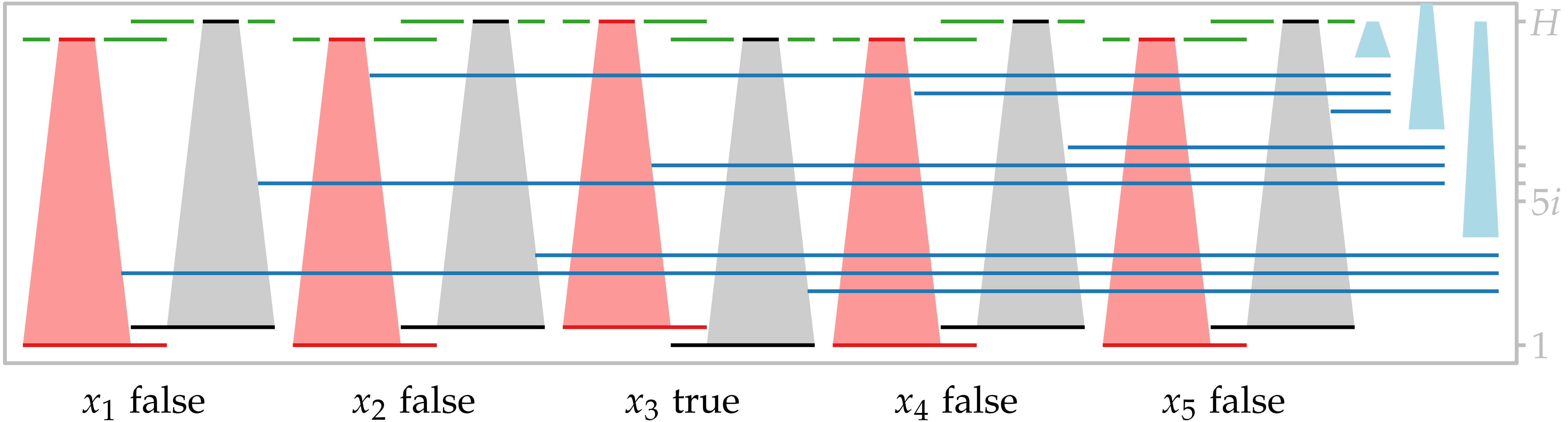
Let $\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ be an instance of 3-SAT with variables $\{x_1, x_2, \ldots, x_n\}$, and let $H = 5m + 1$. We construct a set $\mathcal{I}_\varphi$ of intervals.

Clause Gadget



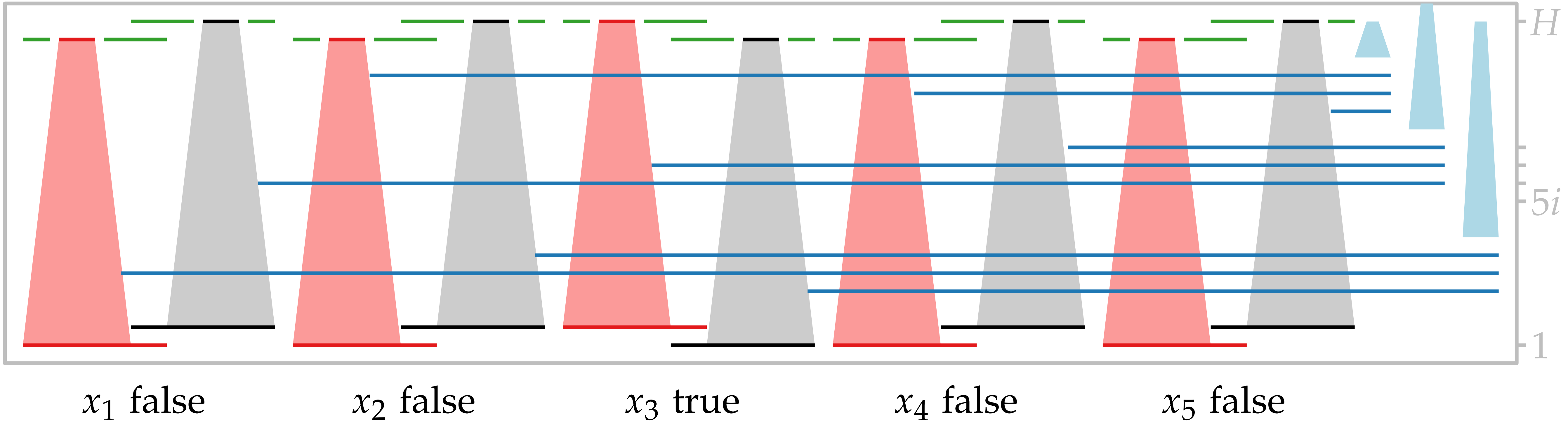
Example for $(\neg x_2 \vee \neg x_4 \vee x_5) \wedge (x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3)$.

Clause Gadget



Example for $(\neg x_2 \vee \neg x_4 \vee x_5) \wedge (x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3)$.

Clause Gadget



Example for $(\neg x_2 \vee \neg x_4 \vee x_5) \wedge (x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3)$.

The graph $\mathcal{C}[\mathcal{I}_\varphi]$ admits a coloring with H colors $\Leftrightarrow$ φ is satisfiable. □

Bidirectional Intervals

Theorem. Given a set $\mathcal{I}$ of intervals, $\varphi \colon \mathcal{I} \to \{\mathsf{left}, \mathsf{right}\}$, and $k \in \mathbb{N}$, it is NP-hard to decide whether $\chi(\mathcal{B}[\mathcal{I}, \varphi]) \leq k$.

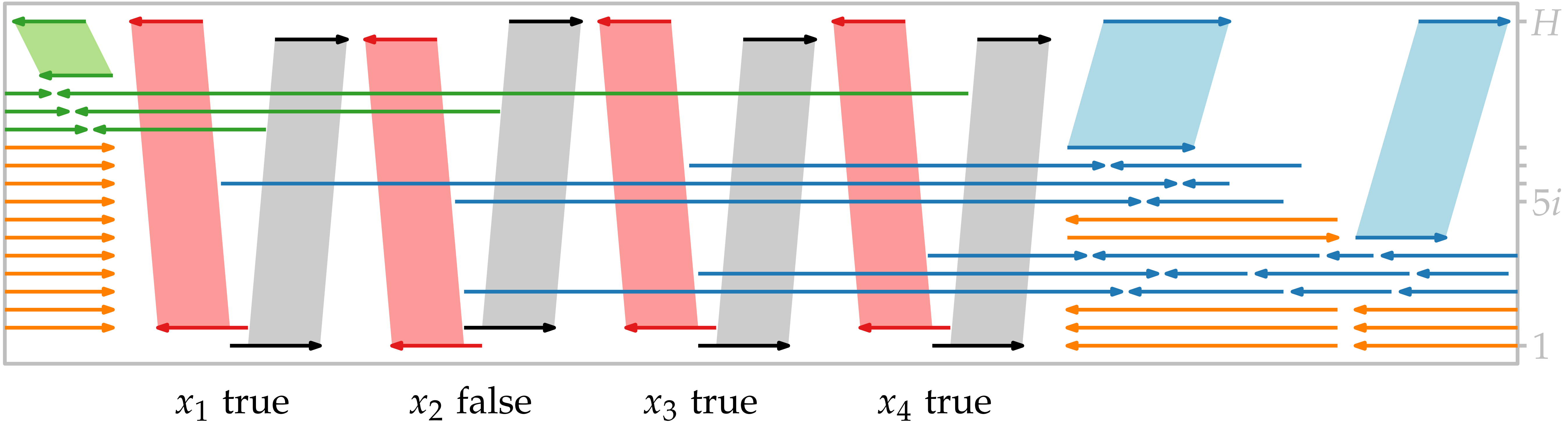
Bidirectional Intervals

Theorem. Given a set $\mathcal{I}$ of intervals, $\varphi\colon \mathcal{I} \to \{\mathsf{left}, \mathsf{right}\}$, and $k \in \mathbb{N}$, it is NP-hard to decide whether $\chi(\underbrace{\mathcal{B}[\mathcal{I}, \varphi]}_{\text{mixed intersection graph of bidirectional intervals}}) \leq k$.

Bidirectional Intervals

Theorem. Given a set $\mathcal{I}$ of intervals, $\varphi\colon \mathcal{I} \to \{\text{left}, \text{right}\}$, and $k \in \mathbb{N}$, it is NP-hard to decide whether $\chi(\underbrace{\mathcal{B}[\mathcal{I}, \varphi]}_{\text{mixed intersection graph of bidirectional intervals}}) \leq k$.

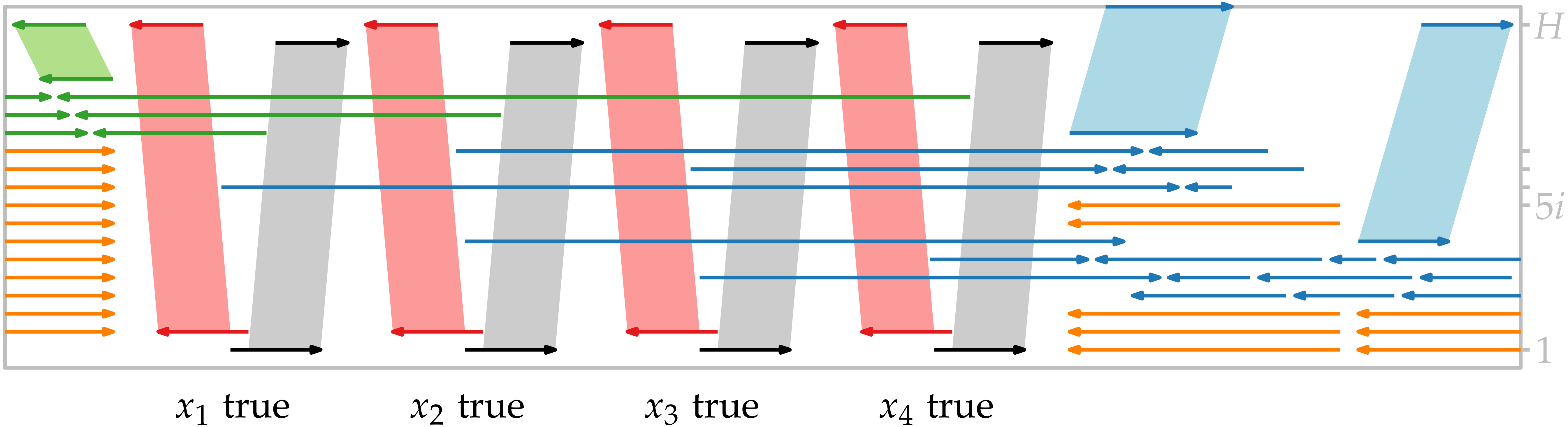
Proof sketch.



Bidirectional Intervals

Theorem. Given a set $\mathcal{I}$ of intervals, $\varphi\colon \mathcal{I} \to \{\text{left}, \text{right}\}$, and $k \in \mathbb{N}$, it is NP-hard to decide whether $\chi(\underbrace{\mathcal{B}[\mathcal{I}, \varphi]}_{\text{mixed intersection graph of bidirectional intervals}}) \leq k$.

Proof sketch.



Mixed Interval Graphs

Recall that a *mixed interval graph* is an interval graph where two intersecting intervals are connected by an edge or an arc in either direction.

Mixed Interval Graphs

Recall that a *mixed interval graph* is an interval graph where two intersecting intervals are connected by an edge or an arc in either direction.

If G is a mixed interval graph, then clearly $\chi(G) \geq \omega(G)$.

Mixed Interval Graphs

Recall that a *mixed interval graph* is an interval graph where two intersecting intervals are connected by an edge or an arc in either direction.

If G is a mixed interval graph, then clearly $\chi(G) \geq \omega(G)$.

Let $\lambda(G)$ denote the length of a longest directed path in G.

Mixed Interval Graphs

Recall that a *mixed interval graph* is an interval graph where two intersecting intervals are connected by an edge or an arc in either direction.

If G is a mixed interval graph, then clearly $\chi(G) \geq \omega(G)$.

Let $\lambda(G)$ denote the length of a longest directed path in G.

Then clearly $\chi(G) \geq \lambda(G) + 1$.

Mixed Interval Graphs

Recall that a *mixed interval graph* is an interval graph where two intersecting intervals are connected by an edge or an arc in either direction.

If G is a mixed interval graph, then clearly $\chi(G) \geq \omega(G)$.

Let $\lambda(G)$ denote the length of a longest directed path in G.

Then clearly $\chi(G) \geq \lambda(G) + 1$. Hence, $\chi(G) \geq \max\{\omega(G), \lambda(G) + 1\}$.

Mixed Interval Graphs

Recall that a *mixed interval graph* is an interval graph where two intersecting intervals are connected by an edge or an arc in either direction.

If G is a mixed interval graph, then clearly $\chi(G) \geq \omega(G)$.

Let $\lambda(G)$ denote the length of a longest directed path in G.

Then clearly $\chi(G) \geq \lambda(G) + 1$. Hence, $\chi(G) \geq \max\{\omega(G), \lambda(G) + 1\}$.

Theorem. Let G be a mixed interval graph without directed cycles. Then $\chi(G) \leq (\lambda(G) + 1) \cdot \omega(G)$.

Mixed Interval Graphs

Recall that a *mixed interval graph* is an interval graph where two intersecting intervals are connected by an edge or an arc in either direction.

If G is a mixed interval graph, then clearly $\chi(G) \geq \omega(G)$.

Let $\lambda(G)$ denote the length of a longest directed path in G.

Then clearly $\chi(G) \geq \lambda(G) + 1$. Hence, $\chi(G) \geq \max\{\omega(G), \lambda(G) + 1\}$.

Theorem. Let G be a mixed interval graph without directed cycles. Then $\chi(G) \leq (\lambda(G) + 1) \cdot \omega(G)$.

Our constructive proof yields a $\min\{\omega(G), \lambda(G) + 1\}$-approximation algorithm.

A Constructive Proof

Theorem. Let G be a mixed interval graph without directed cycles. Then $\chi(G) \leq (\lambda(G)+1) \cdot \omega(G)$.

A Constructive Proof

Theorem. Let G be a mixed interval graph without directed cycles. Then $\chi(G) \leq (\lambda(G)+1) \cdot \omega(G)$.

Proof. Let $G^{\rightarrow}$ be the graph obtained from G by removing all edges.

A Constructive Proof

Theorem. Let G be a mixed interval graph without directed cycles. Then $\chi(G) \leq (\lambda(G) + 1) \cdot \omega(G)$.

Proof. Let $G^{\rightarrow}$ be the graph obtained from G by removing all edges.

Clearly $G^{\rightarrow}$ is a DAG.

A Constructive Proof

Theorem. Let G be a mixed interval graph without directed cycles. Then $\chi(G) \leq (\lambda(G)+1) \cdot \omega(G)$.

Proof. Let $G^{\rightarrow}$ be the graph obtained from G by removing all edges.

Clearly $G^{\rightarrow}$ is a DAG. Partition $V(G)$ into *layers* $L_0, L_1, \ldots$ as follows.

A Constructive Proof

Theorem. Let G be a mixed interval graph without directed cycles. Then $\chi(G) \leq (\lambda(G) + 1) \cdot \omega(G)$.

Proof. Let $G^{\rightarrow}$ be the graph obtained from G by removing all edges.

Clearly $G^{\rightarrow}$ is a DAG. Partition $V(G)$ into *layers* $L_0, L_1, \ldots$ as follows.

Let L_0 be the set of sources in $G^{\rightarrow}$, i.e., the vertices without incoming arcs.

A Constructive Proof

Theorem. Let G be a mixed interval graph without directed cycles. Then $\chi(G) \leq (\lambda(G) + 1) \cdot \omega(G)$.

Proof. Let $G^{\rightarrow}$ be the graph obtained from G by removing all edges.

Clearly $G^{\rightarrow}$ is a DAG. Partition $V(G)$ into *layers* $L_0, L_1, \ldots$ as follows.

Let L_0 be the set of sources in $G^{\rightarrow}$, i.e., the vertices without incoming arcs.

For $i = 1, 2, \ldots$, let L_i be the set of sources in $G^{\rightarrow} \setminus \bigcup_{j=0}^{i-1} L_j$.

A Constructive Proof

Theorem. Let G be a mixed interval graph without directed cycles. Then $\chi(G) \leq (\lambda(G)+1) \cdot \omega(G)$.

Proof. Let $G^{\rightarrow}$ be the graph obtained from G by removing all edges.

Clearly $G^{\rightarrow}$ is a DAG. Partition $V(G)$ into *layers* $L_0, L_1, \ldots$ as follows.

Let L_0 be the set of sources in $G^{\rightarrow}$, i.e., the vertices without incoming arcs.

For $i = 1, 2, \ldots$, let L_i be the set of sources in $G^{\rightarrow} \setminus \bigcup_{j=0}^{i-1} L_j$.

Note that $\lambda(G) = \max\{i \colon L_i \neq \emptyset\}$.

A Constructive Proof

Theorem. Let G be a mixed interval graph without directed cycles. Then $\chi(G) \leq (\lambda(G)+1) \cdot \omega(G)$.

Proof. Let $G^{\rightarrow}$ be the graph obtained from G by removing all edges.

Clearly $G^{\rightarrow}$ is a DAG. Partition $V(G)$ into *layers* $L_0, L_1, \dots$ as follows.

Let L_0 be the set of sources in $G^{\rightarrow}$, i.e., the vertices without incoming arcs.

For $i = 1, 2, \dots$, let L_i be the set of sources in $G^{\rightarrow} \setminus \bigcup_{j=0}^{i-1} L_j$.

Note that $\lambda(G) = \max\{i \colon L_i \neq \emptyset\}$.

For $x \in V(G)$, let $\ell(x) \in \{0, \dots, \lambda(G)\}$ be the *layer* of x.

A Constructive Proof

Theorem. Let G be a mixed interval graph without directed cycles. Then $\chi(G) \leq (\lambda(G)+1) \cdot \omega(G)$.

Proof. Let $G^{\rightarrow}$ be the graph obtained from G by removing all edges.

Clearly $G^{\rightarrow}$ is a DAG. Partition $V(G)$ into *layers* $L_0, L_1, \dots$ as follows.

Let L_0 be the set of sources in $G^{\rightarrow}$, i.e., the vertices without incoming arcs.

For $i = 1, 2, \dots$, let L_i be the set of sources in $G^{\rightarrow} \setminus \bigcup_{j=0}^{i-1} L_j$.

Note that $\lambda(G) = \max\{i \colon L_i \neq \emptyset\}$.

For $x \in V(G)$, let $\ell(x) \in \{0, \dots, \lambda(G)\}$ be the *layer* of x.

Let $U(G)$ be the *underlying undirected graph* of G.

A Constructive Proof

Theorem. Let G be a mixed interval graph without directed cycles. Then $\chi(G) \leq (\lambda(G)+1) \cdot \omega(G)$.

Proof. Let $G^{\rightarrow}$ be the graph obtained from G by removing all edges.

Clearly $G^{\rightarrow}$ is a DAG. Partition $V(G)$ into *layers* $L_0, L_1, \dots$ as follows.

Let L_0 be the set of sources in $G^{\rightarrow}$, i.e., the vertices without incoming arcs.

For $i = 1, 2, \dots$, let L_i be the set of sources in $G^{\rightarrow} \setminus \bigcup_{j=0}^{i-1} L_j$.

Note that $\lambda(G) = \max\{i \colon L_i \neq \emptyset\}$.

For $x \in V(G)$, let $\ell(x) \in \{0, \dots, \lambda(G)\}$ be the *layer* of x.

Let $U(G)$ be the *underlying undirected graph* of G.

$U(G)$ is an interval graph,

A Constructive Proof

Theorem. Let G be a mixed interval graph without directed cycles. Then $\chi(G) \leq (\lambda(G) + 1) \cdot \omega(G)$.

Proof. Let $G^{\rightarrow}$ be the graph obtained from G by removing all edges.

Clearly $G^{\rightarrow}$ is a DAG. Partition $V(G)$ into *layers* $L_0, L_1, \ldots$ as follows.

Let L_0 be the set of sources in $G^{\rightarrow}$, i.e., the vertices without incoming arcs.

For $i = 1, 2, \ldots$, let L_i be the set of sources in $G^{\rightarrow} \setminus \bigcup_{j=0}^{i-1} L_j$.

Note that $\lambda(G) = \max\{i \colon L_i \neq \emptyset\}$.

For $x \in V(G)$, let $\ell(x) \in \{0, \ldots, \lambda(G)\}$ be the *layer* of x.

Let $U(G)$ be the *underlying undirected graph* of G.

$U(G)$ is an interval graph, hence $\chi(U(G)) = \omega(U(G)) = \omega(G)$.

A Constructive Proof

Theorem. Let G be a mixed interval graph without directed cycles. Then $\chi(G) \leq (\lambda(G)+1) \cdot \omega(G)$.

Proof. Let $c \colon V \to \{1, 2, \dots, \omega(U(G))\}$ be an optimal coloring of $U(G)$.

$U(G)$ is an interval graph, hence $\chi(U(G)) = \omega(U(G)) = \omega(G)$.

A Constructive Proof

Theorem. Let G be a mixed interval graph without directed cycles. Then $\chi(G) \leq (\lambda(G)+1) \cdot \omega(G)$.

Proof. Let $c \colon V \to \{1, 2, \dots, \omega(U(G))\}$ be an optimal coloring of $U(G)$.

Define a mapping f. For a vertex x of G, let $f(x) =$

A Constructive Proof

Theorem. Let G be a mixed interval graph without directed cycles. Then $\chi(G) \leq (\lambda(G)+1) \cdot \omega(G)$.

Proof. Let $c\colon V \to \{1,2,\ldots,\omega(U(G))\}$ be an optimal coloring of $U(G)$.

Define a mapping f. For a vertex x of G, let $f(x) = \ell(x) \cdot \omega(G) + c(x)$.

A Constructive Proof

Theorem. Let G be a mixed interval graph without directed cycles. Then $\chi(G) \leq (\lambda(G) + 1) \cdot \omega(G)$.

Proof. Let $c\colon V \to \{1, 2, \ldots, \omega(U(G))\}$ be an optimal coloring of $U(G)$.

Define a mapping f. For a vertex x of G, let $f(x) = \ell(x) \cdot \omega(G) + c(x)$.

Note that $1 \leq f(x) \leq (\lambda(G) + 1) \cdot \omega(G)$.

A Constructive Proof

Theorem. Let G be a mixed interval graph without directed cycles. Then $\chi(G) \leq (\lambda(G) + 1) \cdot \omega(G)$.

Proof. Let $c \colon V \to \{1, 2, \ldots, \omega(U(G))\}$ be an optimal coloring of $U(G)$.

Define a mapping f. For a vertex x of G, let $f(x) = \ell(x) \cdot \omega(G) + c(x)$.

Note that $1 \leq f(x) \leq (\lambda(G) + 1) \cdot \omega(G)$. We claim that f colors G.

A Constructive Proof

Theorem. Let G be a mixed interval graph without directed cycles. Then $\chi(G) \leq (\lambda(G)+1) \cdot \omega(G)$.

Proof. Let $c\colon V \to \{1,2,\ldots,\omega(U(G))\}$ be an optimal coloring of $U(G)$.

Define a mapping f. For a vertex x of G, let $f(x) = \ell(x) \cdot \omega(G) + c(x)$.

Note that $1 \leq f(x) \leq (\lambda(G)+1) \cdot \omega(G)$. We claim that f colors G.

If $\{x,y\}$ is an edge of G, then

A Constructive Proof

Theorem. Let G be a mixed interval graph without directed cycles. Then $\chi(G) \leq (\lambda(G) + 1) \cdot \omega(G)$.

Proof. Let $c\colon V \to \{1, 2, \ldots, \omega(U(G))\}$ be an optimal coloring of $U(G)$.

Define a mapping f. For a vertex x of G, let $f(x) = \ell(x) \cdot \omega(G) + c(x)$.

Note that $1 \leq f(x) \leq (\lambda(G) + 1) \cdot \omega(G)$. We claim that f colors G.

If $\{x, y\}$ is an edge of G, then $c(x) \neq c(y)$

A Constructive Proof

Theorem. Let G be a mixed interval graph without directed cycles. Then $\chi(G) \leq (\lambda(G) + 1) \cdot \omega(G)$.

Proof. Let $c\colon V \to \{1, 2, \dots, \omega(U(G))\}$ be an optimal coloring of $U(G)$.

Define a mapping f. For a vertex x of G, let $f(x) = \ell(x) \cdot \omega(G) + c(x)$.

Note that $1 \leq f(x) \leq (\lambda(G) + 1) \cdot \omega(G)$. We claim that f colors G.

If $\{x, y\}$ is an edge of G, then $c(x) \neq c(y)$ and hence, $f(x) \neq f(y)$.

A Constructive Proof

Theorem. Let G be a mixed interval graph without directed cycles. Then $\chi(G) \leq (\lambda(G)+1) \cdot \omega(G)$.

Proof. Let $c\colon V \to \{1,2,\ldots,\omega(U(G))\}$ be an optimal coloring of $U(G)$.

Define a mapping f. For a vertex x of G, let $f(x) = \ell(x) \cdot \omega(G) + c(x)$.

Note that $1 \leq f(x) \leq (\lambda(G)+1) \cdot \omega(G)$. We claim that f colors G.

If $\{x,y\}$ is an edge of G, then $c(x) \neq c(y)$ and hence, $f(x) \neq f(y)$.
If (x,y) is an arc of G, then

A Constructive Proof

Theorem. Let G be a mixed interval graph without directed cycles. Then $\chi(G) \leq (\lambda(G)+1) \cdot \omega(G)$.

Proof. Let $c\colon V \to \{1,2,\ldots,\omega(U(G))\}$ be an optimal coloring of $U(G)$.

Define a mapping f. For a vertex x of G, let $f(x) = \ell(x) \cdot \omega(G) + c(x)$.

Note that $1 \leq f(x) \leq (\lambda(G)+1) \cdot \omega(G)$. We claim that f colors G.

If $\{x,y\}$ is an edge of G, then $c(x) \neq c(y)$ and hence, $f(x) \neq f(y)$.
If (x,y) is an arc of G, then $\ell(x) < \ell(y)$

A Constructive Proof

Theorem. Let G be a mixed interval graph without directed cycles. Then $\chi(\mathrm{G}) \leq (\lambda(\mathrm{G})+1) \cdot \omega(\mathrm{G})$.

Proof. Let $c\colon V \to \{1,2,\ldots,\omega(U(G))\}$ be an optimal coloring of $U(G)$.

Define a mapping f. For a vertex x of G, let $f(x) = \ell(x) \cdot \omega(\mathrm{G}) + c(x)$.

Note that $1 \leq f(x) \leq (\lambda(\mathrm{G})+1) \cdot \omega(\mathrm{G})$. We claim that f colors G.

If $\{x,y\}$ is an edge of G, then $c(x) \neq c(y)$ and hence, $f(x) \neq f(y)$.
If (x,y) is an arc of G, then $\ell(x) < \ell(y)$ and hence, $f(x) < f(y)$. □

A Lower Bound Example

Proposition. There is an infinite family $(G_k)_{k \geq 1}$ of mixed interval graphs with $|V(G_k)| = 2k^2$, $\lambda(G_k) = k - 1$, $\omega(G_k) = 2k$, and $\chi(G_k) = (k+1) \cdot k = (\lambda(G_k) + 2) \cdot \omega(G_k)/2$.

A Lower Bound Example

Proposition. There is an infinite family $(G_k)_{k\geq 1}$ of mixed interval graphs with $|V(G_k)| = 2k^2$, $\lambda(G_k) = k-1$, $\omega(G_k) = 2k$, and $\chi(G_k) = (k+1)\cdot k = (\lambda(G_k)+2)\cdot\omega(G_k)/2$.

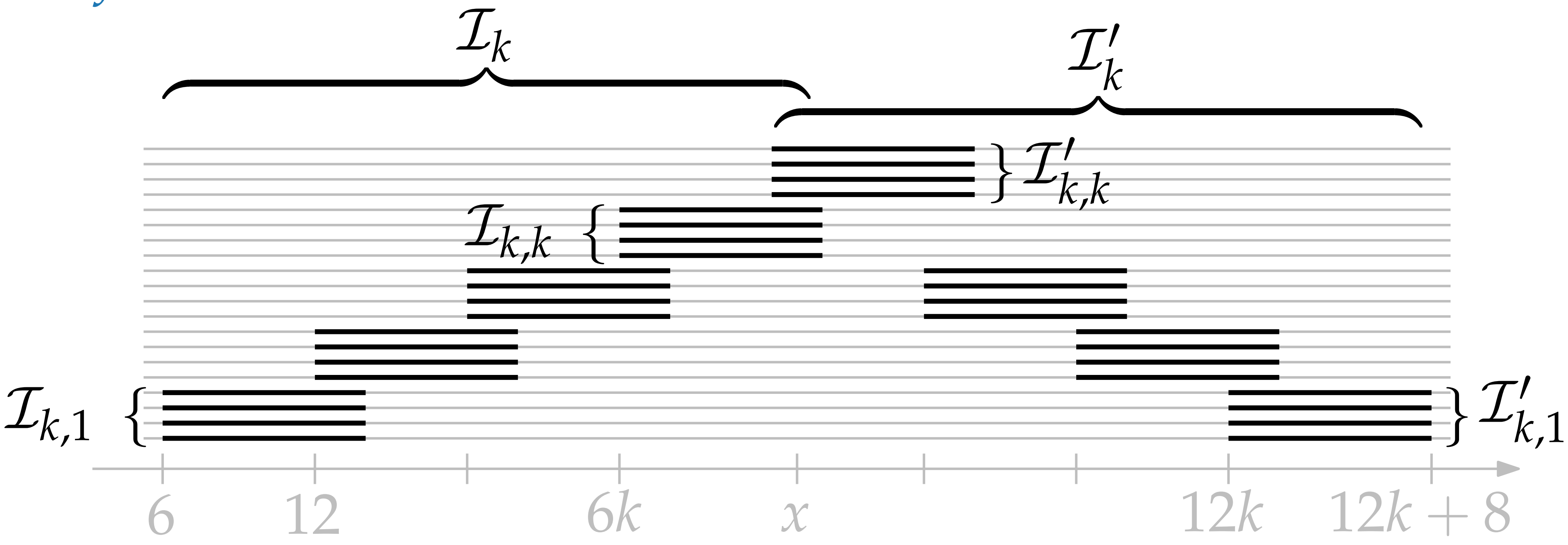
That is, our upper bound for $\chi(G)$, $(\lambda(G)+1)\cdot\omega(G)$, is asymptotically tight.

A Lower Bound Example

Proposition. There is an infinite family $(G_k)_{k\geq 1}$ of mixed interval graphs with $|V(G_k)| = 2k^2$, $\lambda(G_k) = k-1$, $\omega(G_k) = 2k$, and $\chi(G_k) = (k+1)\cdot k = (\lambda(G_k)+2)\cdot\omega(G_k)/2$.

That is, our upper bound for $\chi(G)$, $(\lambda(G)+1)\cdot\omega(G)$, is asymptotically tight.

Proof.

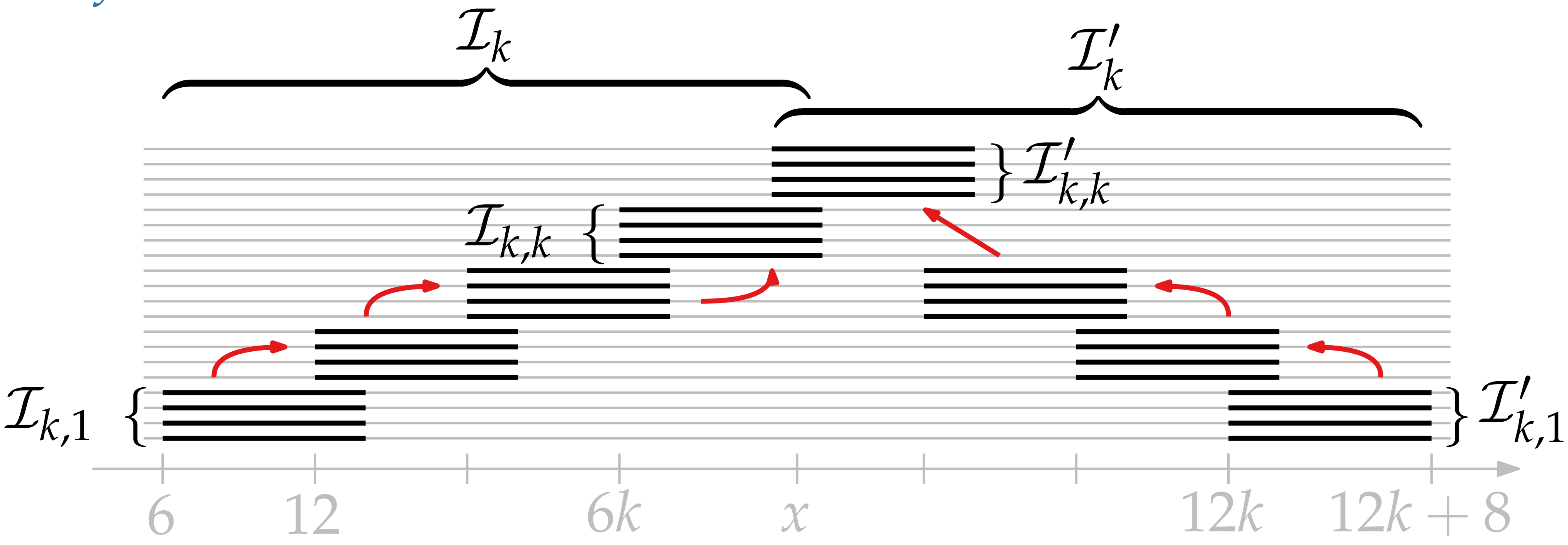


A Lower Bound Example

Proposition. There is an infinite family $(G_k)_{k\geq 1}$ of mixed interval graphs with $|V(G_k)| = 2k^2$, $\lambda(G_k) = k-1$, $\omega(G_k) = 2k$, and $\chi(G_k) = (k+1)\cdot k = (\lambda(G_k)+2)\cdot\omega(G_k)/2$.

That is, our upper bound for $\chi(G)$, $(\lambda(G)+1)\cdot\omega(G)$, is asymptotically tight.

Proof.



Summary

Mixed interval	Coloring				Recognition
graph class	complexity	lower bound	upper bound	approximation	
containment	NP-hard	$2\omega-1$	$2\omega-1$	2	$O(nm)$
directional	$O(n \log n)$			1	$O(n^2)$
bidirectional	NP-hard			2	open
general	NP-hard	$(\lambda+2)\omega/2$	$(\lambda+1)\omega$	$\min\{\omega, \lambda+1\}$	$O(n+m)$ [LB79]

Summary

Mixed interval	Coloring				Recognition
graph class	complexity	lower bound	upper bound	approximation	
containment	NP-hard	$2\omega-1$	$2\omega-1$	2	$O(nm)$
directional	$O(n\log n)$			1	$O(n^2)$
bidirectional	NP-hard			2	open
general	NP-hard	$(\lambda+2)\omega/2$	$(\lambda+1)\omega$	$\min\{\omega,\lambda+1\}$	$O(n+m)$ [LB79]

Summary

Mixed interval	Coloring				Recognition
graph class	complexity	lower bound	upper bound	approximation	
containment	NP-hard	$2\omega-1$	$2\omega-1$	2	$O(nm)$
directional	$O(n \log n)$			1	$O(n^2)$
bidirectional	NP-hard			2	open
general	NP-hard	$(\lambda+2)\omega/2$	$(\lambda+1)\omega$	$\min\{\omega, \lambda+1\}$	$O(n+m)$ [LB79]

Summary

Mixed interval		Coloring			Recognition
graph class	complexity	lower bound	upper bound	approximation	
containment	NP-hard	$2\omega-1$	$2\omega-1$	2	$O(nm)$
directional	$O(n \log n)$			1	$O(n^2)$
bidirectional	NP-hard			2	open
general	NP-hard	$(\lambda+2)\omega/2$	$(\lambda+1)\omega$	$\min\{\omega, \lambda+1\}$	$O(n+m)$ [LB79]

Follow-up Work

- Given a mixed graph G with an orientation φ, we can decide in linear time whether G admits an oriented interval representation that complies with φ.

Summary

Mixed interval	Coloring				Recognition
graph class	complexity	lower bound	upper bound	approximation	
containment	NP-hard	$2\omega-1$	$2\omega-1$	2	$O(nm)$
directional	$O(n\log n)$			1	$O(n^2)$
bidirectional	NP-hard			2	open
general	NP-hard	$(\lambda+2)\omega/2$	$(\lambda+1)\omega$	$\min\{\omega,\lambda+1\}$	$O(n+m)$ [LB79]

Follow-up Work

- Given a mixed graph G with an orientation φ, we can decide in linear time whether G admits an oriented interval representation that complies with φ.
- In particular, we can recognize directional interval graphs in linear time.