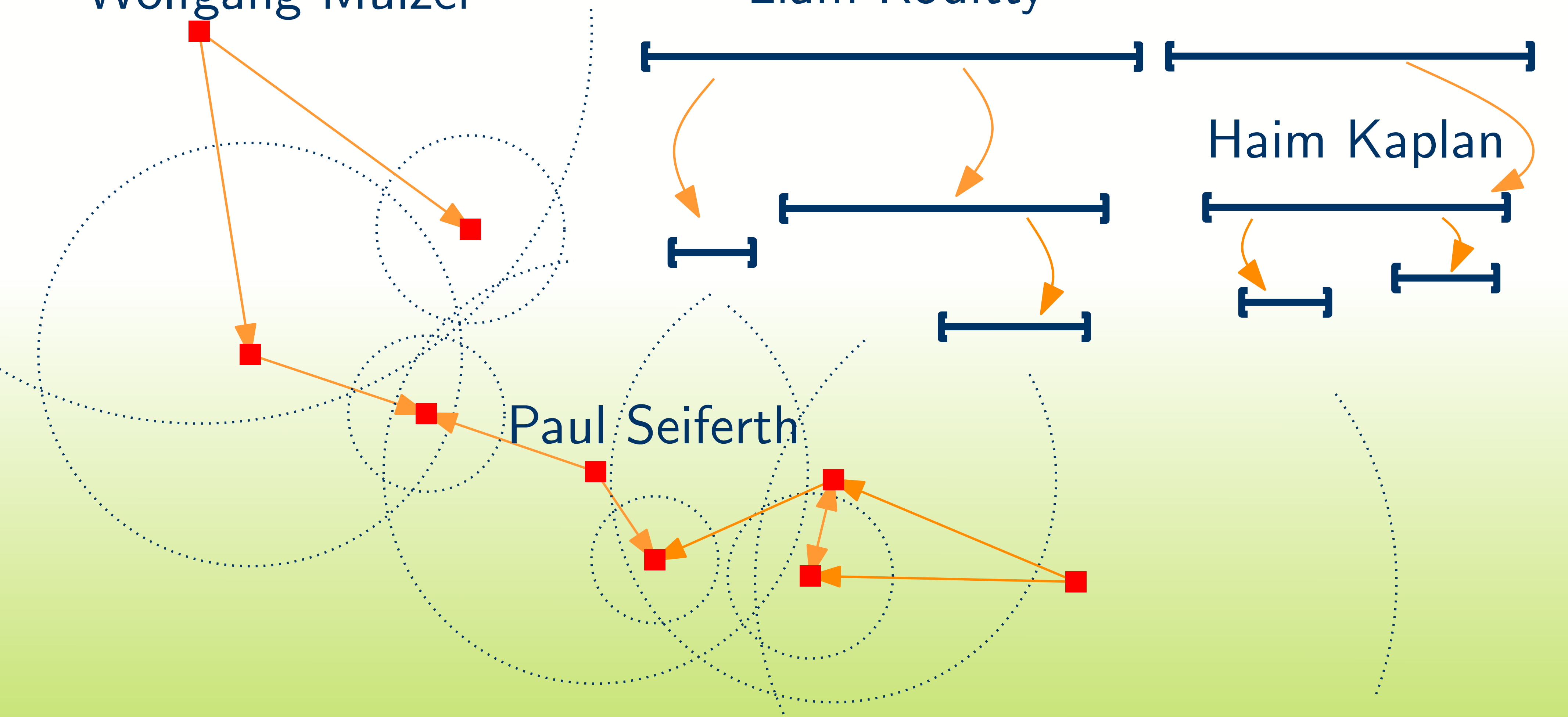
Reachability Oracles for Disk Transmission Graphs

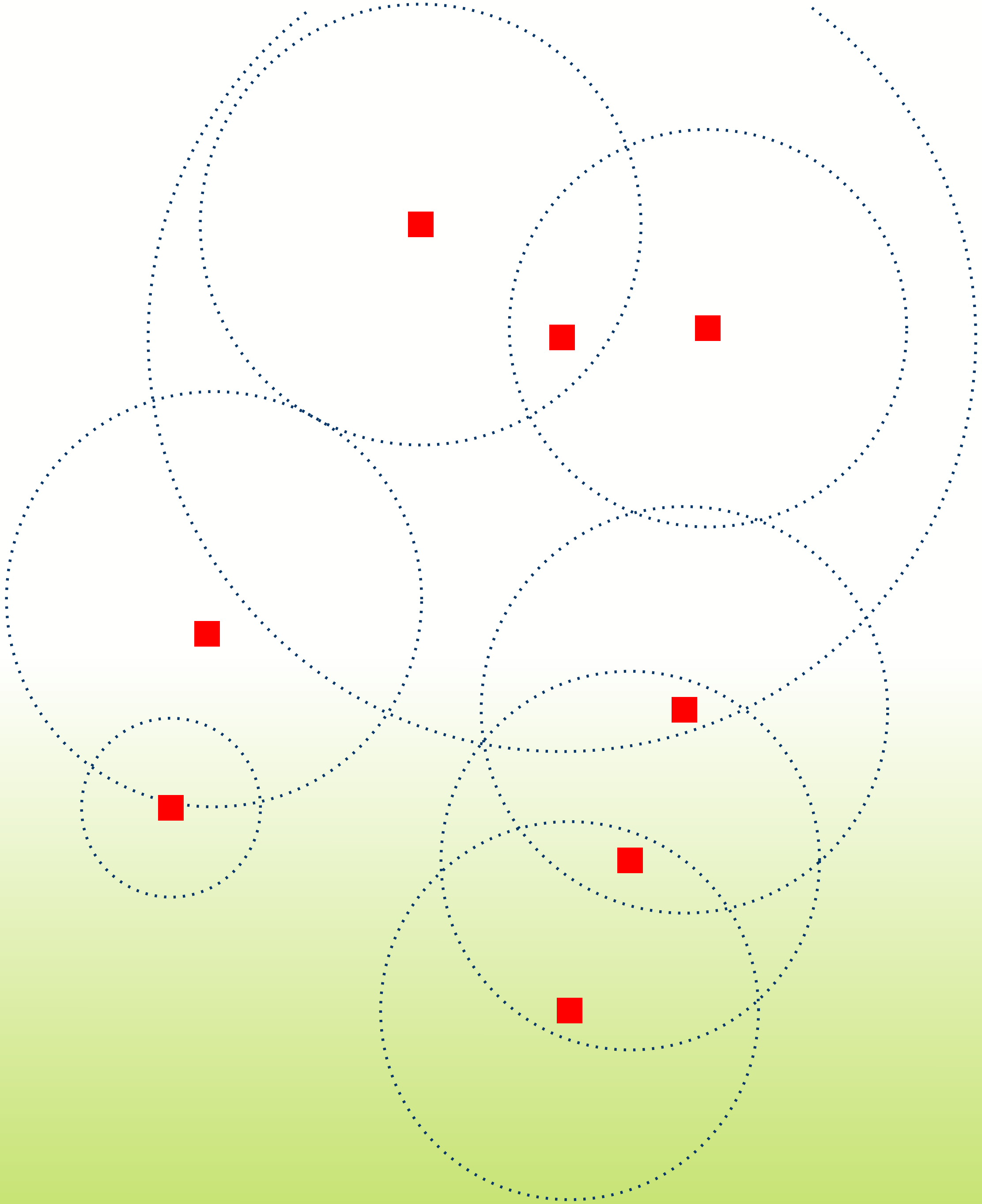
Wolfgang Mulzer
Liam Roditty
Haim Kaplan
Paul Seiferth



The Problem

Given: point set $P \subseteq \mathbb{R}^d$

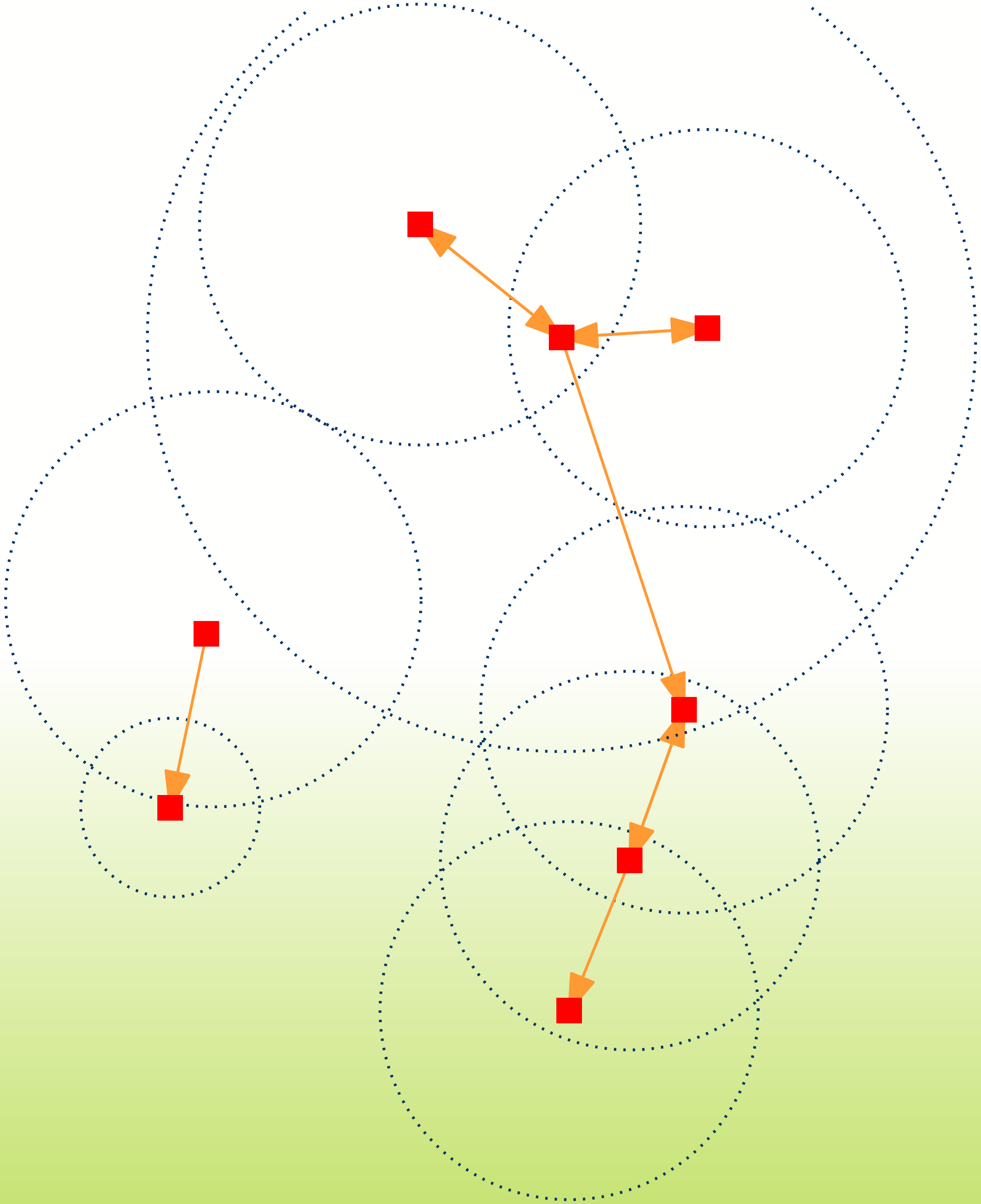
The Problem



Given: point set $P \subseteq \mathbb{R}^d$

each $p \in P$ has radius r_p

The Problem

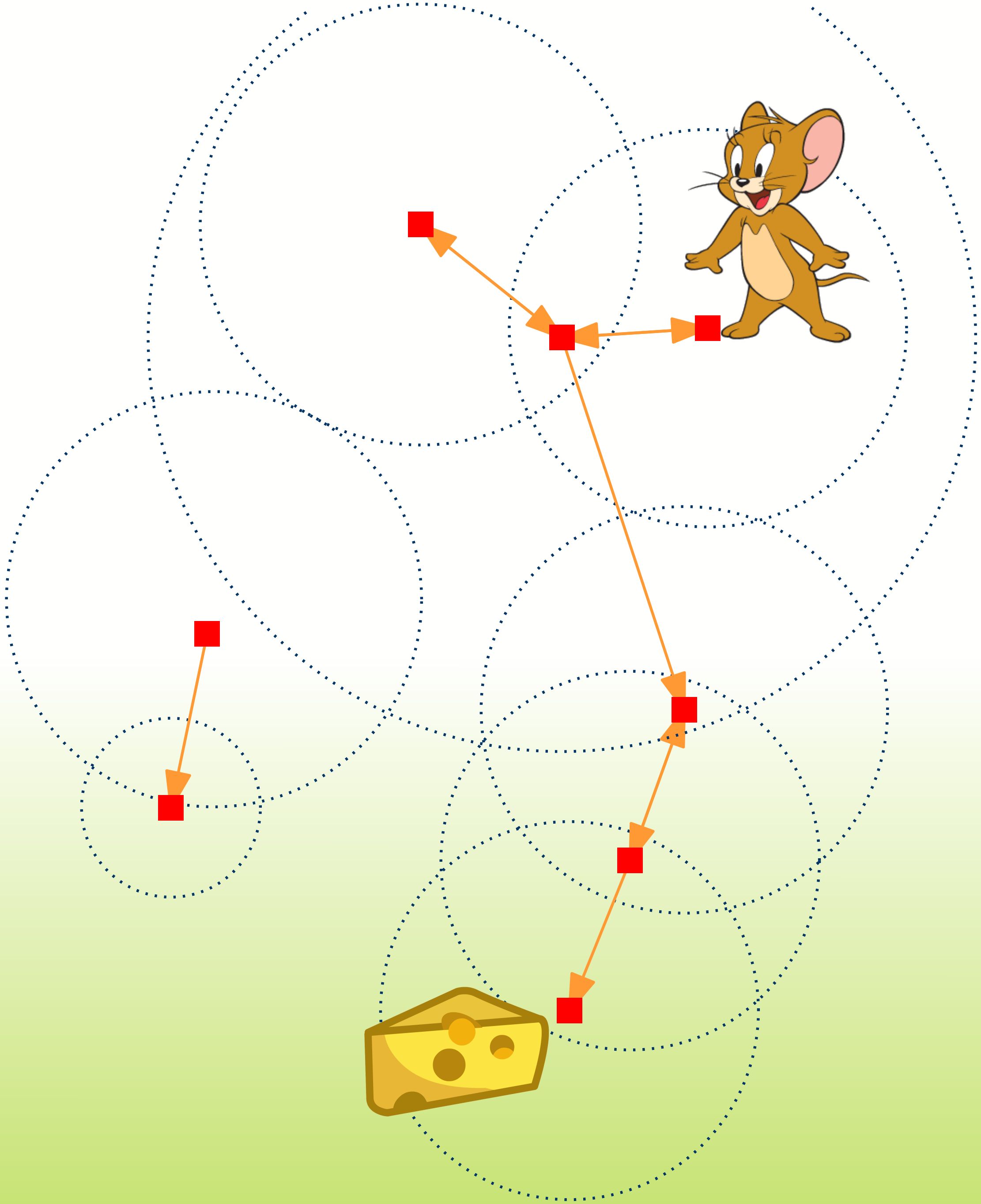


Given: point set $P \subseteq \mathbb{R}^d$

each $p \in P$ has radius r_p

$\Rightarrow$ directed graph on P

The Problem



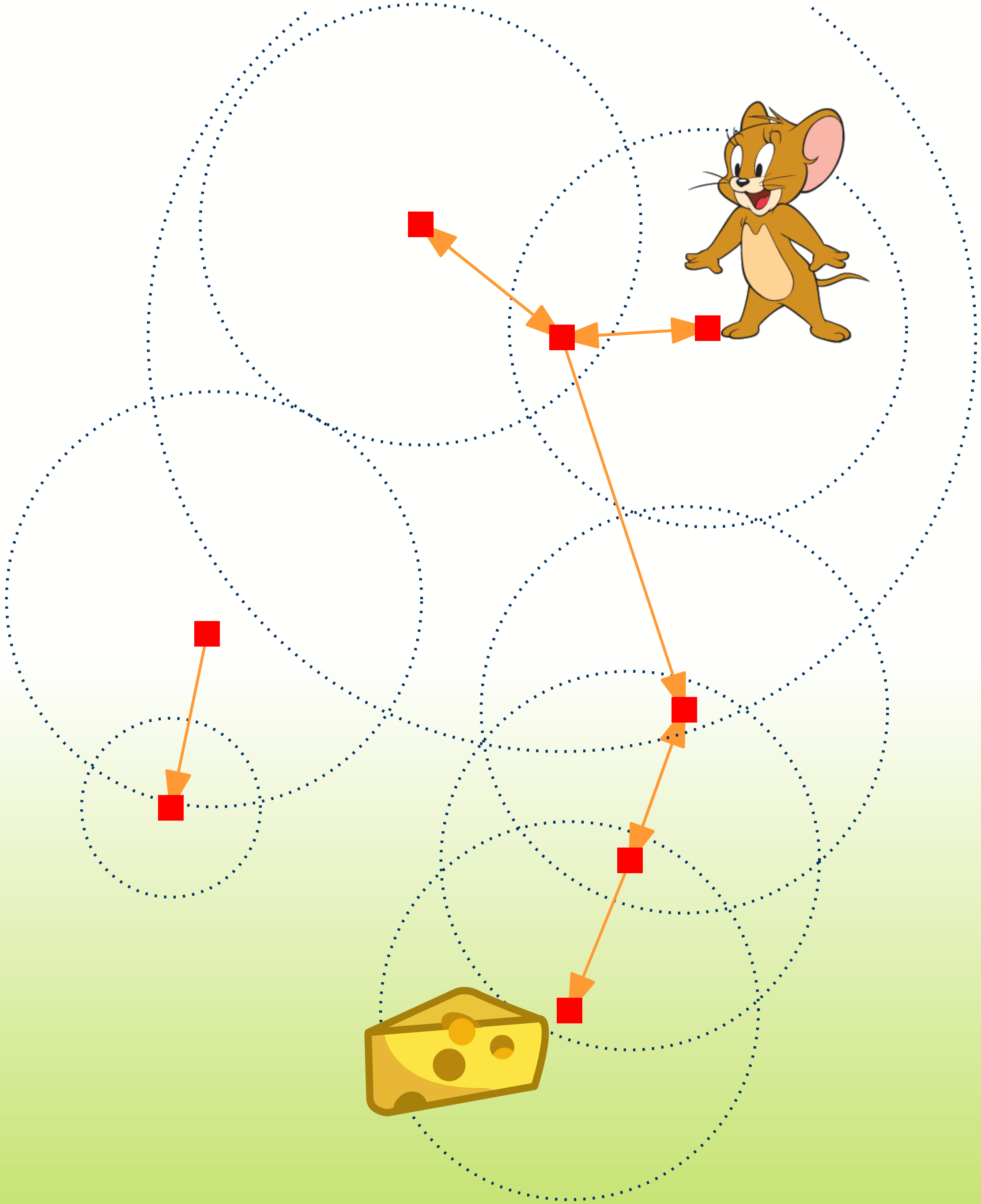
Given: point set $P \subseteq \mathbb{R}^d$

each $p \in P$ has radius r_p

$\Rightarrow$ directed graph on P

Want: A datastructure to answer *reachability queries*: is there a directed path between two vertices u, v?

The Problem



Given: point set $P \subseteq \mathbb{R}^d$

each $p \in P$ has radius r_p

$\Rightarrow$ directed graph on P

Want: A datastructure to answer *reachability queries*: is there a directed path between two vertices u, v?

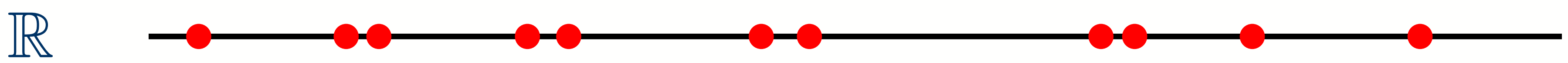
Quality measured in:

- Space S(n)
- Query Time Q(n)
- Preprocessing P(n)

1-dimensional case

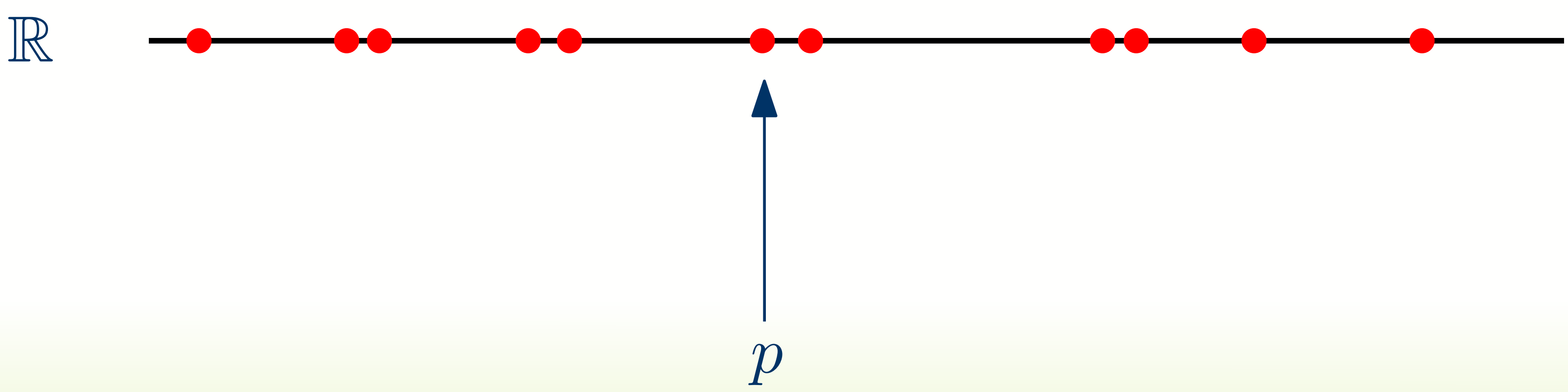
$\mathbb{R}$ 

1-dimensional case



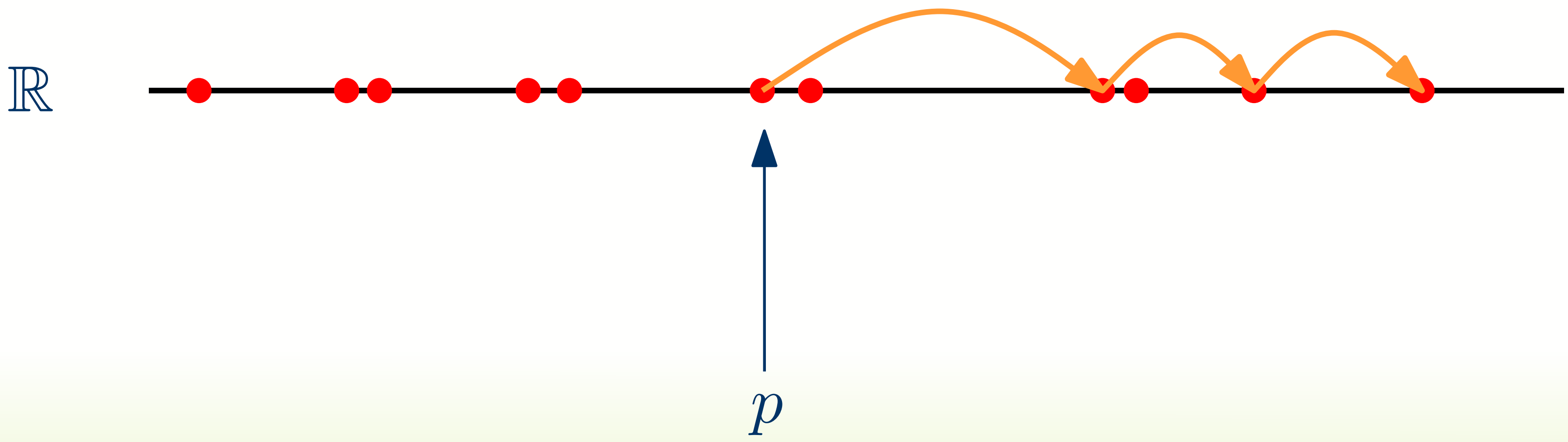
Observation: All points reachable from $p \in P$ lie in an interval.

1-dimensional case



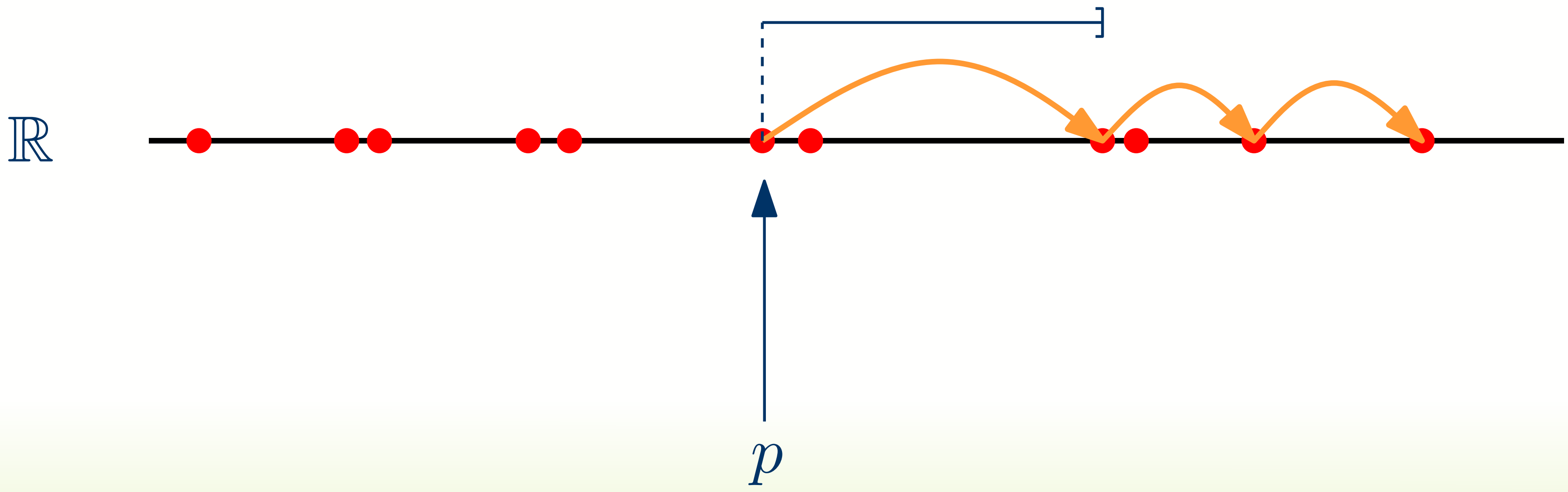
Observation: All points reachable from $p \in P$ lie in an interval.

1-dimensional case



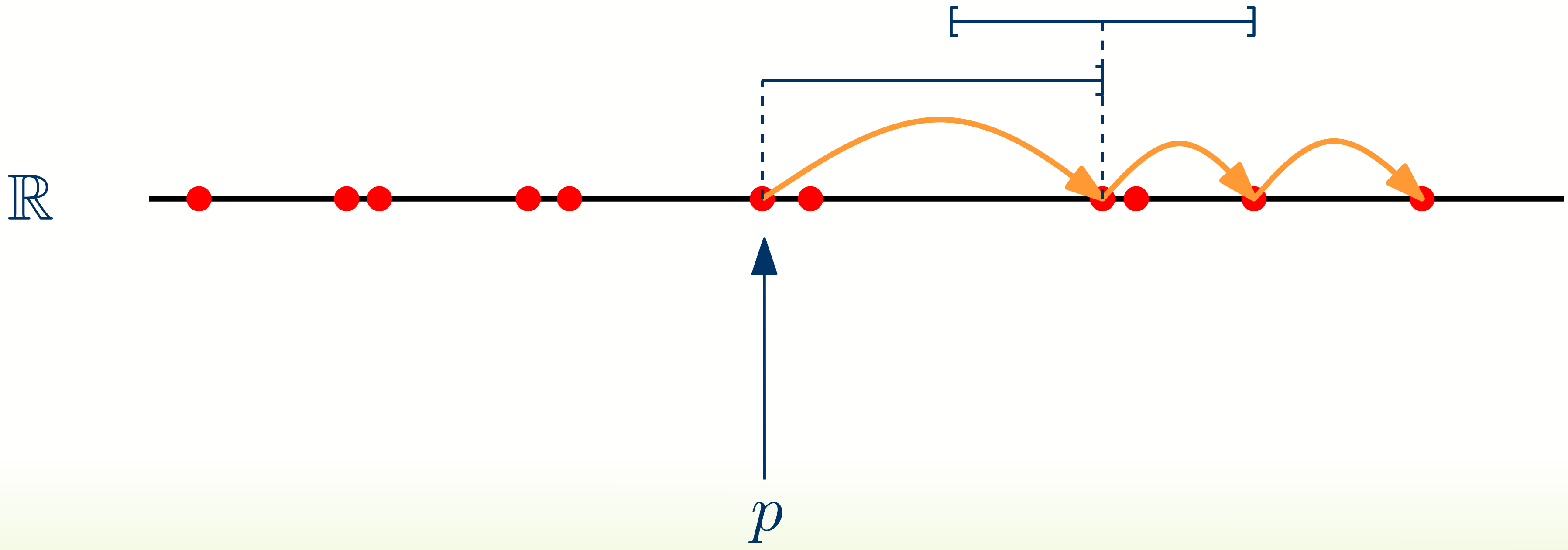
Observation: All points reachable from $p \in P$ lie in an interval.

1-dimensional case



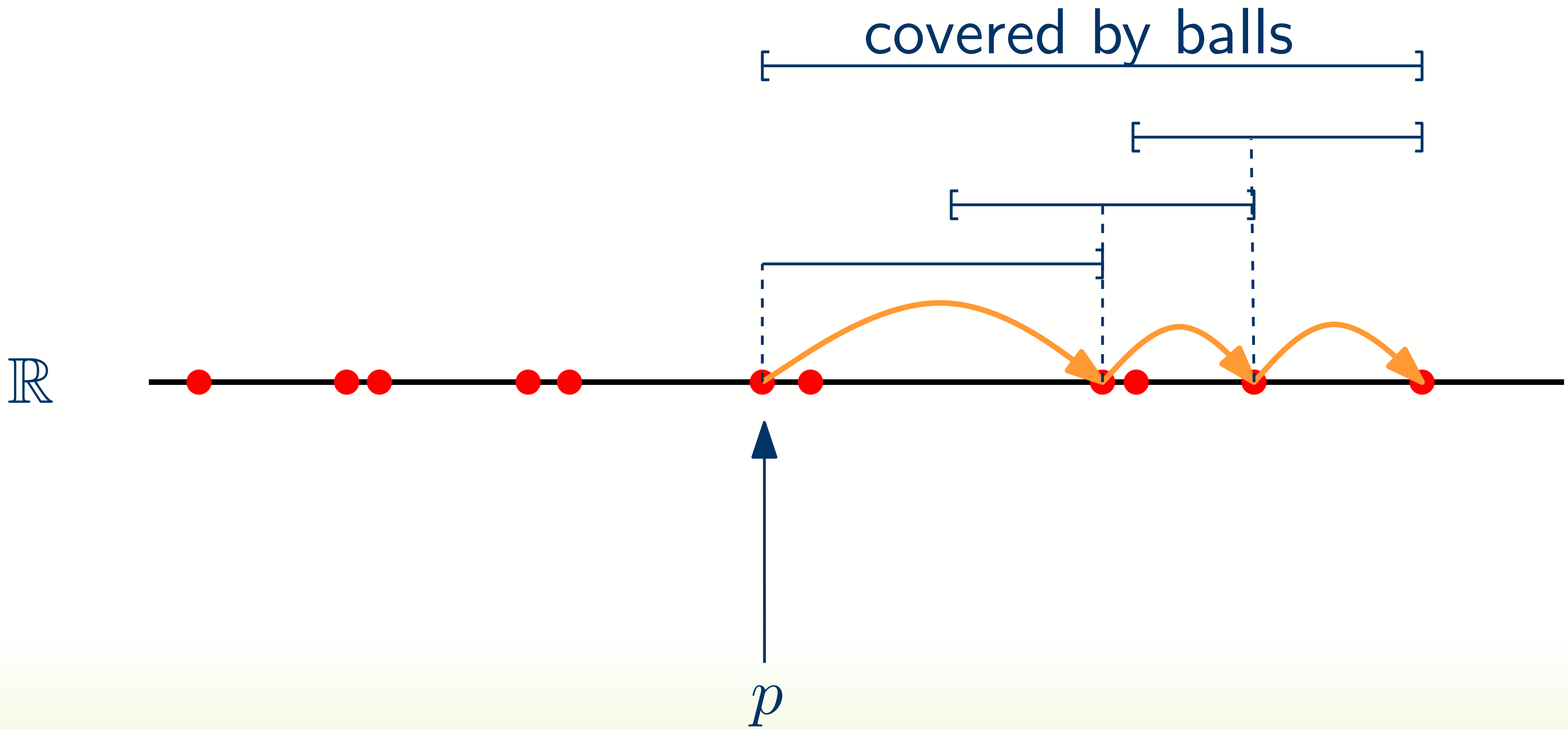
Observation: All points reachable from $p \in P$ lie in an interval.

1-dimensional case



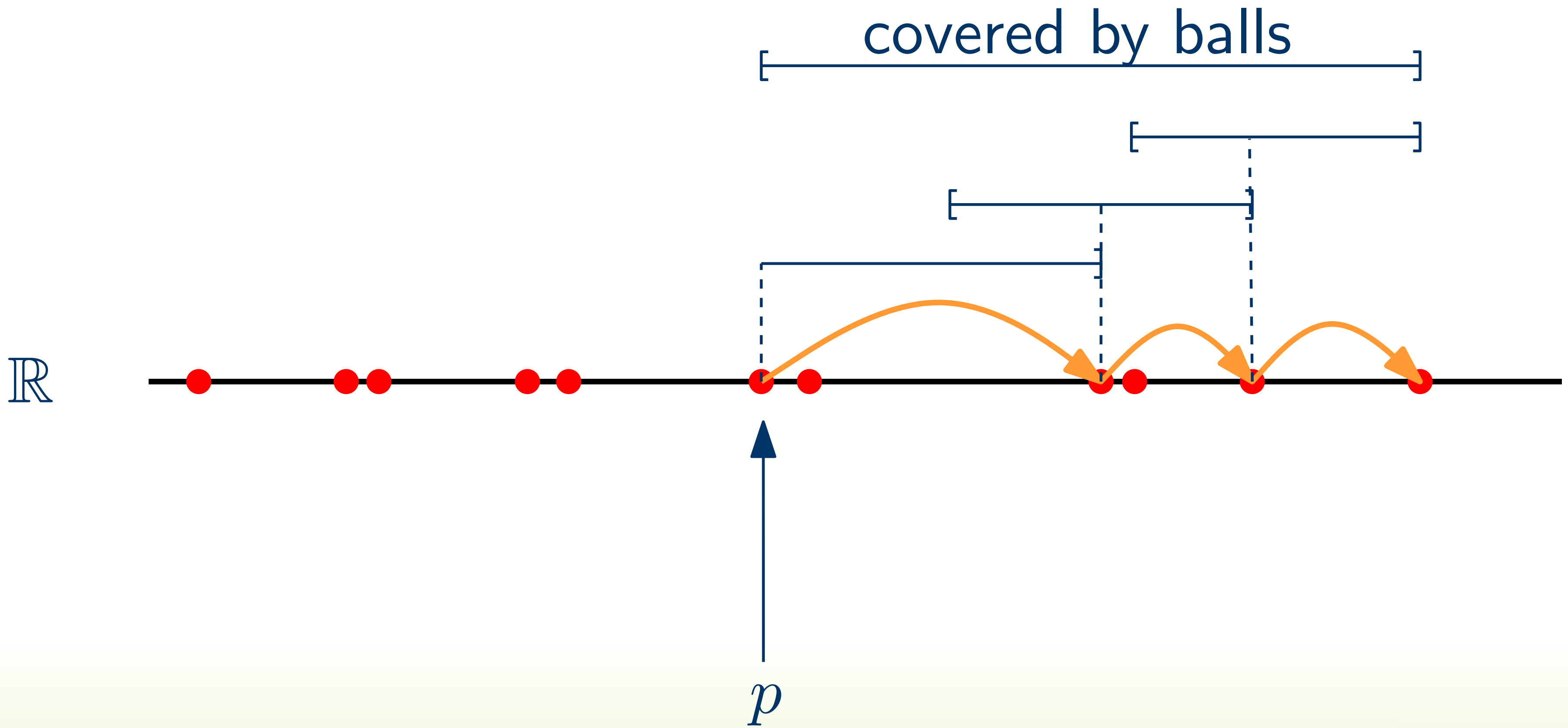
Observation: All points reachable from $p \in P$ lie in an interval.

1-dimensional case



Observation: All points reachable from $p \in P$ lie in an interval.

1-dimensional case



Observation: All points reachable from $p \in P$ lie in an interval.

$\Longrightarrow$

DS: Save for each $p \in P$ the boundary points of this *reachability* interval.

Computing reachbility intervals

Computing reachbility intervals

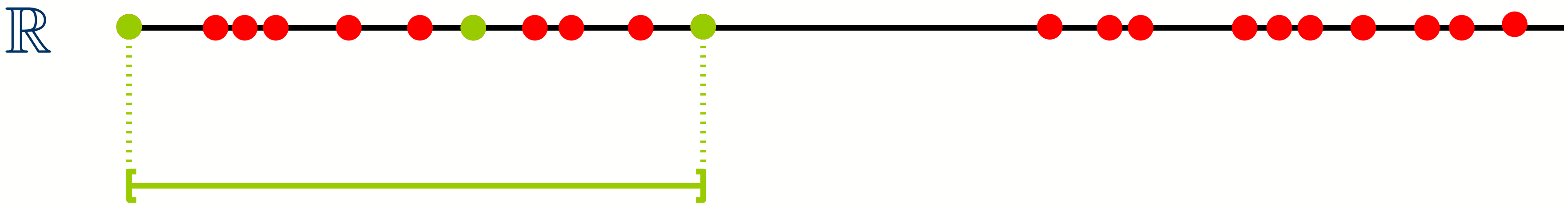
Look at the strongly connected components!

Computing reachbility intervals

Look at the strongly connected components!

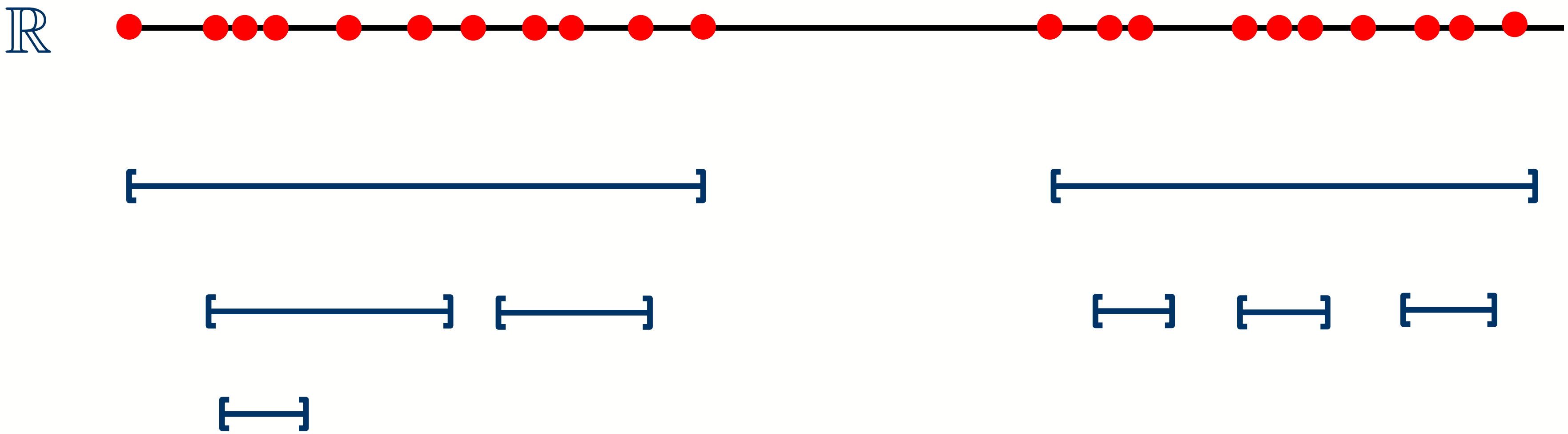
Computing reachbility intervals

Look at the strongly connected components!



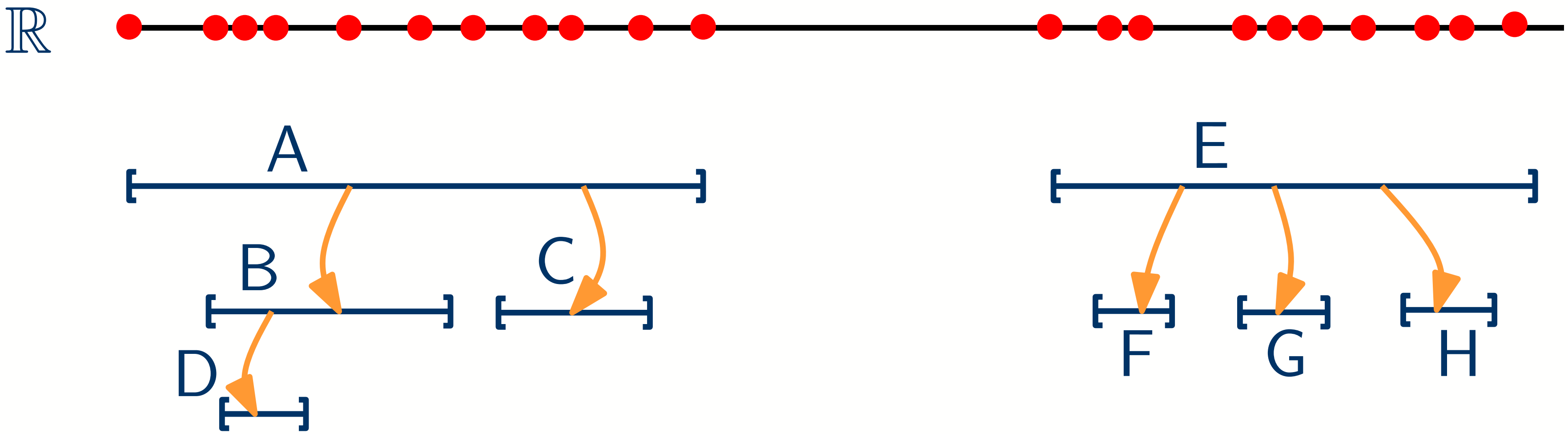
Computing reachbility intervals

Look at the strongly connected components!



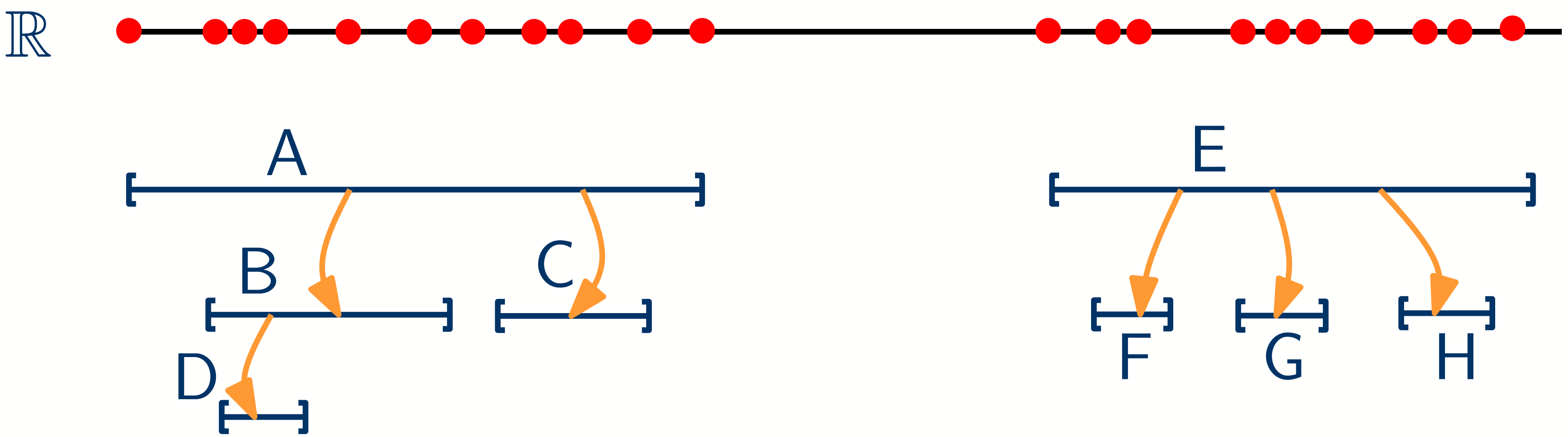
Computing reachbility intervals

Look at the strongly connected components!



Computing reachbility intervals

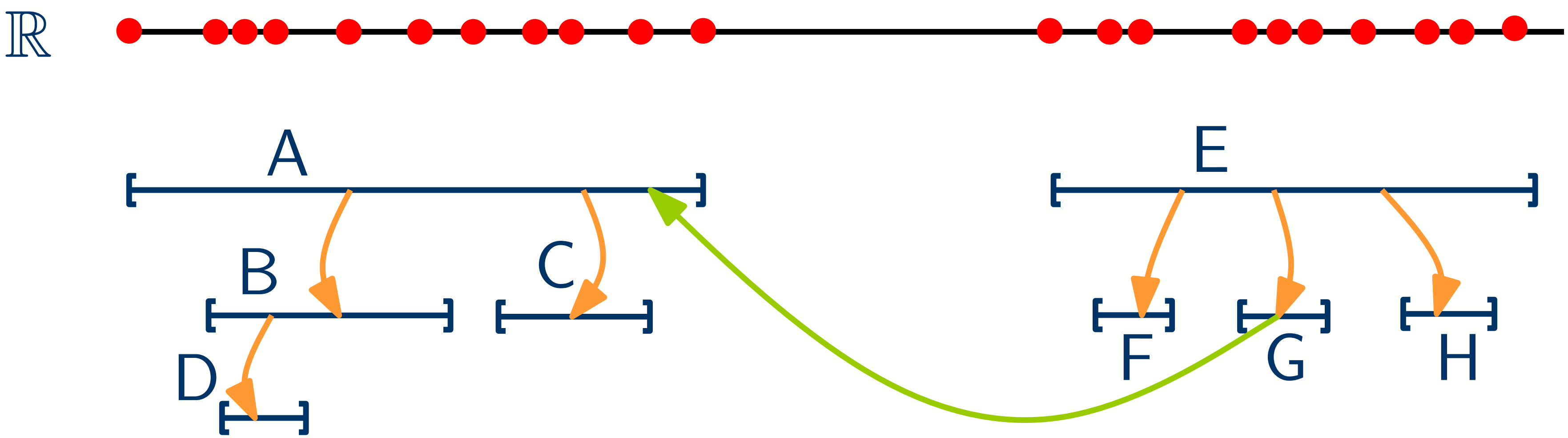
Look at the strongly connected components!



Observation: It suffices to look at siblings

Computing reachbility intervals

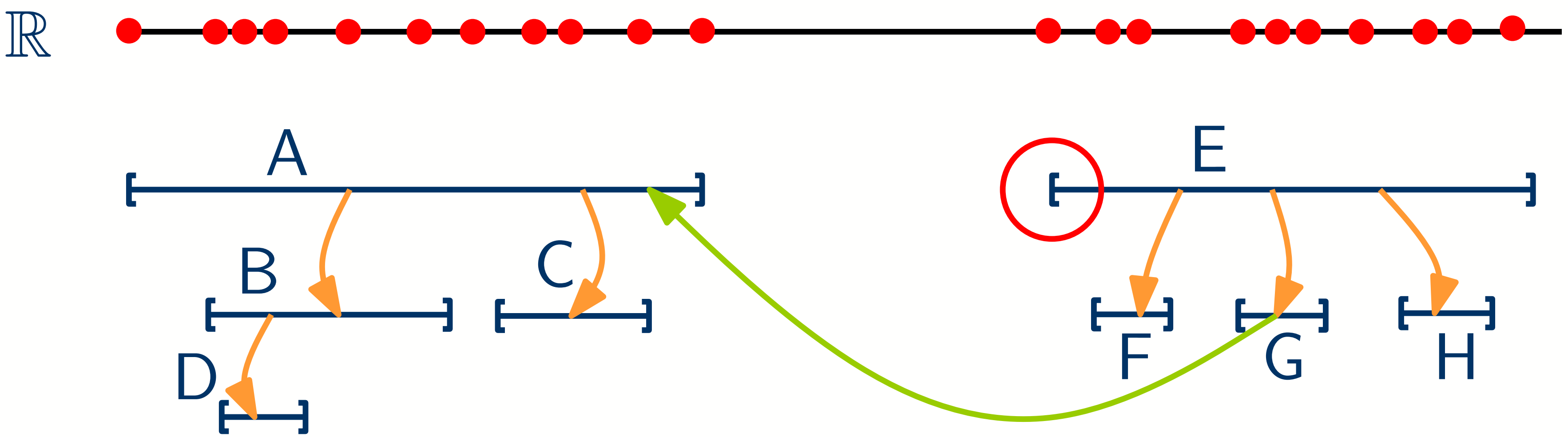
Look at the strongly connected components!



Observation: It suffices to look at siblings

Computing reachbility intervals

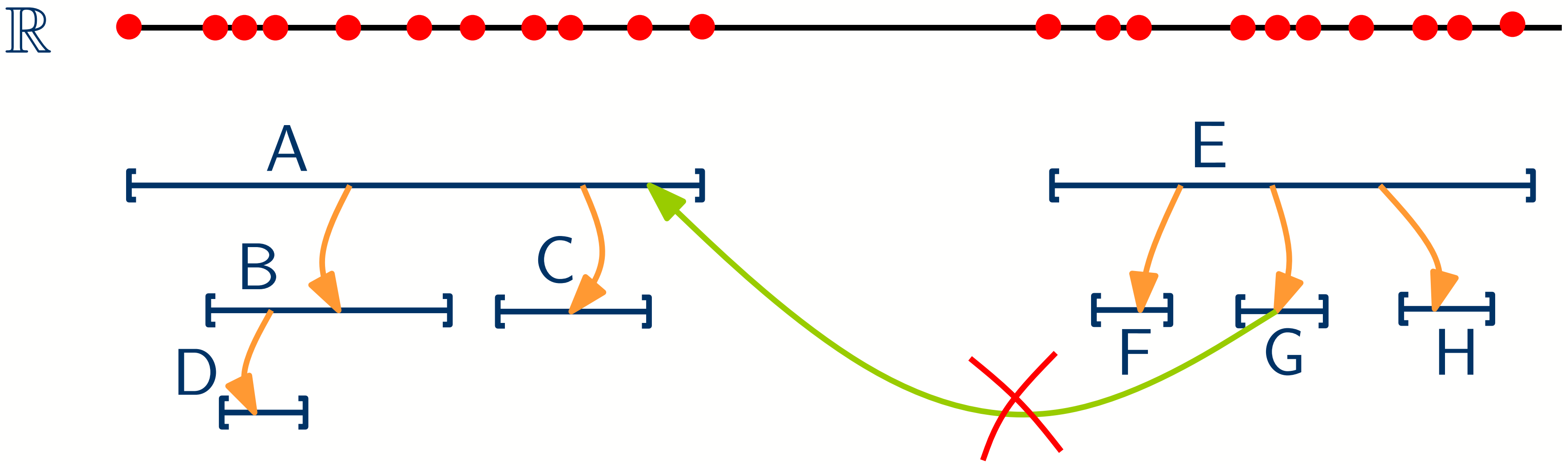
Look at the strongly connected components!



Observation: It suffices to look at siblings

Computing reachbility intervals

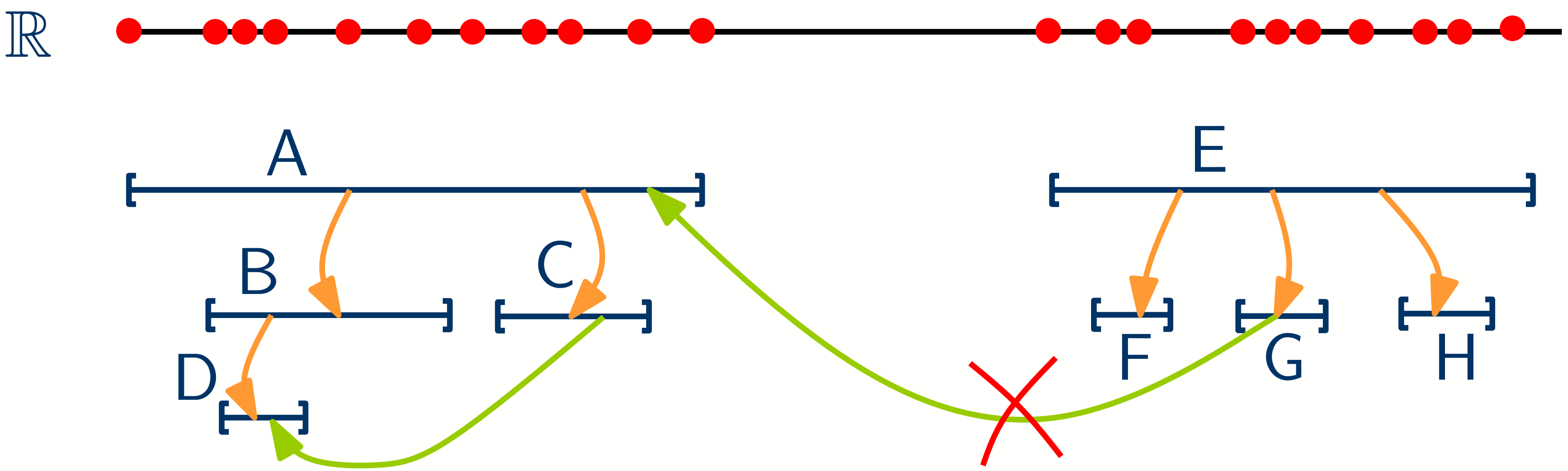
Look at the strongly connected components!



Observation: It suffices to look at siblings

Computing reachbility intervals

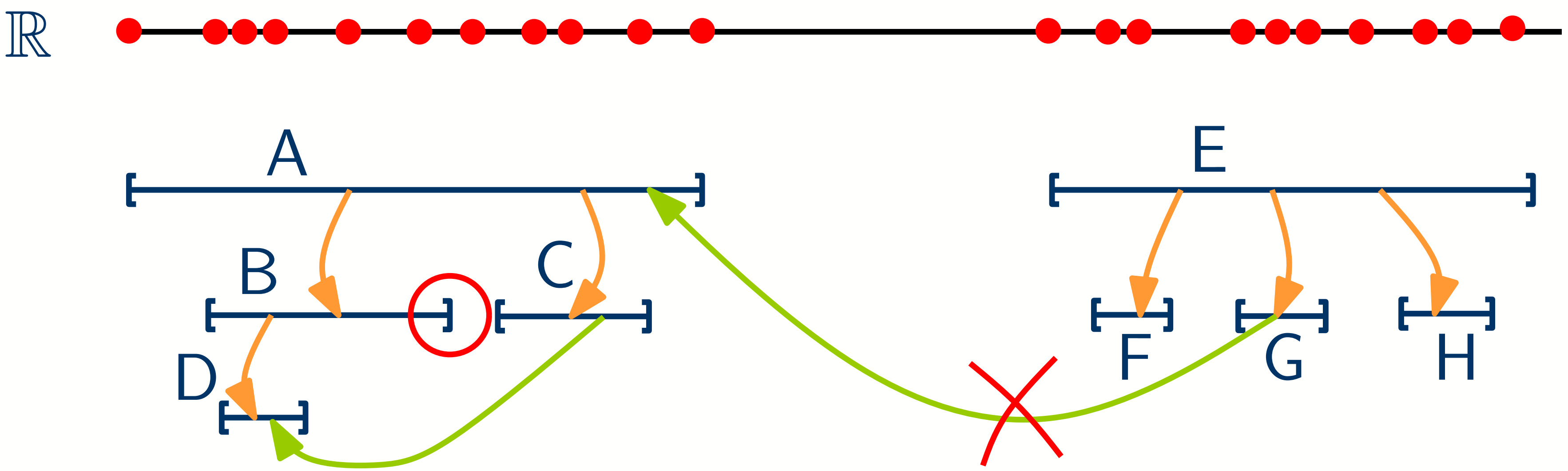
Look at the strongly connected components!



Observation: It suffices to look at siblings

Computing reachbility intervals

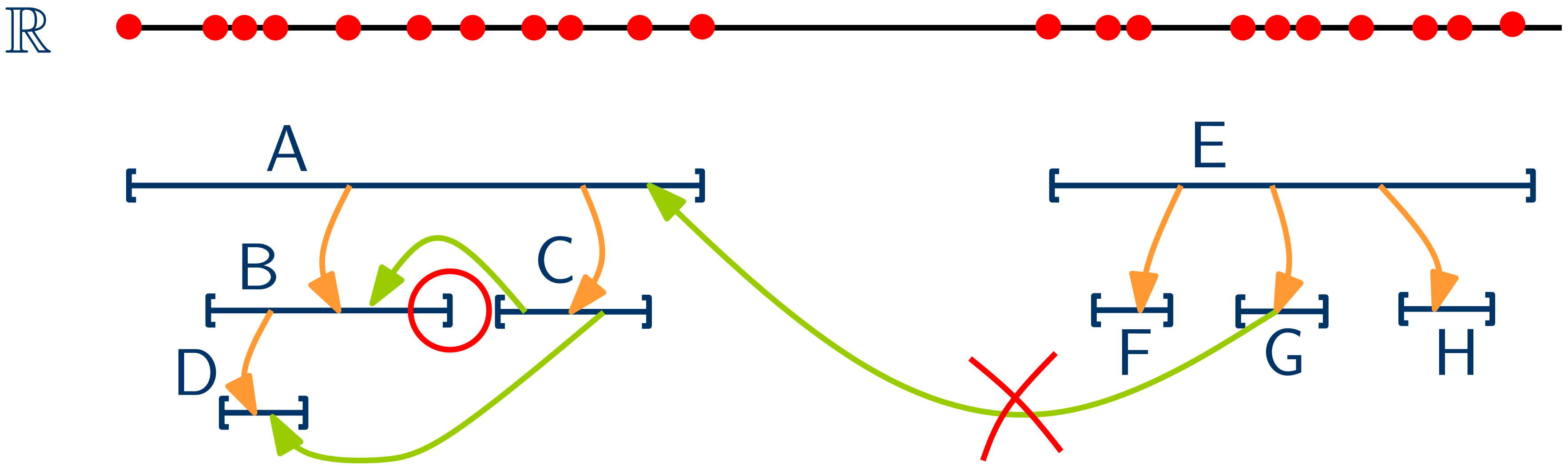
Look at the strongly connected components!



Observation: It suffices to look at siblings

Computing reachbility intervals

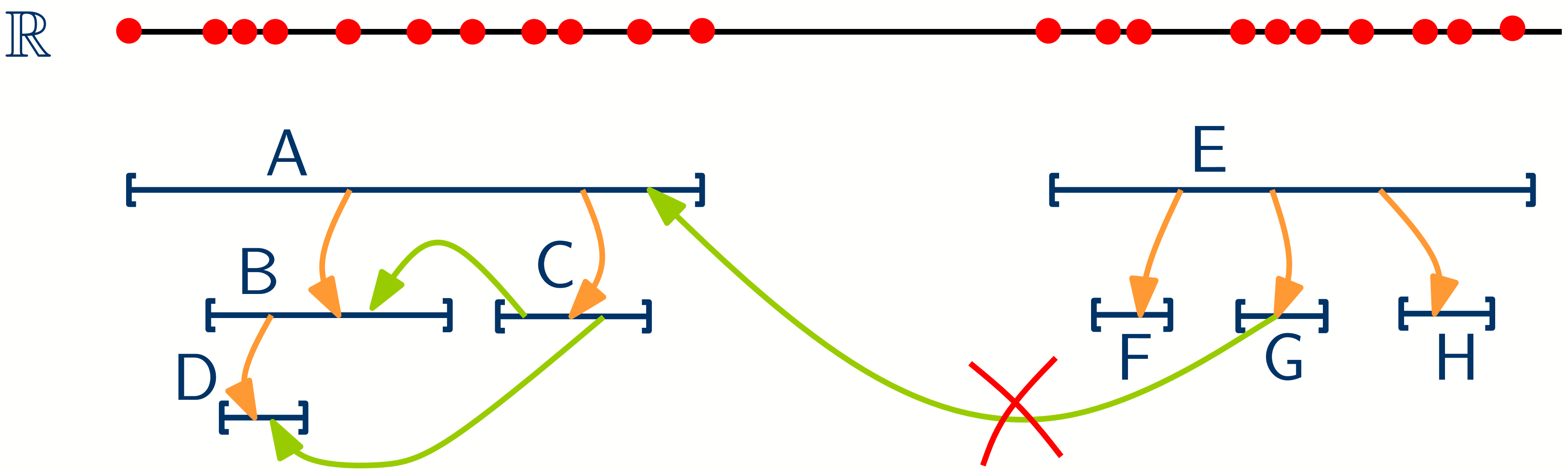
Look at the strongly connected components!



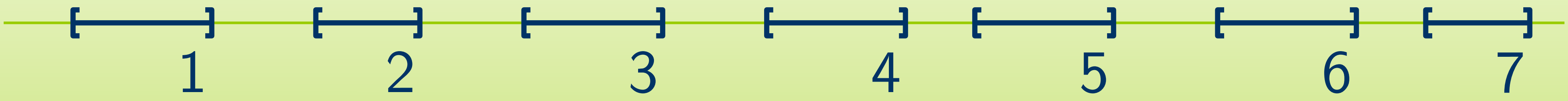
Observation: It suffices to look at siblings

Computing reachbility intervals

Look at the strongly connected components!

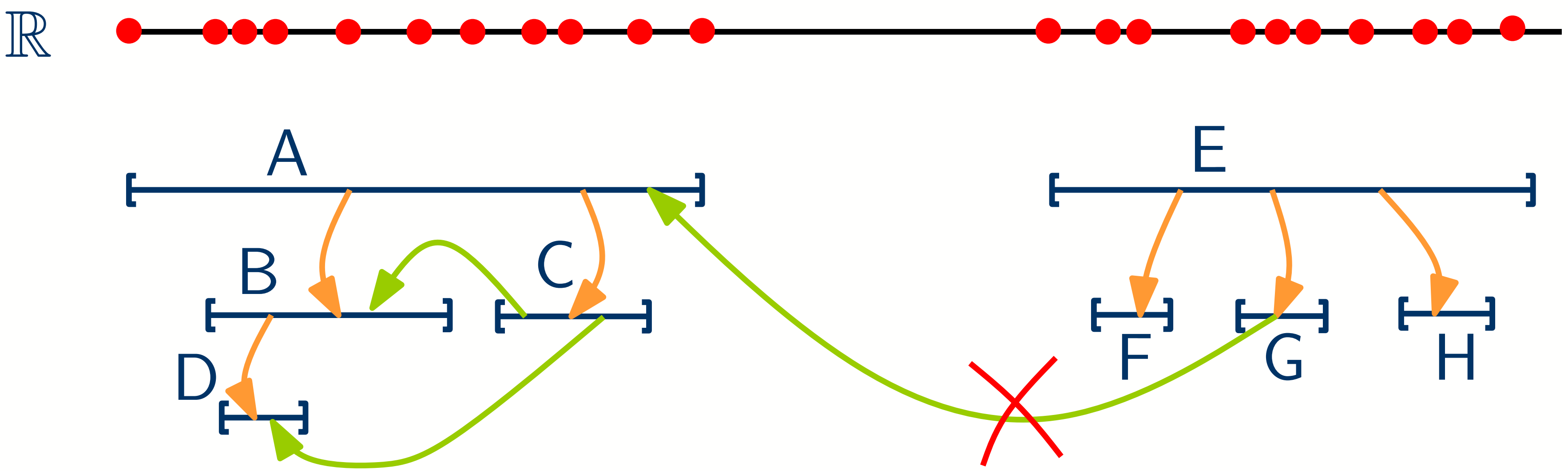


Observation: It suffices to look at siblings

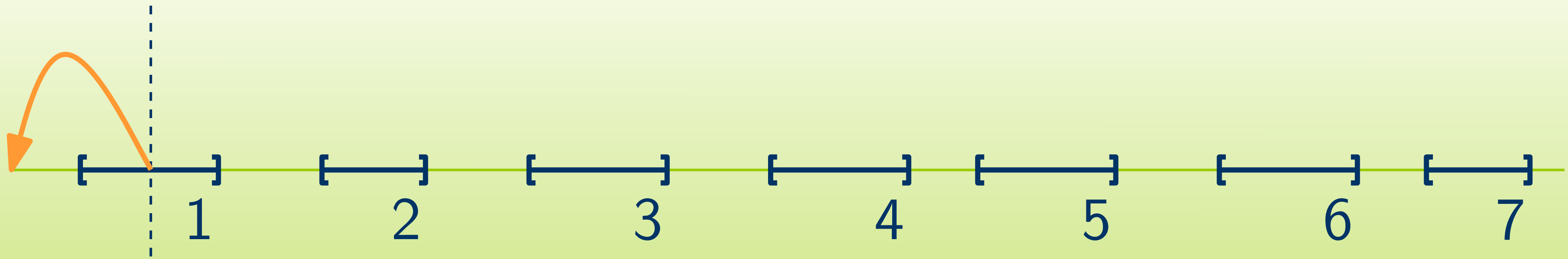


Computing reachbility intervals

Look at the strongly connected components!

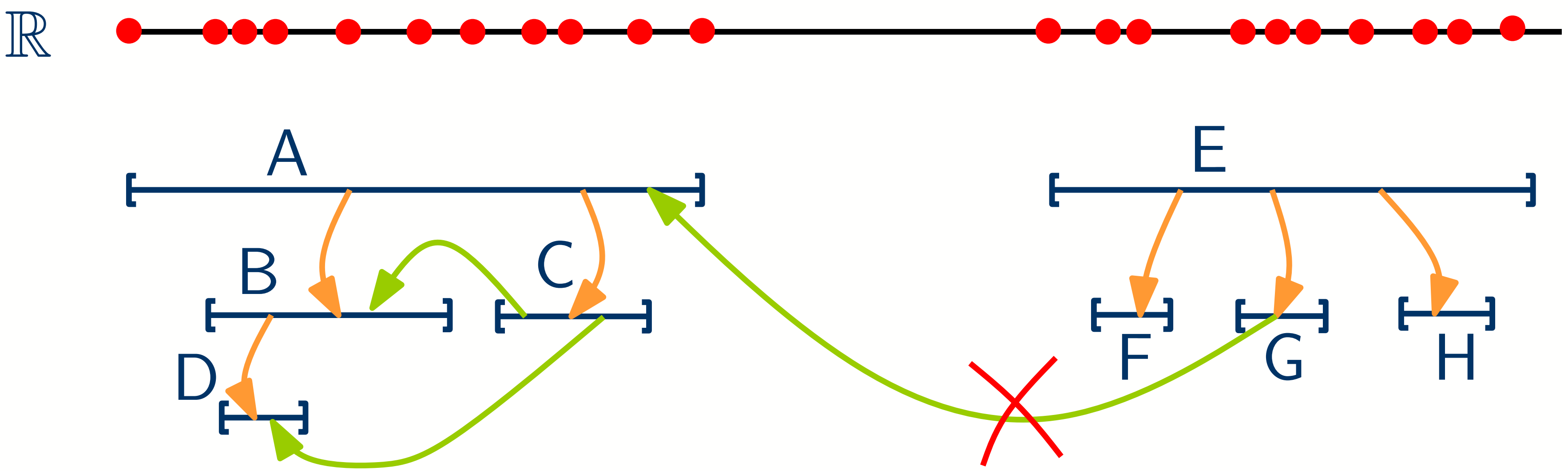


Observation: It suffices to look at siblings

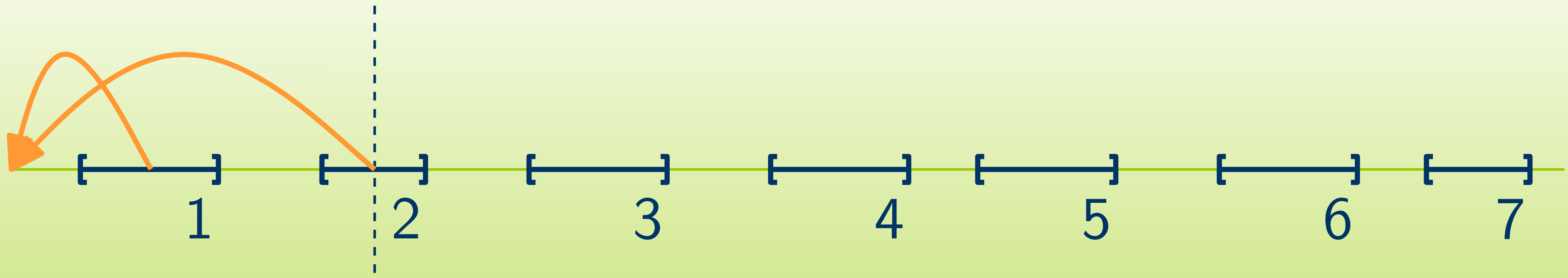


Computing reachbility intervals

Look at the strongly connected components!

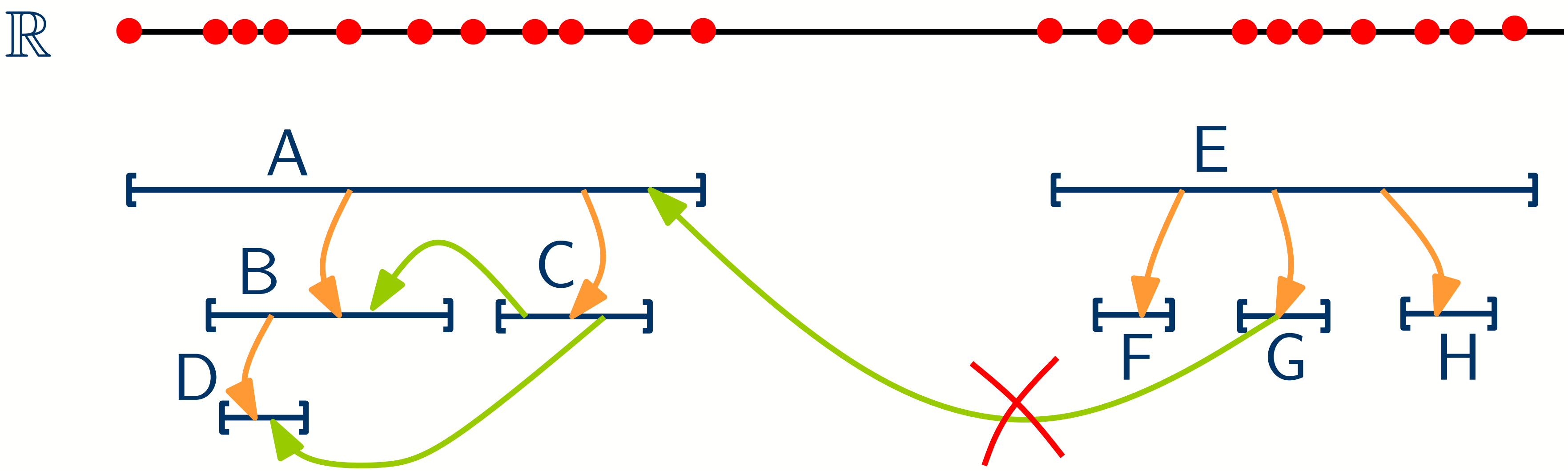


Observation: It suffices to look at siblings

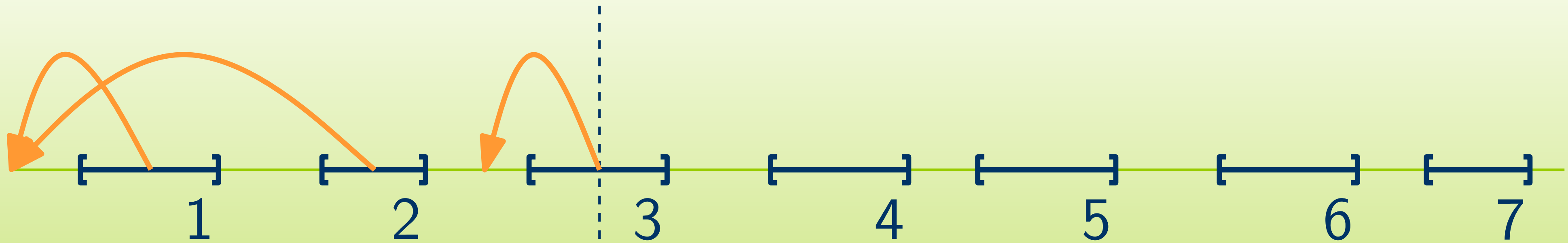


Computing reachbility intervals

Look at the strongly connected components!

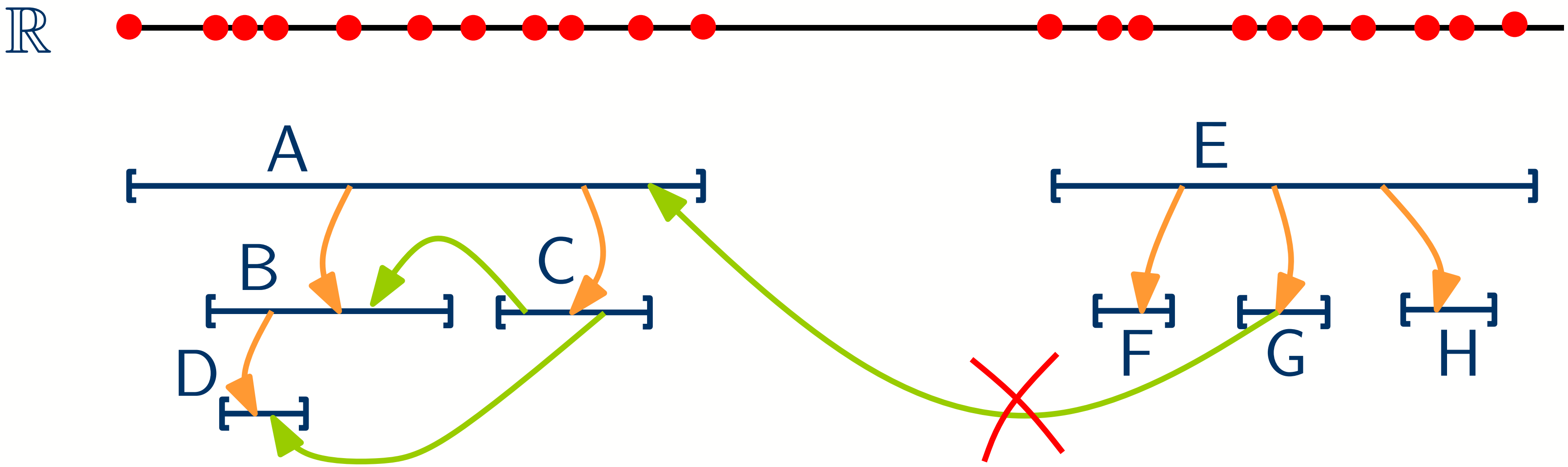


Observation: It suffices to look at siblings

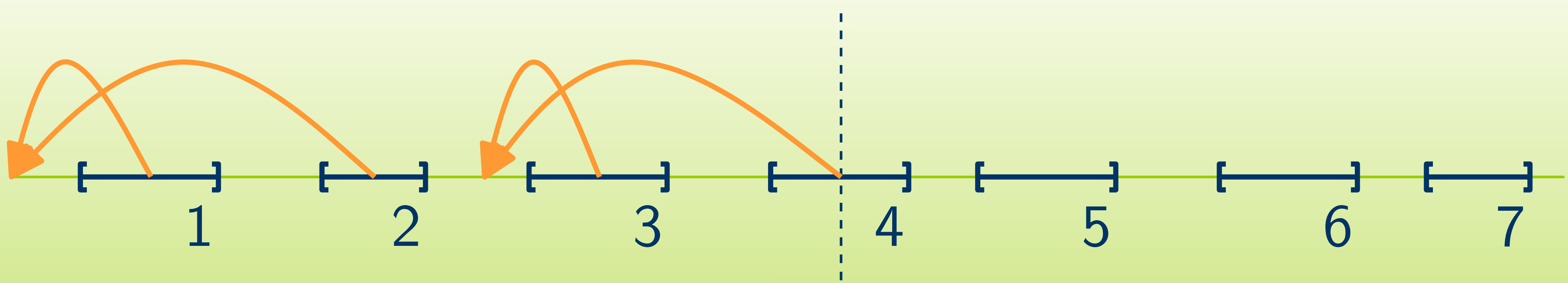


Computing reachbility intervals

Look at the strongly connected components!

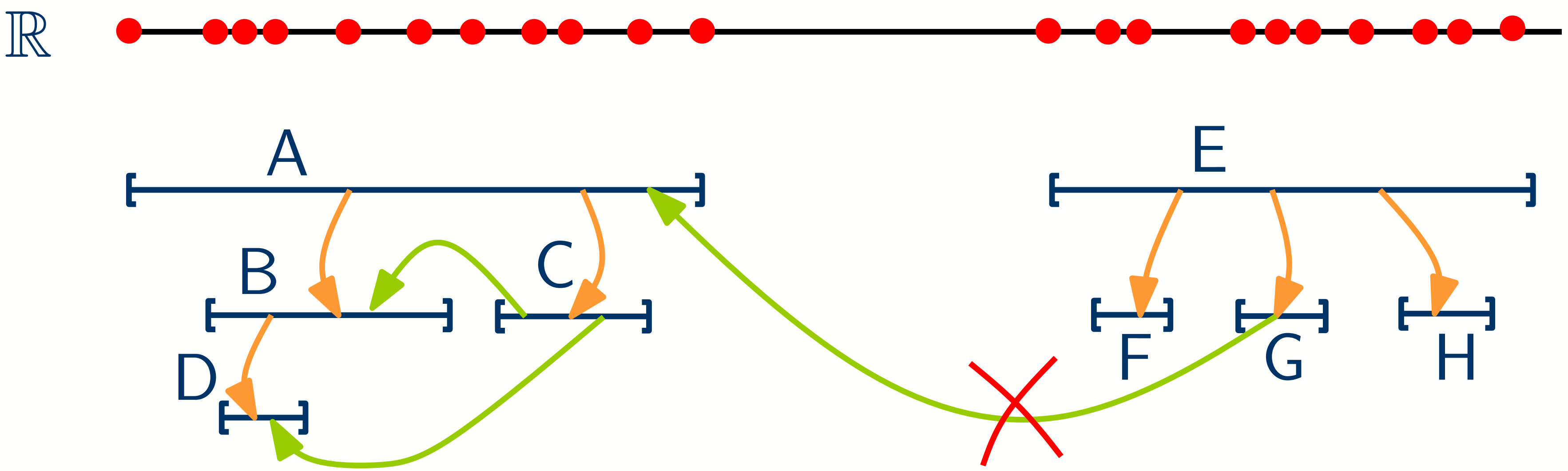


Observation: It suffices to look at siblings

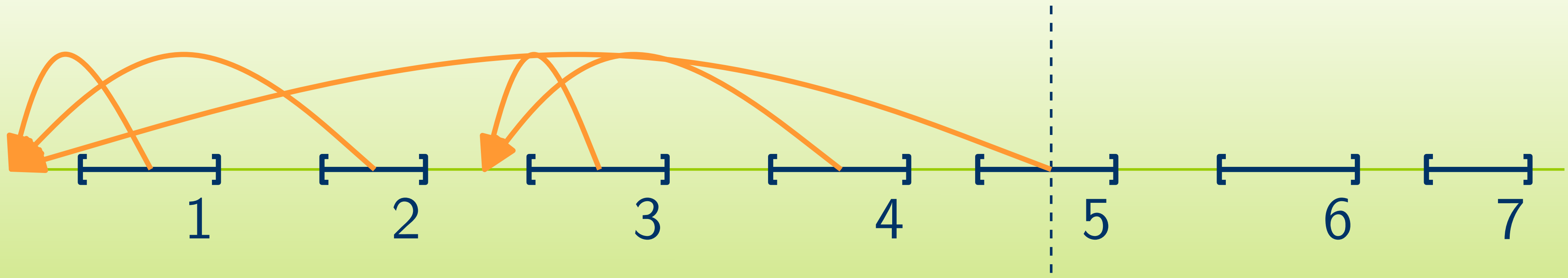


Computing reachbility intervals

Look at the strongly connected components!

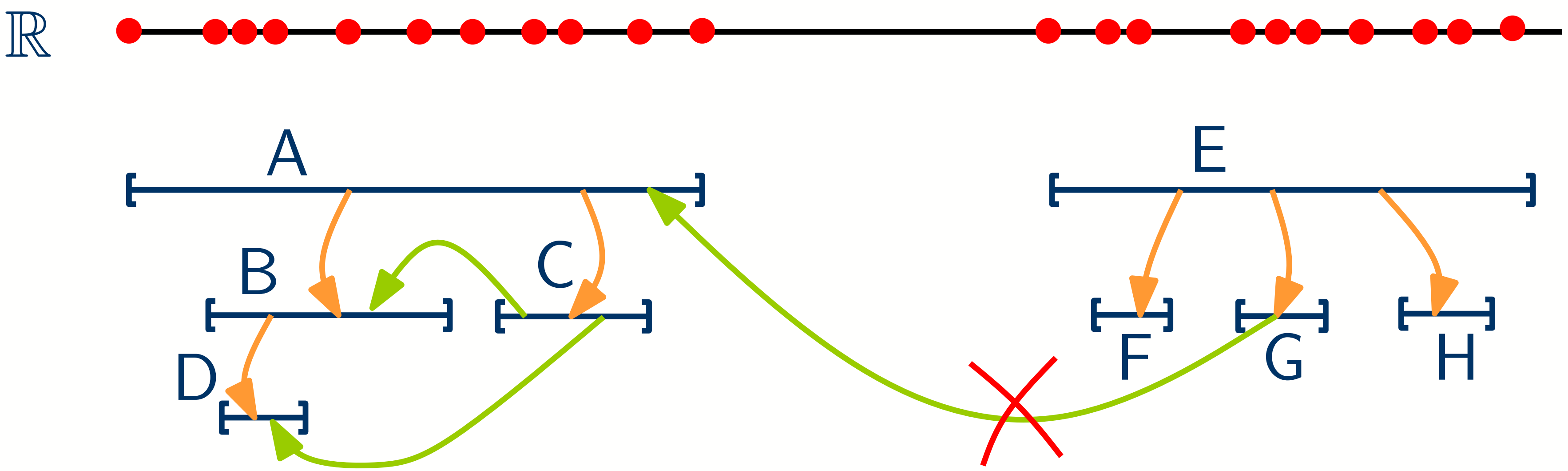


Observation: It suffices to look at siblings

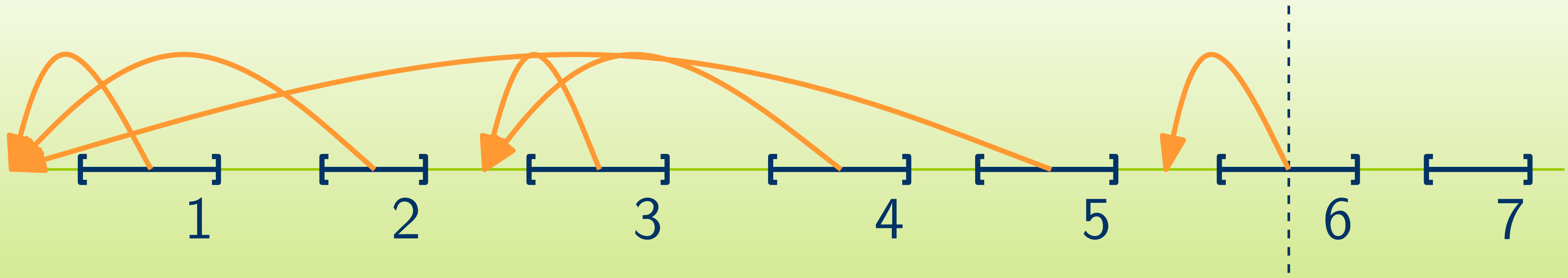


Computing reachbility intervals

Look at the strongly connected components!

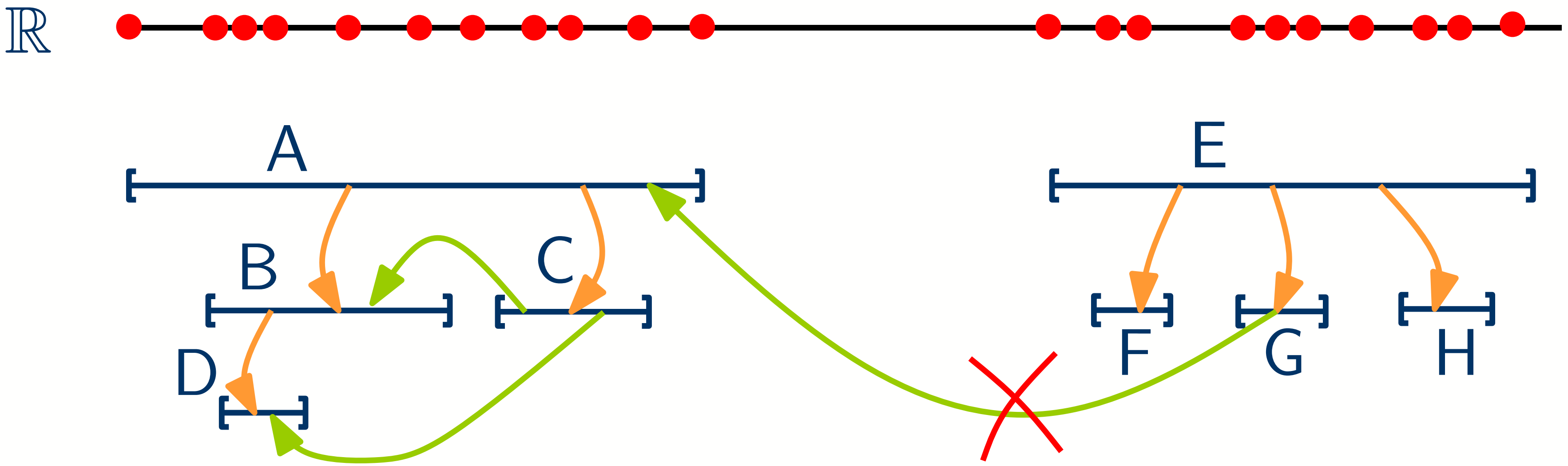


Observation: It suffices to look at siblings

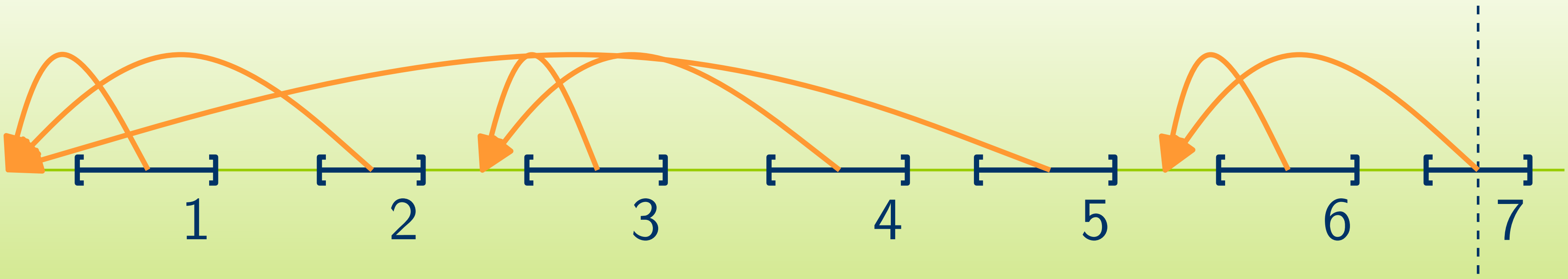


Computing reachbility intervals

Look at the strongly connected components!



Observation: It suffices to look at siblings



Results

	$P(n)$	$S(n)$	$Q(n)$	Restrictions
$d = 1$	$O(n \log n)$	$O(n)$	$O(1)$	none
$d = 2$				

2-dimensional case for radii in $[1, \sqrt{3})$

Theorem (Thorup): For **planar** graphs we can compute a reachability oracle with $S(n) = O(n \log n)$ and $Q(n) = O(1)$ in time $O(n \log n)$

2-dimensional case for radii in $[1, \sqrt{3})$

Theorem (Thorup): For **planar** graphs we can compute a reachability oracle with $S(n) = O(n \log n)$ and $Q(n) = O(1)$ in time $O(n \log n)$

Plan:

1. Make graph planar without changing reachability
2. Use Thorup's Theorem

2-dimensional case for radii in $[1, \sqrt{3})$

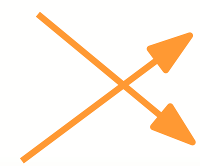
Theorem (Thorup): For **planar** graphs we can compute a reachability oracle with $S(n) = O(n \log n)$ and $Q(n) = O(1)$ in time $O(n \log n)$

Plan:

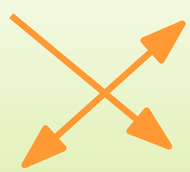
1. Make graph planar without changing reachability
2. Use Thorup's Theorem

Possible crossings:

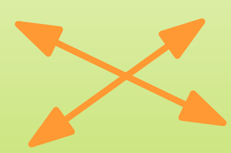
i)



ii)



iii)



2-dimensional case for radii in $[1, \sqrt{3})$

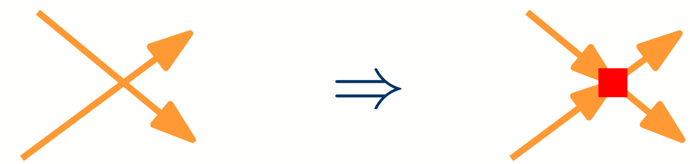
Theorem (Thorup): For **planar** graphs we can compute a reachability oracle with $S(n) = O(n \log n)$ and $Q(n) = O(1)$ in time $O(n \log n)$

Plan:

1. Make graph planar without changing reachability
2. Use Thorup's Theorem

Possible crossings:

i)



ii)

iii)

2-dimensional case for radii in $[1, \sqrt{3})$

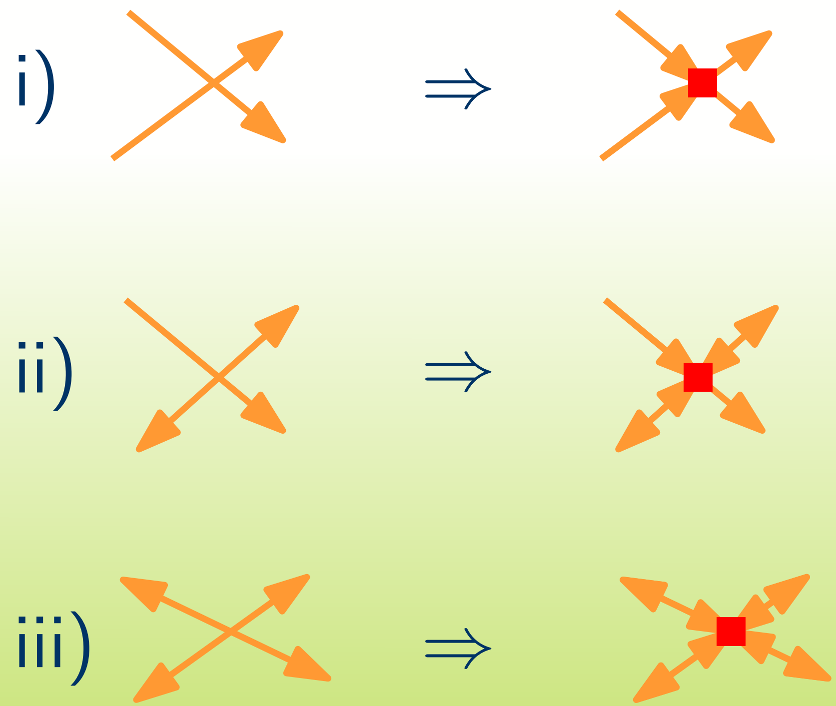
Theorem (Thorup): For **planar** graphs we can compute a reachability oracle with $S(n) = O(n \log n)$ and $Q(n) = O(1)$ in time $O(n \log n)$

Plan:

1. Make graph planar without changing reachability
2. Use Thorup's Theorem

Possible crossings:

up to $O(n^2)$



2-dimensional case for radii in $[1, \sqrt{3})$

Theorem (Thorup): For **planar** graphs we can compute a reachability oracle with $S(n) = O(n \log n)$ and $Q(n) = O(1)$ in time $O(n \log n)$

Plan:

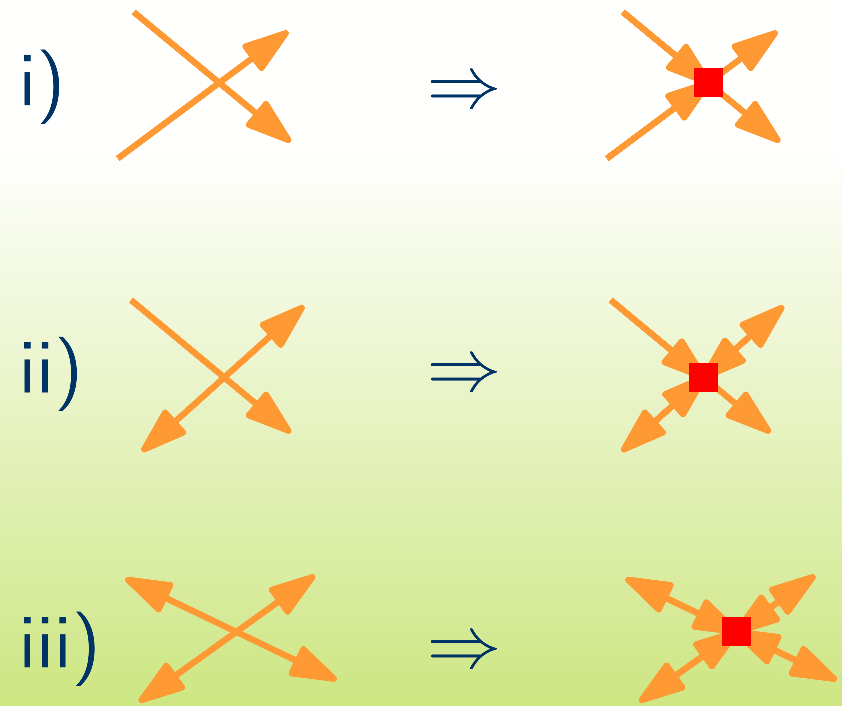
1. Prune G to reduce crossings

2. ~~1.~~ Make graph planar without changing reachability

3. ~~2.~~ Use Thorup's Theorem

Possible crossings:

up to $O(n^2)$



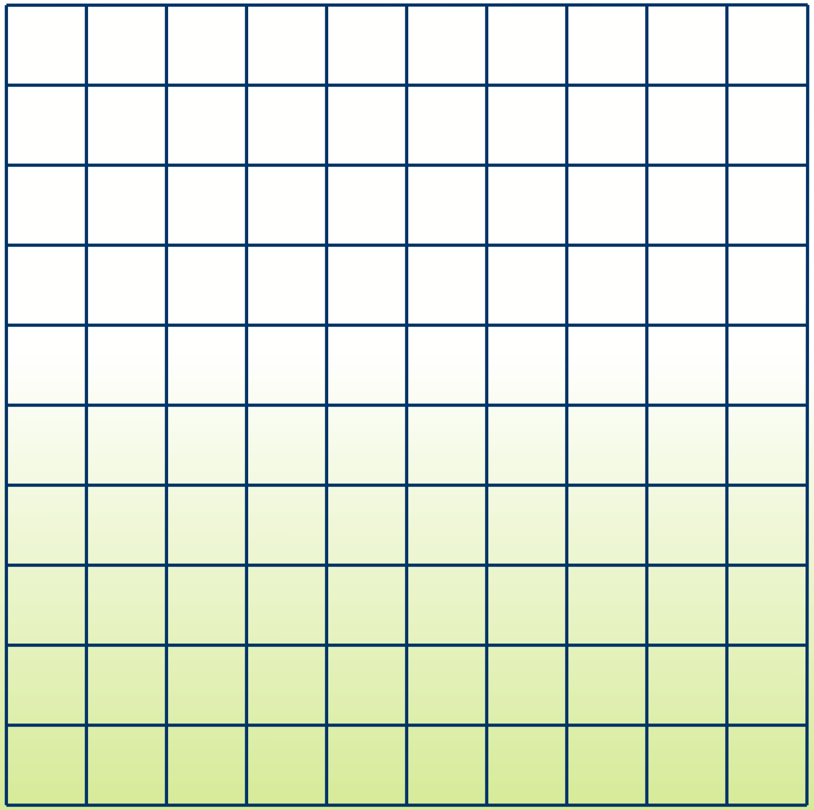
Pruning G

Want: O(n) edges; O(n) crossings; same reachability

Pruning G

Want: O(n) edges; O(n) crossings; same reachability

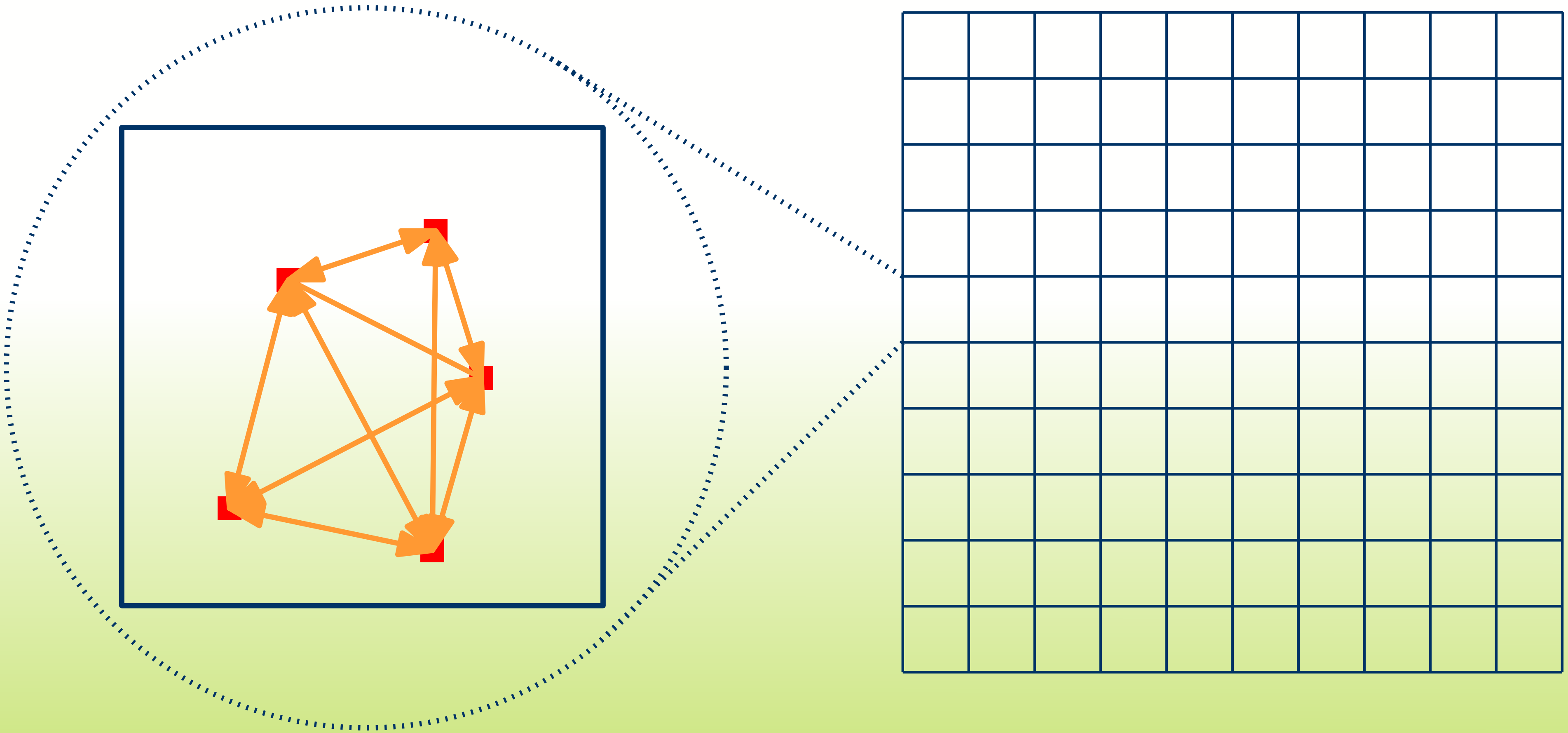
Use grid with side length $1/2$



Pruning G

Want: O(n) edges; O(n) crossings; same reachability

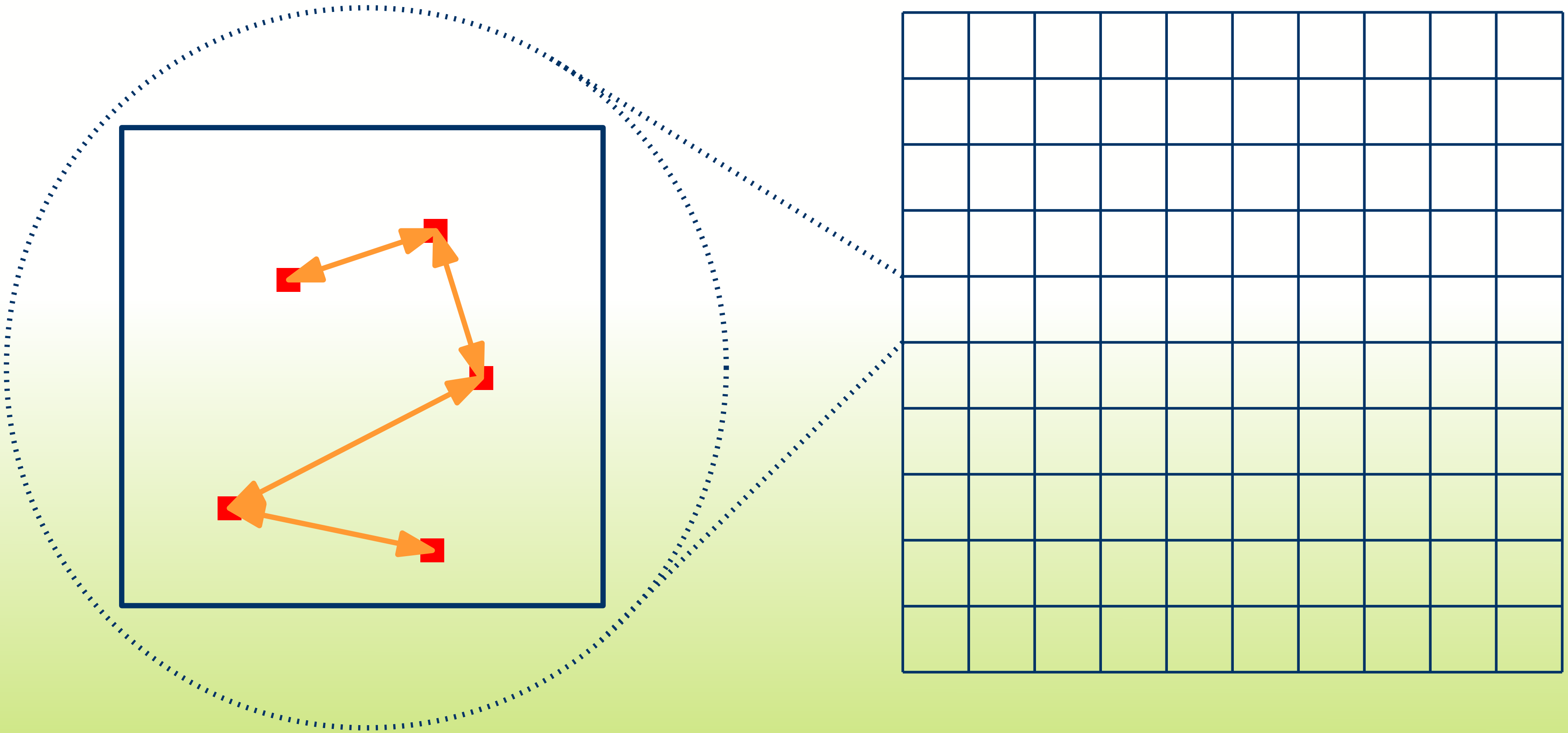
Use grid with side length $1/2$



Pruning G

Want: O(n) edges; O(n) crossings; same reachability

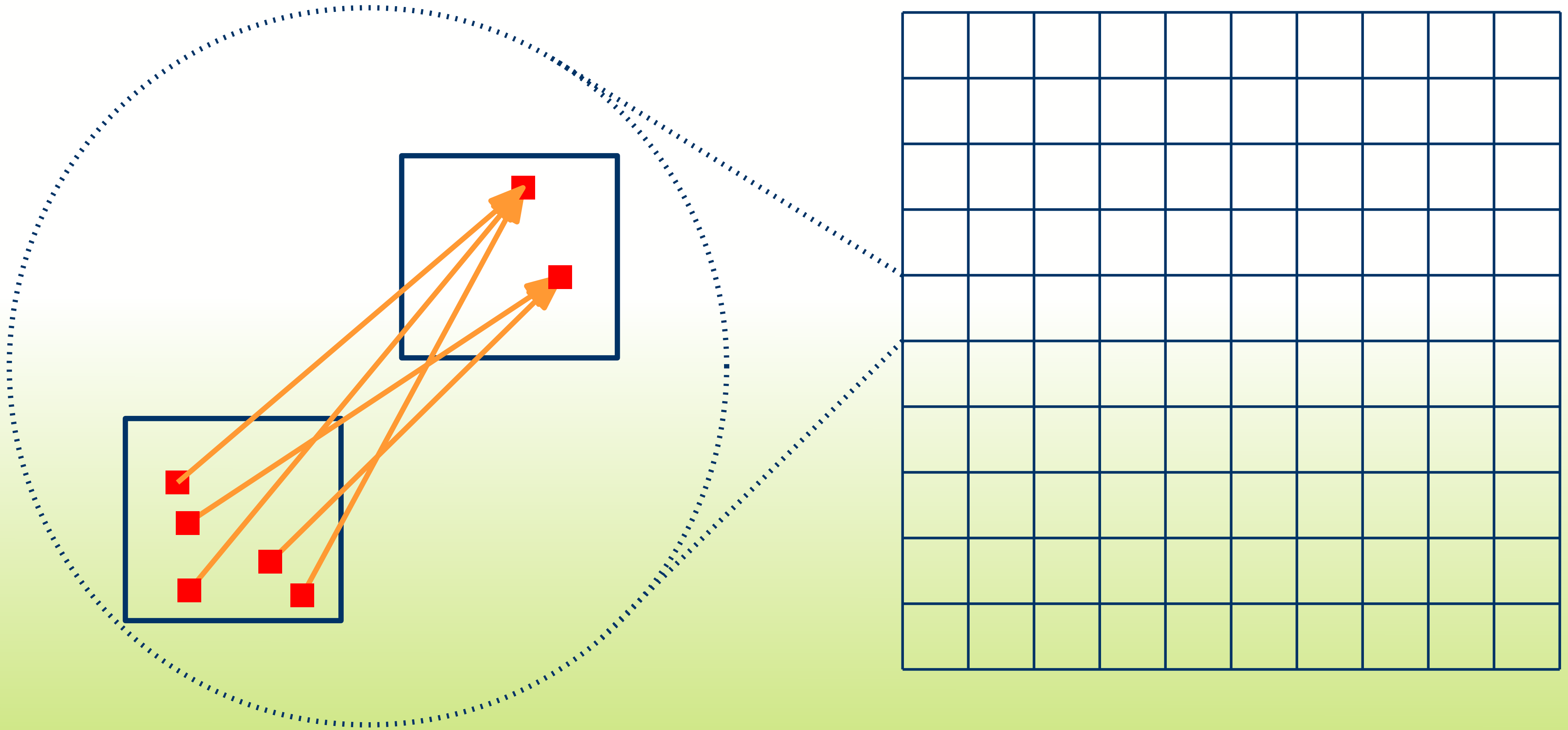
Use grid with side length $1/2$



Pruning G

Want: O(n) edges; O(n) crossings; same reachability

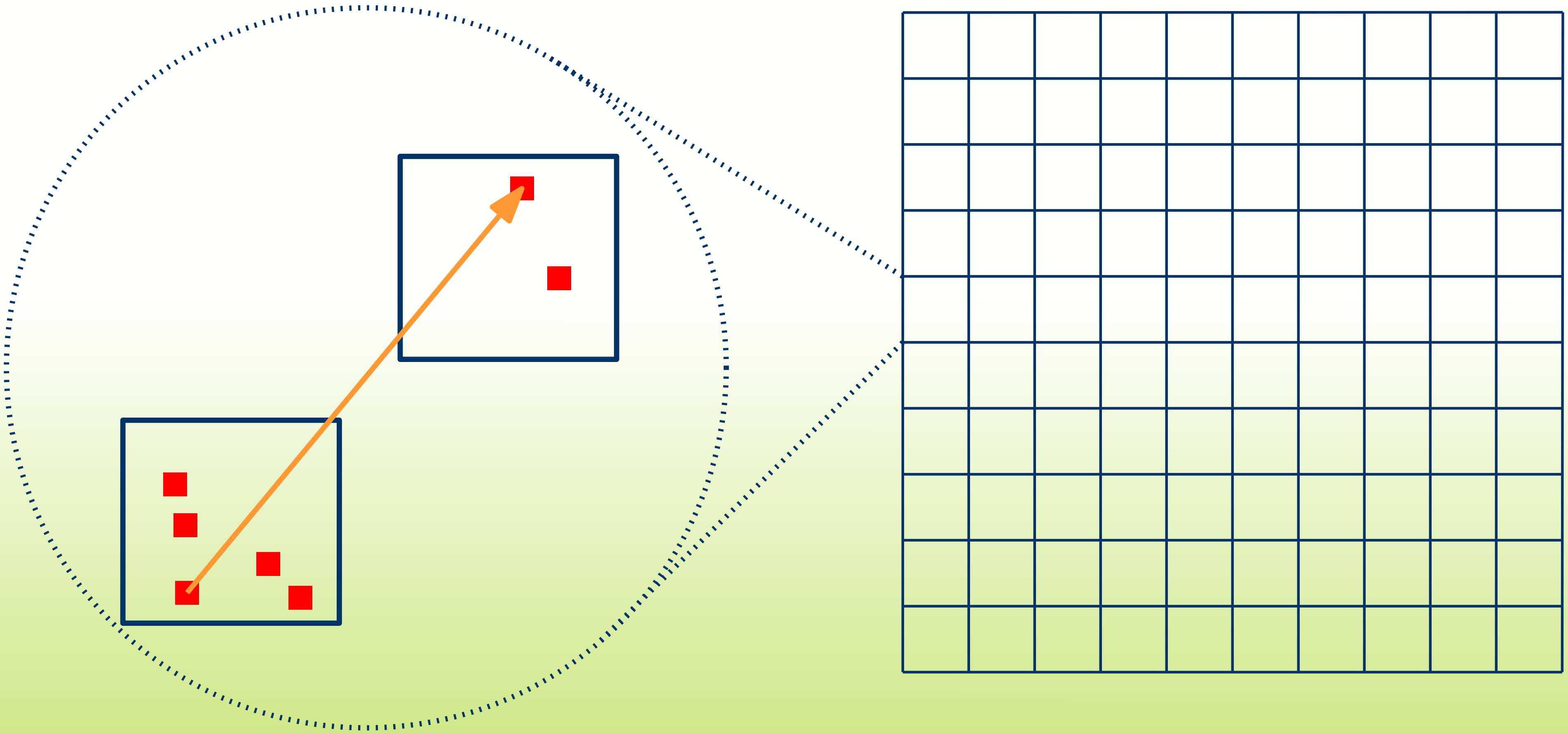
Use grid with side length $1/2$



Pruning G

Want: O(n) edges; O(n) crossings; same reachability

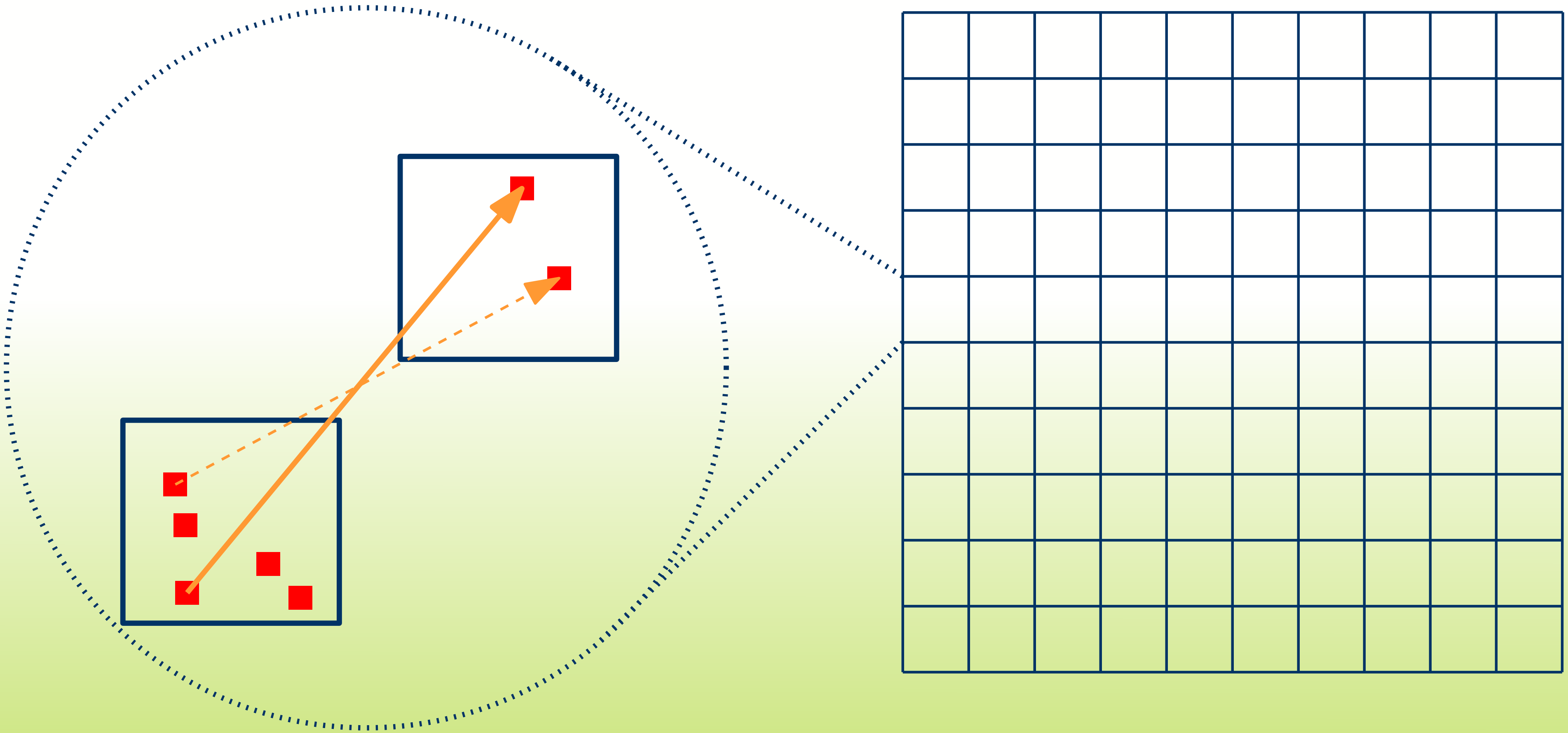
Use grid with side length $1/2$



Pruning G

Want: O(n) edges; O(n) crossings; same reachability

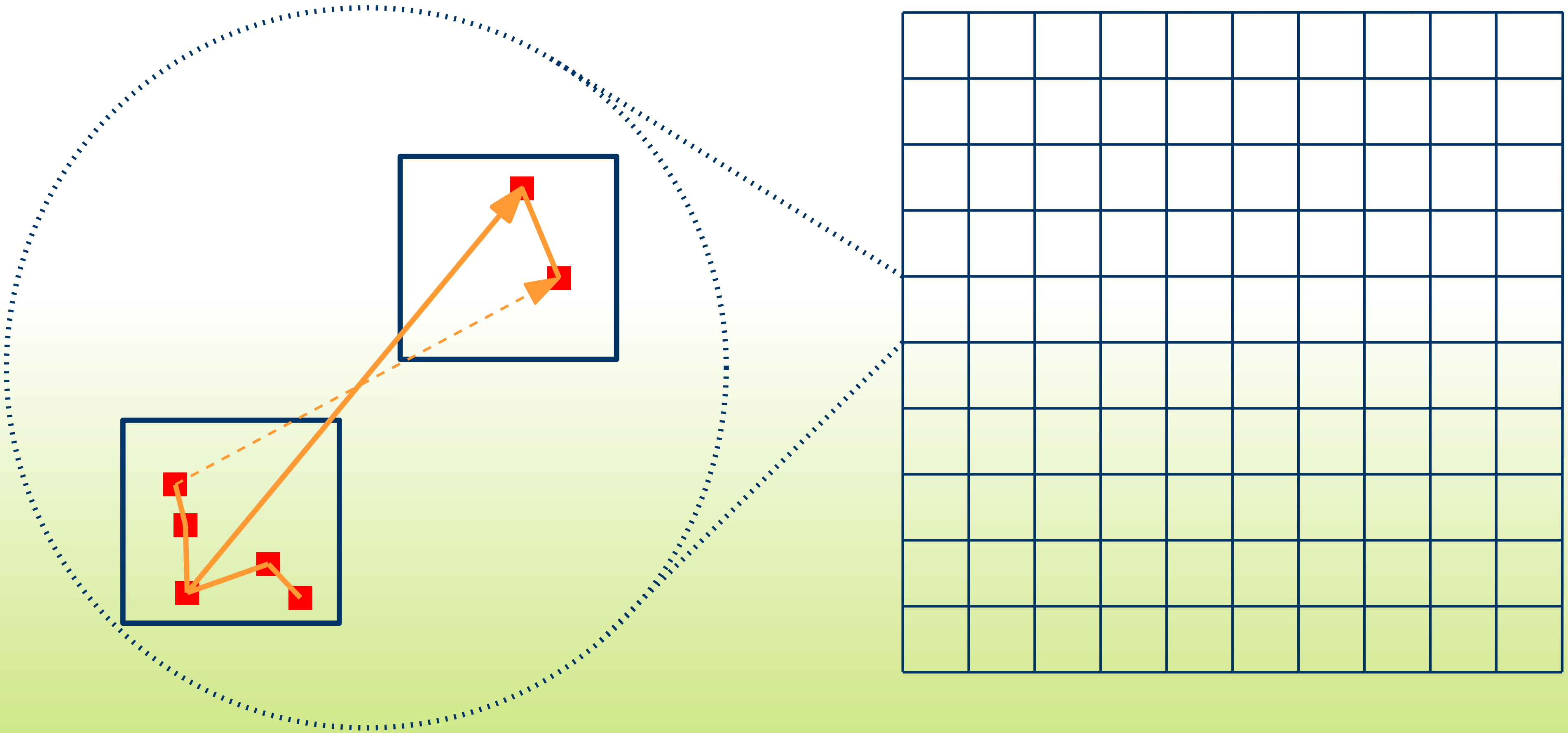
Use grid with side length $1/2$



Pruning G

Want: O(n) edges; O(n) crossings; same reachability

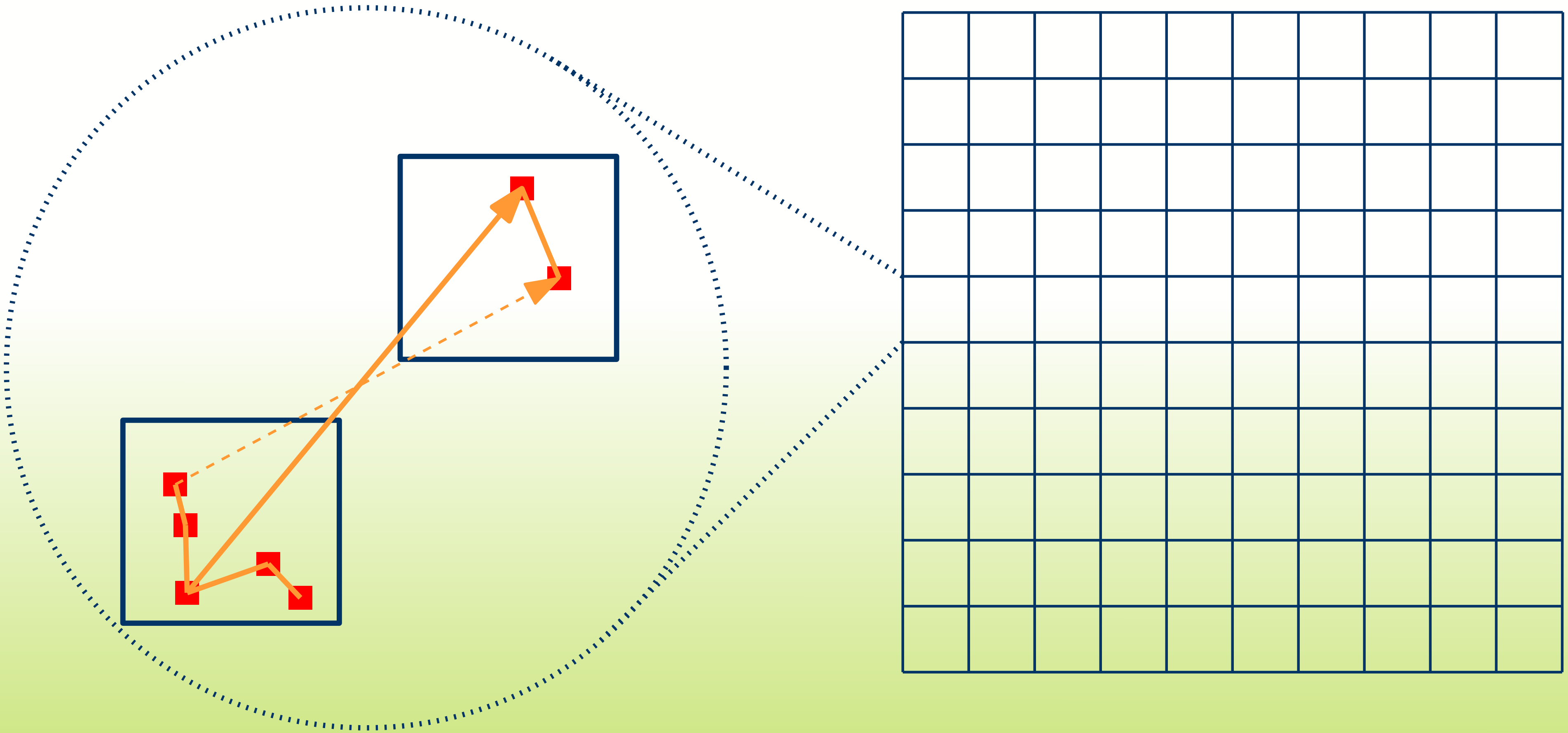
Use grid with side length $1/2$



Pruning G

Want: O(n) edges; O(n) crossings; same reachability

Use grid with side length $1/2$

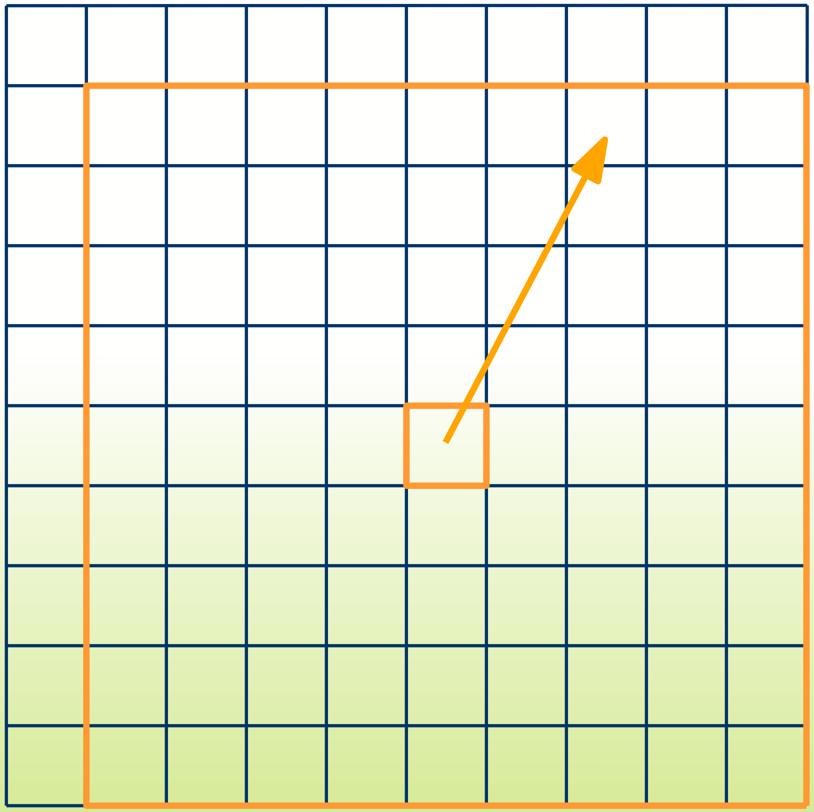


Pruning G

Want: O(n) edges; O(n) crossings; same reachability

Use grid with side length $1/2$

Def: *Neighborhood* of $\square$:
9×9 grid $\square$ is centered at

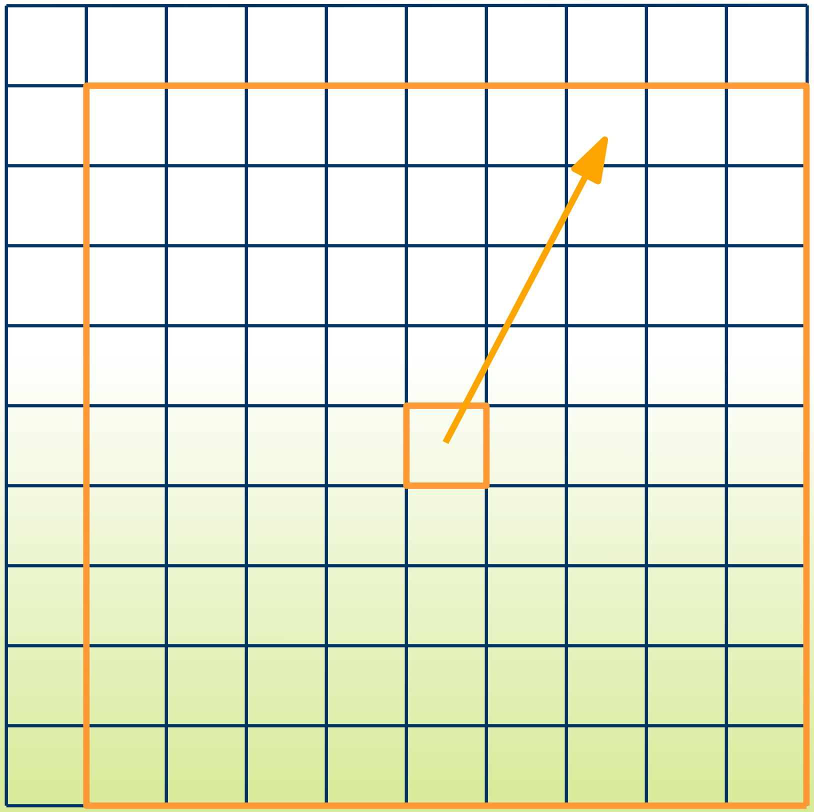


Pruning G

Want: O(n) edges; O(n) crossings; same reachability

Use grid with side length $1/2$

Def: *Neighborhood* of $\square$:
9×9 grid $\square$ is centered at

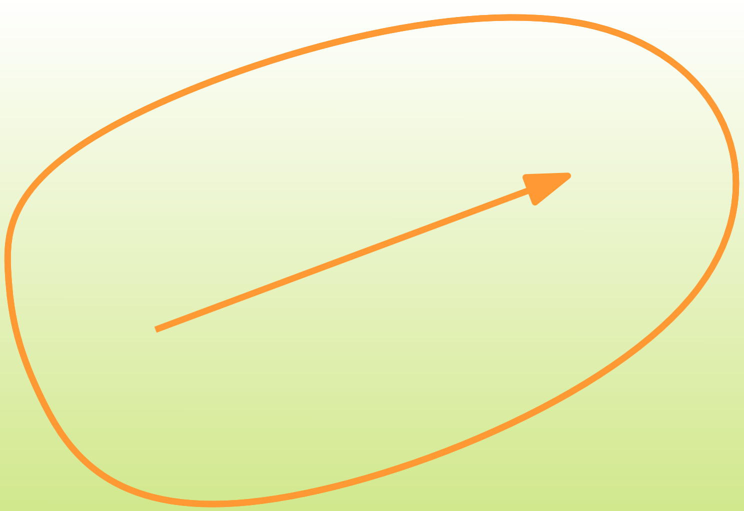


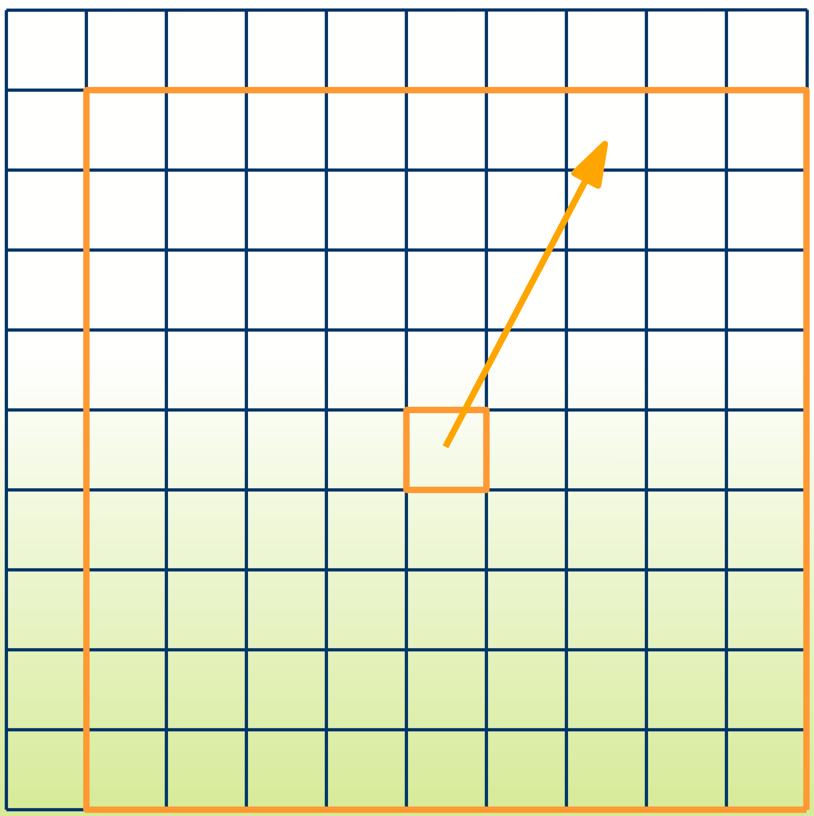
Pruning G

Want: O(n) edges; O(n) crossings; same reachability

Use grid with side length $1/2$

Def: *Neighborhood* of $\square$:
9×9 grid $\square$ is centered at



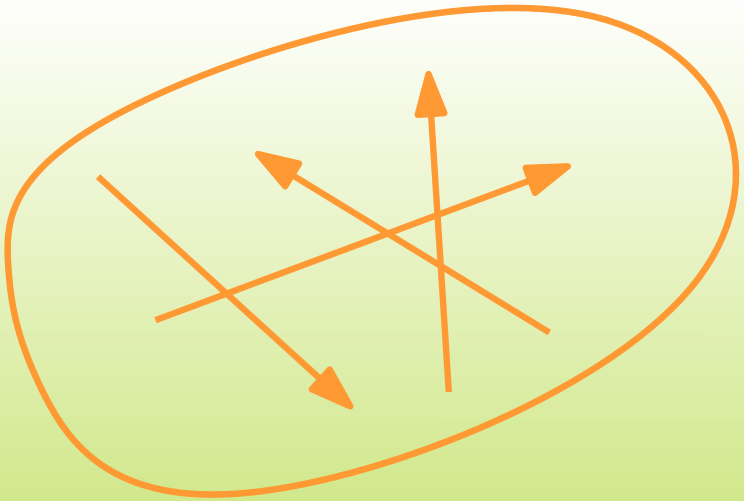


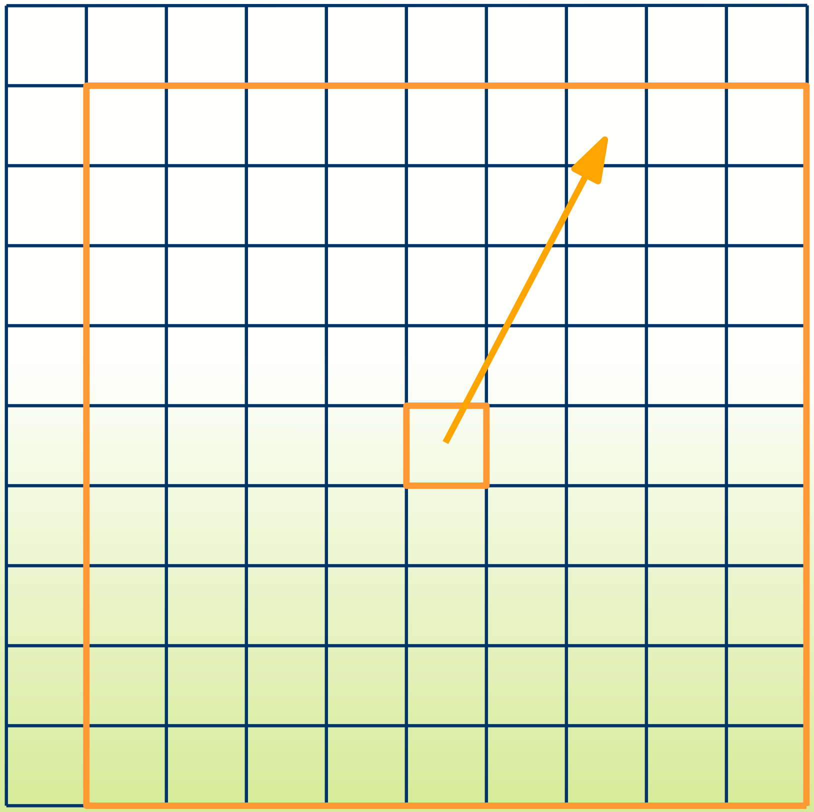
Pruning G

Want: O(n) edges; O(n) crossings; same reachability

Use grid with side length $1/2$

Def: *Neighborhood* of $\square$:
9×9 grid $\square$ is centered at



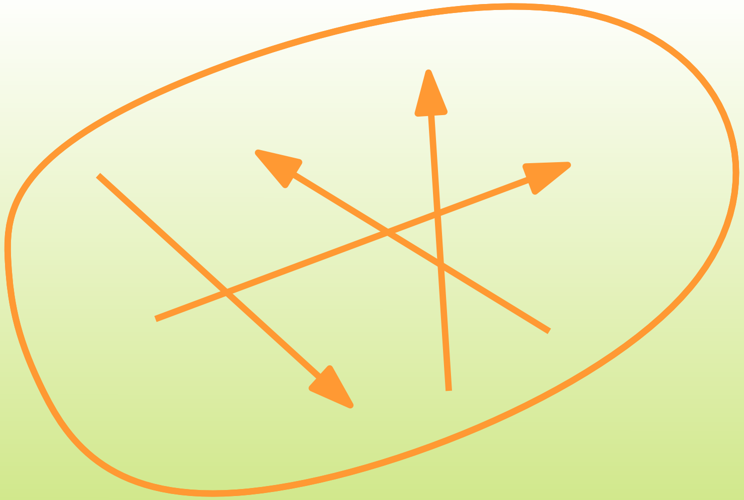


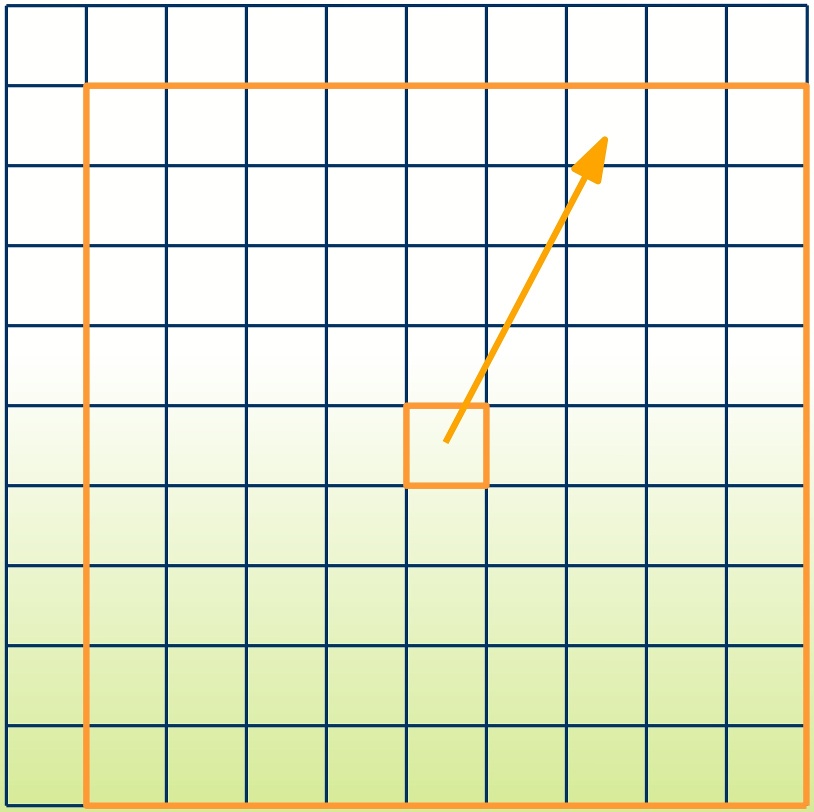
Pruning G

Want: O(n) edges; O(n) crossings; same reachability

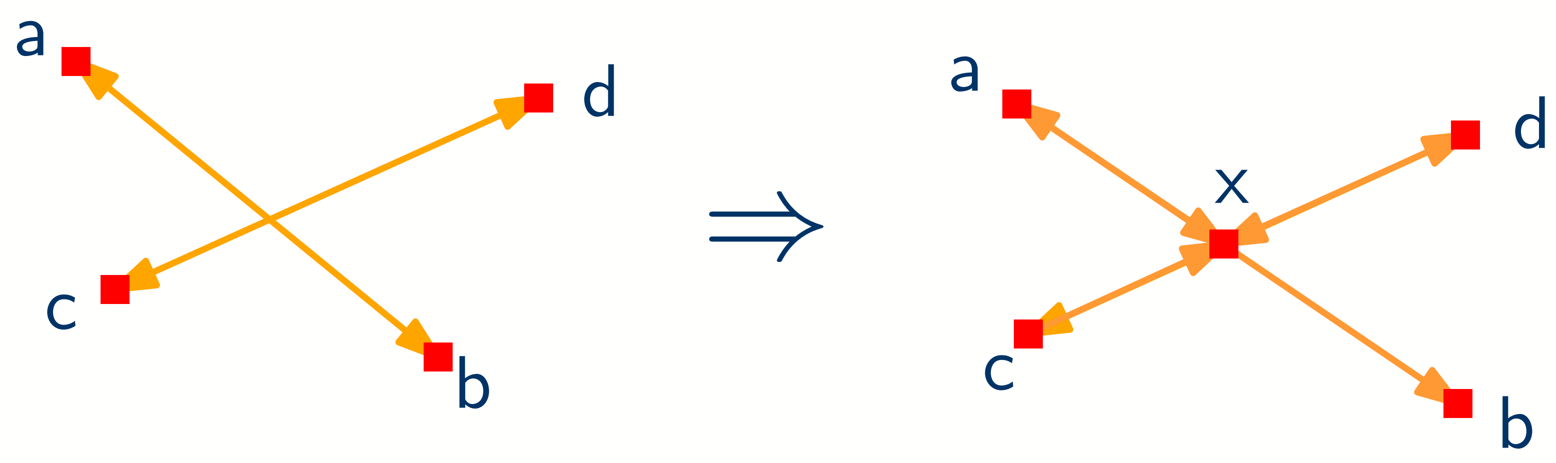
Use grid with side length $1/2$

Def: *Neighborhood* of $\square$:
9×9 grid $\square$ is centered at

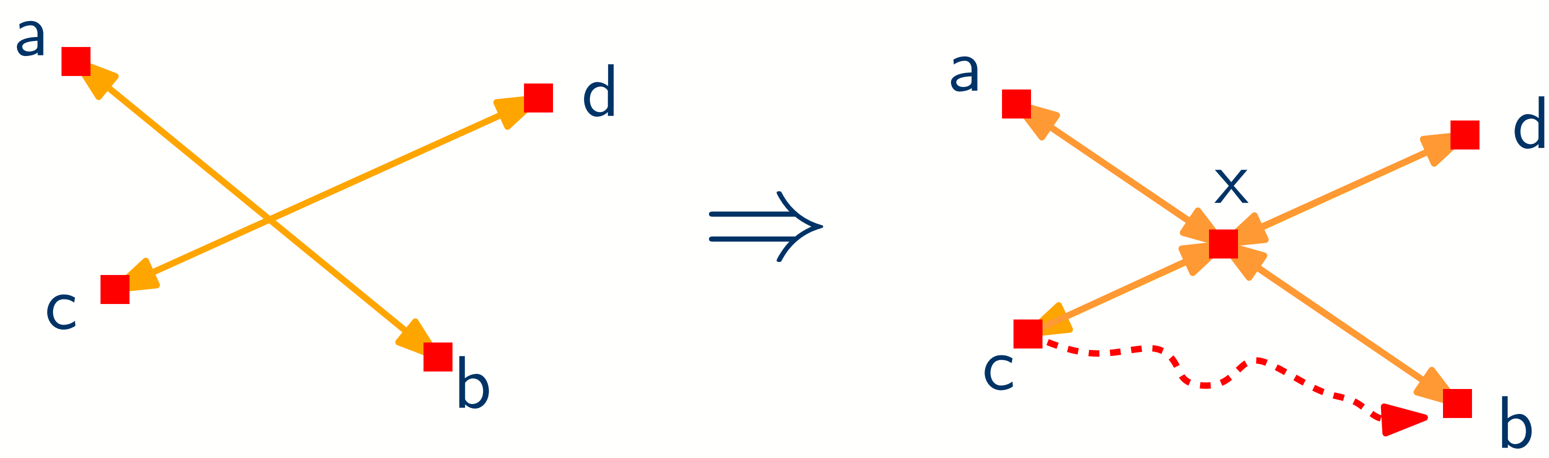




Resolving the crossings

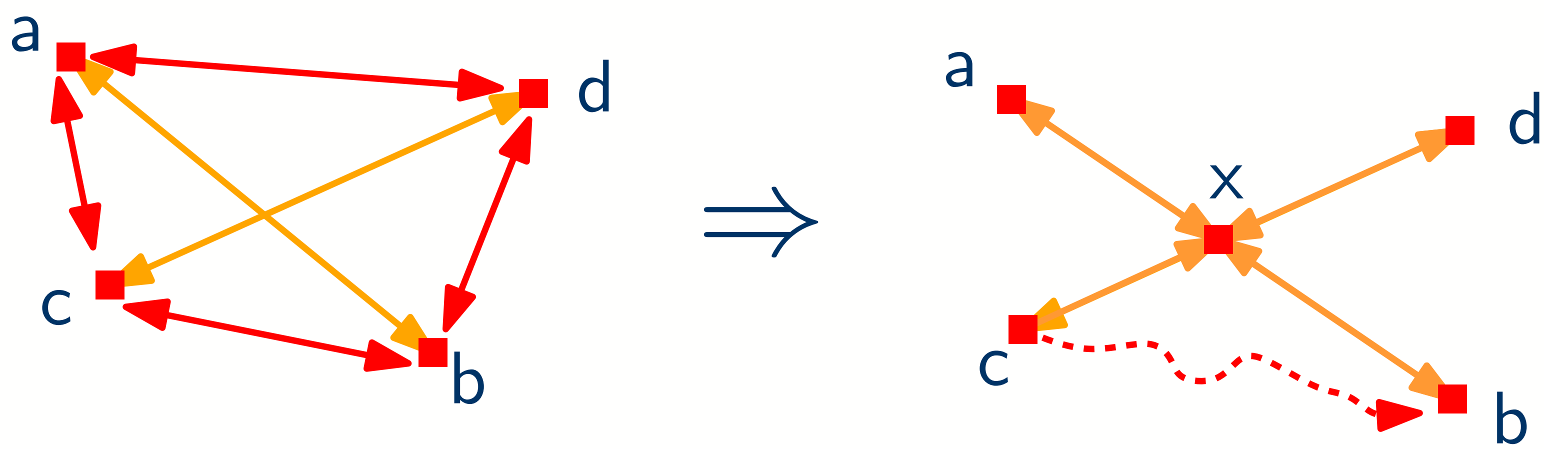


Resolving the crossings



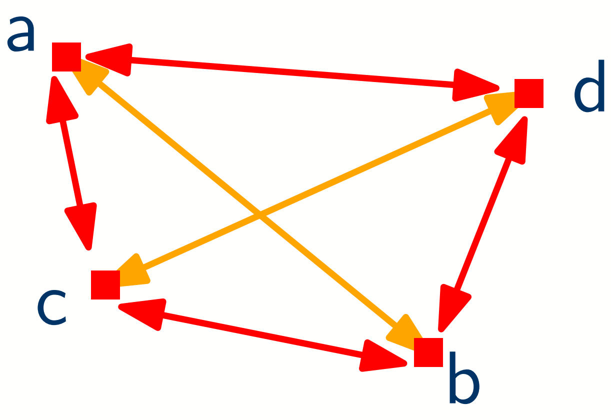
Why does this not change reachabilty?

Resolving the crossings

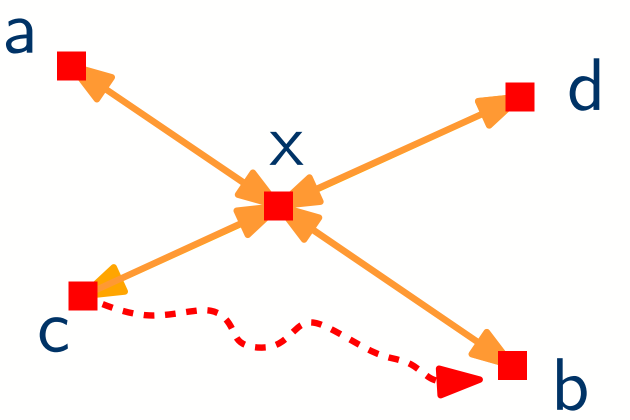


Why does this not change reachabilty?

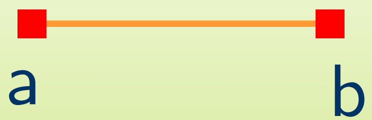
Resolving the crossings



$\Rightarrow$

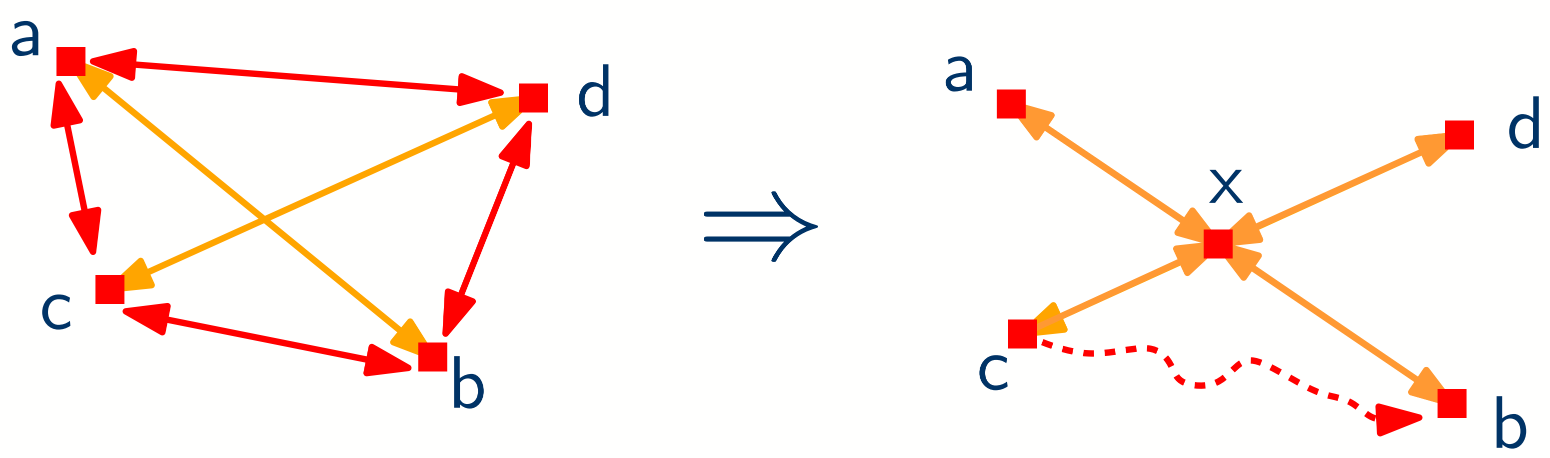


Why does this not change reachabilty?

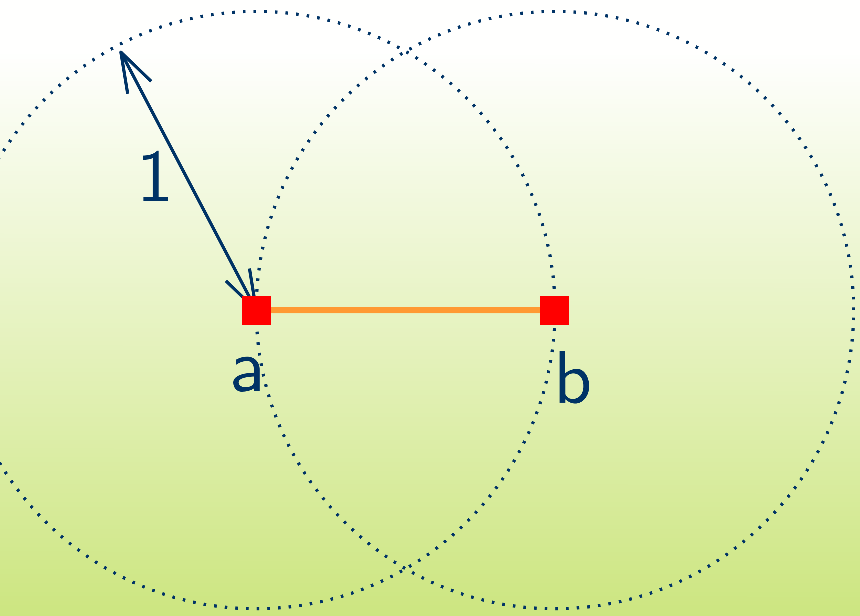


$$d(a, b) \leq d(c, d)$$
$$d(a, b) = 1$$

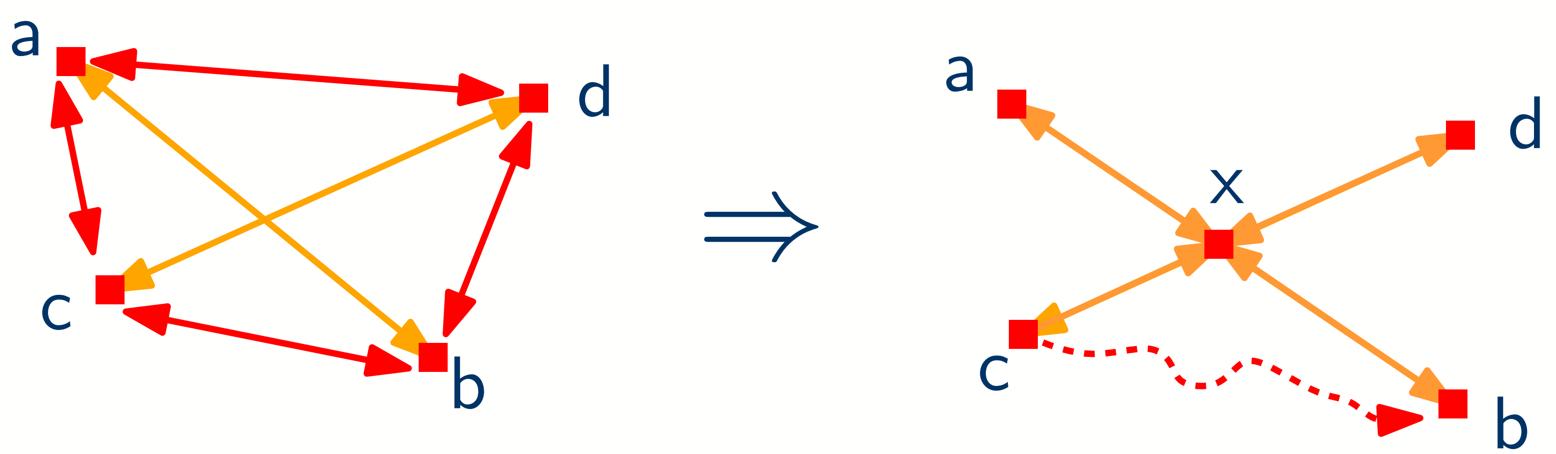
Resolving the crossings



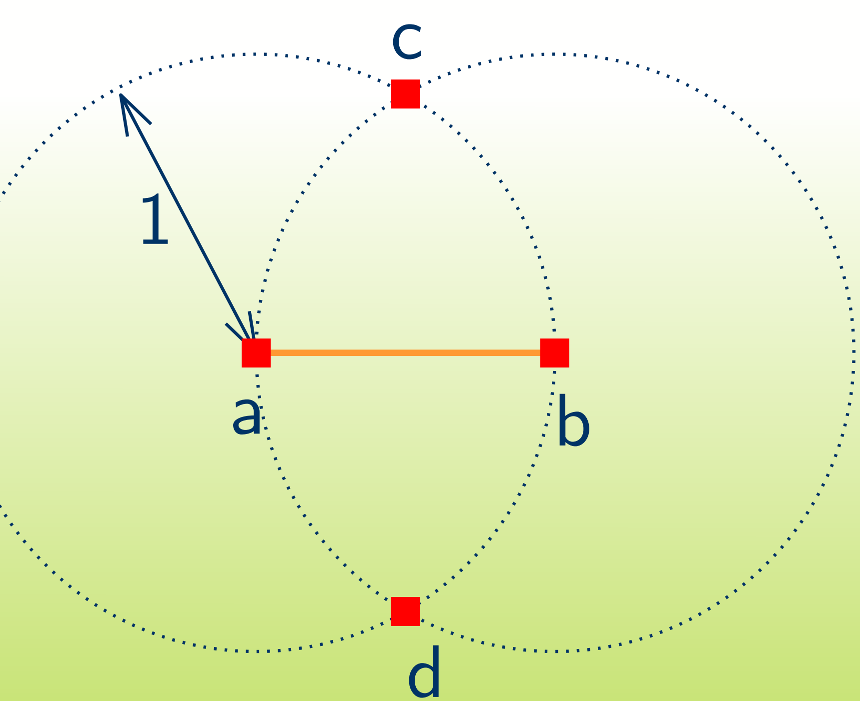
Why does this not change reachabilty?



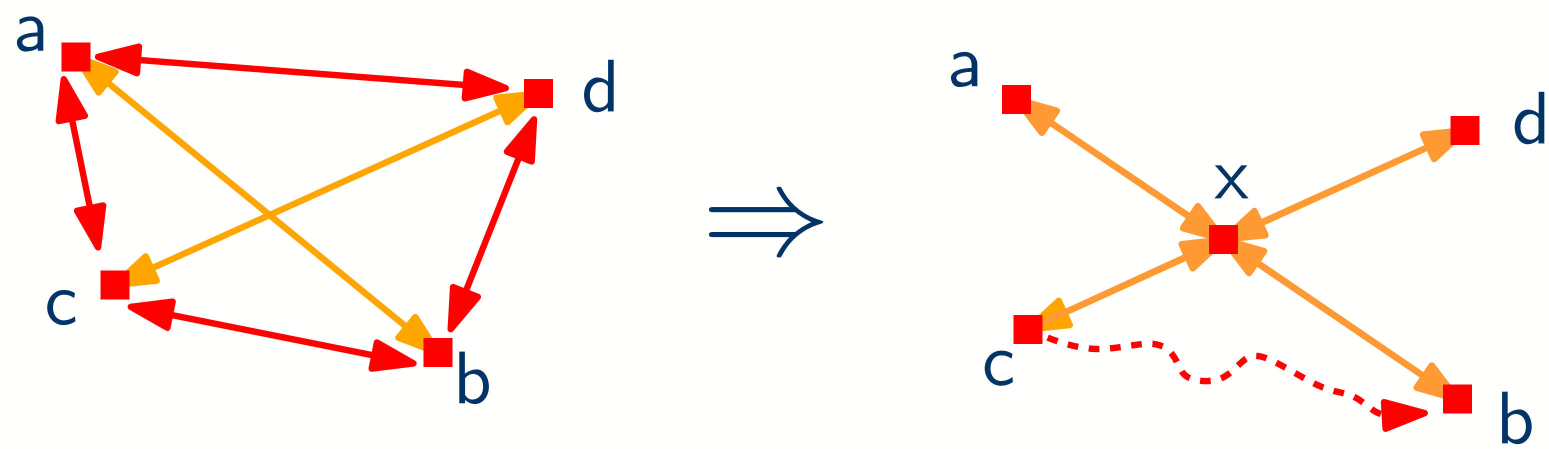
Resolving the crossings



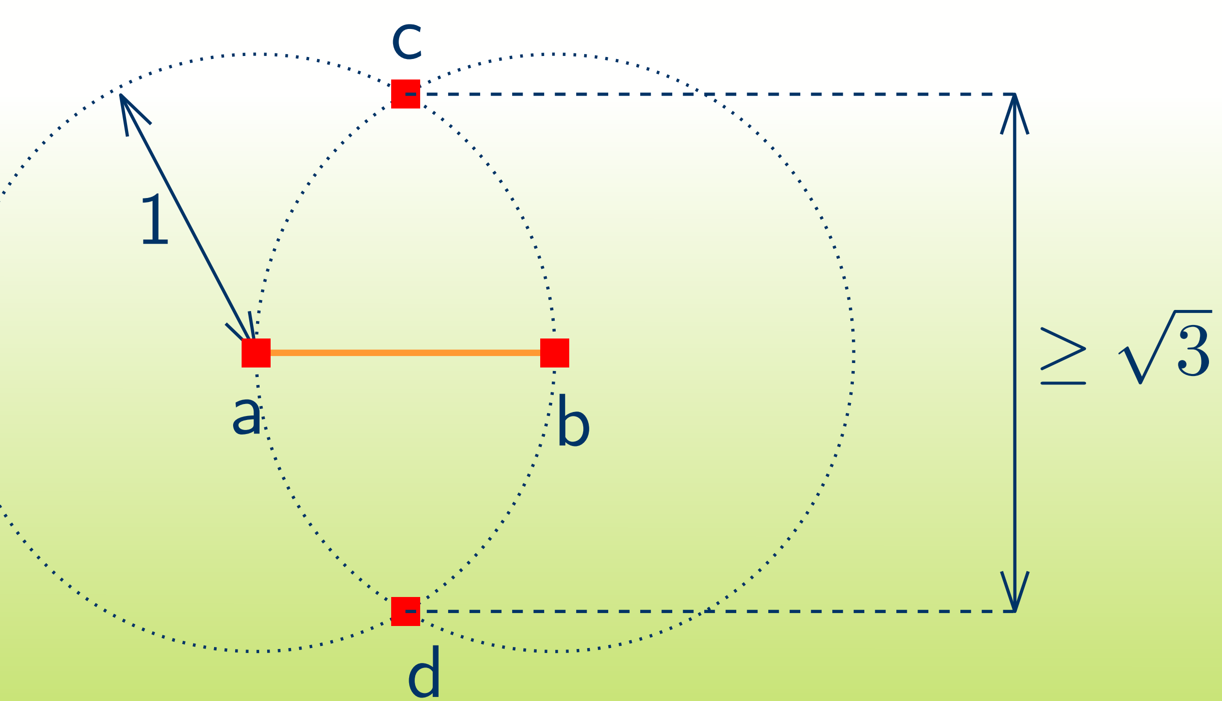
Why does this not change reachabilty?



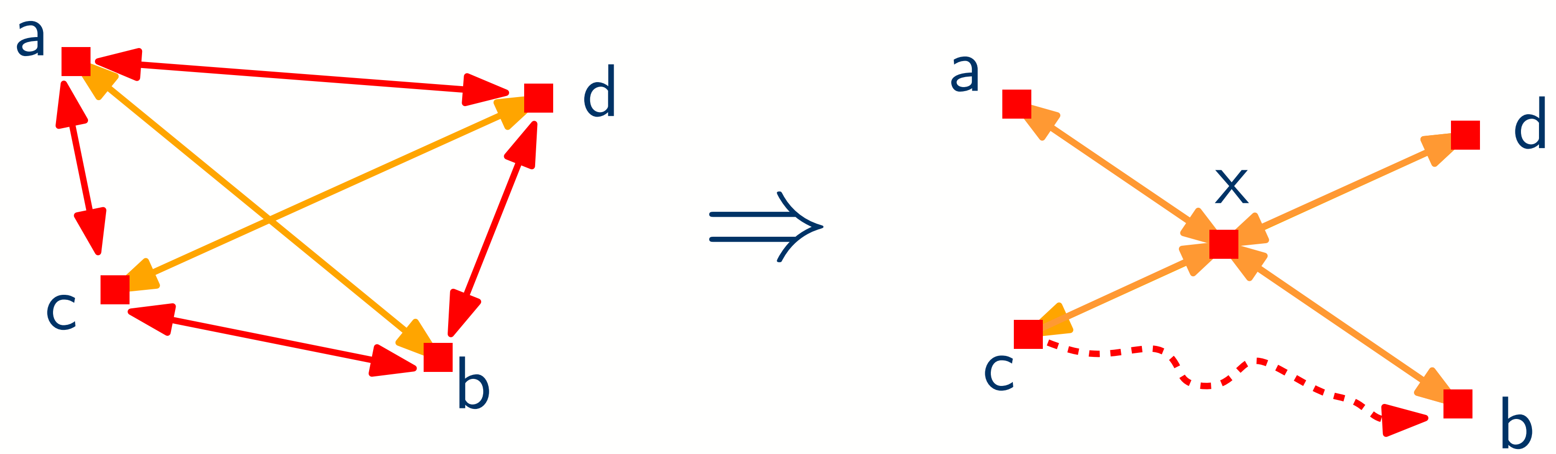
Resolving the crossings



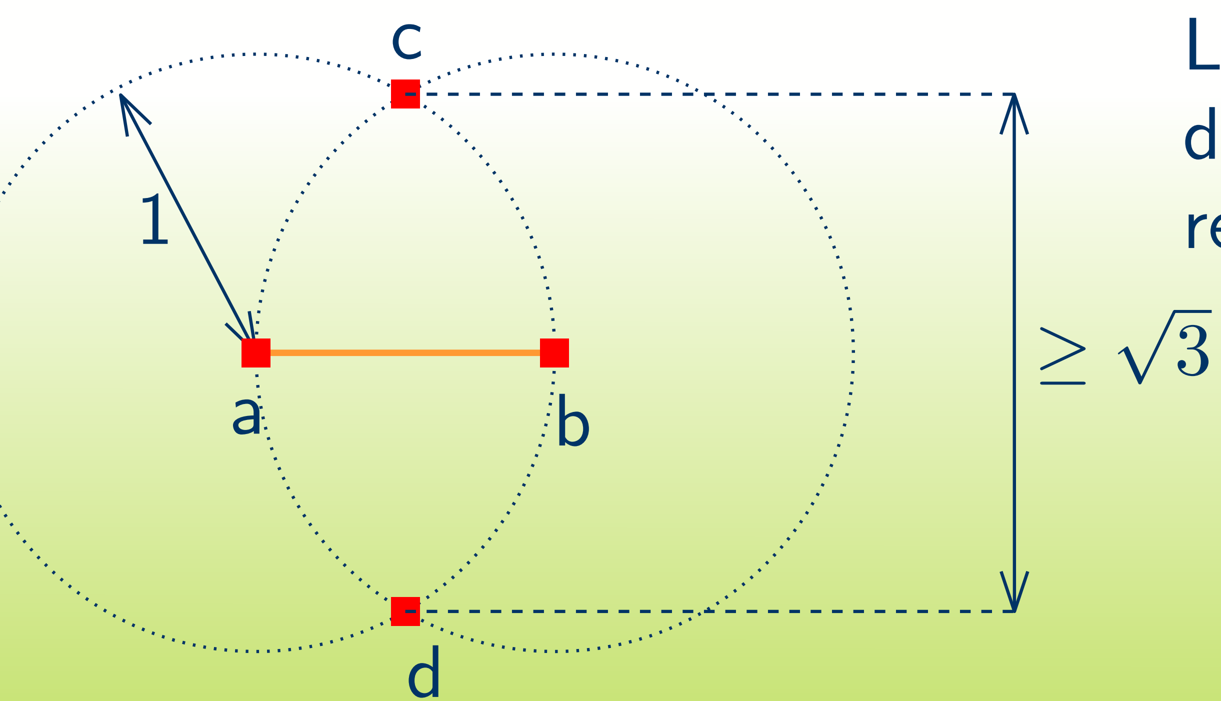
Why does this not change reachabilty?



Resolving the crossings

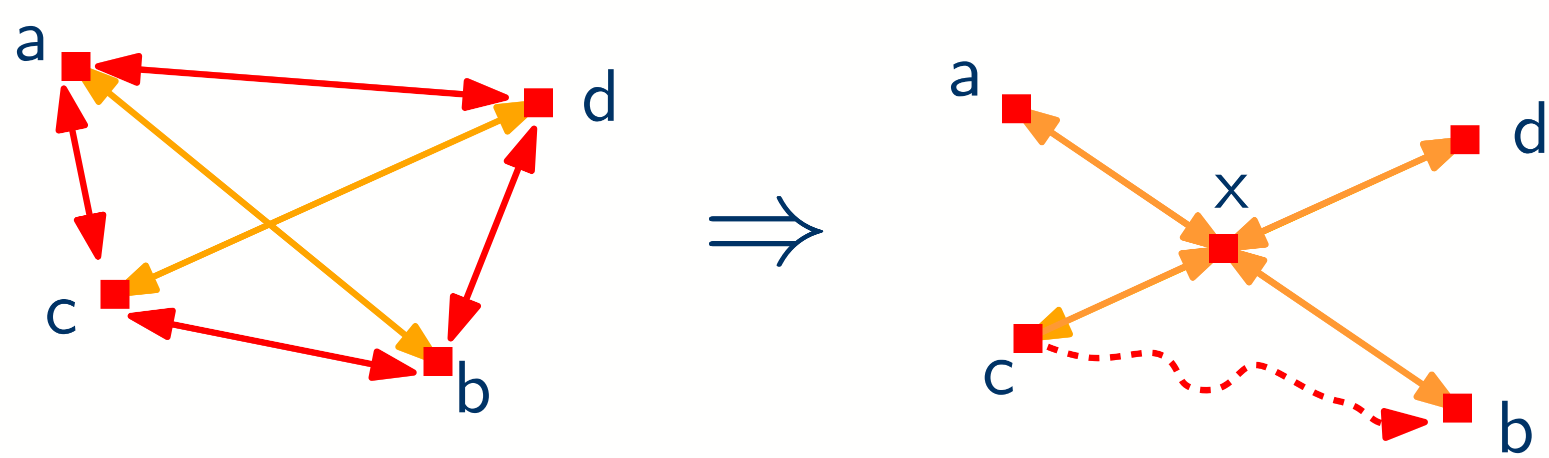


Why does this not change reachabilty?

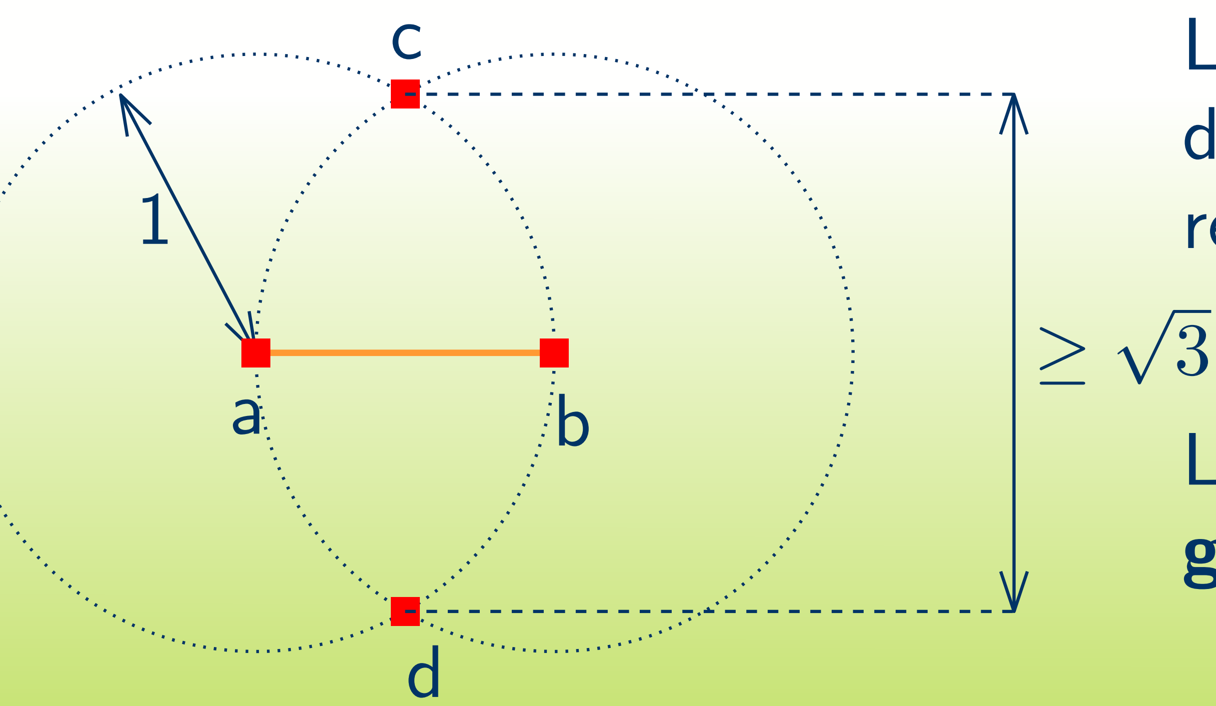


Lemma: Resolving crossings does not change the reachability **locally**.

Resolving the crossings



Why does this not change reachabilty?



Lemma: Resolving crossings does not change the reachability **locally**.

$\Downarrow$

Lemma 2: This is also true **globally**.

Results

	$P(n)$	$S(n)$	$Q(n)$	Restrictions
$d = 1$	$O(n \log n)$	$O(n)$	$O(1)$	none
$d = 2$	$O(n \log n)$	$O(n \log n)$	$O(1)$	radii in $[1, \sqrt{3})$