Parameterized Algorithms

Exercise 6 – Sheet 5, Evaluation, Treewidth

Elly, Jean-Pierre, Wendy

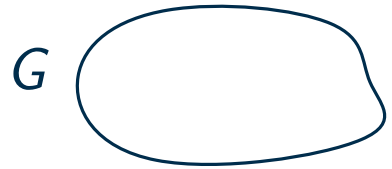
Sheet 5: Longest Cycle

Given a graph G and parameter k,
does G contain a cycle of length at least k?

Sheet 5: Longest Cycle

Given a graph G and parameter k,
does G contain a cycle of length at least k?

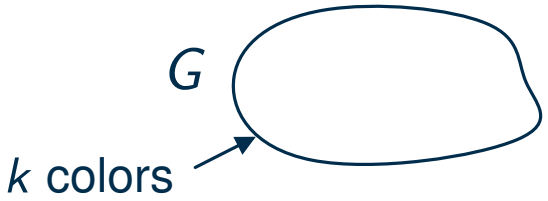
Idea 1: Find a cycle of length exactly k



Sheet 5: Longest Cycle

Given a graph G and parameter k,
does G contain a cycle of length at least k?

Idea 1: Find a cycle of length exactly k

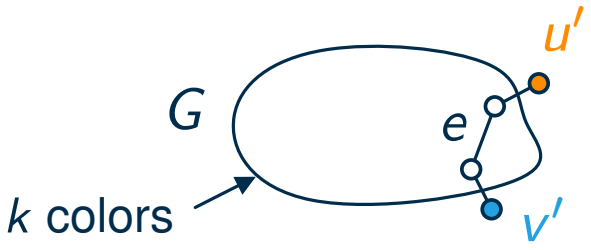


Sheet 5: Longest Cycle

Given a graph G and parameter k,
does G contain a cycle of length at least k?

Idea 1: Find a cycle of length exactly k

- attach new leaves u' and v' with new colors to edge $e = \{u, v\}$

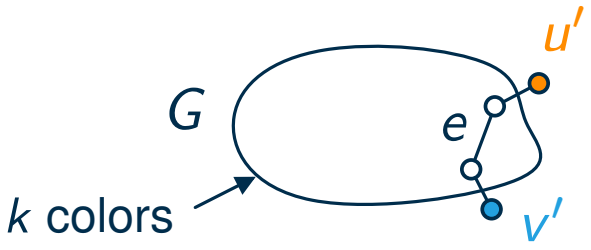


Sheet 5: Longest Cycle

Given a graph G and parameter k,
does G contain a cycle of length at least k?

Idea 1: Find a cycle of length exactly k

- attach new leaves u' and v' with new colors to edge $e = \{u, v\}$
- solve colorful longest path for $k + 2$

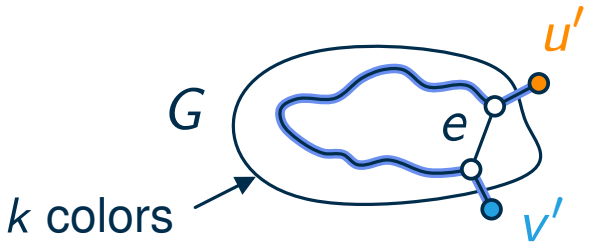


Sheet 5: Longest Cycle

Given a graph G and parameter k,
does G contain a cycle of length at least k?

Idea 1: Find a cycle of length exactly k

- attach new leaves u' and v' with new colors to edge $e = \{u, v\}$
- solve colorful longest path for $k + 2$

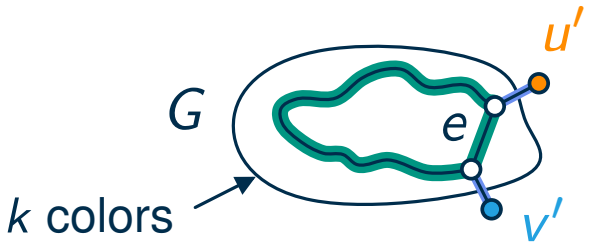


Sheet 5: Longest Cycle

Given a graph G and parameter k,
does G contain a cycle of length at least k?

Idea 1: Find a cycle of length exactly k

- attach new leaves u' and v' with new colors to edge $e = \{u, v\}$
- solve colorful longest path for $k + 2$



Sheet 5: Longest Cycle

Given a graph G and parameter k,
does G contain a cycle of length at least k?

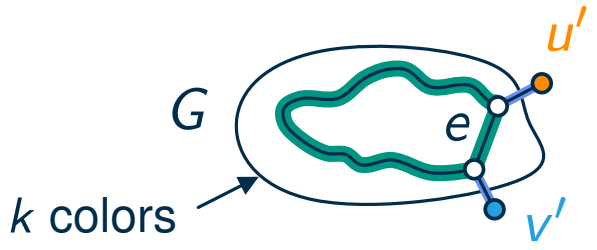
Idea 1: Find a cycle of length exactly k

- attach new leaves u' and v' with new colors to edge $e = \{u, v\}$
- solve colorful longest path for $k + 2$

algo from lecture

Colorful Longest Path: Given a graph $G = (V, E)$, a parameter k, and a partition (coloring) $V_1, \ldots, V_k$ of V, is there a colorful k-vertex path?

$\rightarrow$ solvable in $O(2^k km)$ time.



Sheet 5: Longest Cycle

Given a graph G and parameter k,
does G contain a cycle of length at least k?

Idea 1: Find a cycle of length exactly k

- attach new leaves u' and v' with new colors to edge $e = \{u, v\}$
- solve colorful longest path for $k + 2$ — algo from lecture
- colorings: (n, k)-perfect family of hash functions

Colorful Longest Path: Given a graph $G = (V, E)$, a parameter k, and a partition (coloring) $V_1, \ldots, V_k$ of V, is there a colorful k-vertex path?

$\rightarrow$ solvable in $O(2^k km)$ time.

Sheet 5: Longest Cycle

Given a graph G and parameter k,
does G contain a cycle of length at least k?

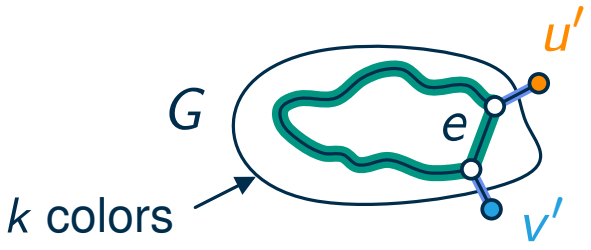
Colorful Longest Path: Given a graph $G = (V, E)$, a parameter k, and a partition (coloring) $V_1, \ldots, V_k$ of V, is there a colorful k-vertex path?

$\rightarrow$ solvable in $O(2^k km)$ time.

Idea 1: Find a cycle of length exactly k

- attach new leaves u' and v' with new colors to edge $e = \{u, v\}$
- solve colorful longest path for $k + 2$ — algo from lecture
- colorings: (n, k)-perfect family of hash functions

} once per edge



Sheet 5: Longest Cycle

Given a graph G and parameter k,
does G contain a cycle of length at least k?

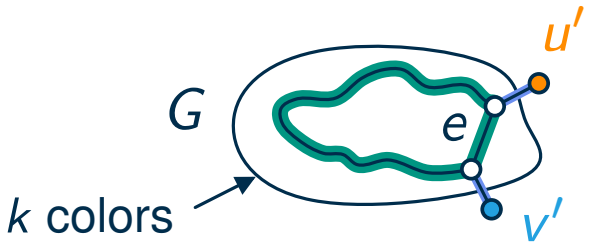
Colorful Longest Path: Given a graph $G = (V, E)$, a parameter k, and a partition (coloring) $V_1, \ldots, V_k$ of V, is there a colorful k-vertex path?

$\rightarrow$ solvable in $O(2^k km)$ time.

Idea 1: Find a cycle of length exactly k

- attach new leaves u' and v' with new colors to edge $e = \{u, v\}$
- solve colorful longest path for $k + 2$ (algo from lecture)
- colorings: (n, k)-perfect family of hash functions

} once per edge



Idea 2: Iteratively shrink cycles

Sheet 5: Longest Cycle

Given a graph G and parameter k,
does G contain a cycle of length at least k?

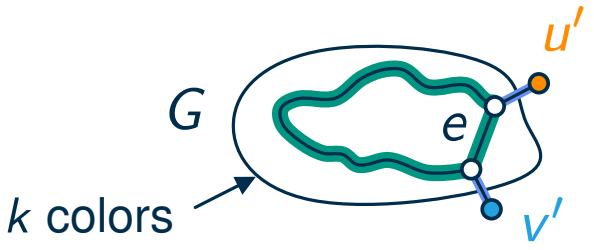
Colorful Longest Path: Given a graph $G = (V, E)$, a parameter k, and a partition (coloring) $V_1, \ldots, V_k$ of V, is there a colorful k-vertex path?

$\rightarrow$ solvable in $O(2^k km)$ time.

Idea 1: Find a cycle of length exactly k

- attach new leaves u' and v' with new colors to edge $e = \{u, v\}$
- solve colorful longest path for $k + 2$ (algo from lecture)
- colorings: (n, k)-perfect family of hash functions

} once per edge



Idea 2: Iteratively shrink cycles

- contract one edge

Sheet 5: Longest Cycle

Given a graph G and parameter k,
does G contain a cycle of length at least k?

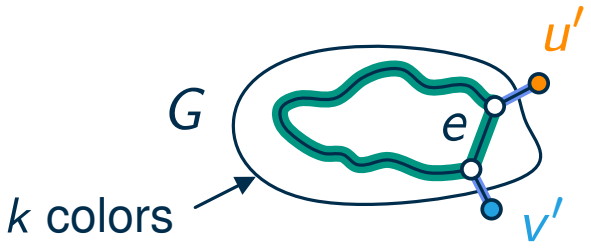
Colorful Longest Path: Given a graph $G = (V, E)$, a parameter k, and a partition (coloring) $V_1, \ldots, V_k$ of V, is there a colorful k-vertex path?

$\rightarrow$ solvable in $O(2^k km)$ time.

Idea 1: Find a cycle of length exactly k

- attach new leaves u' and v' with new colors to edge $e = \{u, v\}$
- solve colorful longest path for $k + 2$ (algo from lecture)
- colorings: (n, k)-perfect family of hash functions

} once per edge



Idea 2: Iteratively shrink cycles

- contract one edge

how does the length of the longest cycle change?

Sheet 5: Longest Cycle

Given a graph G and parameter k,
does G contain a cycle of length at least k?

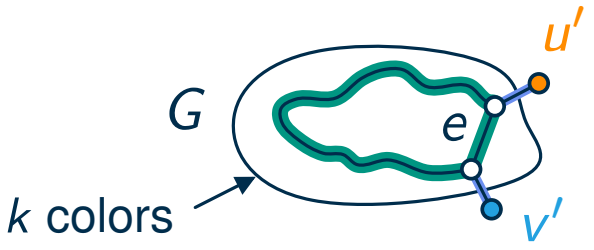
Colorful Longest Path: Given a graph $G = (V, E)$, a parameter k, and a partition (coloring) $V_1, \ldots, V_k$ of V, is there a colorful k-vertex path?

$\rightarrow$ solvable in $O(2^k km)$ time.

Idea 1: Find a cycle of length exactly k

- attach new leaves u' and v' with new colors to edge $e = \{u, v\}$
- solve colorful longest path for $k + 2$ — algo from lecture
- colorings: (n, k)-perfect family of hash functions

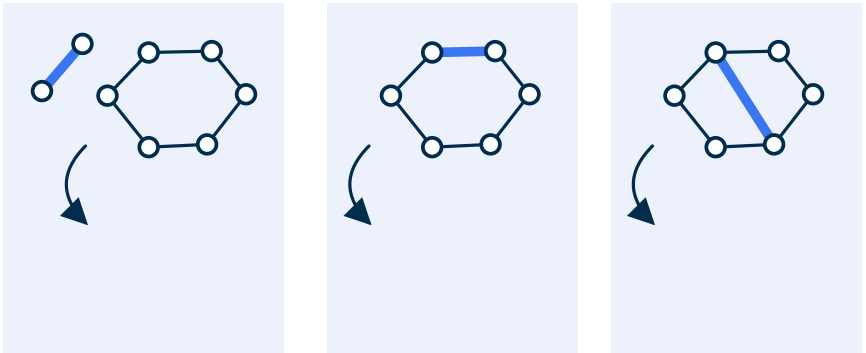
} once per edge



Idea 2: Iteratively shrink cycles

- contract one edge

how does the length of the longest cycle change?



Sheet 5: Longest Cycle

Given a graph G and parameter k,
does G contain a cycle of length at least k?

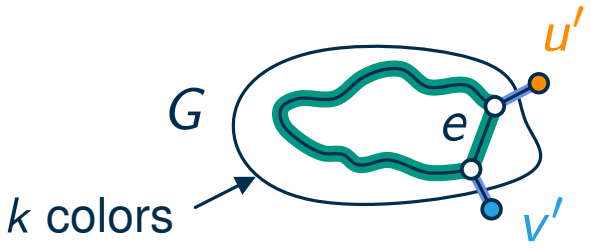
Colorful Longest Path: Given a graph $G = (V, E)$, a parameter k, and a partition (coloring) $V_1, \ldots, V_k$ of V, is there a colorful k-vertex path?

$\rightarrow$ solvable in $O(2^k km)$ time.

Idea 1: Find a cycle of length exactly k

- attach new leaves u' and v' with new colors to edge $e = \{u, v\}$
- solve colorful longest path for $k + 2$ (algo from lecture)
- colorings: (n, k)-perfect family of hash functions

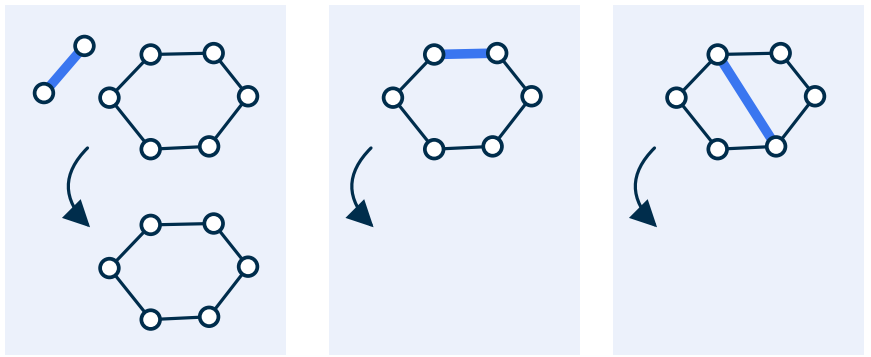
} once per edge



Idea 2: Iteratively shrink cycles

- contract one edge

how does the length of the longest cycle change?



Sheet 5: Longest Cycle

Given a graph G and parameter k,
does G contain a cycle of length at least k?

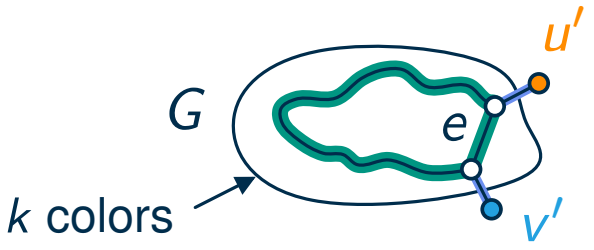
Colorful Longest Path: Given a graph $G = (V, E)$, a parameter k, and a partition (coloring) $V_1, \ldots, V_k$ of V, is there a colorful k-vertex path?

$\rightarrow$ solvable in $O(2^k km)$ time.

Idea 1: Find a cycle of length exactly k

- attach new leaves u' and v' with new colors to edge $e = \{u, v\}$
- solve colorful longest path for $k + 2$ (algo from lecture)
- colorings: (n, k)-perfect family of hash functions

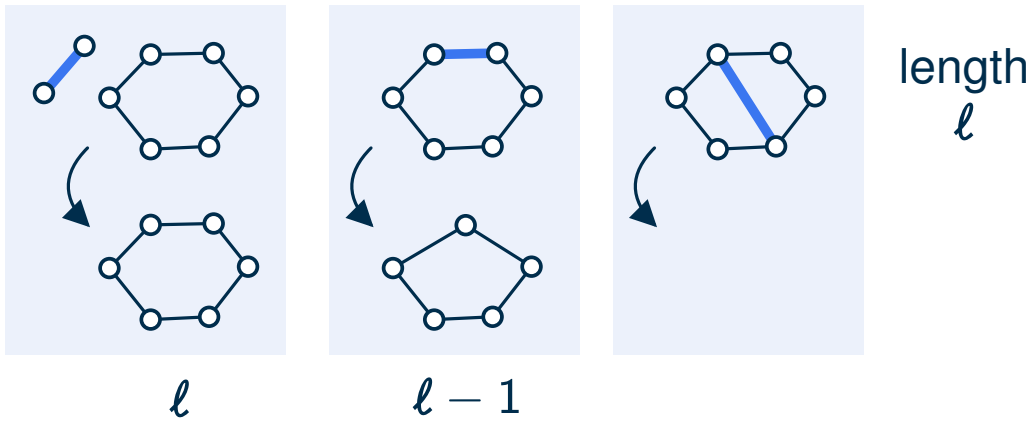
} once per edge



Idea 2: Iteratively shrink cycles

- contract one edge

how does the length of the longest cycle change?



Sheet 5: Longest Cycle

Given a graph G and parameter k,
does G contain a cycle of length at least k?

Colorful Longest Path: Given a graph $G = (V, E)$, a parameter k, and a partition (coloring) $V_1, \ldots, V_k$ of V, is there a colorful k-vertex path?

$\rightarrow$ solvable in $O(2^k km)$ time.

Idea 1: Find a cycle of length exactly k

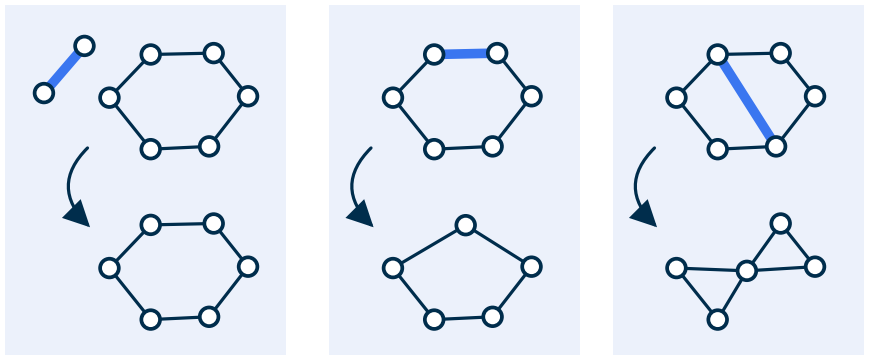
- attach new leaves u' and v' with new colors to edge $e = \{u, v\}$
- solve colorful longest path for $k + 2$ (algo from lecture)
- colorings: (n, k)-perfect family of hash functions

} once per edge

Idea 2: Iteratively shrink cycles

- contract one edge

how does the length of the longest cycle change?



length ℓ

Sheet 5: Longest Cycle

Given a graph G and parameter k,
does G contain a cycle of length at least k?

Colorful Longest Path: Given a graph $G = (V, E)$, a parameter k, and a partition (coloring) $V_1, \ldots, V_k$ of V, is there a colorful k-vertex path?

$\rightarrow$ solvable in $O(2^k km)$ time.

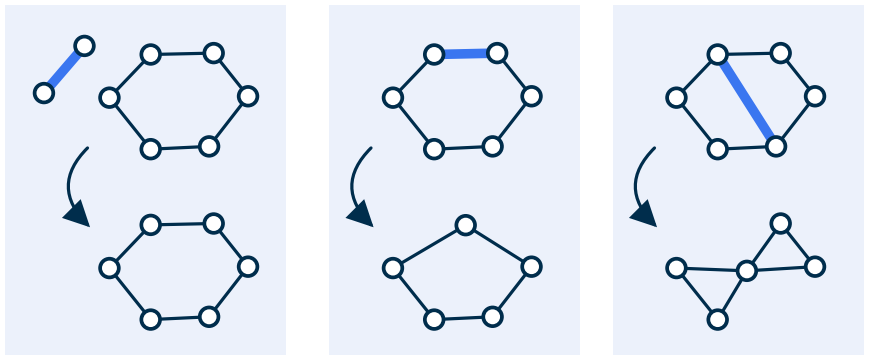
Idea 1: Find a cycle of length exactly k

- attach new leaves u' and v' with new colors to edge $e = \{u, v\}$
- solve colorful longest path for $k + 2$ (algo from lecture)
- colorings: (n, k)-perfect family of hash functions

} once per edge

Idea 2: Iteratively shrink cycles

- contract one edge $\rightarrow$ reduces length of longest cycle by at most a factor of 2



Sheet 5: Longest Cycle

Given a graph G and parameter k,
does G contain a cycle of length at least k?

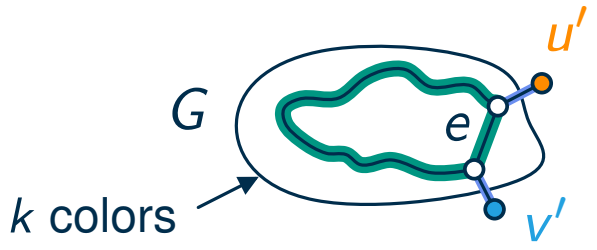
Colorful Longest Path: Given a graph $G = (V, E)$, a parameter k, and a partition (coloring) $V_1, \ldots, V_k$ of V, is there a colorful k-vertex path?

$\rightarrow$ solvable in $O(2^k km)$ time.

Idea 1: Find a cycle of length exactly k

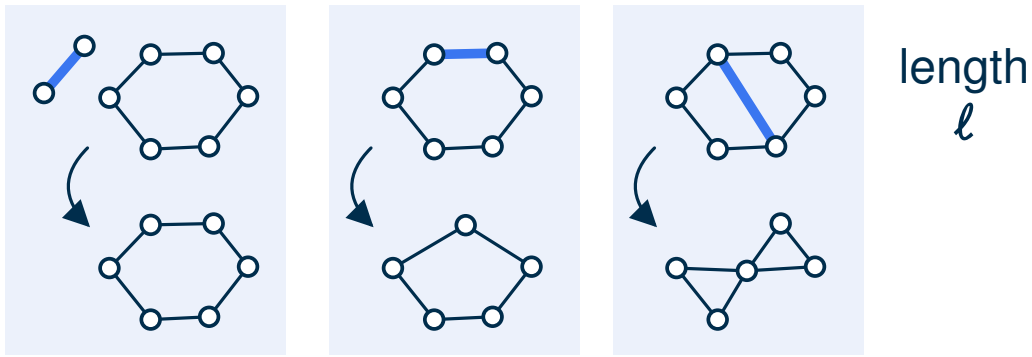
- attach new leaves u' and v' with new colors to edge $e = \{u, v\}$
- solve colorful longest path for $k + 2$ (algo from lecture)
- colorings: (n, k)-perfect family of hash functions

} once per edge



Idea 2: Iteratively shrink cycles

- contract one edge $\rightarrow$ reduces length of longest cycle by at most a factor of 2
- check for cycles of all lengths from k to $2k$



Sheet 5: Longest Cycle

Given a graph G and parameter k,
does G contain a cycle of length at least k?

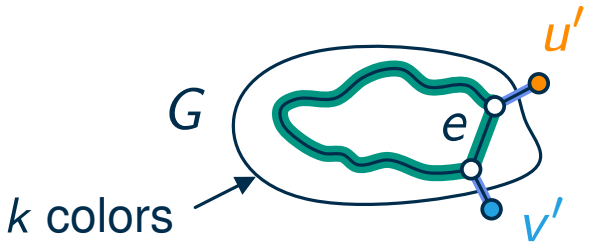
Colorful Longest Path: Given a graph $G = (V, E)$, a parameter k, and a partition (coloring) $V_1, \ldots, V_k$ of V, is there a colorful k-vertex path?

$\rightarrow$ solvable in $O(2^k km)$ time.

Idea 1: Find a cycle of length exactly k

- attach new leaves u' and v' with new colors to edge $e = \{u, v\}$
- solve colorful longest path for $k + 2$ (algo from lecture)
- colorings: (n, k)-perfect family of hash functions

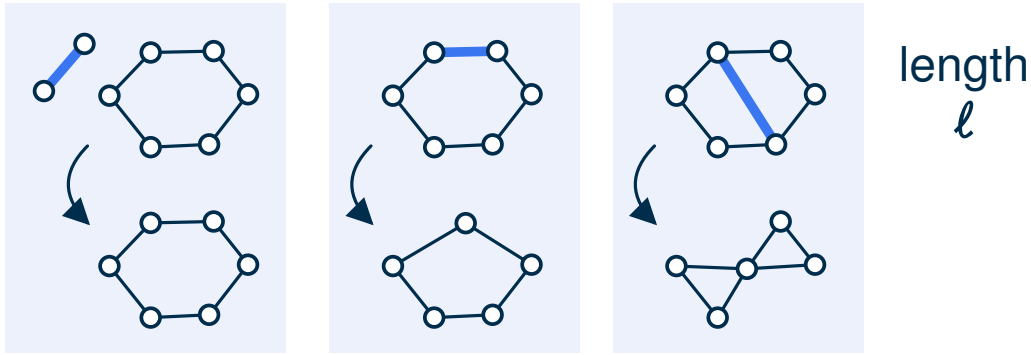
} once per edge



Idea 2: Iteratively shrink cycles

- contract one edge $\rightarrow$ reduces length of longest cycle by at most a factor of 2
- check for cycles of all lengths from k to $2k$

} iterate

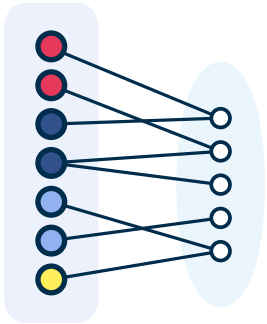


Sheet 5: Colorful Left-Right Dominating Set

Given a bipartite graph G with sides L and R and a coloring of L into ℓ colors. Is there a colorful set $X \subseteq L$ of size exactly ℓ that dominates all vertices in R?

Sheet 5: Colorful Left-Right Dominating Set

Given a bipartite graph G with sides L and R and a coloring of L into ℓ colors. Is there a colorful set $X \subseteq L$ of size exactly ℓ that dominates all vertices in R?

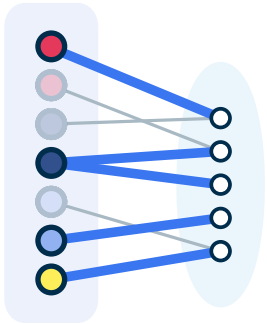


00

$\ell = 4$

Sheet 5: Colorful Left-Right Dominating Set

Given a bipartite graph G with sides L and R and a coloring of L into ℓ colors. Is there a colorful set $X \subseteq L$ of size exactly ℓ that dominates all vertices in R?



00

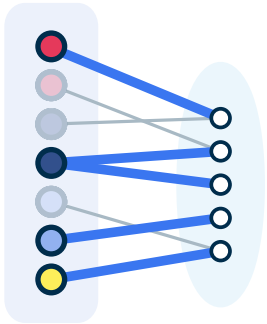
$\ell = 4$

Sheet 5: Colorful Left-Right Dominating Set

Given a bipartite graph G with sides L and R and a coloring of L into ℓ colors. Is there a colorful set $X \subseteq L$ of size exactly ℓ that dominates all vertices in R?

Show: no poly kernelization unless NP $\subseteq$ coNP/ poly

parameter: $|R| + \ell$



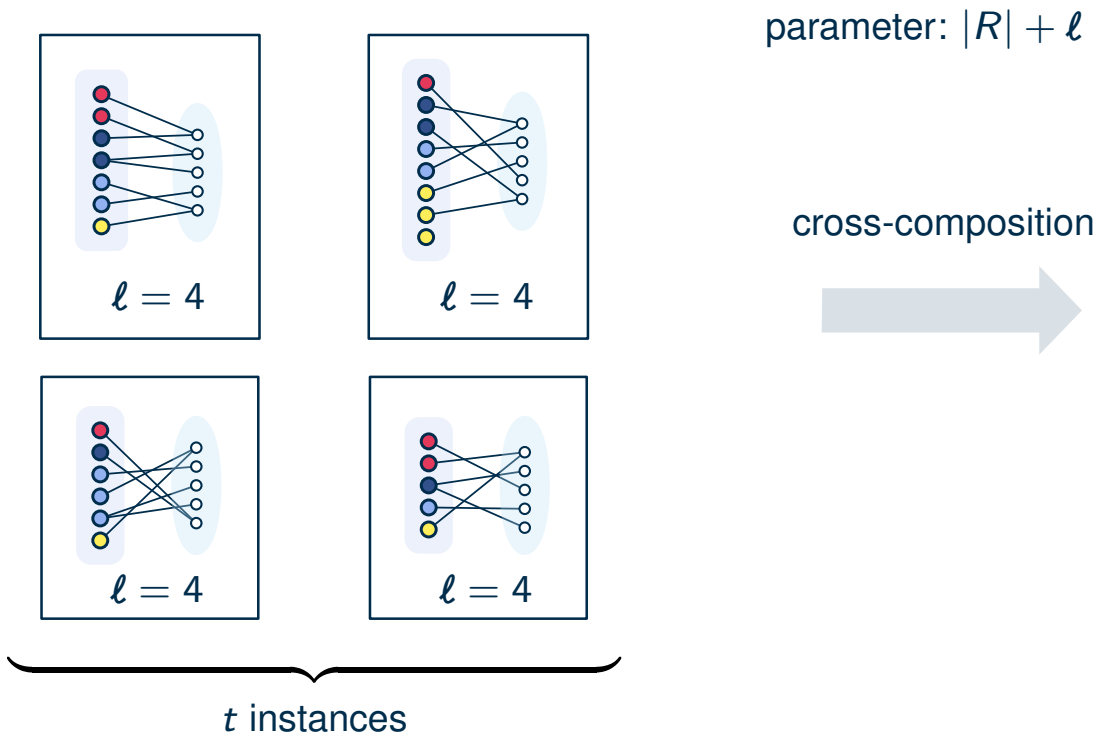
00

$\ell = 4$

Sheet 5: Colorful Left-Right Dominating Set

Given a bipartite graph G with sides L and R and a coloring of L into ℓ colors. Is there a colorful set $X \subseteq L$ of size exactly ℓ that dominates all vertices in R?

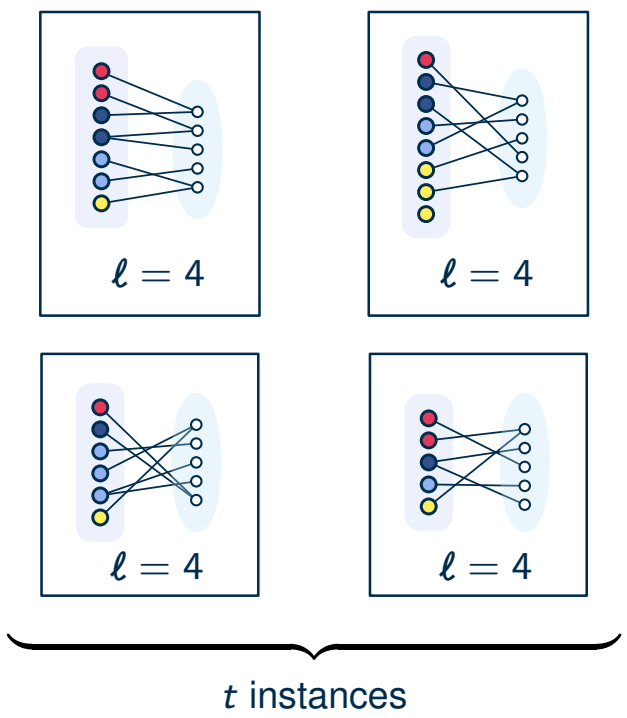
Show: no poly kernelization unless NP $\subseteq$ coNP/ poly



Sheet 5: Colorful Left-Right Dominating Set

Given a bipartite graph G with sides L and R and a coloring of L into ℓ colors. Is there a colorful set $X \subseteq L$ of size exactly ℓ that dominates all vertices in R?

Show: no poly kernelization unless NP $\subseteq$ coNP/ poly



parameter: $|R| + \ell$

cross-composition

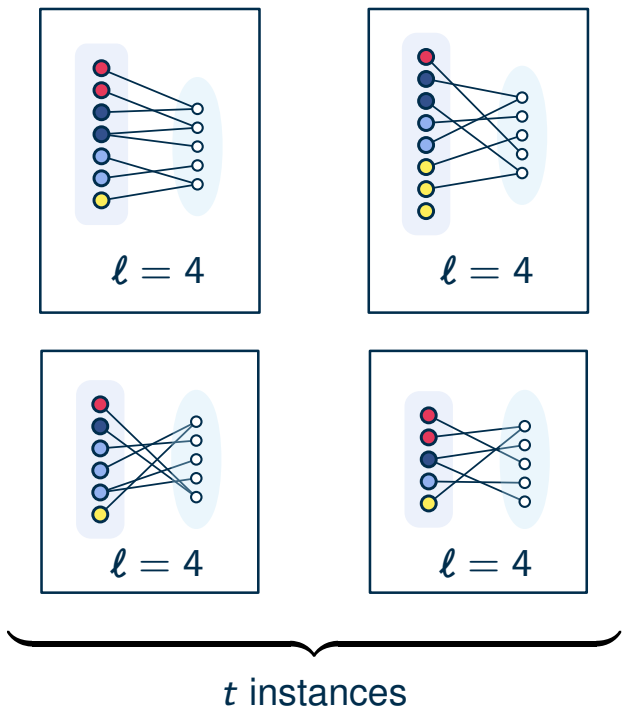
“similar”?

- same $|R|$ and ℓ

Sheet 5: Colorful Left-Right Dominating Set

Given a bipartite graph G with sides L and R and a coloring of L into ℓ colors. Is there a colorful set $X \subseteq L$ of size exactly ℓ that dominates all vertices in R?

Show: no poly kernelization unless NP $\subseteq$ coNP/ poly



parameter: $|R| + \ell$

cross-composition

"similar"?

- same $|R|$ and ℓ

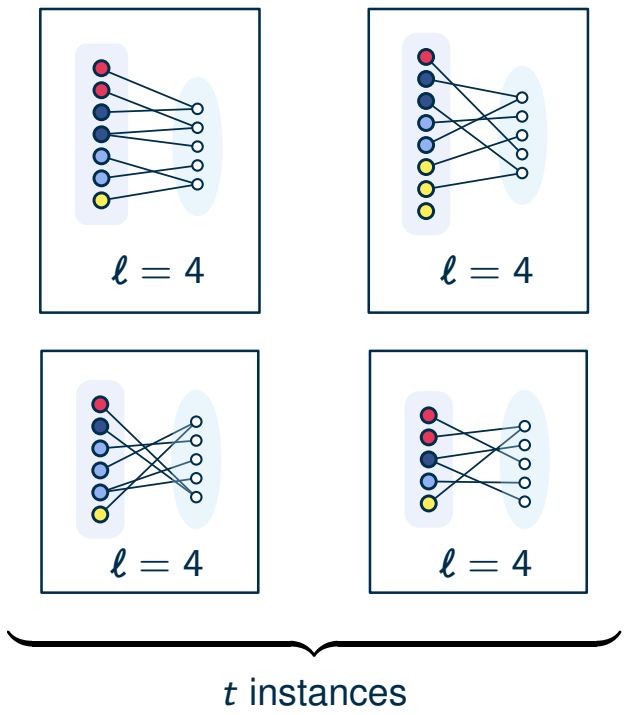
What do we need?

- instance that has a solution if and only if at least one input instance has a solution
- parameter polynomial in $\max_{i \in [t]} |x_i| + \log t$
- construction runs polynomial in $\sum_{i=1}^{t} |x_i|$

Sheet 5: Colorful Left-Right Dominating Set

Given a bipartite graph G with sides L and R and a coloring of L into ℓ colors. Is there a colorful set $X \subseteq L$ of size exactly ℓ that dominates all vertices in R?

Show: no poly kernelization unless NP $\subseteq$ coNP/ poly



parameter: $|R| + \ell$

cross-composition

"similar"?

- same $|R|$ and ℓ

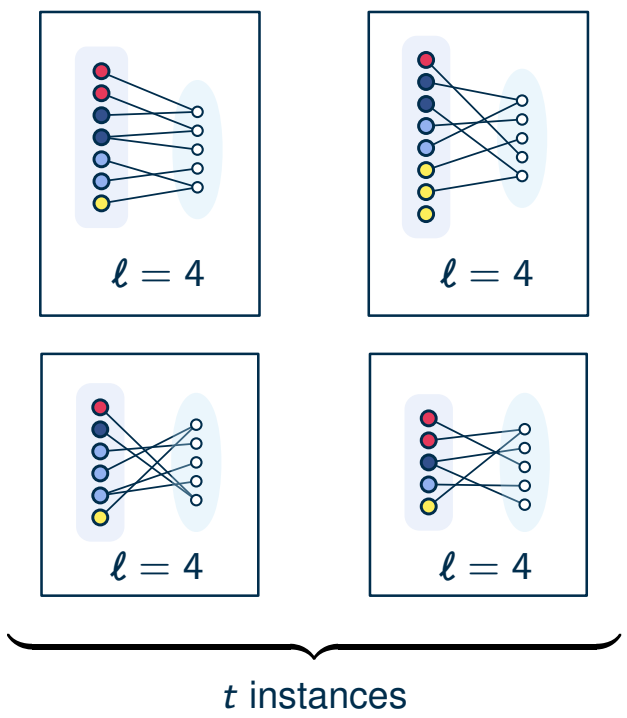
What do we need?

- instance that has a solution if and only if at least one input instance has a solution
- parameter polynomial in $\max_{i \in [t]} |x_i| + \log t$
- construction runs polynomial in $\sum_{i=1}^{t} |x_i|$

Sheet 5: Colorful Left-Right Dominating Set

Given a bipartite graph G with sides L and R and a coloring of L into ℓ colors. Is there a colorful set $X \subseteq L$ of size exactly ℓ that dominates all vertices in R?

Show: no poly kernelization unless NP $\subseteq$ coNP/ poly



parameter: $|R| + \ell$

cross-composition

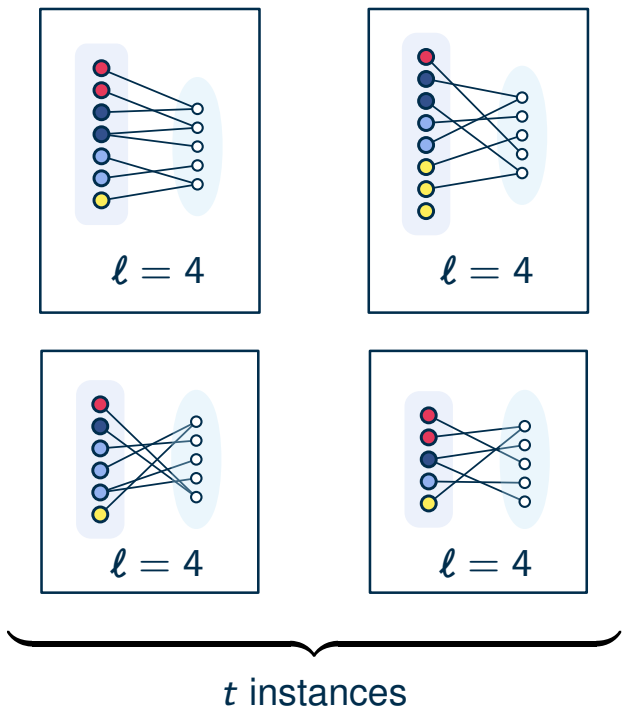
"similar"?

- same $|R|$ and ℓ

Sheet 5: Colorful Left-Right Dominating Set

Given a bipartite graph G with sides L and R and a coloring of L into ℓ colors. Is there a colorful set $X \subseteq L$ of size exactly ℓ that dominates all vertices in R?

Show: no poly kernelization unless NP $\subseteq$ coNP/ poly



parameter: $|R| + \ell$

cross-composition

"similar"?

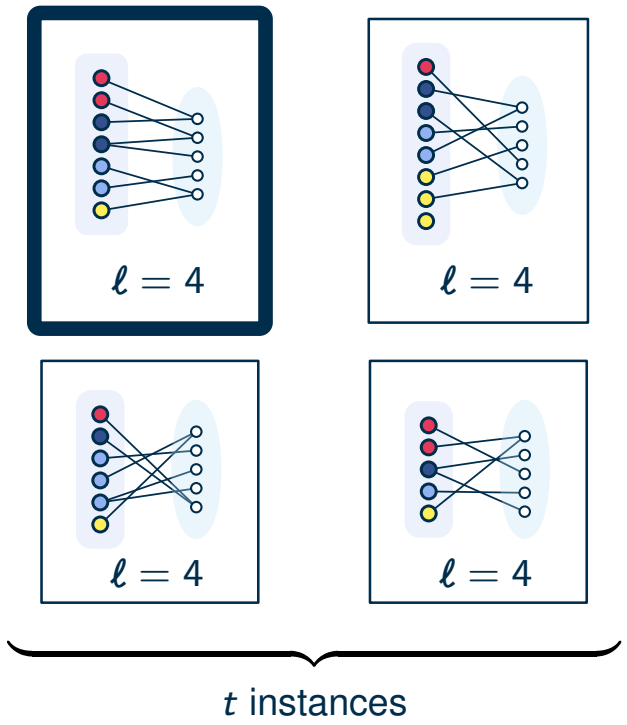
- same $|R|$ and ℓ

R

Sheet 5: Colorful Left-Right Dominating Set

Given a bipartite graph G with sides L and R and a coloring of L into ℓ colors. Is there a colorful set $X \subseteq L$ of size exactly ℓ that dominates all vertices in R?

Show: no poly kernelization unless NP $\subseteq$ coNP/ poly

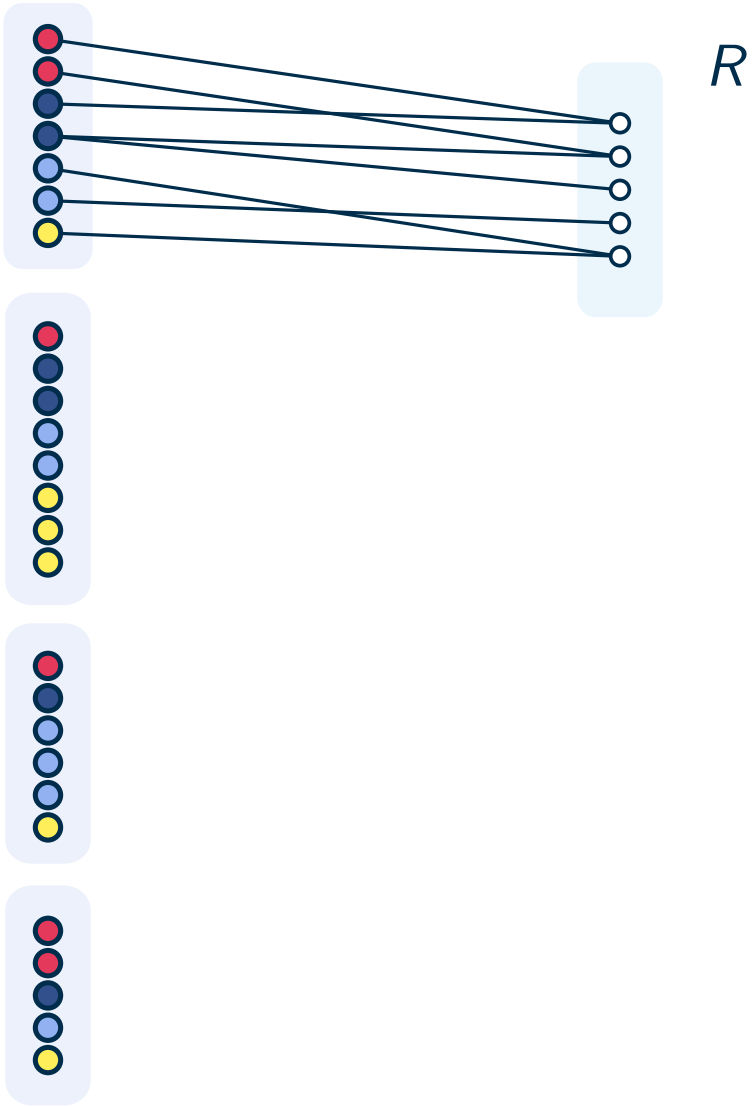


parameter: $|R| + \ell$

cross-composition

"similar"?

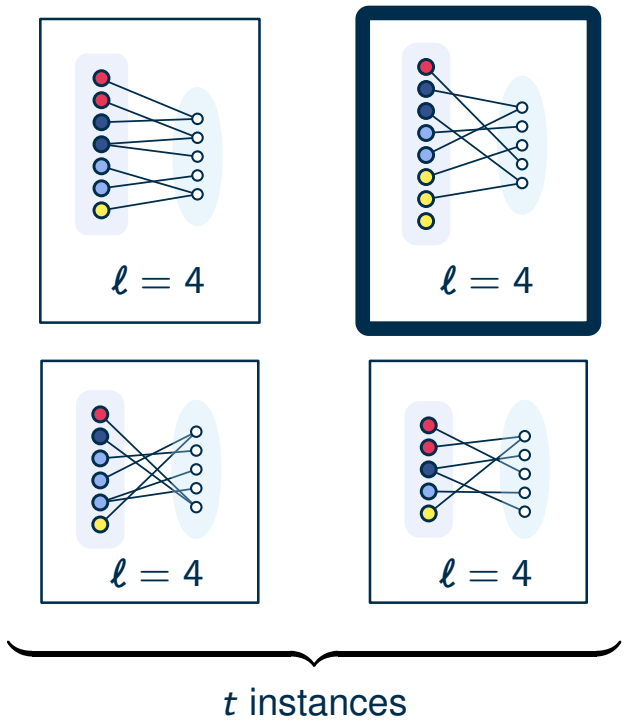
- same $|R|$ and ℓ



Sheet 5: Colorful Left-Right Dominating Set

Given a bipartite graph G with sides L and R and a coloring of L into ℓ colors. Is there a colorful set $X \subseteq L$ of size exactly ℓ that dominates all vertices in R?

Show: no poly kernelization unless NP $\subseteq$ coNP/ poly

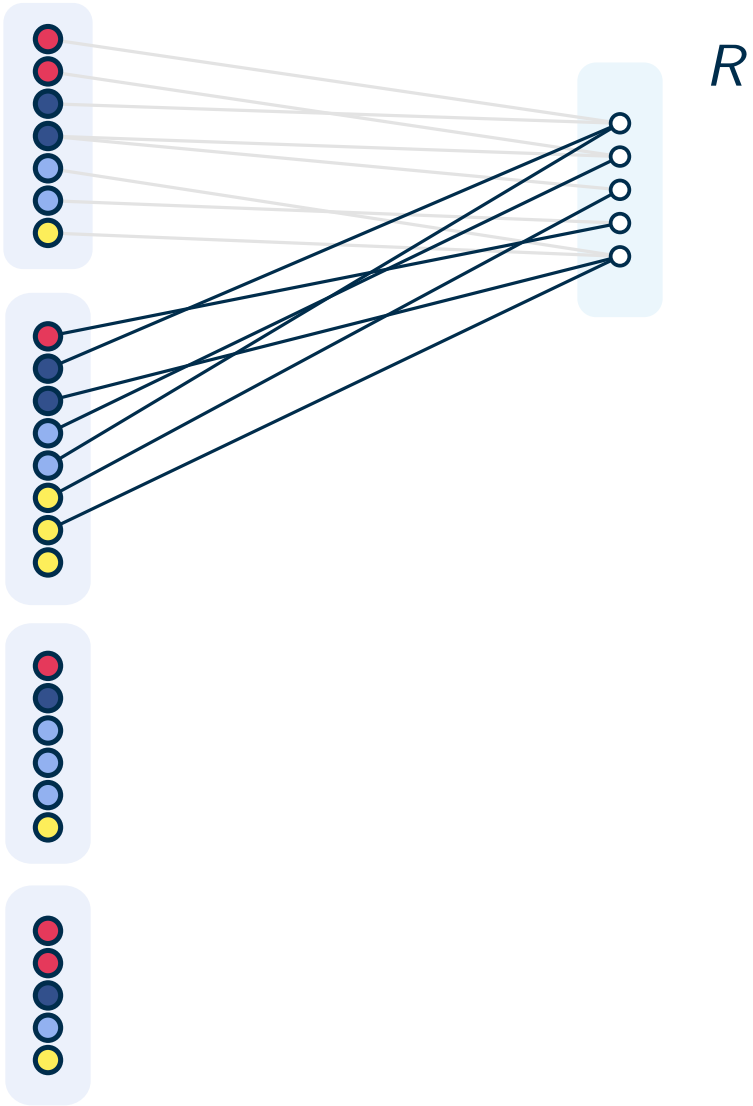


parameter: $|R| + \ell$

cross-composition

“similar”?

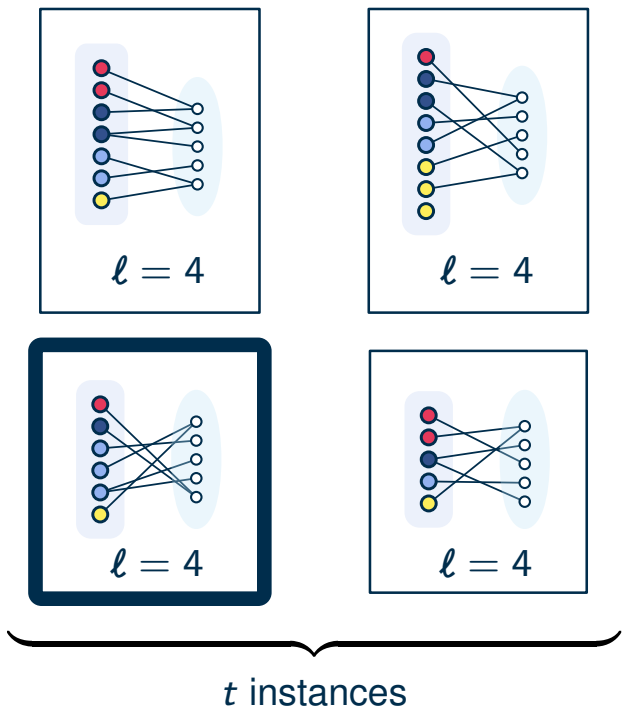
- same $|R|$ and ℓ



Sheet 5: Colorful Left-Right Dominating Set

Given a bipartite graph G with sides L and R and a coloring of L into ℓ colors. Is there a colorful set $X \subseteq L$ of size exactly ℓ that dominates all vertices in R?

Show: no poly kernelization unless NP $\subseteq$ coNP/ poly

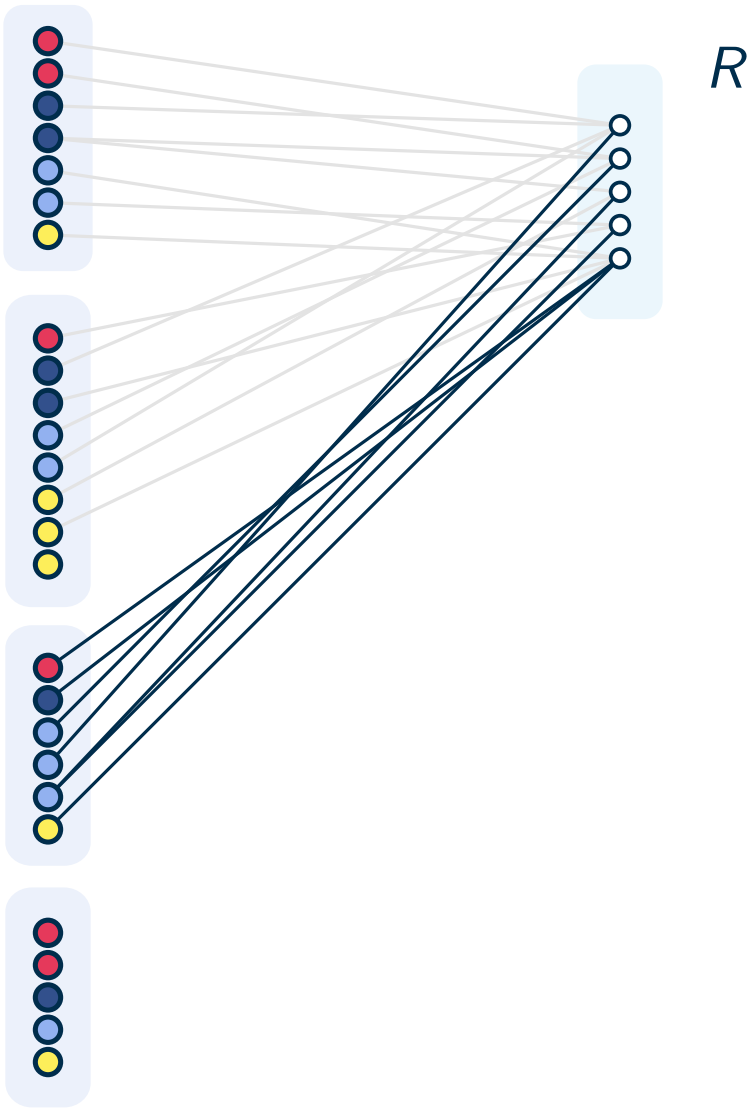


parameter: $|R| + \ell$

cross-composition

“similar”?

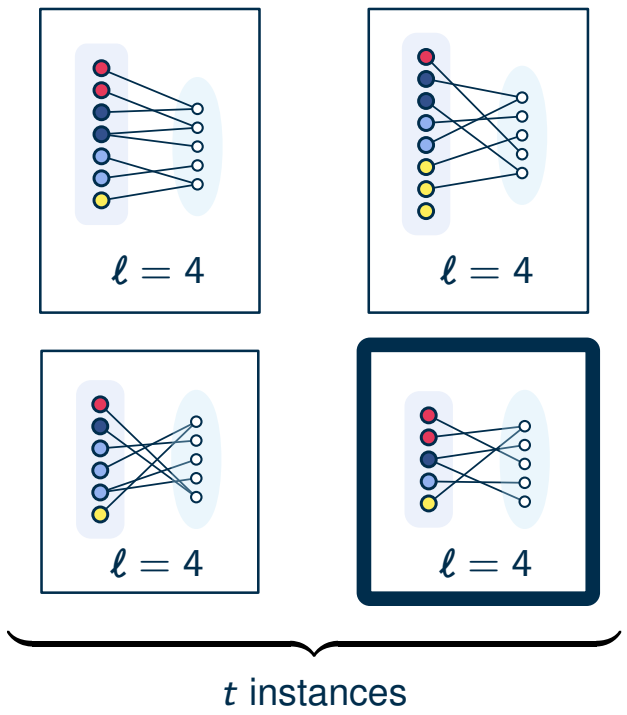
- same $|R|$ and ℓ



Sheet 5: Colorful Left-Right Dominating Set

Given a bipartite graph G with sides L and R and a coloring of L into ℓ colors. Is there a colorful set $X \subseteq L$ of size exactly ℓ that dominates all vertices in R?

Show: no poly kernelization unless NP $\subseteq$ coNP/ poly

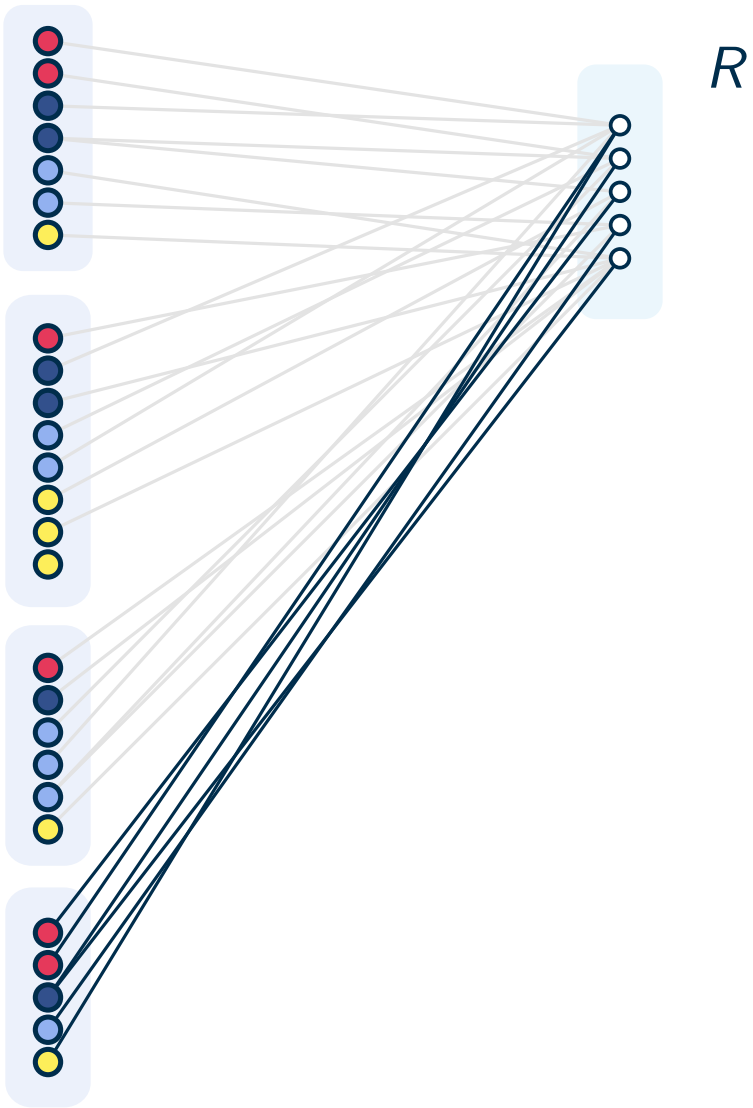


parameter: $|R| + \ell$

cross-composition

“similar”?

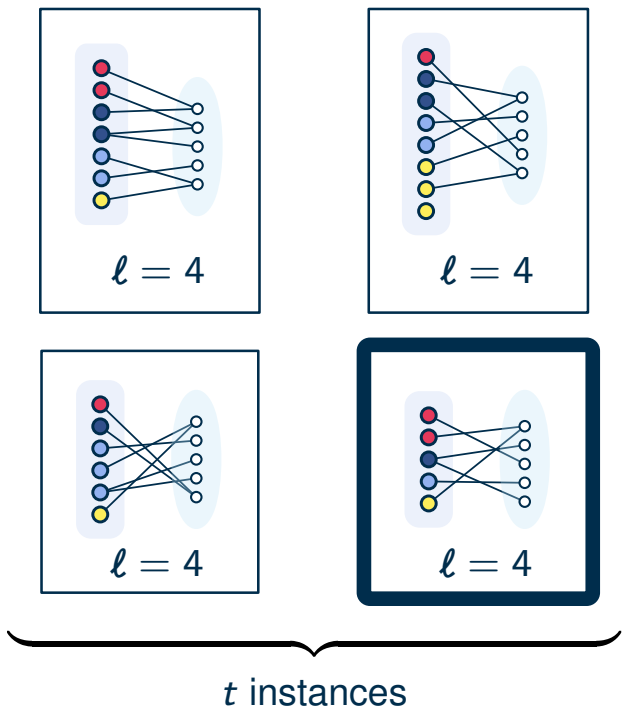
- same $|R|$ and ℓ



Sheet 5: Colorful Left-Right Dominating Set

Given a bipartite graph G with sides L and R and a coloring of L into ℓ colors. Is there a colorful set $X \subseteq L$ of size exactly ℓ that dominates all vertices in R?

Show: no poly kernelization unless NP $\subseteq$ coNP/ poly

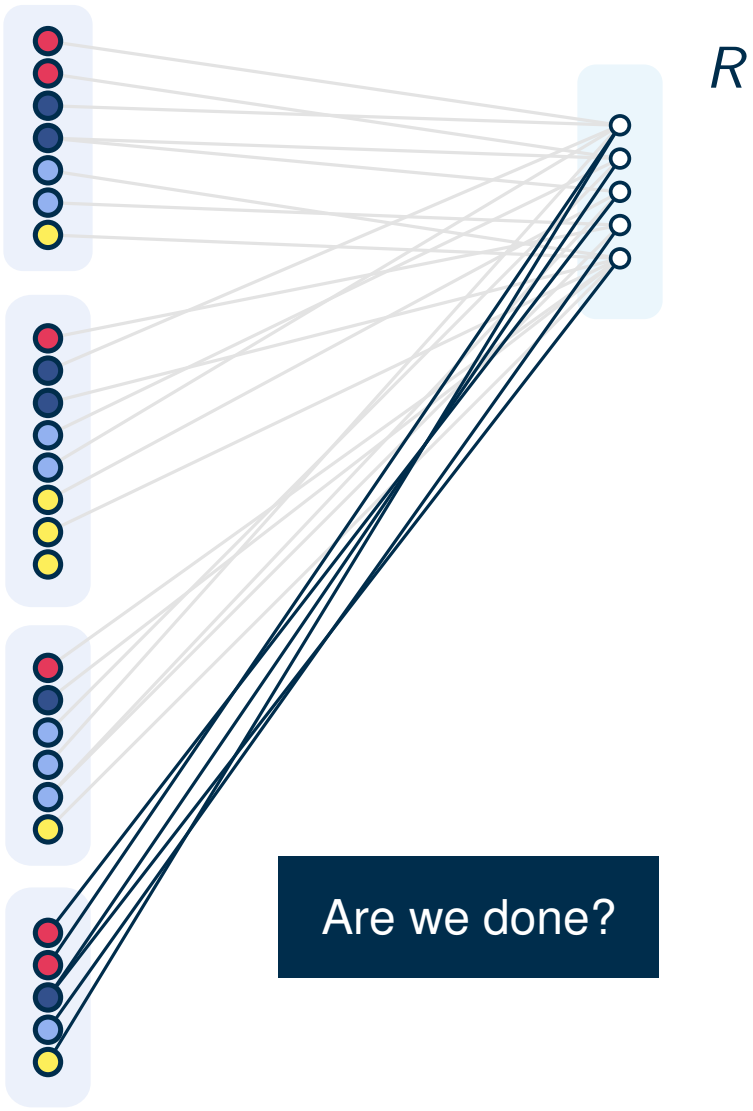


parameter: $|R| + \ell$

cross-composition

“similar”?

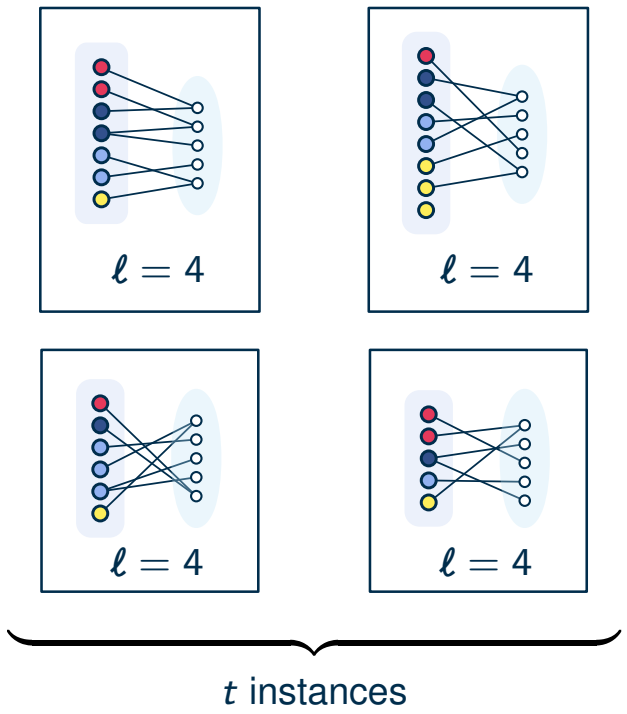
- same $|R|$ and ℓ



Sheet 5: Colorful Left-Right Dominating Set

Given a bipartite graph G with sides L and R and a coloring of L into ℓ colors. Is there a colorful set $X \subseteq L$ of size exactly ℓ that dominates all vertices in R?

Show: no poly kernelization unless NP $\subseteq$ coNP/ poly

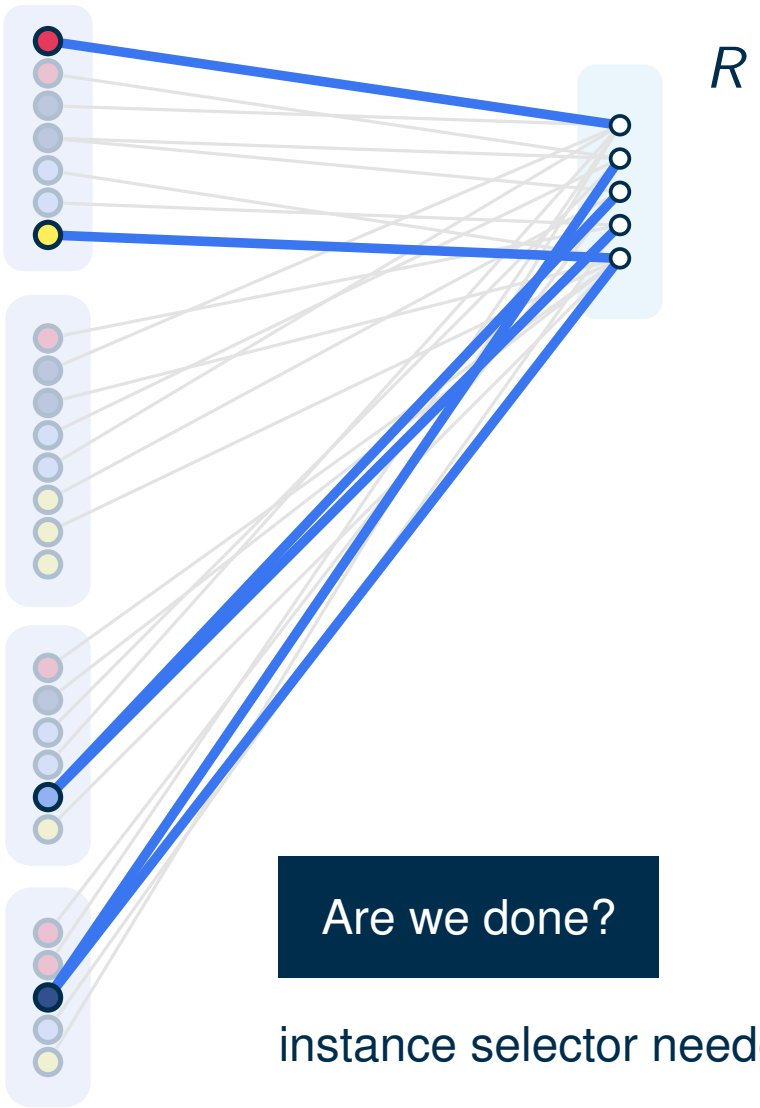


parameter: $|R| + \ell$

cross-composition

"similar"?

- same $|R|$ and ℓ



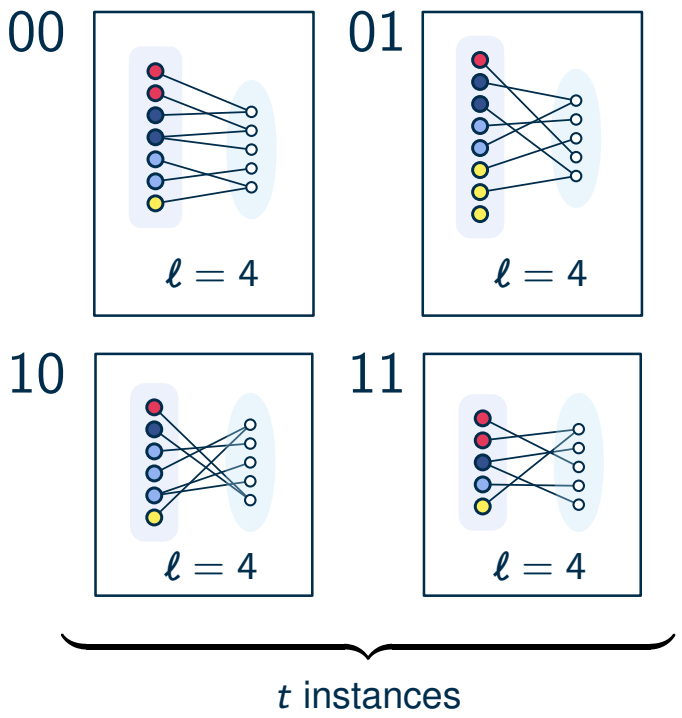
Are we done?

instance selector needed

Sheet 5: Colorful Left-Right Dominating Set

Given a bipartite graph G with sides L and R and a coloring of L into ℓ colors. Is there a colorful set $X \subseteq L$ of size exactly ℓ that dominates all vertices in R?

Show: no poly kernelization unless NP $\subseteq$ coNP/ poly

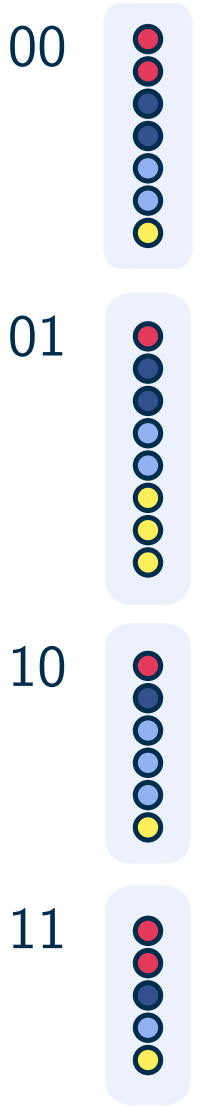


parameter: $|R| + \ell$

cross-composition

"similar"?

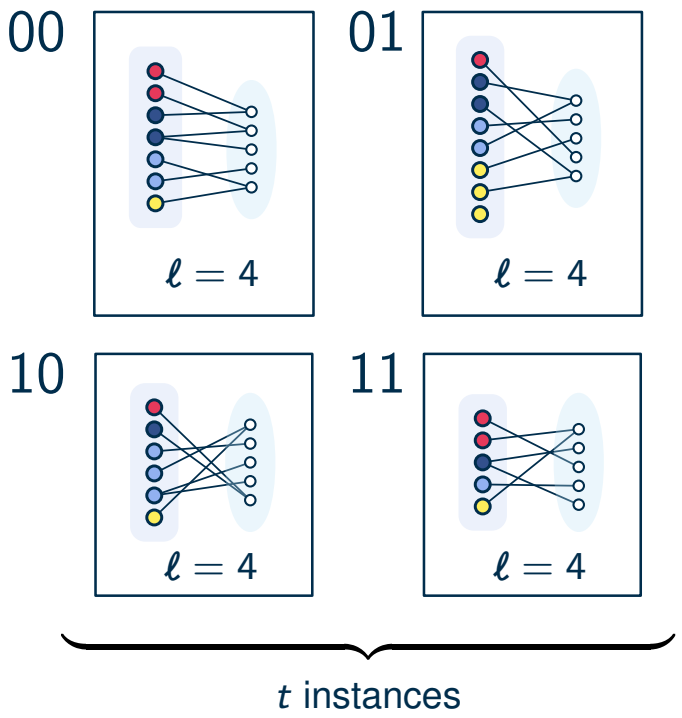
- same $|R|$ and ℓ



Sheet 5: Colorful Left-Right Dominating Set

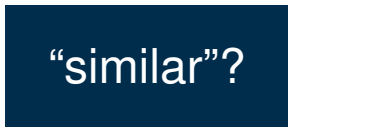
Given a bipartite graph G with sides L and R and a coloring of L into ℓ colors. Is there a colorful set $X \subseteq L$ of size exactly ℓ that dominates all vertices in R?

Show: no poly kernelization unless NP $\subseteq$ coNP/ poly



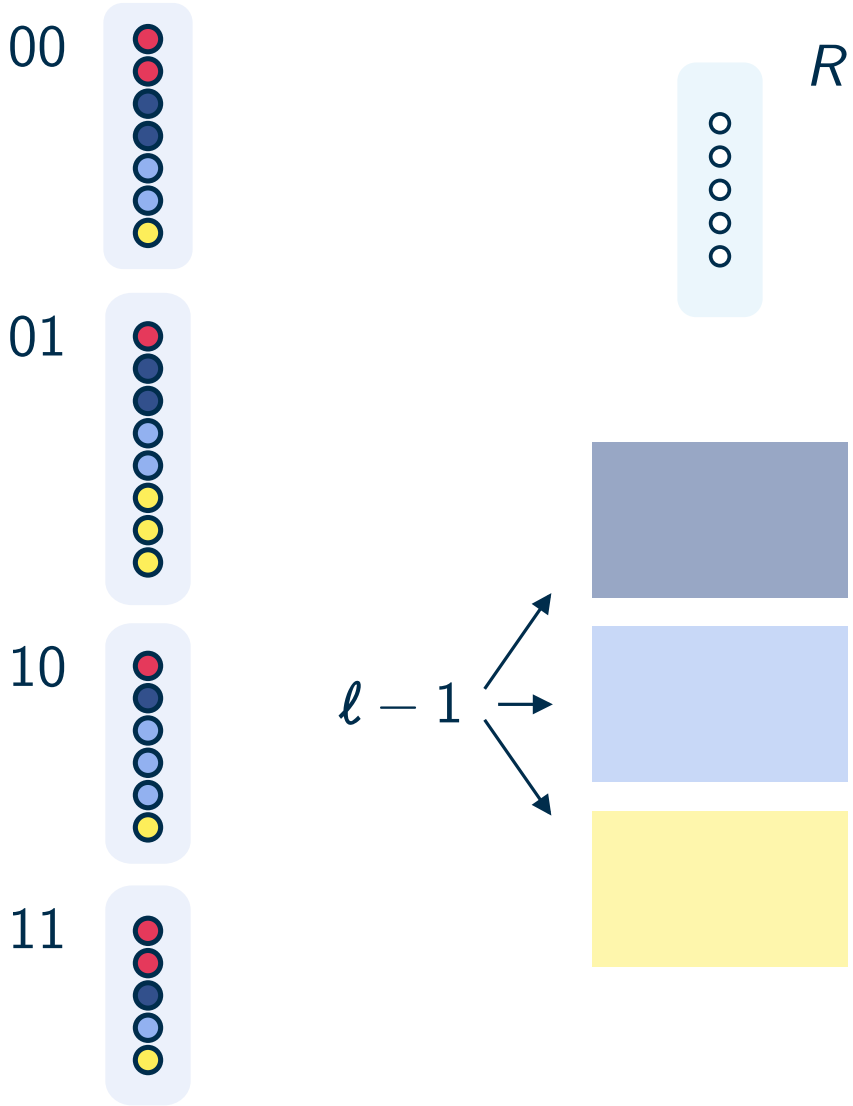
parameter: $|R| + \ell$

cross-composition



"similar"?

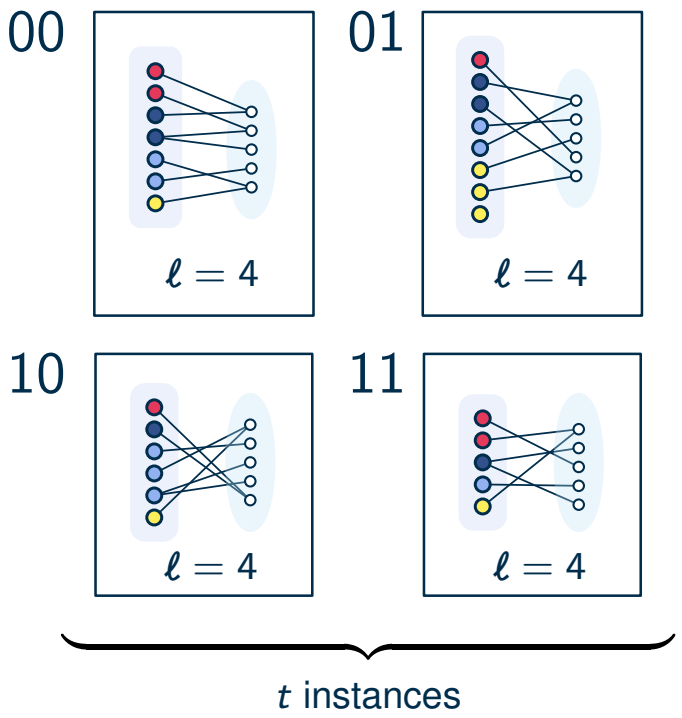
- same $|R|$ and ℓ



Sheet 5: Colorful Left-Right Dominating Set

Given a bipartite graph G with sides L and R and a coloring of L into ℓ colors. Is there a colorful set $X \subseteq L$ of size exactly ℓ that dominates all vertices in R?

Show: no poly kernelization unless NP $\subseteq$ coNP/ poly

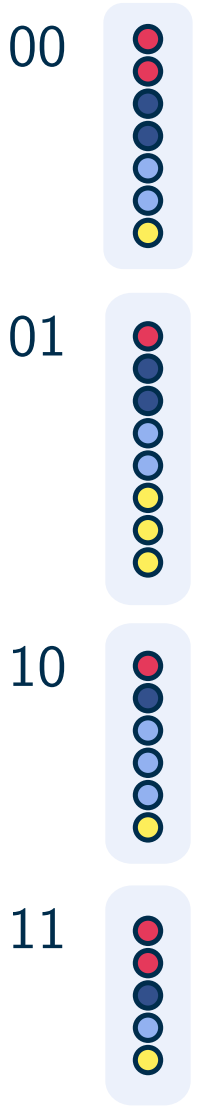


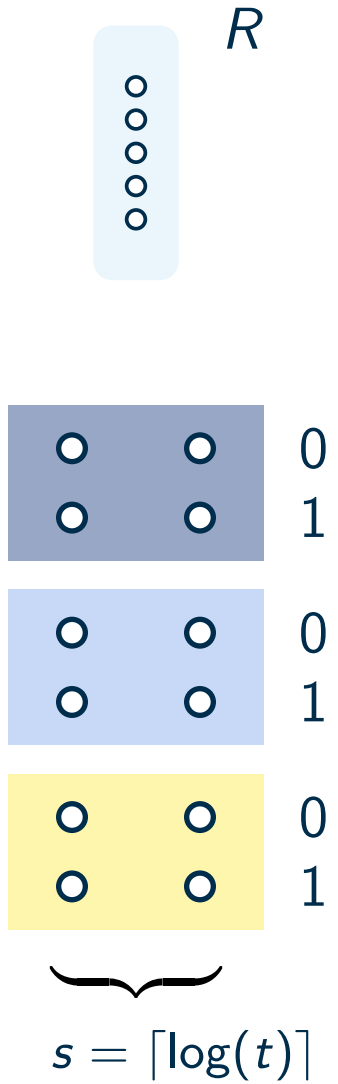
parameter: $|R| + \ell$

cross-composition

"similar"?

- same $|R|$ and ℓ



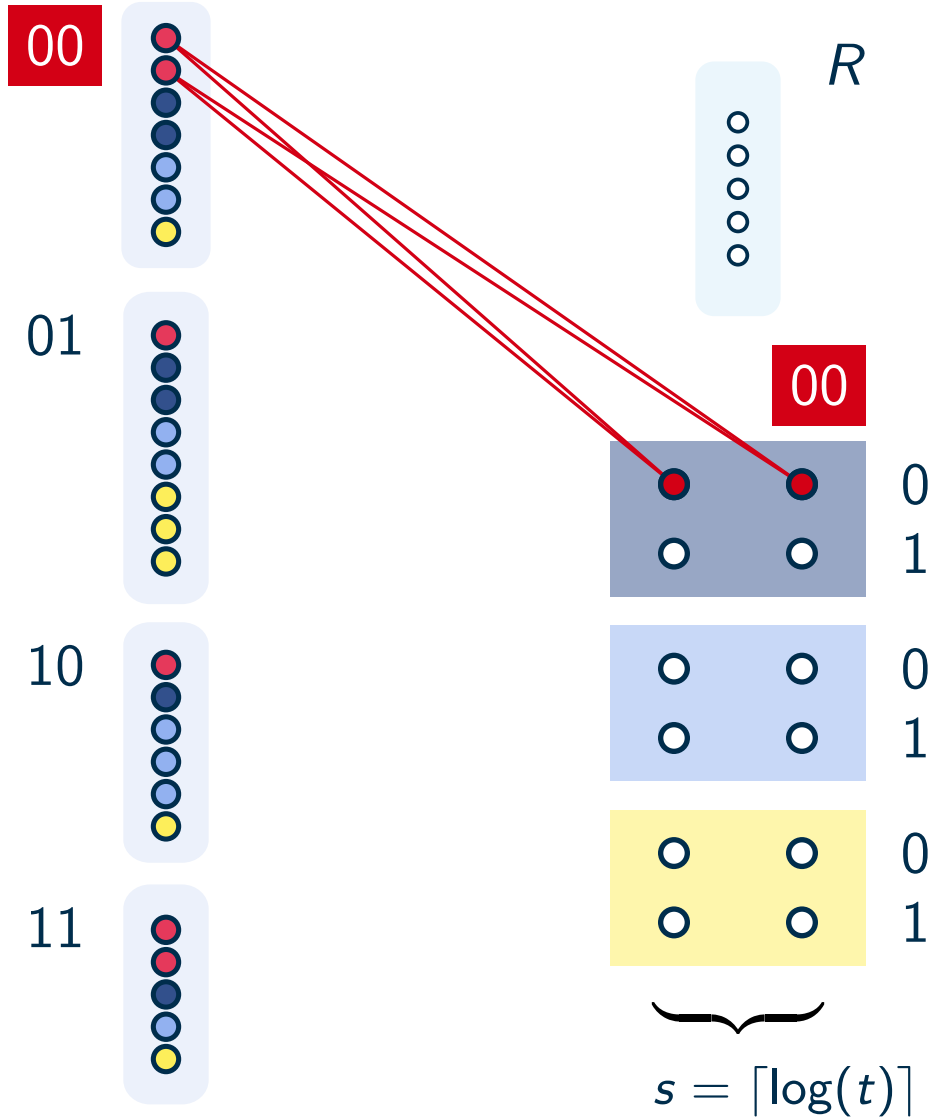


Sheet 5: Colorful Left-Right Dominating Set

Given a bipartite graph G with sides L and R and a coloring of L into ℓ colors. Is there a colorful set $X \subseteq L$ of size exactly ℓ that dominates all vertices in R?

Show: no poly kernelization unless NP $\subseteq$ coNP/ poly

- connect each red vertex to binary representations of its instance index

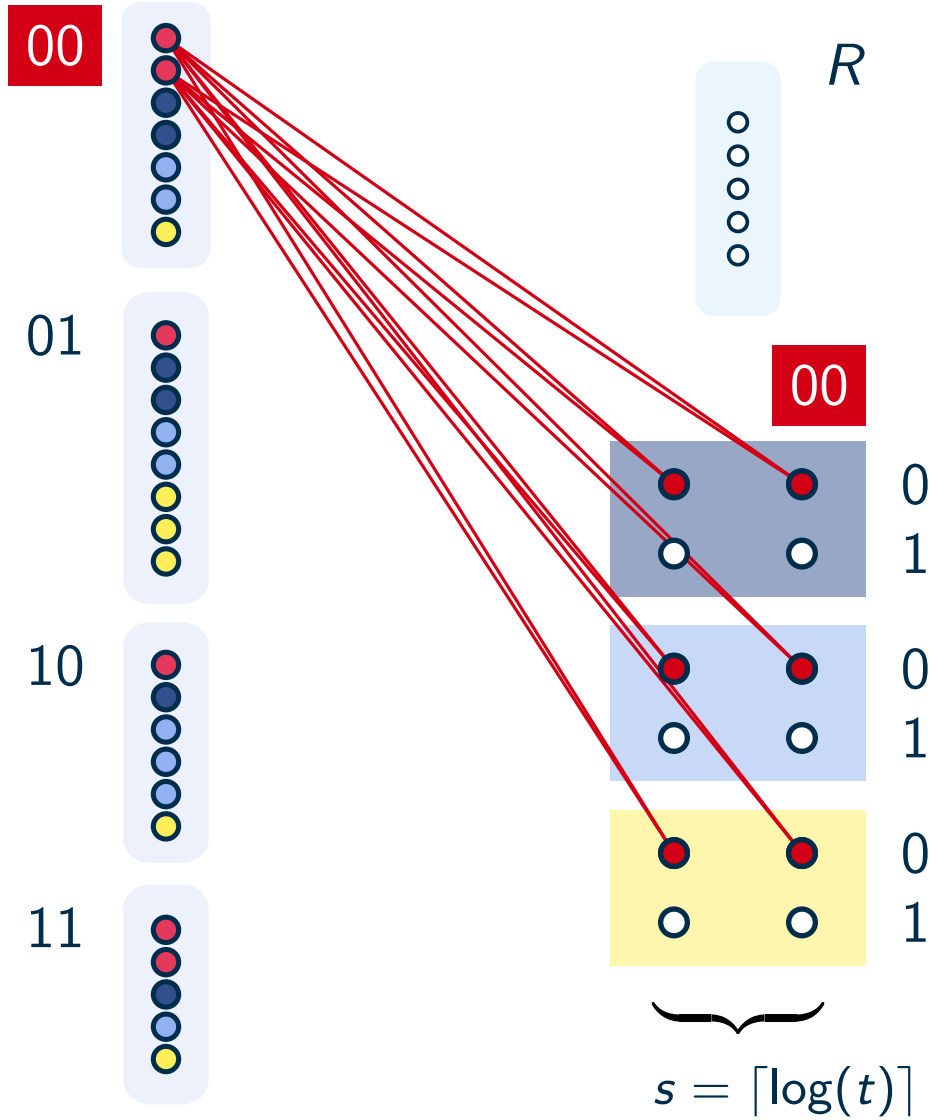


Sheet 5: Colorful Left-Right Dominating Set

Given a bipartite graph G with sides L and R and a coloring of L into ℓ colors. Is there a colorful set $X \subseteq L$ of size exactly ℓ that dominates all vertices in R?

Show: no poly kernelization unless NP $\subseteq$ coNP/ poly

- connect each red vertex to binary representations of its instance index
  - ↳ for **each** color part of selector gadget

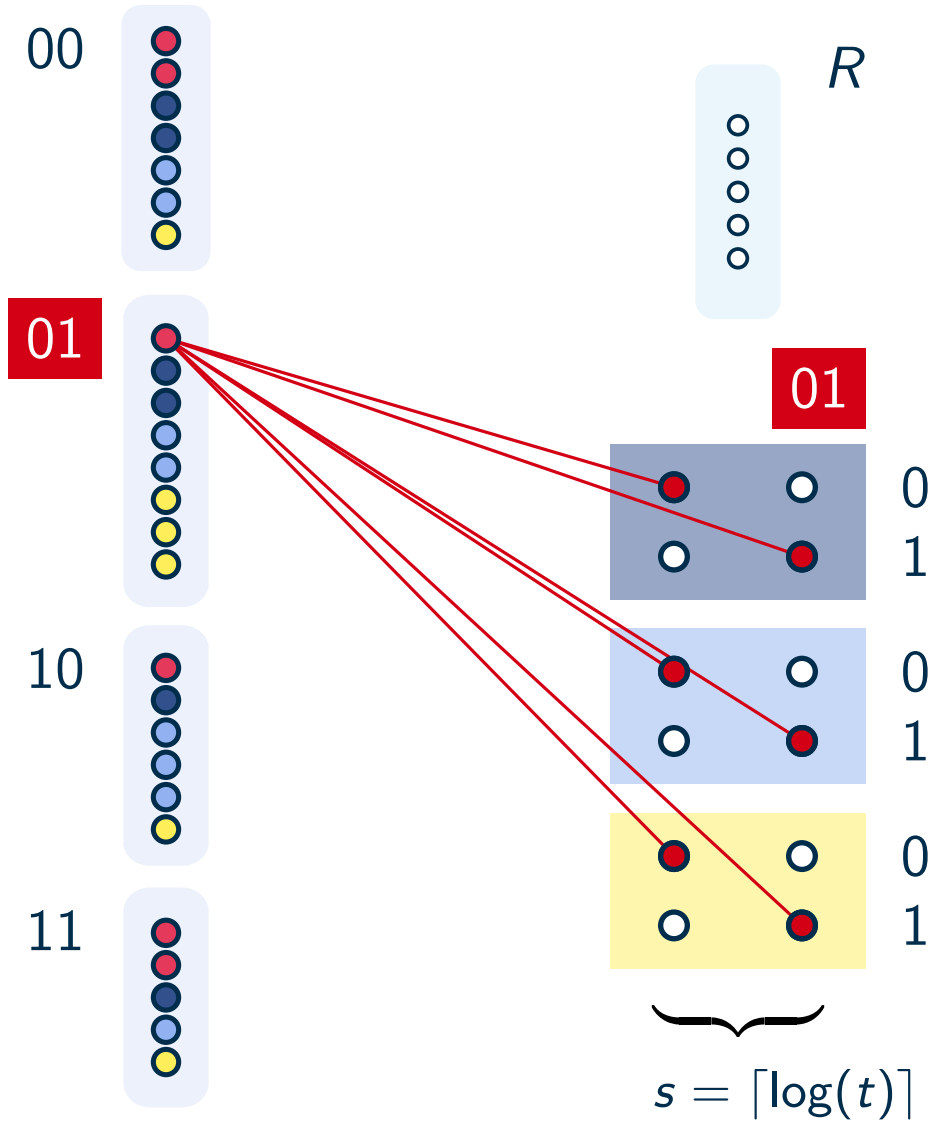


Sheet 5: Colorful Left-Right Dominating Set

Given a bipartite graph G with sides L and R and a coloring of L into ℓ colors. Is there a colorful set $X \subseteq L$ of size exactly ℓ that dominates all vertices in R?

Show: no poly kernelization unless NP $\subseteq$ coNP/ poly

- connect each red vertex to binary representations of its instance index
  - ↳ for **each** color part of selector gadget

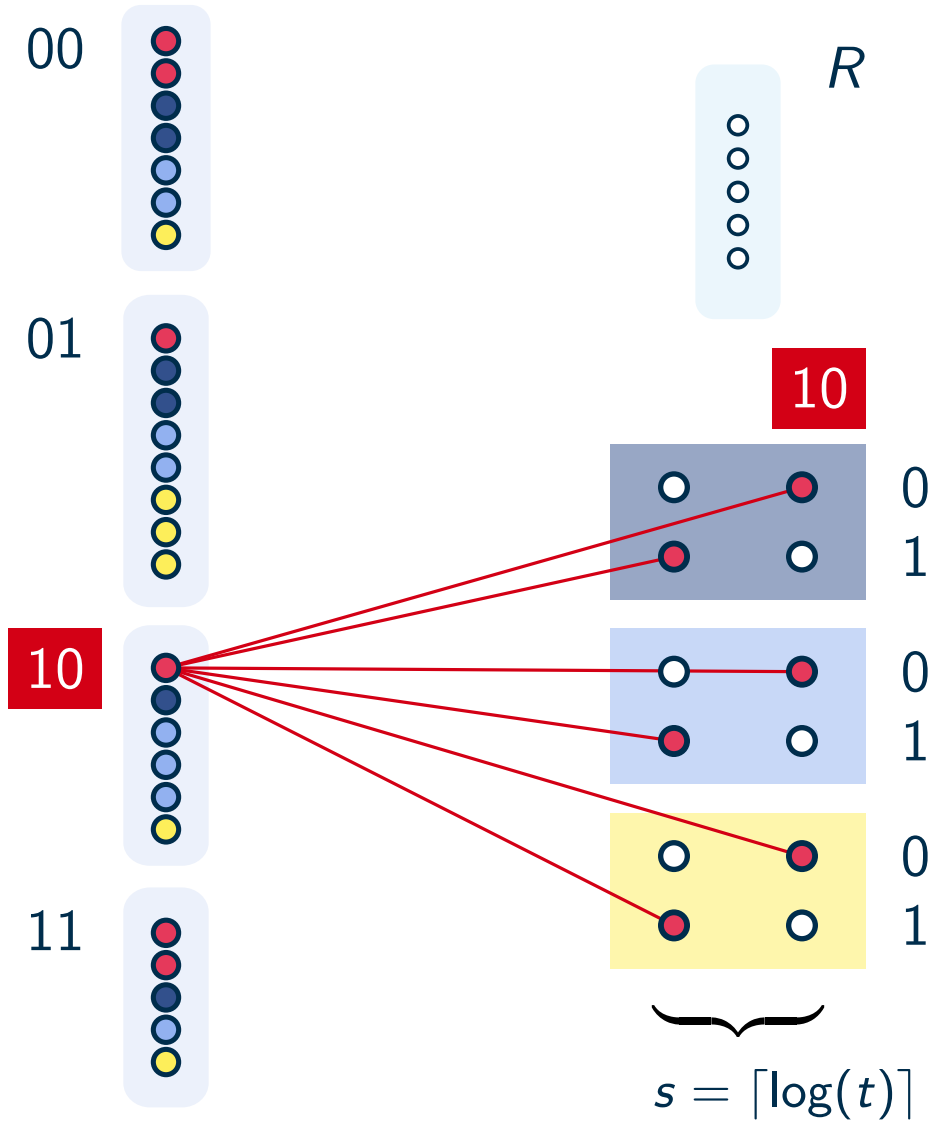


Sheet 5: Colorful Left-Right Dominating Set

Given a bipartite graph G with sides L and R and a coloring of L into ℓ colors. Is there a colorful set $X \subseteq L$ of size exactly ℓ that dominates all vertices in R?

Show: no poly kernelization unless NP $\subseteq$ coNP/ poly

- connect each red vertex to binary representations of its instance index
  - ↳ for **each** color part of selector gadget

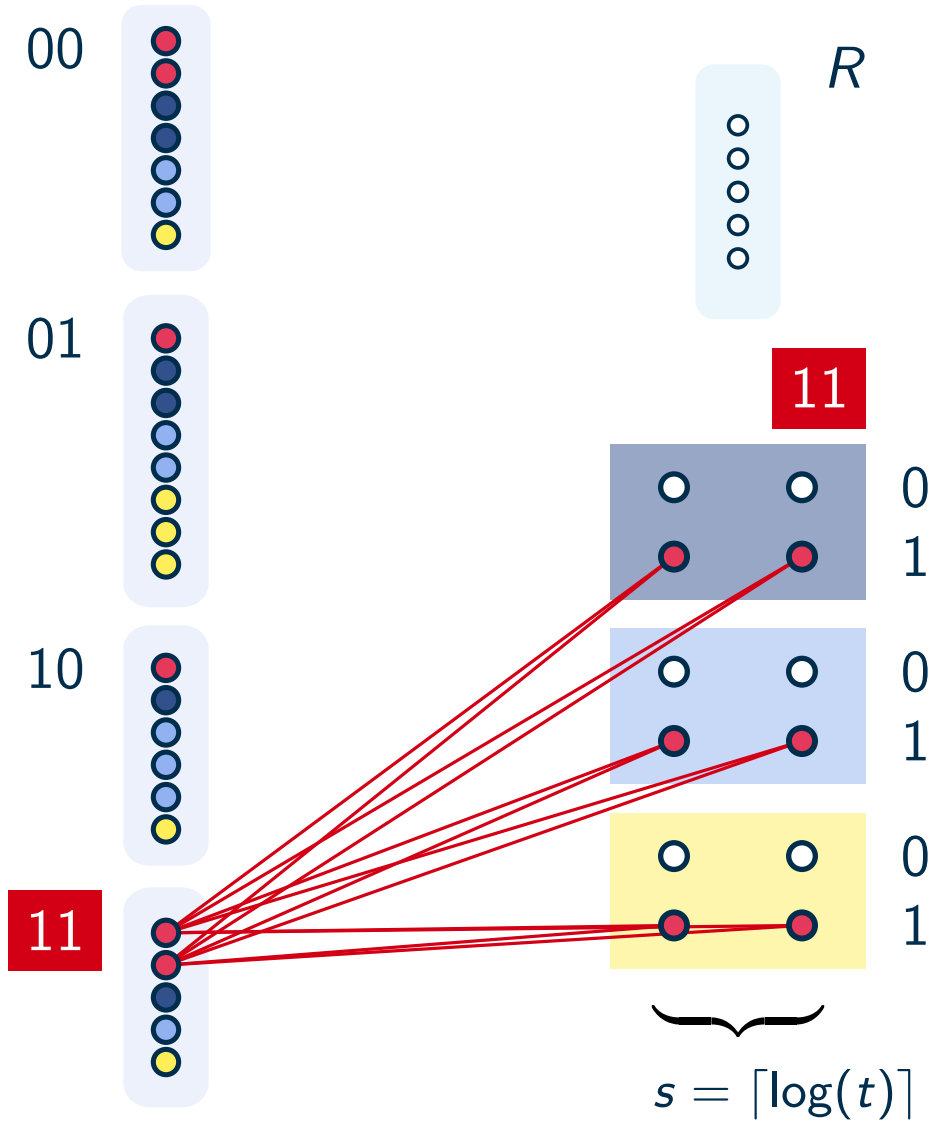


Sheet 5: Colorful Left-Right Dominating Set

Given a bipartite graph G with sides L and R and a coloring of L into ℓ colors. Is there a colorful set $X \subseteq L$ of size exactly ℓ that dominates all vertices in R?

Show: no poly kernelization unless NP $\subseteq$ coNP/ poly

- connect each red vertex to binary representations of its instance index
  - ↳ for **each** color part of selector gadget

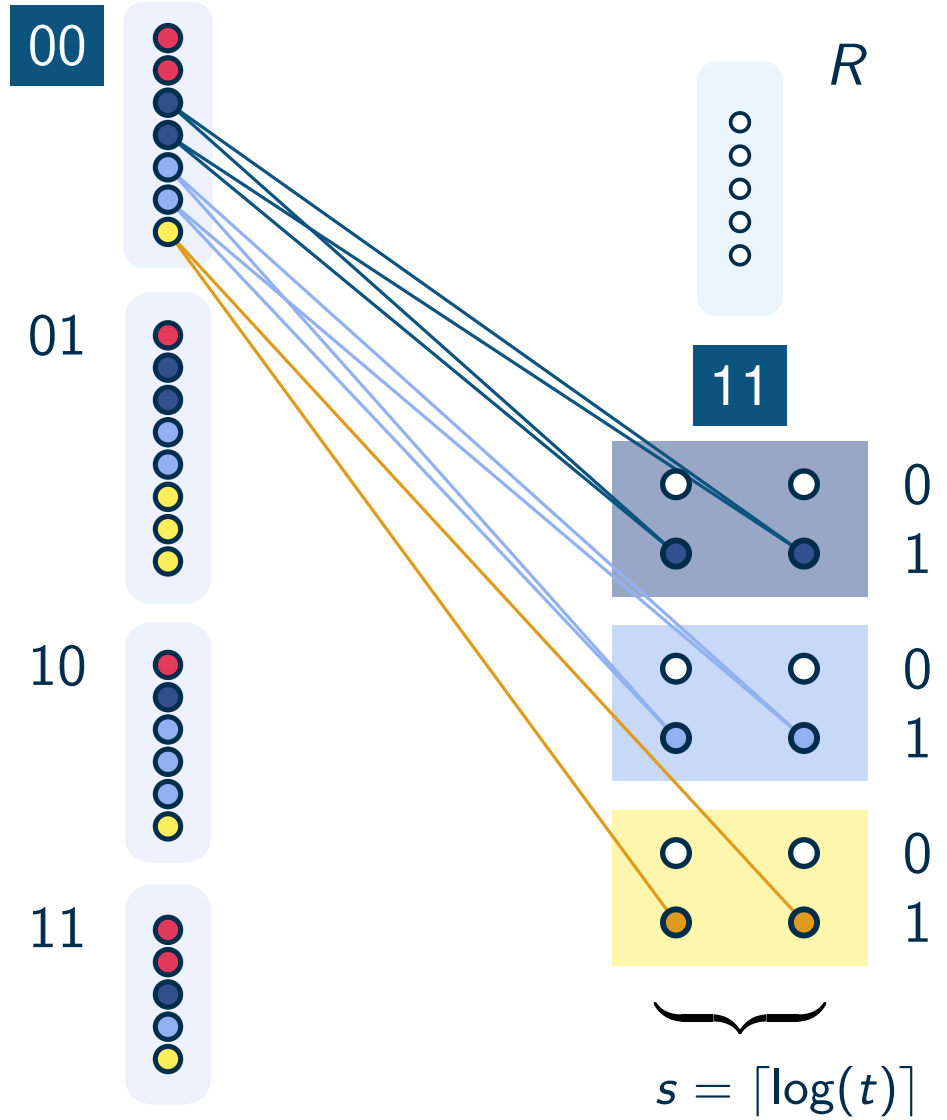


Sheet 5: Colorful Left-Right Dominating Set

Given a bipartite graph G with sides L and R and a coloring of L into ℓ colors. Is there a colorful set $X \subseteq L$ of size exactly ℓ that dominates all vertices in R?

Show: no poly kernelization unless NP $\subseteq$ coNP/ poly

- connect each red vertex to binary representations of its instance index
  - ↳ for **each** color part of selector gadget
- other colors: connect left vertices to *complement* of binary representations of their instance indices
  - ↳ only to part of selector gadget with same color

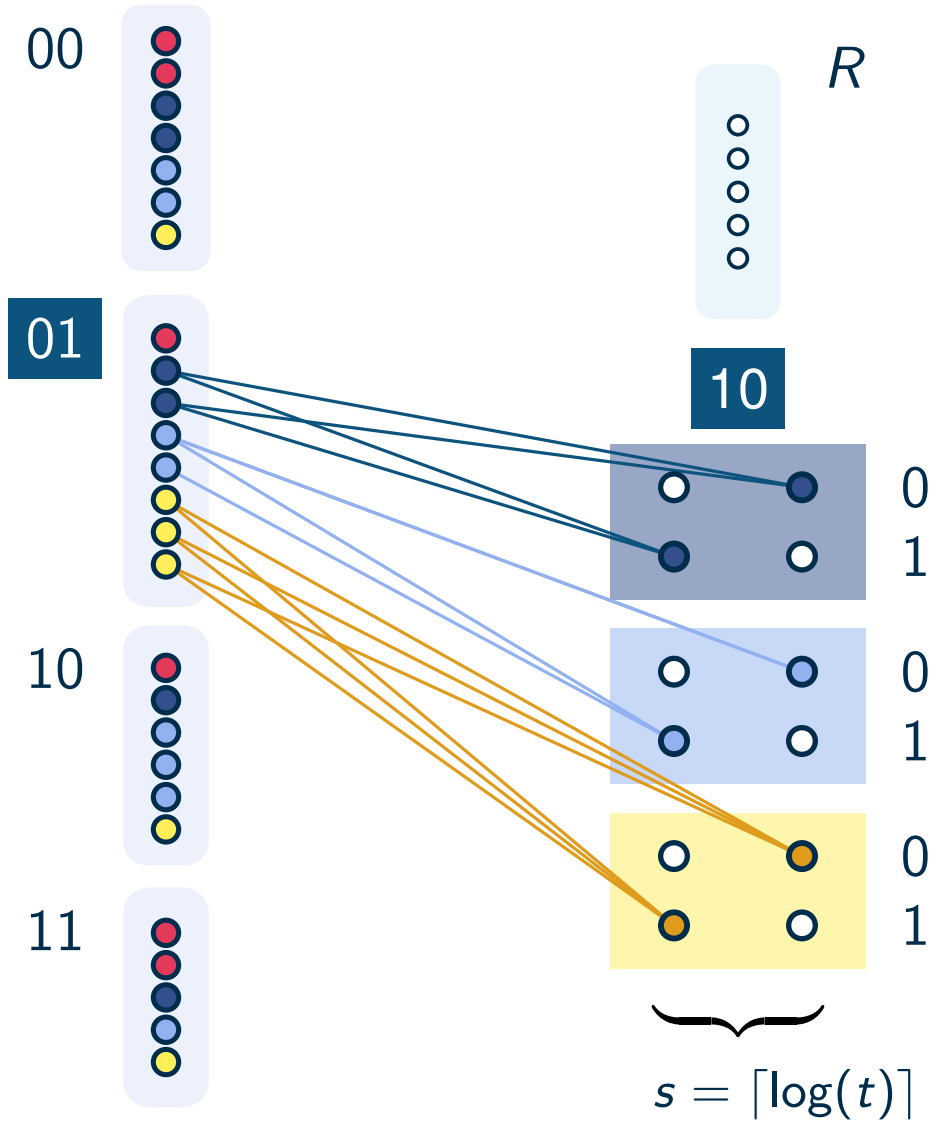


Sheet 5: Colorful Left-Right Dominating Set

Given a bipartite graph G with sides L and R and a coloring of L into ℓ colors. Is there a colorful set $X \subseteq L$ of size exactly ℓ that dominates all vertices in R?

Show: no poly kernelization unless NP $\subseteq$ coNP/ poly

- connect each red vertex to binary representations of its instance index
  - ↳ for **each** color part of selector gadget
- other colors: connect left vertices to *complement* of binary representations of their instance indices
  - ↳ only to part of selector gadget with same color

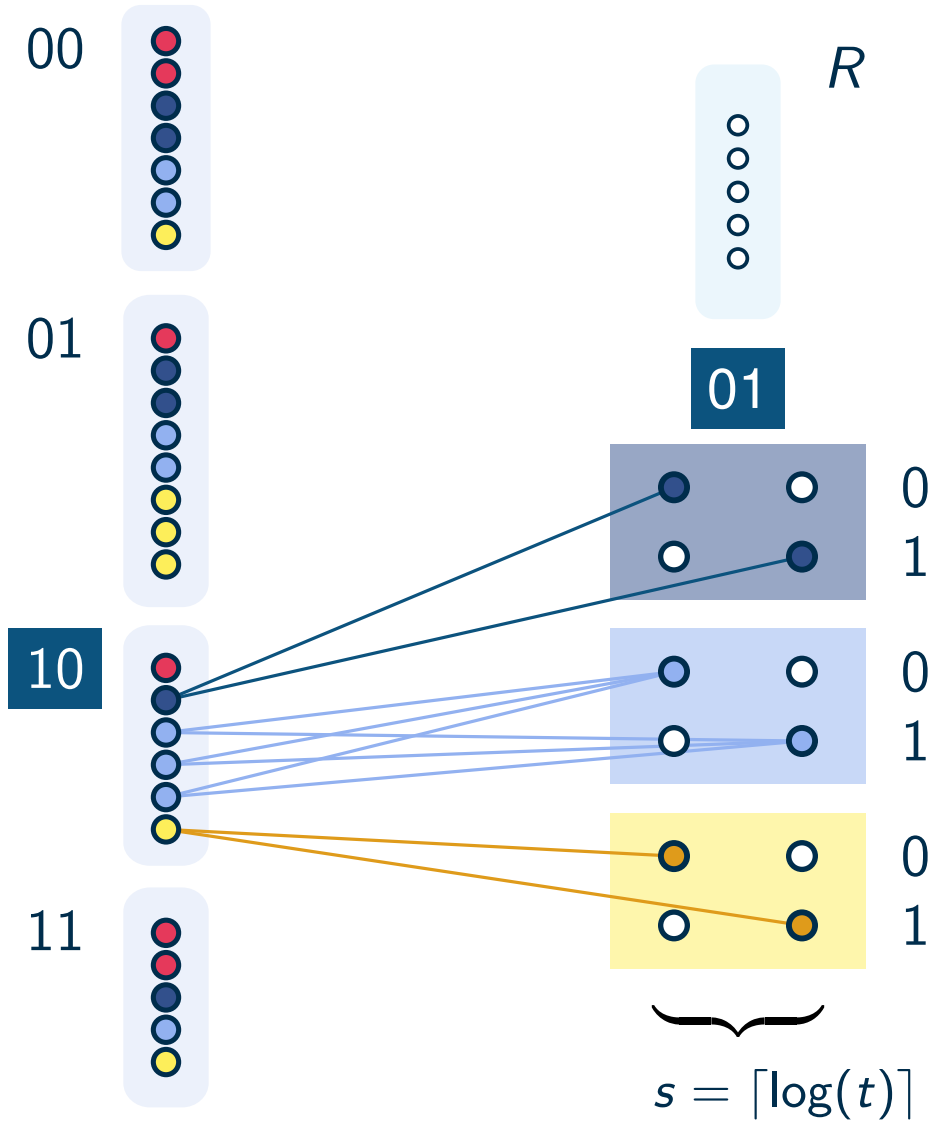


Sheet 5: Colorful Left-Right Dominating Set

Given a bipartite graph G with sides L and R and a coloring of L into ℓ colors. Is there a colorful set $X \subseteq L$ of size exactly ℓ that dominates all vertices in R?

Show: no poly kernelization unless NP $\subseteq$ coNP/ poly

- connect each red vertex to binary representations of its instance index
  - ↳ for **each** color part of selector gadget
- other colors: connect left vertices to *complement* of binary representations of their instance indices
  - ↳ only to part of selector gadget with same color

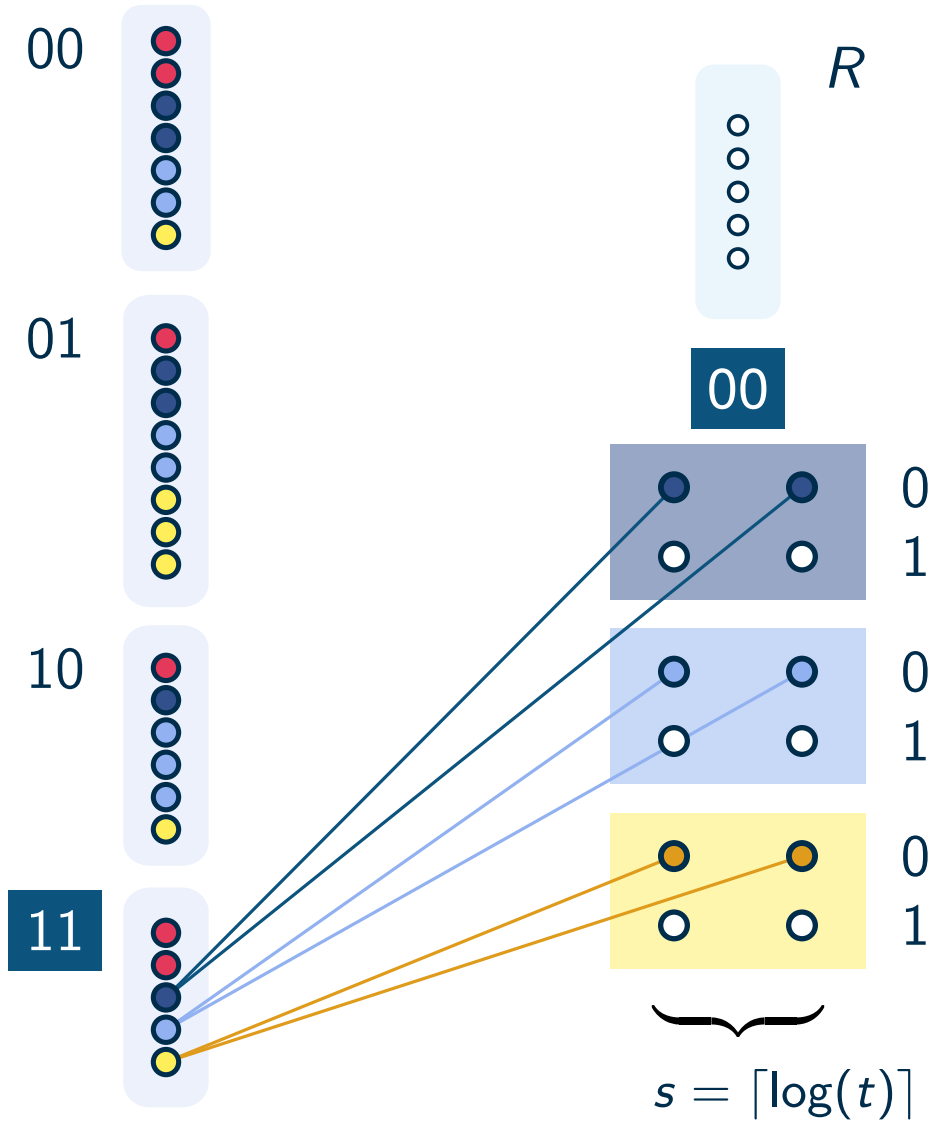


Sheet 5: Colorful Left-Right Dominating Set

Given a bipartite graph G with sides L and R and a coloring of L into ℓ colors. Is there a colorful set $X \subseteq L$ of size exactly ℓ that dominates all vertices in R?

Show: no poly kernelization unless NP $\subseteq$ coNP/ poly

- connect each red vertex to binary representations of its instance index
  - ↳ for **each** color part of selector gadget
- other colors: connect left vertices to *complement* of binary representations of their instance indices
  - ↳ only to part of selector gadget with same color

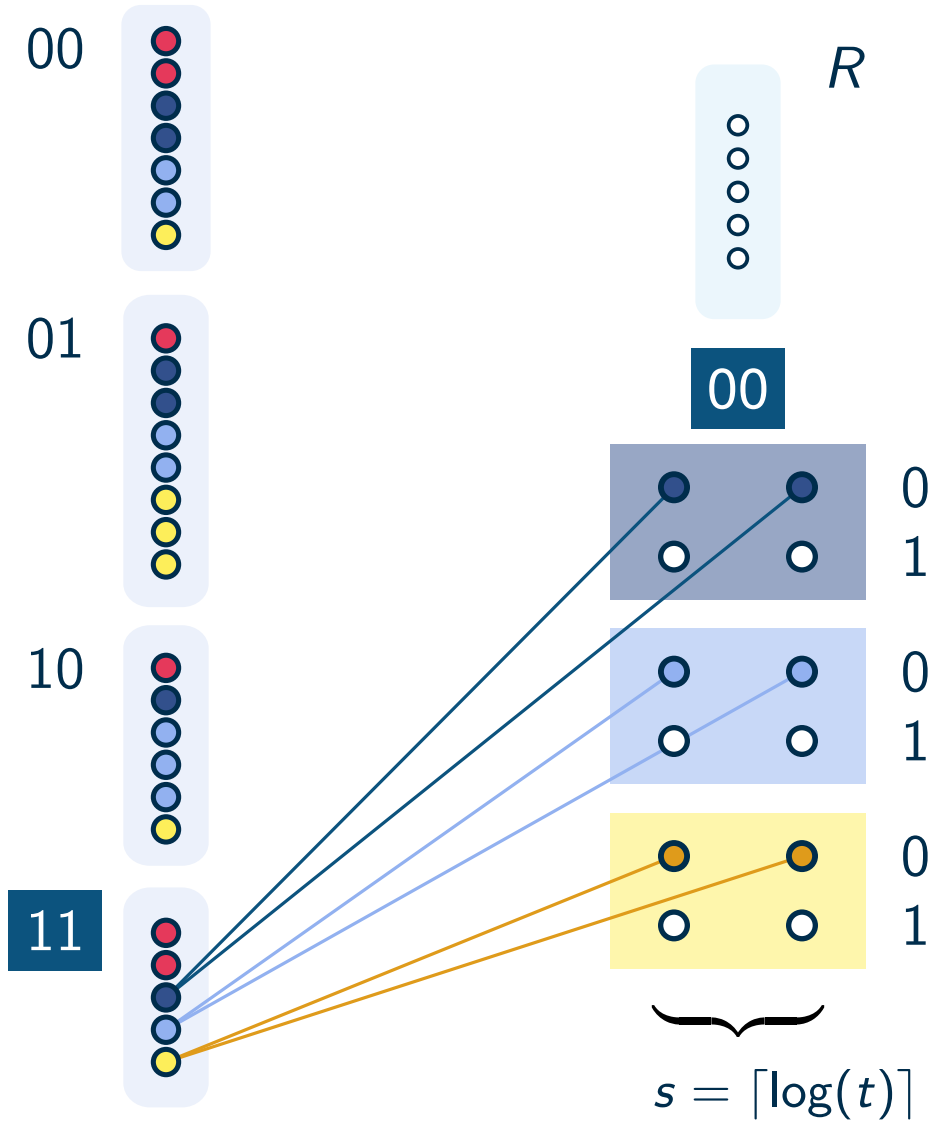


Sheet 5: Colorful Left-Right Dominating Set

Given a bipartite graph G with sides L and R and a coloring of L into ℓ colors. Is there a colorful set $X \subseteq L$ of size exactly ℓ that dominates all vertices in R?

Show: no poly kernelization unless NP $\subseteq$ coNP/ poly

- connect each red vertex to binary representations of its instance index
  - ↳ for **each** color part of selector gadget
- other colors: connect left vertices to *complement* of binary representations of their instance indices
  - ↳ only to part of selector gadget with same color
- only a colorful set from same input instance can dominate entire selector gadget

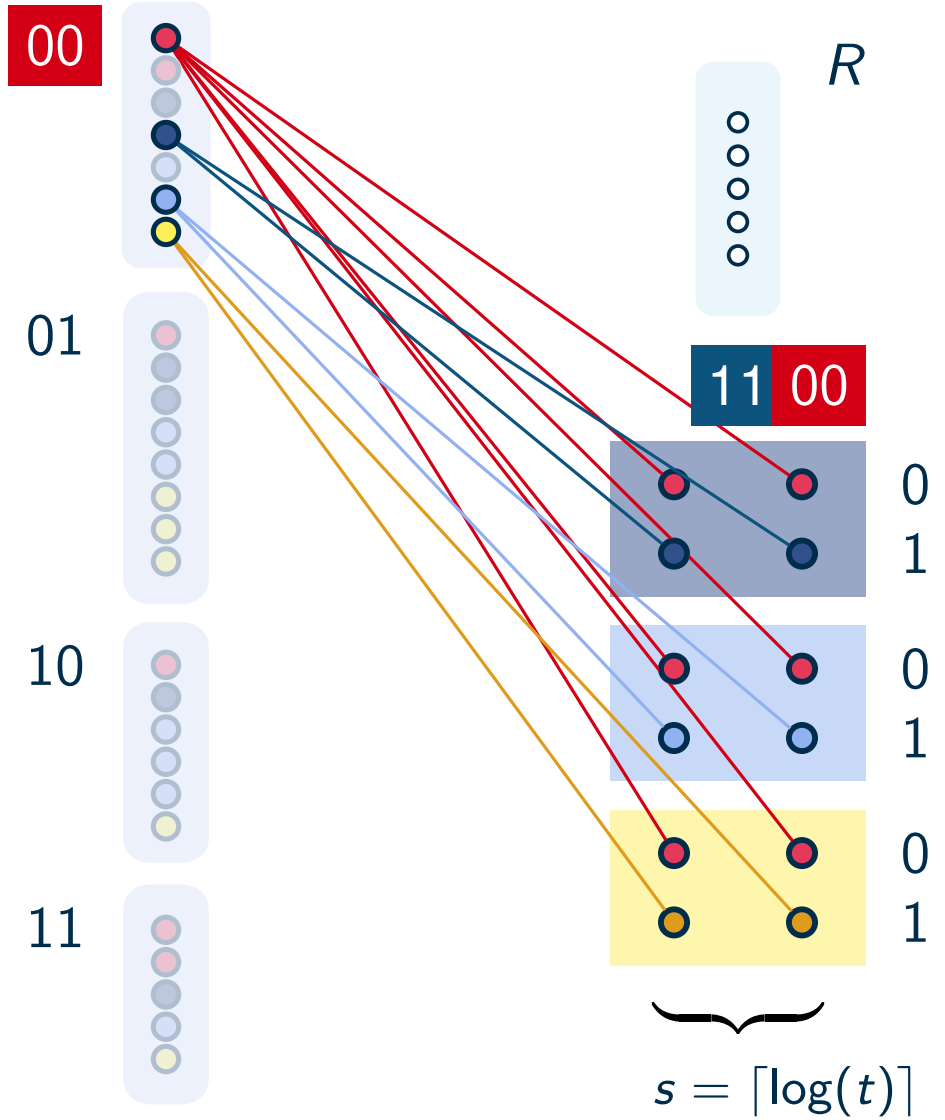


Sheet 5: Colorful Left-Right Dominating Set

Given a bipartite graph G with sides L and R and a coloring of L into ℓ colors. Is there a colorful set $X \subseteq L$ of size exactly ℓ that dominates all vertices in R?

Show: no poly kernelization unless NP $\subseteq$ coNP/ poly

- connect each red vertex to binary representations of its instance index
  - ↳ for **each** color part of selector gadget
- other colors: connect left vertices to *complement* of binary representations of their instance indices
  - ↳ only to part of selector gadget with same color
- only a colorful set from same input instance can dominate entire selector gadget



What do you think about things?

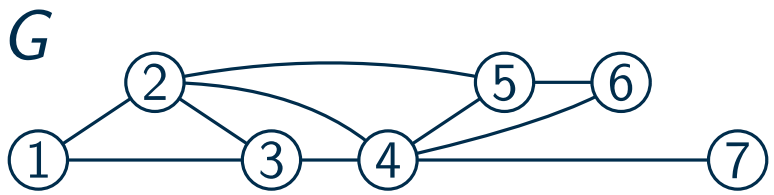
- ratio: sheets vs problem solving vs solutions?
  (should one of them be shorter/longer?)
- difficulty level of problems during exercises?
- topics you would have wanted more/less focus on?
- comments on problem sheets and grading/correction?
- . . .
- don’t forget telling us what you liked :)

otherwise we might change stuff because only people who disliked it commented on it

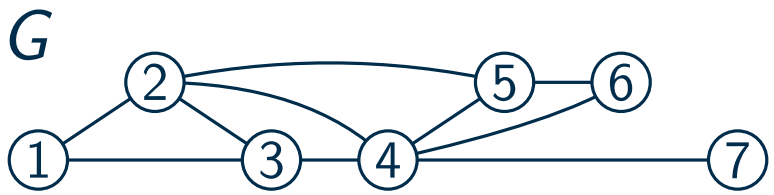
`https://onlineumfrage.kit.edu/evasys/online.php?p=6F1T3`

Recap: Treewidth

Tree decomposition of a graph $G = (V, E)$

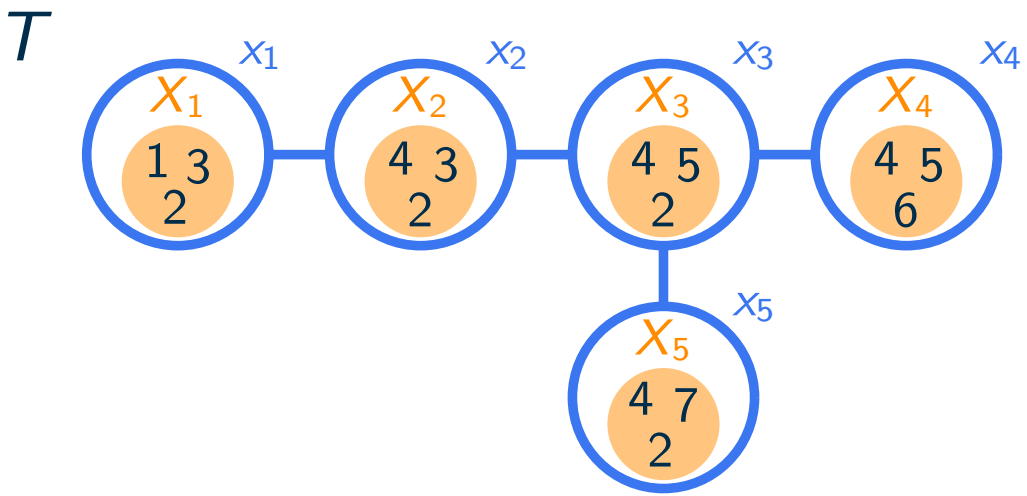


Recap: Treewidth

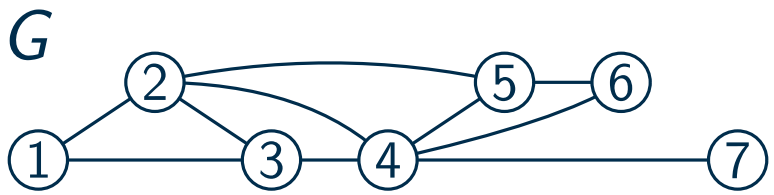


Tree decomposition of a graph $G = (V, E)$

- tree T on nodes $x_1, \ldots, x_r$ with bags $X_1, \ldots, X_r$, such that:

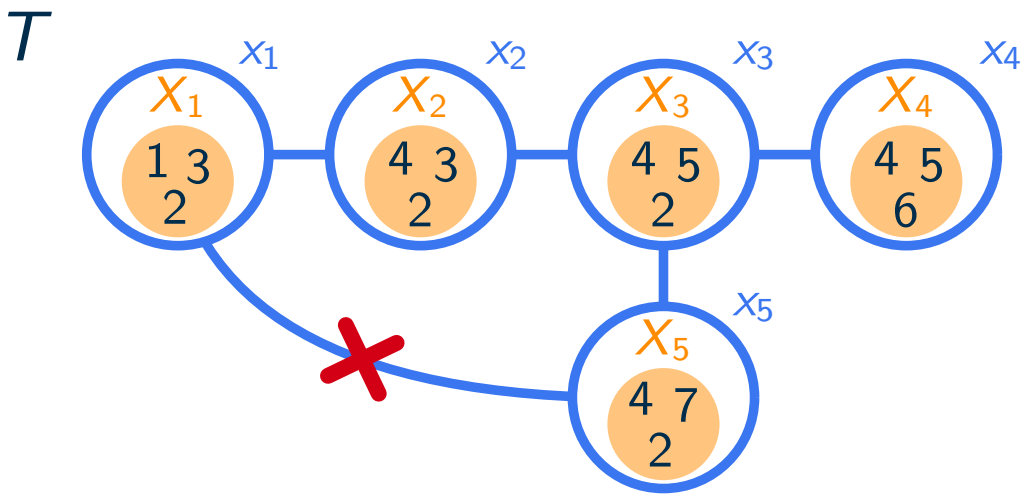


Recap: Treewidth

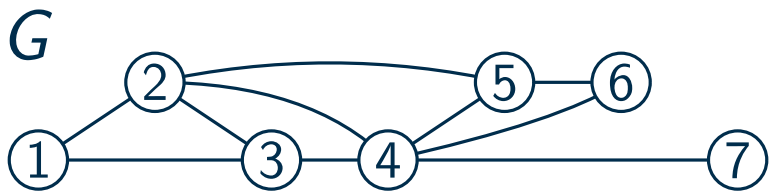


Tree decomposition of a graph $G = (V, E)$

- tree T on nodes $x_1, \ldots, x_r$ with bags $X_1, \ldots, X_r$, such that:

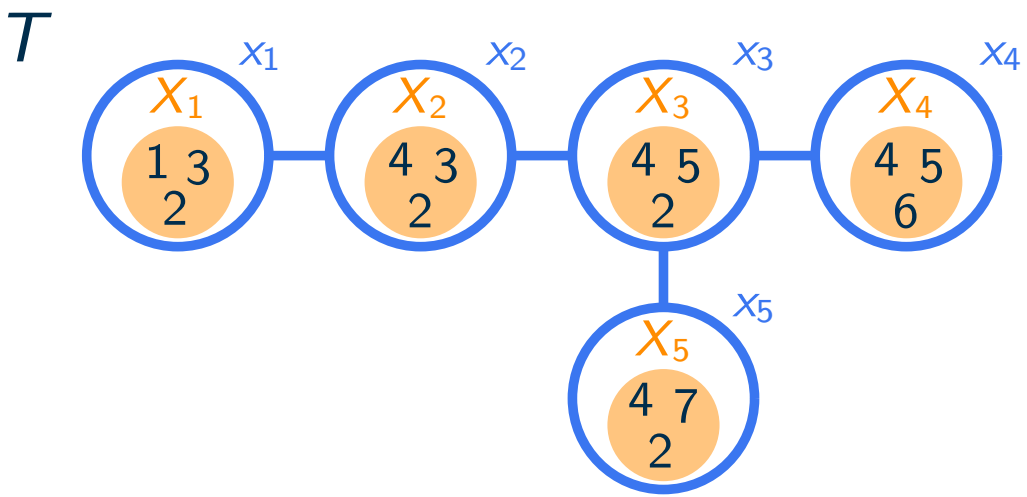


Recap: Treewidth

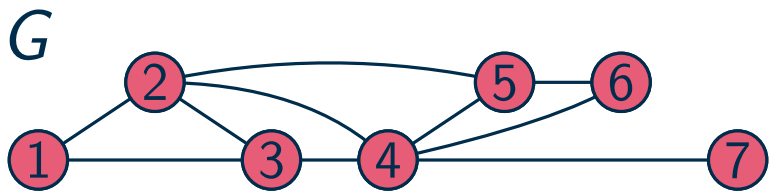


Tree decomposition of a graph $G = (V, E)$

- tree T on nodes $x_1, \ldots, x_r$ with bags $X_1, \ldots, X_r$, such that:
  - $X_1 \cup \cdots \cup X_r = V$

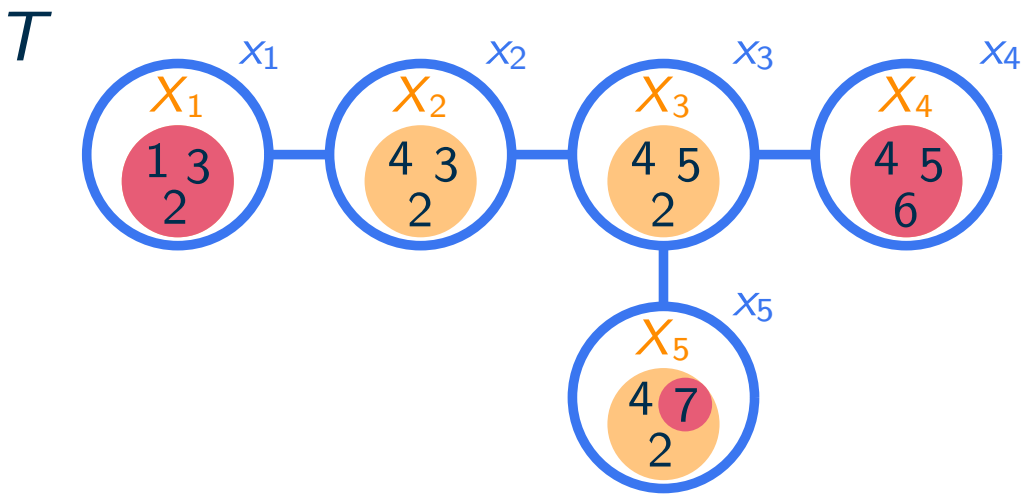


Recap: Treewidth

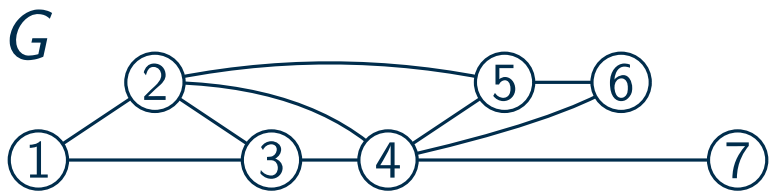


Tree decomposition of a graph $G = (V, E)$

- tree T on nodes $x_1, \ldots, x_r$ with bags $X_1, \ldots, X_r$, such that:
  - $X_1 \cup \cdots \cup X_r = V$

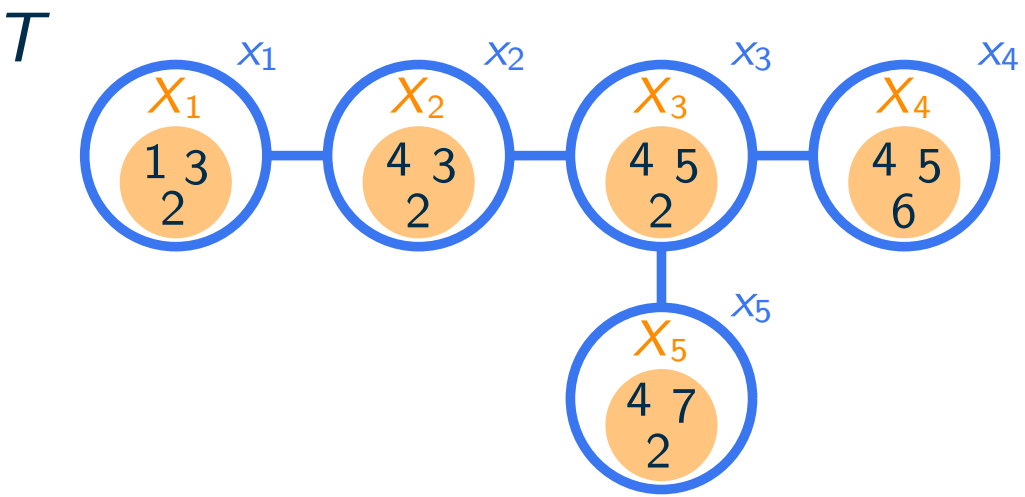


Recap: Treewidth

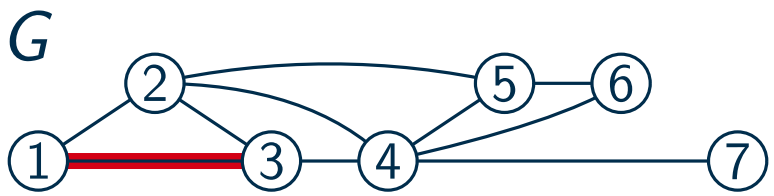


Tree decomposition of a graph $G = (V, E)$

- tree T on nodes $x_1, \ldots, x_r$ with bags $X_1, \ldots, X_r$, such that:
  - $X_1 \cup \cdots \cup X_r = V$
  - $\{u, v\} \in E \Rightarrow u, v$ share a bag

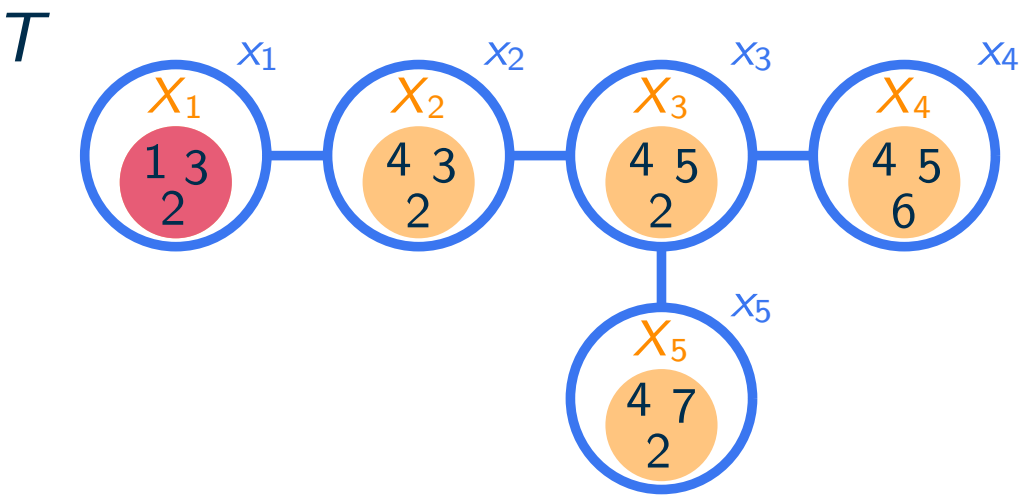


Recap: Treewidth

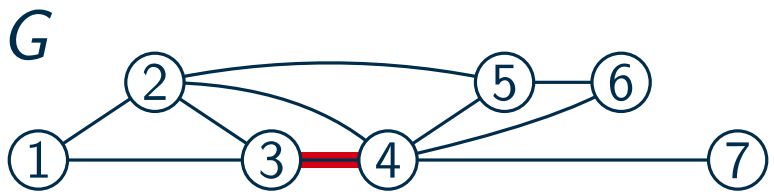


Tree decomposition of a graph $G = (V, E)$

- tree T on nodes $x_1, \ldots, x_r$ with bags $X_1, \ldots, X_r$, such that:
  - $X_1 \cup \cdots \cup X_r = V$
  - $\{u, v\} \in E \Rightarrow u, v$ share a bag

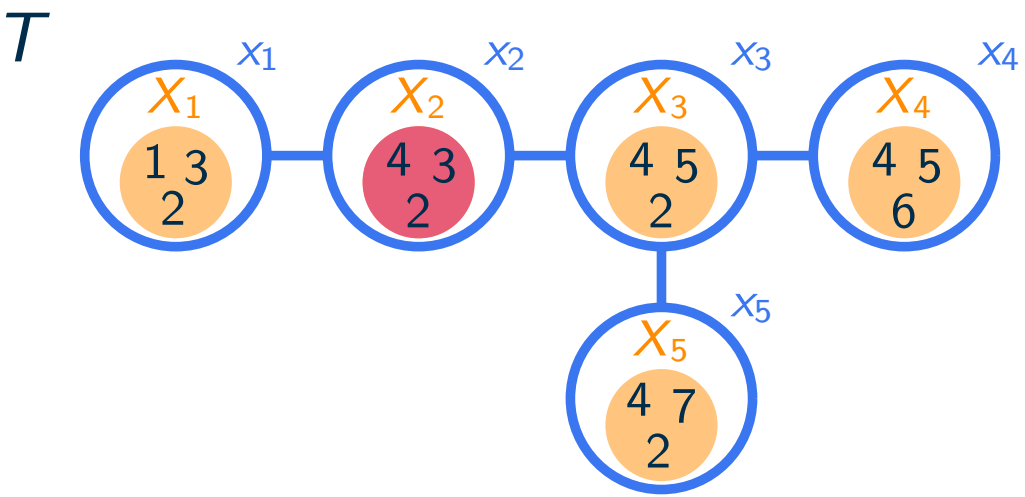


Recap: Treewidth

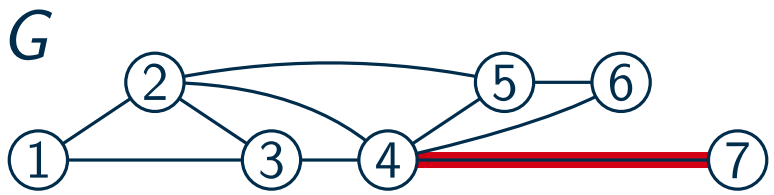


Tree decomposition of a graph $G = (V, E)$

- tree T on nodes $x_1, \ldots, x_r$ with bags $X_1, \ldots, X_r$, such that:
  - $X_1 \cup \cdots \cup X_r = V$
  - $\{u, v\} \in E \Rightarrow u, v$ share a bag

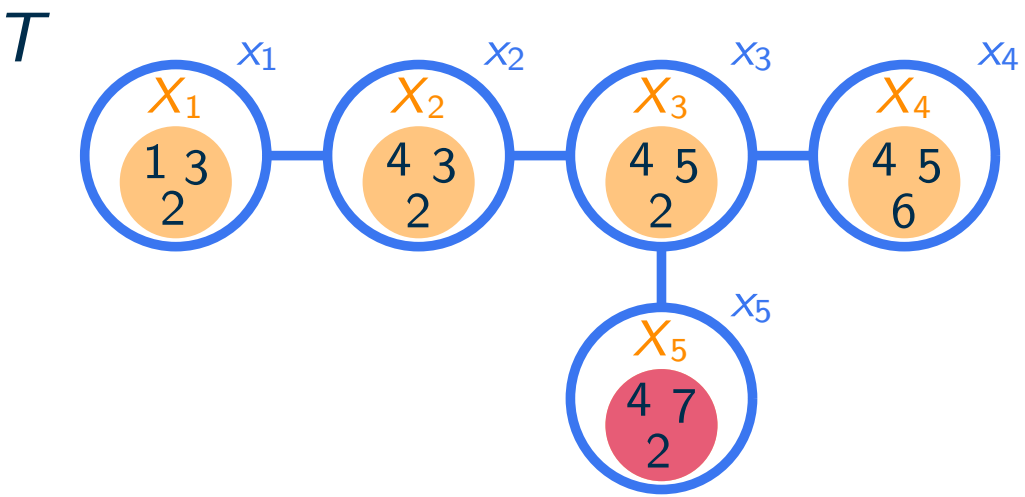


Recap: Treewidth

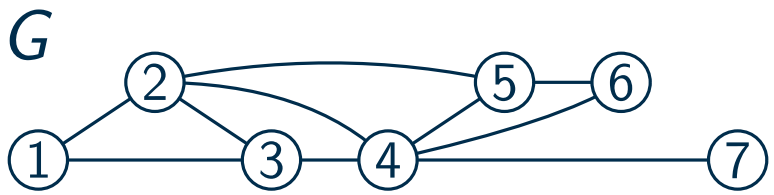


Tree decomposition of a graph $G = (V, E)$

- tree T on nodes $x_1, \ldots, x_r$ with bags $X_1, \ldots, X_r$, such that:
  - $X_1 \cup \cdots \cup X_r = V$
  - $\{u, v\} \in E \Rightarrow u, v$ share a bag

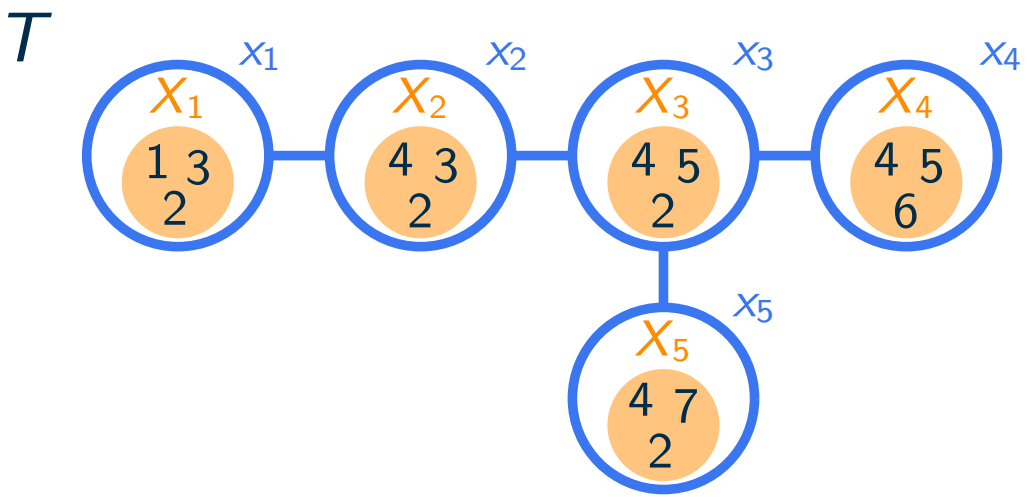


Recap: Treewidth

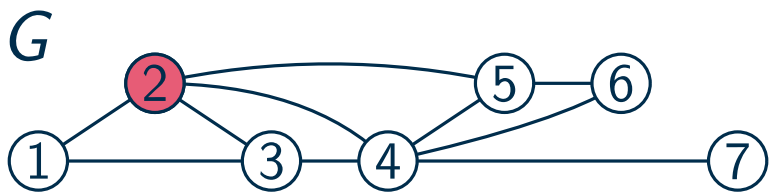


Tree decomposition of a graph $G = (V, E)$

- tree T on nodes $x_1, \ldots, x_r$ with bags $X_1, \ldots, X_r$, such that:
  - $X_1 \cup \cdots \cup X_r = V$
  - $\{u, v\} \in E \Rightarrow u, v$ share a bag
  - the bags of every vertex form a subtree

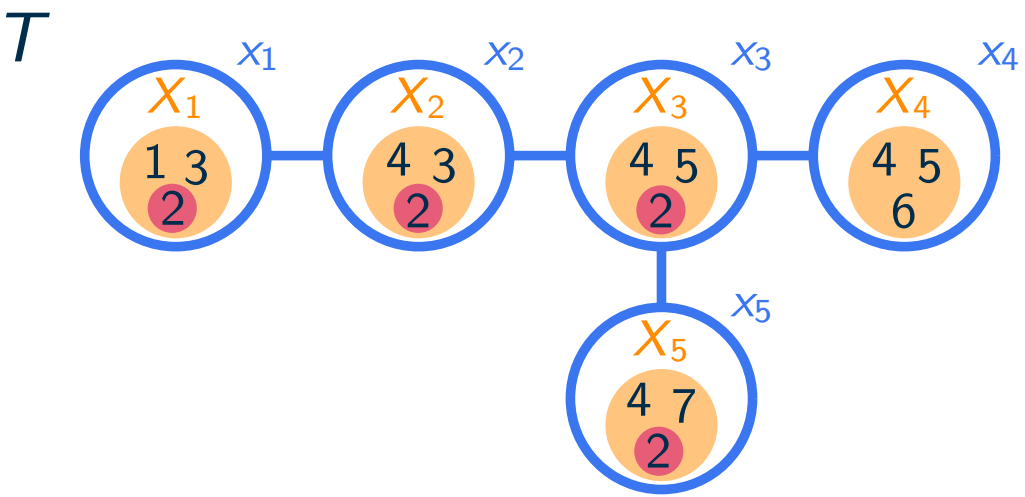


Recap: Treewidth

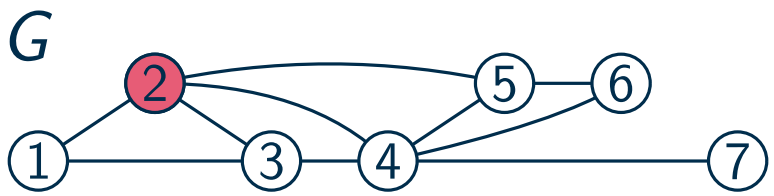


Tree decomposition of a graph $G = (V, E)$

- tree T on nodes $x_1, \ldots, x_r$ with bags $X_1, \ldots, X_r$, such that:
  - $X_1 \cup \cdots \cup X_r = V$
  - $\{u, v\} \in E \Rightarrow u, v$ share a bag
  - the bags of every vertex form a subtree

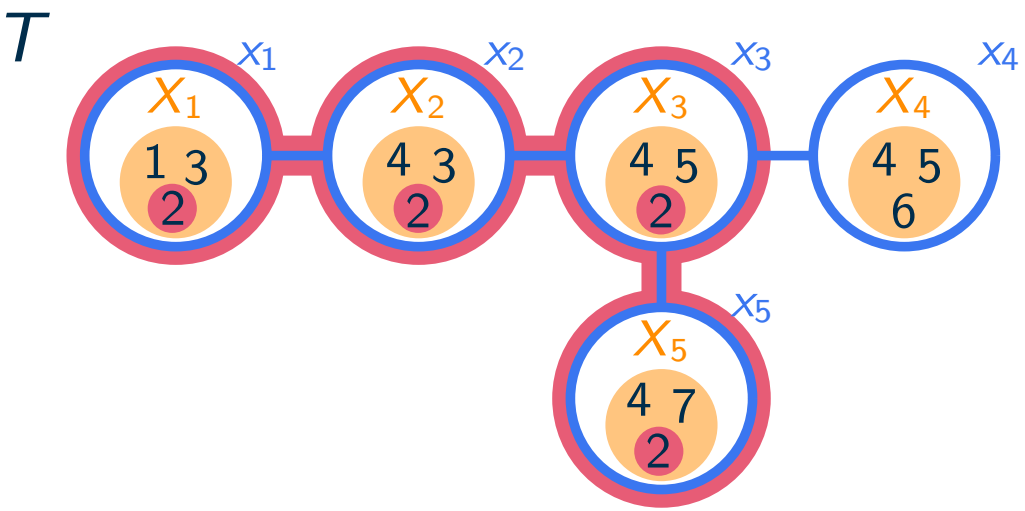


Recap: Treewidth

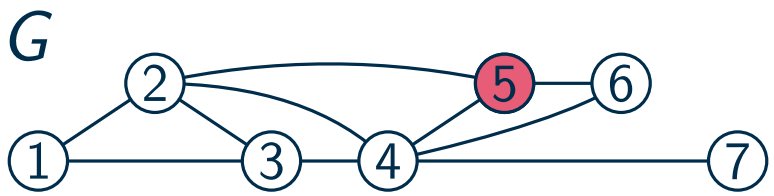


Tree decomposition of a graph $G = (V, E)$

- tree T on nodes $x_1, \ldots, x_r$ with bags $X_1, \ldots, X_r$, such that:
  - $X_1 \cup \cdots \cup X_r = V$
  - $\{u, v\} \in E \Rightarrow u, v$ share a bag
  - the bags of every vertex form a subtree

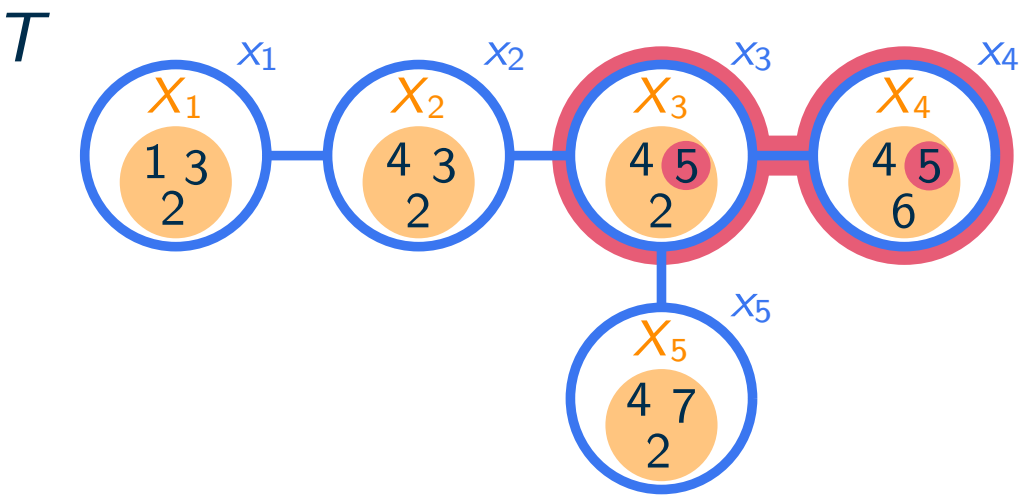


Recap: Treewidth

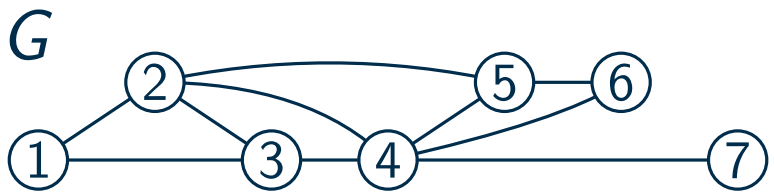


Tree decomposition of a graph $G = (V, E)$

- tree T on nodes $x_1, \ldots, x_r$ with bags $X_1, \ldots, X_r$, such that:
  - $X_1 \cup \cdots \cup X_r = V$
  - $\{u, v\} \in E \Rightarrow u, v$ share a bag
  - the bags of every vertex form a subtree



Recap: Treewidth

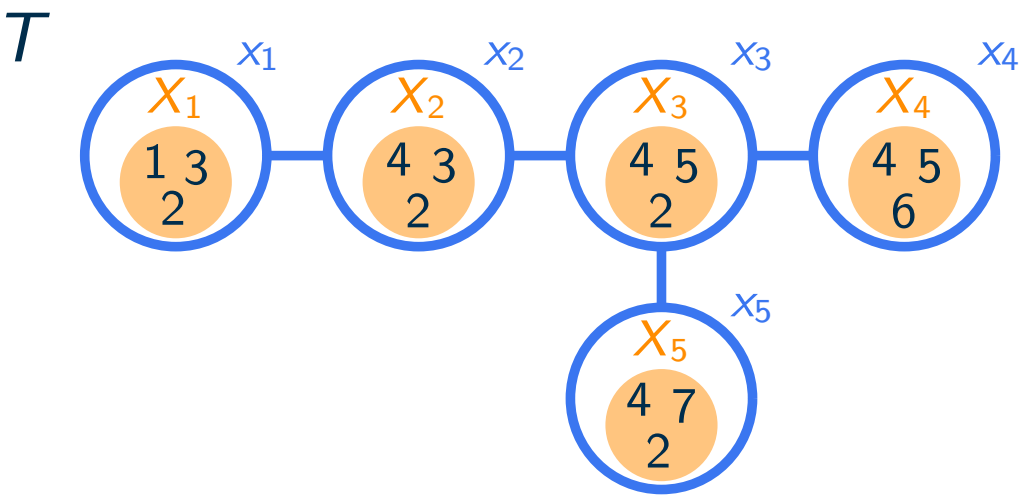


Tree decomposition of a graph $G = (V, E)$

- tree T on nodes $x_1, \ldots, x_r$ with bags $X_1, \ldots, X_r$, such that:
  - $X_1 \cup \cdots \cup X_r = V$
  - $\{u, v\} \in E \Rightarrow u, v$ share a bag
  - the bags of every vertex form a subtree

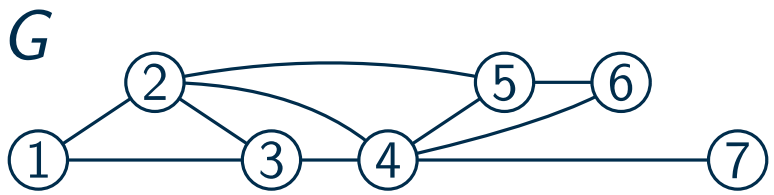
Treewidth

- width of a specific decomposition: $\max\{|X_i|\} - 1$



width 2

Recap: Treewidth

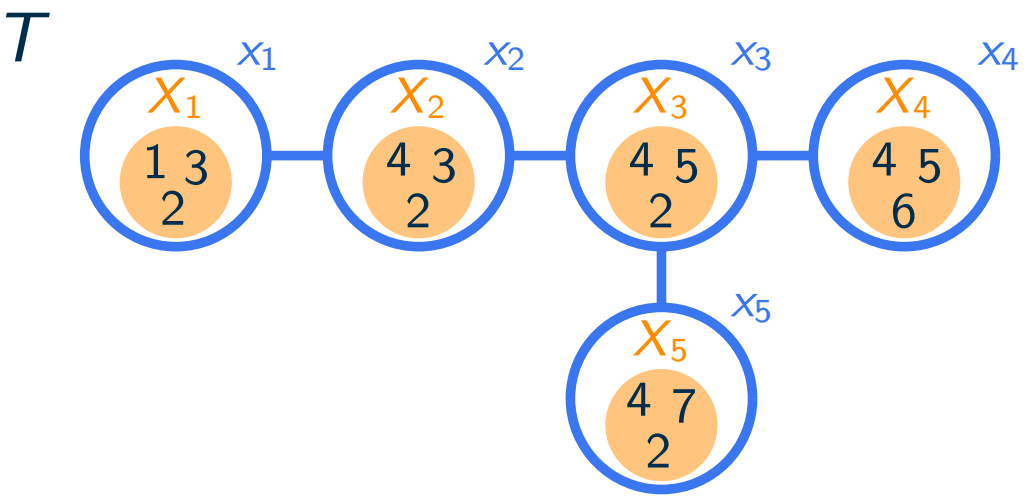


Tree decomposition of a graph $G = (V, E)$

- tree T on nodes $x_1, \ldots, x_r$ with bags $X_1, \ldots, X_r$, such that:
  - $X_1 \cup \cdots \cup X_r = V$
  - $\{u, v\} \in E \Rightarrow u, v$ share a bag
  - the bags of every vertex form a subtree

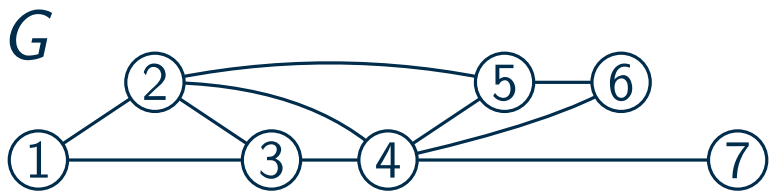
Treewidth

- width of a specific decomposition: $\max\{|X_i|\} - 1$
- treewidth of a graph: minimum width of a decomposition



width 2

Recap: Treewidth

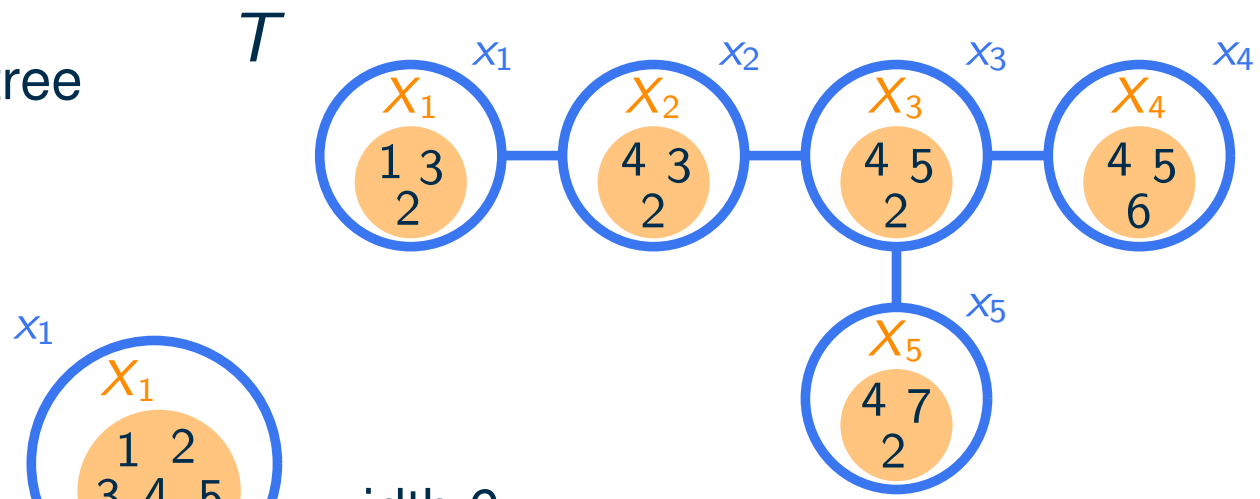


Tree decomposition of a graph $G = (V, E)$

- tree T on nodes $x_1, \ldots, x_r$ with bags $X_1, \ldots, X_r$, such that:
  - $X_1 \cup \cdots \cup X_r = V$
  - $\{u, v\} \in E \Rightarrow u, v$ share a bag
  - the bags of every vertex form a subtree

Treewidth

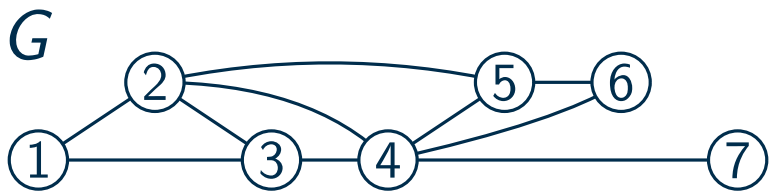
- width of a specific decomposition: $\max\{|X_i|\} - 1$
- treewidth of a graph: minimum width of a decomposition



width 6

width 2

Recap: Treewidth

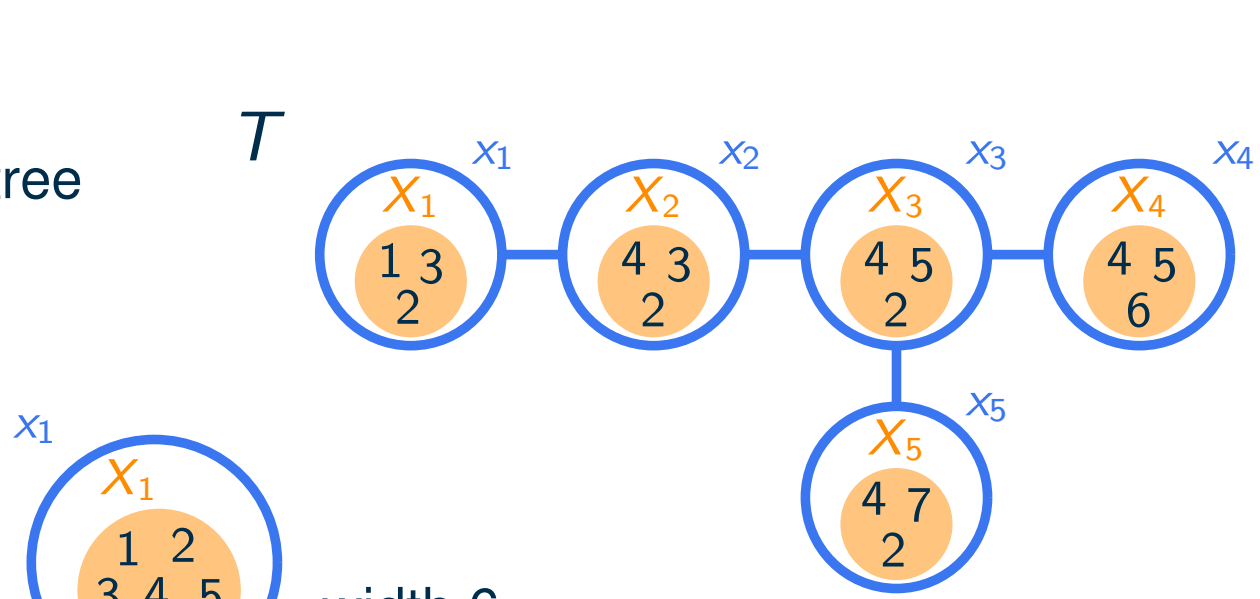


Tree decomposition of a graph $G = (V, E)$

- tree T on nodes $x_1, \ldots, x_r$ with bags $X_1, \ldots, X_r$, such that:
  - $X_1 \cup \cdots \cup X_r = V$
  - $\{u, v\} \in E \Rightarrow u, v$ share a bag
  - the bags of every vertex form a subtree

Treewidth

- width of a specific decomposition: $\max\{|X_i|\} - 1$
- treewidth of a graph: minimum width of a decomposition



width 6

width 2

$\Rightarrow$ treewidth ≤ 2

Treewidth Problems

Show the following statements:

Treewidth Problems

Show the following statements:

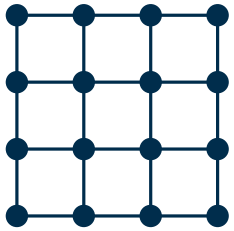
Every tree with more than one vertex has treewidth 1. ⋆

Treewidth Problems

Show the following statements:

Every tree with more than one vertex has treewidth 1. ⋆

For each $k \in \mathbb{N}$, the $k \times k$-grid has treewidth at most k. ⋆⋆



4×4-grid

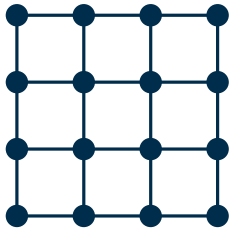
Treewidth Problems

Show the following statements:

Every tree with more than one vertex has treewidth 1. ⋆

For each $k \in \mathbb{N}$, the $k \times k$-grid has treewidth at most k. ⋆⋆

. . . at least k. ⁂



https://en.wikipedia.org/wiki/Bramble_(graph_theory)

4×4-grid

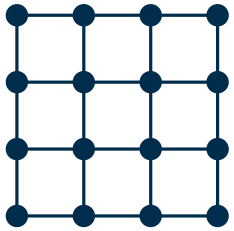
Treewidth Problems

Show the following statements:

Every tree with more than one vertex has treewidth 1. ⋆

For each $k \in \mathbb{N}$, the $k \times k$-grid has treewidth at most k. ⋆⋆

. . . at least k. ⋆⋆ ⋆⋆



https://en.wikipedia.org/wiki/Bramble_(graph_theory)

4×4-grid

For each clique C in a graph G, every tree decomposition of G has a bag X_i with $C \subseteq X_i$ ⋆⋆

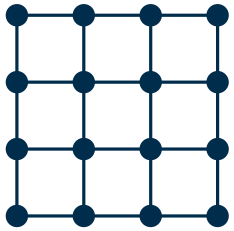
Treewidth Problems

Show the following statements:

Every tree with more than one vertex has treewidth 1. ⋆

For each $k \in \mathbb{N}$, the $k \times k$-grid has treewidth at most k. ⋆⋆

. . . at least k. ⋆⋆



https://en.wikipedia.org/wiki/Bramble_(graph_theory)

4×4-grid

For each clique C in a graph G, every tree decomposition of G has a bag X_i with $C \subseteq X_i$ ⋆⋆

Cops and Robber

- two-player game on graphs, where c cops want to catch one robber

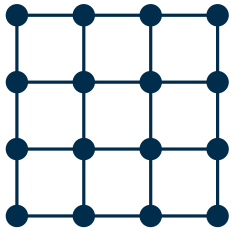
Treewidth Problems

Show the following statements:

Every tree with more than one vertex has treewidth 1. ⋆

For each $k \in \mathbb{N}$, the $k \times k$-grid has treewidth at most k. ⋆⋆

. . . at least k. ⋆⋆



https://en.wikipedia.org/wiki/Bramble_(graph_theory)

4×4-grid

For each clique C in a graph G, every tree decomposition of G has a bag X_i with $C \subseteq X_i$ ⋆⋆

Cops and Robber

- two-player game on graphs, where c cops want to catch one robber
- beginning: cops choose their starting vertices, then robber chooses a vertex

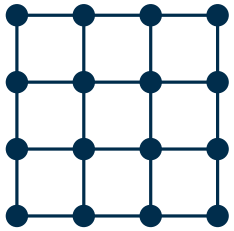
Treewidth Problems

Show the following statements:

Every tree with more than one vertex has treewidth 1. ⋆

For each $k \in \mathbb{N}$, the $k \times k$-grid has treewidth at most k. ⋆⋆

. . . at least k. ⋆⋆



https://en.wikipedia.org/wiki/Bramble_(graph_theory)

4×4-grid

For each clique C in a graph G, every tree decomposition of G has a bag X_i with $C \subseteq X_i$ ⋆⋆

Cops and Robber

- two-player game on graphs, where c cops want to catch one robber
- beginning: cops choose their starting vertices, then robber chooses a vertex

In each round:

- one cop announces new position $v \in V$ and enters helicopter

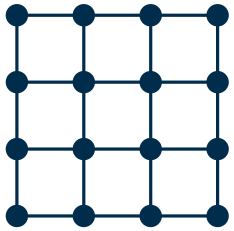
Treewidth Problems

Show the following statements:

Every tree with more than one vertex has treewidth 1. ⋆

For each $k \in \mathbb{N}$, the $k \times k$-grid has treewidth at most k. ⋆⋆

. . . at least k. ⋆⋆



https://en.wikipedia.org/wiki/Bramble_(graph_theory)

4×4-grid

For each clique C in a graph G, every tree decomposition of G has a bag X_i with $C \subseteq X_i$ ⋆⋆

Cops and Robber

- two-player game on graphs, where c cops want to catch one robber
- beginning: cops choose their starting vertices, then robber chooses a vertex

In each round:

- one cop announces new position $v \in V$ and enters helicopter
- robber may move arbitrarily far along edges, but not through cops

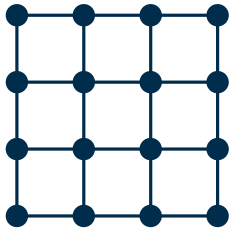
Treewidth Problems

Show the following statements:

Every tree with more than one vertex has treewidth 1. ⋆

For each $k \in \mathbb{N}$, the $k \times k$-grid has treewidth at most k. ⋆⋆

. . . at least k. ⋆⋆



https://en.wikipedia.org/wiki/Bramble_(graph_theory)

4×4-grid

For each clique C in a graph G, every tree decomposition of G has a bag X_i with $C \subseteq X_i$ ⋆⋆

Cops and Robber

- two-player game on graphs, where c cops want to catch one robber
- beginning: cops choose their starting vertices, then robber chooses a vertex

In each round:

- one cop announces new position $v \in V$ and enters helicopter
- robber may move arbitrarily far along edges, but not through cops
- cop lands at position v

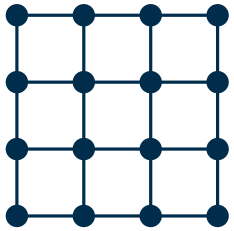
Treewidth Problems

Show the following statements:

Every tree with more than one vertex has treewidth 1. ⋆

For each $k \in \mathbb{N}$, the $k \times k$-grid has treewidth at most k. ⋆⋆

. . . at least k. ⋆⋆



https://en.wikipedia.org/wiki/Bramble_(graph_theory)

4×4-grid

For each clique C in a graph G, every tree decomposition of G has a bag X_i with $C \subseteq X_i$ ⋆⋆

Cops and Robber

- two-player game on graphs, where c cops want to catch one robber
- beginning: cops choose their starting vertices, then robber chooses a vertex

In each round:

- one cop announces new position $v \in V$ and enters helicopter
- robber may move arbitrarily far along edges, but not through cops
- cop lands at position v

Game Over: robber loses if colocated with cop at round end

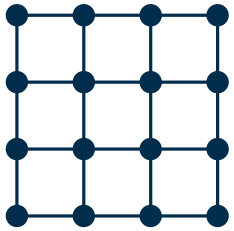
Treewidth Problems

Show the following statements:

Every tree with more than one vertex has treewidth 1. ⋆

For each $k \in \mathbb{N}$, the $k \times k$-grid has treewidth at most k. ⋆⋆

. . . at least k. ⋆⋆



https://en.wikipedia.org/wiki/Bramble_(graph_theory)

4×4-grid

For each clique C in a graph G, every tree decomposition of G has a bag X_i with $C \subseteq X_i$ ⋆⋆

Cops and Robber

- two-player game on graphs, where c cops want to catch one robber
- beginning: cops choose their starting vertices, then robber chooses a vertex

In each round:

- one cop announces new position $v \in V$ and enters helicopter
- robber may move arbitrarily far along edges, but not through cops
- cop lands at position v

Game Over: robber loses if colocated with cop at round end

In a graph with treewidth $\leq k$, there is a winning strategy for $k + 1$ cops. ⋆⋆⋆

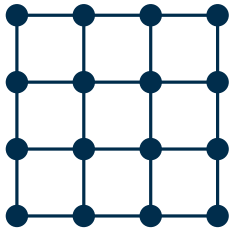
Treewidth Problems

Show the following statements:

Every tree with more than one vertex has treewidth 1. ⋆

For each $k \in \mathbb{N}$, the $k \times k$-grid has treewidth at most k. ⋆⋆

. . . at least k. ⋆⋆



https://en.wikipedia.org/wiki/Bramble_(graph_theory)

4×4-grid

For each clique C in a graph G, every tree decomposition of G has a bag X_i with $C \subseteq X_i$ ⋆⋆

https://www.sciencedirect.com/science/article/pii/S0095895683710270

Cops and Robber

- two-player game on graphs, where c cops want to catch one robber
- beginning: cops choose their starting vertices, then robber chooses a vertex

In each round:

- one cop announces new position $v \in V$ and enters helicopter
- robber may move arbitrarily far along edges, but not through cops
- cop lands at position v

Game Over: robber loses if colocated with cop at round end

In a graph with treewidth $\leq k$, there is a winning strategy for $k + 1$ cops. ⋆⋆⋆

other direction? ⋆⋆⋆

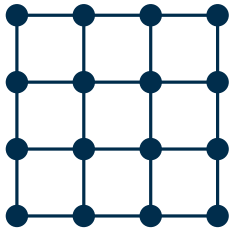
Treewidth Problems

Show the following statements:

Every tree with more than one vertex has treewidth 1. ⋆

For each $k \in \mathbb{N}$, the $k \times k$-grid has treewidth at most k. ⋆⋆

. . . at least k. ⋆⋆



https://en.wikipedia.org/wiki/Bramble_(graph_theory)

4×4-grid

For each clique C in a graph G, every tree decomposition of G has a bag X_i with $C \subseteq X_i$ ⋆⋆

induction on $|C|$

https://www.sciencedirect.com/science/article/pii/S0095895683710270

Cops and Robber

- two-player game on graphs, where c cops want to catch one robber
- beginning: cops choose their starting vertices, then robber chooses a vertex

In each round:

- one cop announces new position $v \in V$ and enters helicopter
- robber may move arbitrarily far along edges, but not through cops
- cop lands at position v

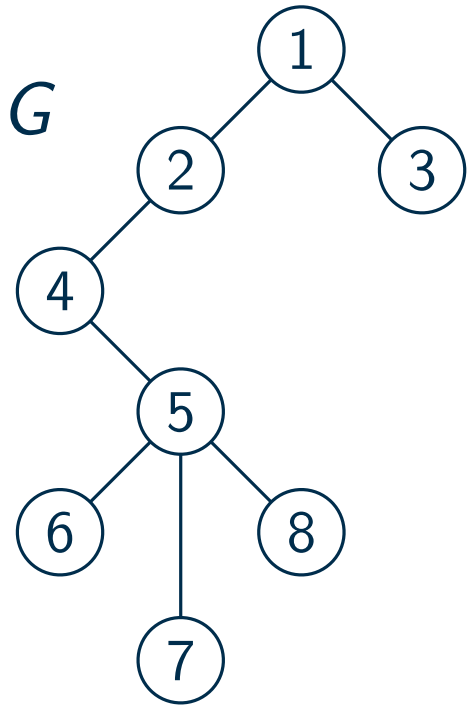
Game Over: robber loses if colocated with cop at round end

In a graph with treewidth $\leq k$, there is a winning strategy for $k + 1$ cops. ⋆⋆⋆

other direction? ⋆⋆⋆

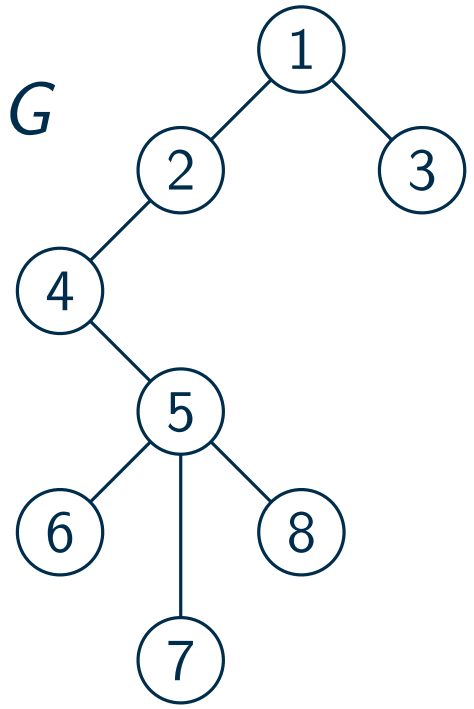
Treewidth of Trees

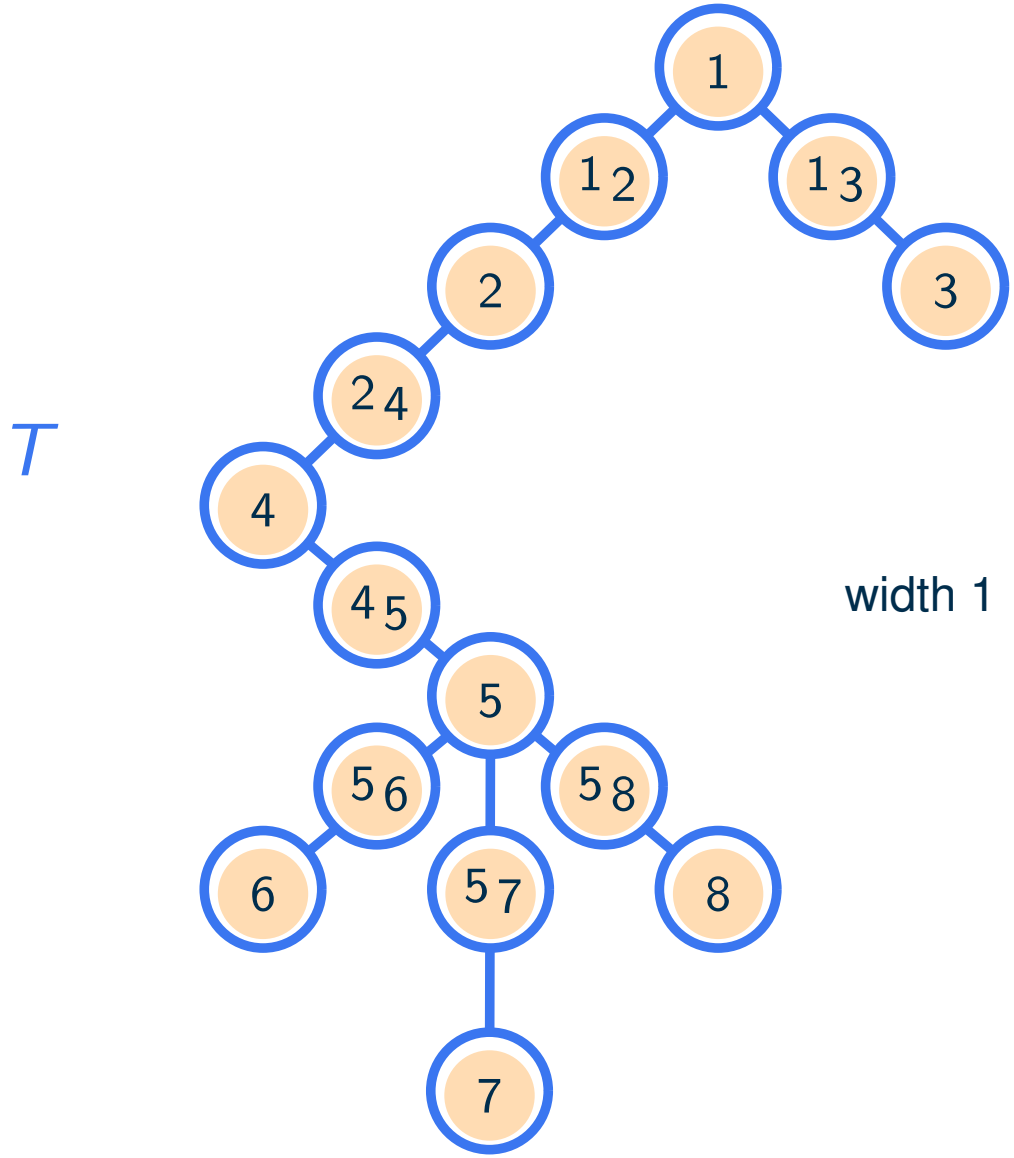
Every tree with more than one vertex has treewidth 1.



Treewidth of Trees

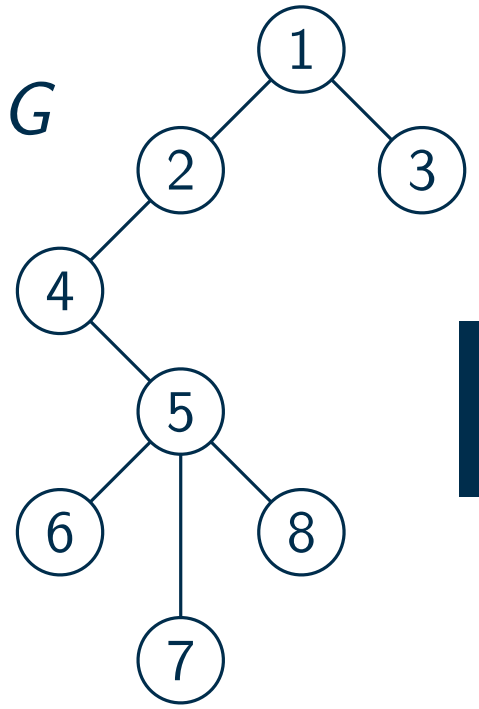
Every tree with more than one vertex has treewidth 1.



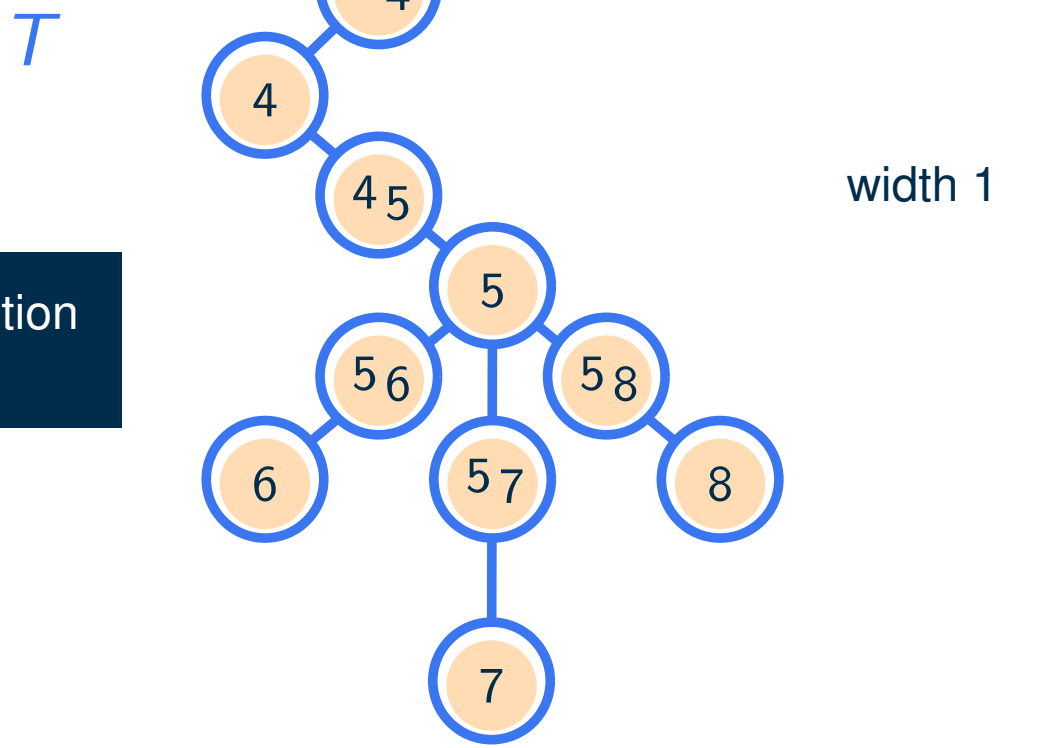


Treewidth of Trees

Every tree with more than one vertex has treewidth 1.

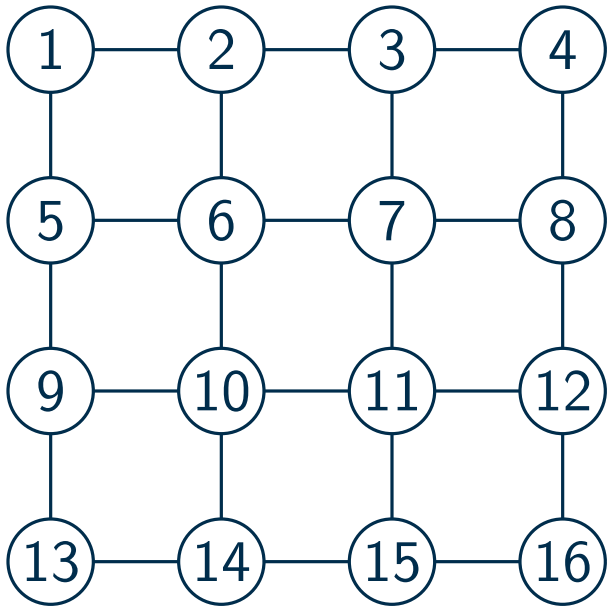


Why is there no decomposition with smaller width?



Treewidth of Grids

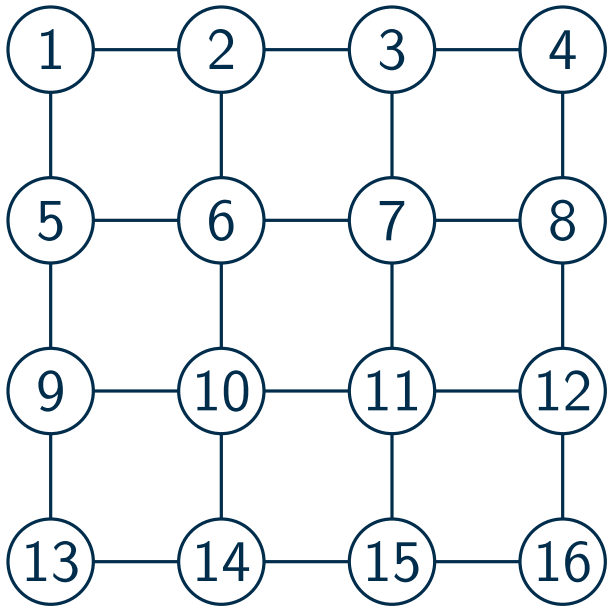
For each $k \in \mathbb{N}$, the $k \times k$-grid has treewidth at most k.



Treewidth of Grids

For each $k \in \mathbb{N}$, the $k \times k$-grid has treewidth at most k.

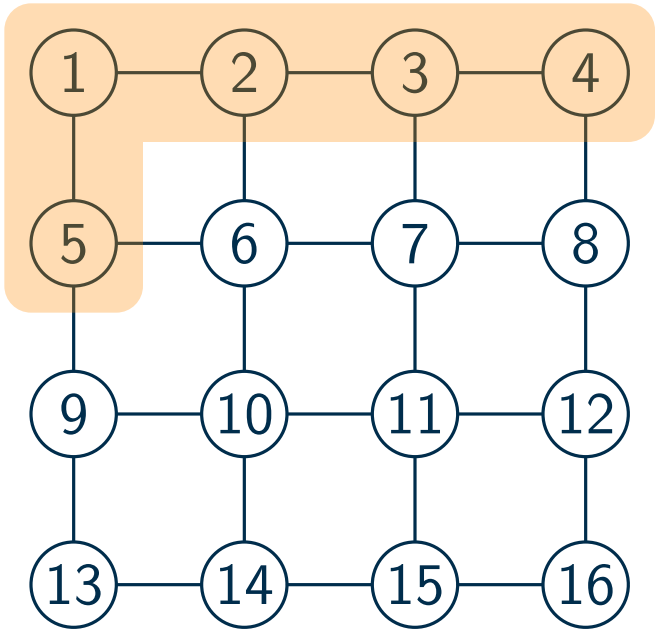
i.e. construct tree decomposition of width k

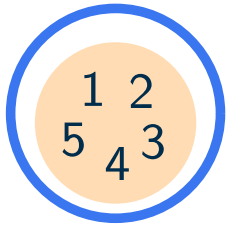


Treewidth of Grids

For each $k \in \mathbb{N}$, the $k \times k$-grid has treewidth at most k.

i.e. construct tree decomposition of width k

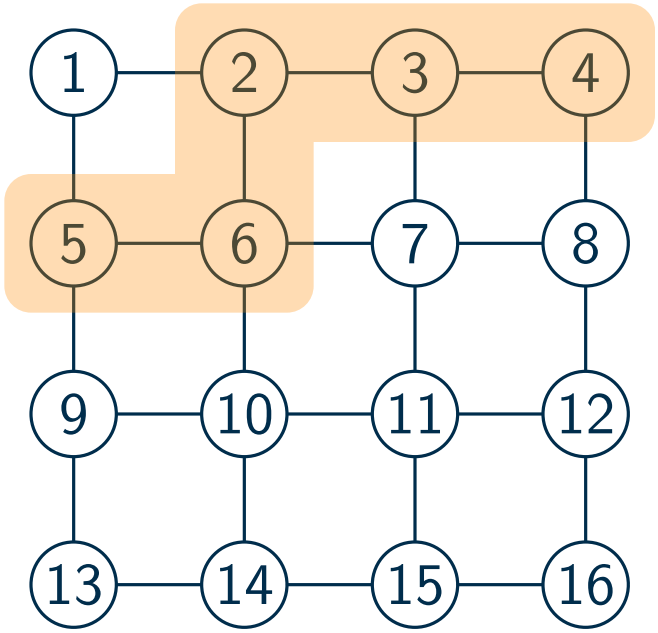


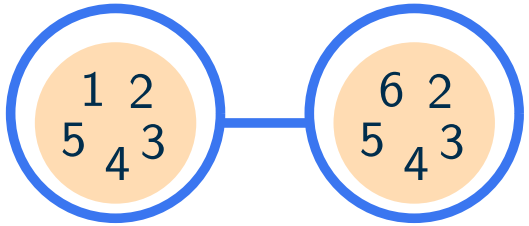


Treewidth of Grids

For each $k \in \mathbb{N}$, the $k \times k$-grid has treewidth at most k.

i.e. construct tree decomposition of width k

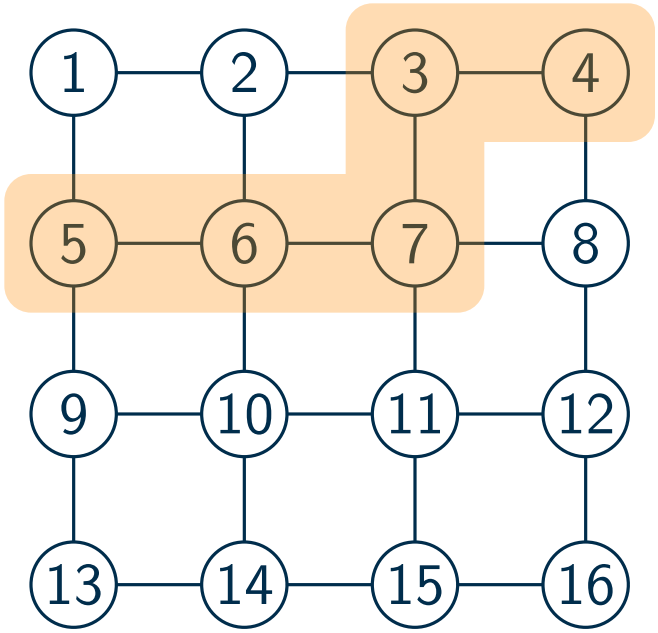


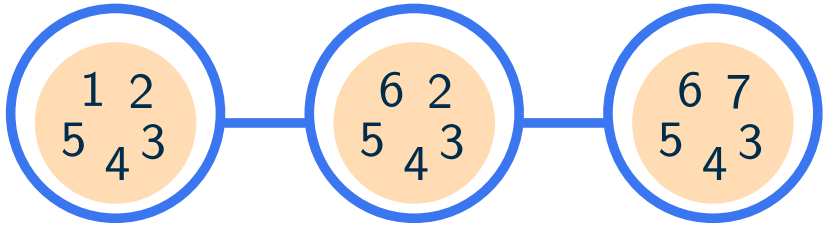


Treewidth of Grids

For each $k \in \mathbb{N}$, the $k \times k$-grid has treewidth at most k.

i.e. construct tree decomposition of width k

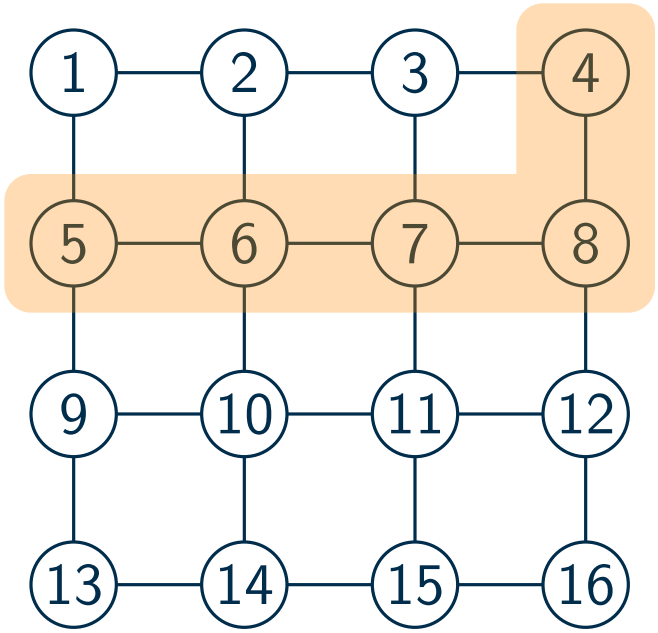


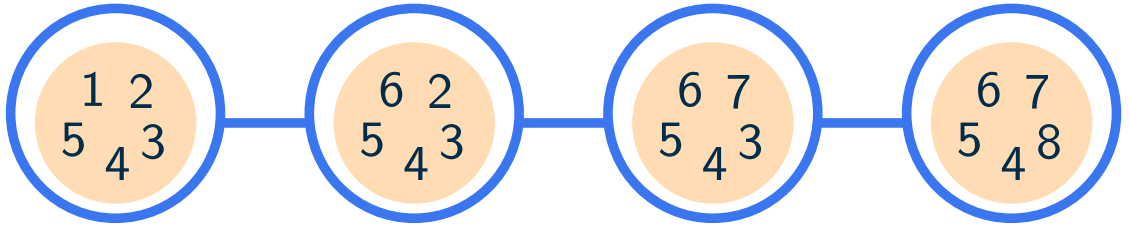


Treewidth of Grids

For each $k \in \mathbb{N}$, the $k \times k$-grid has treewidth at most k.

i.e. construct tree decomposition of width k

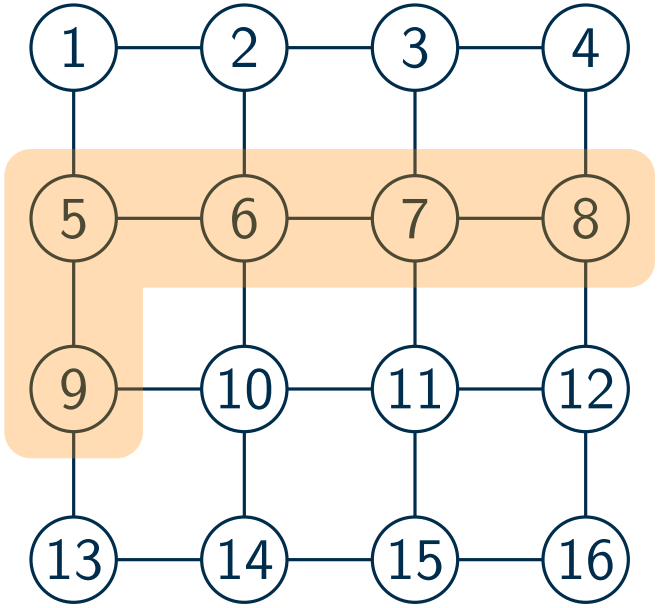


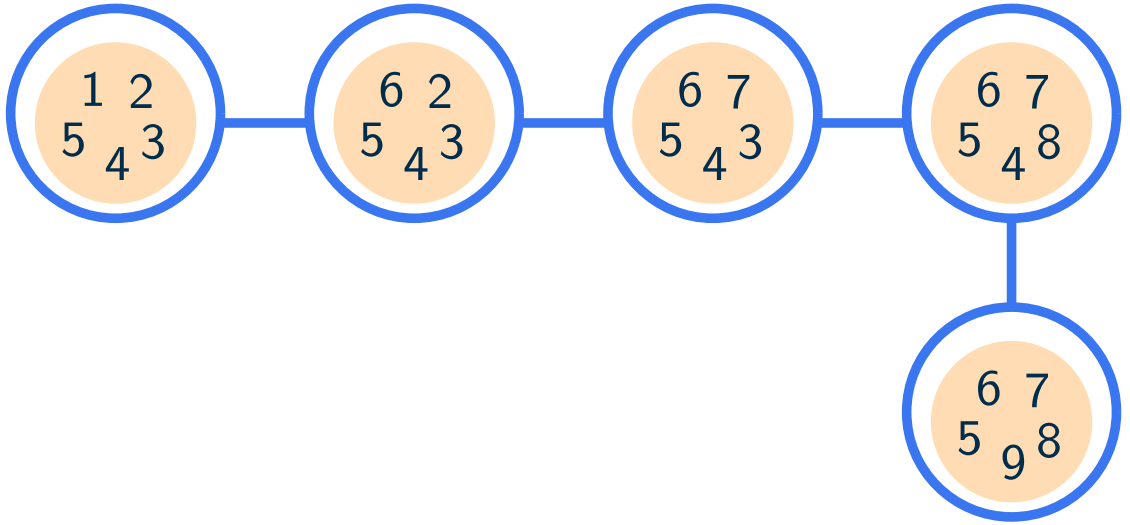


Treewidth of Grids

For each $k \in \mathbb{N}$, the $k \times k$-grid has treewidth at most k.

i.e. construct tree decomposition of width k

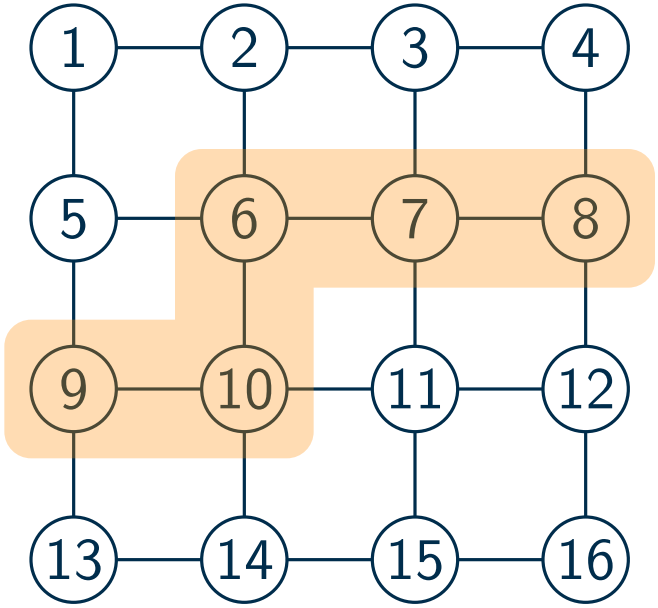


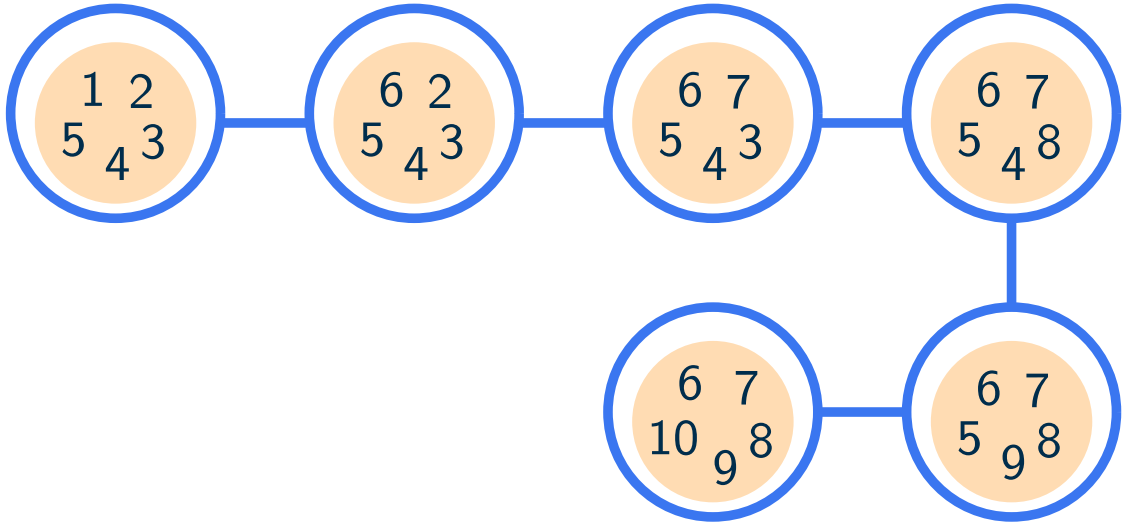


Treewidth of Grids

For each $k \in \mathbb{N}$, the $k \times k$-grid has treewidth at most k.

i.e. construct tree decomposition of width k

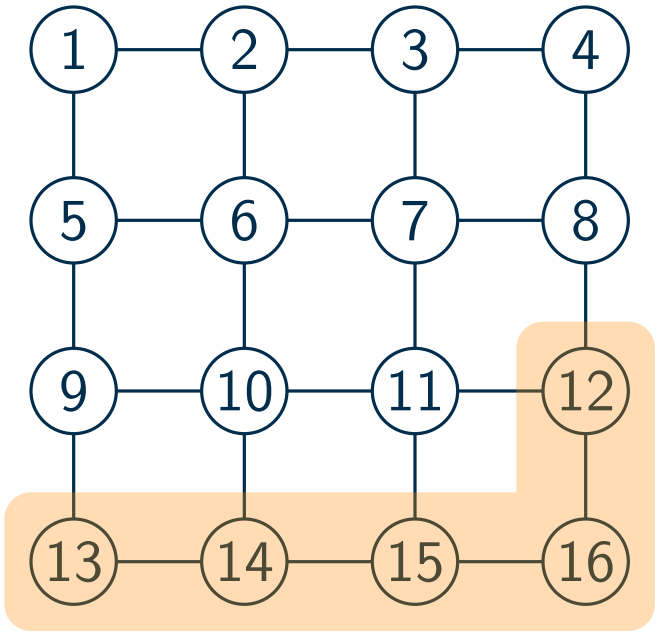


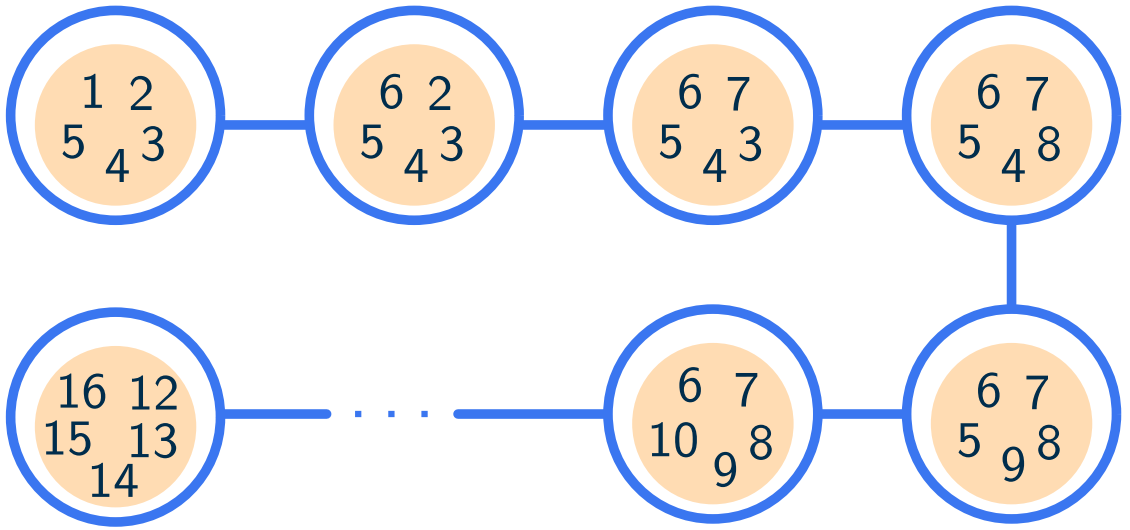


Treewidth of Grids

For each $k \in \mathbb{N}$, the $k \times k$-grid has treewidth at most k.

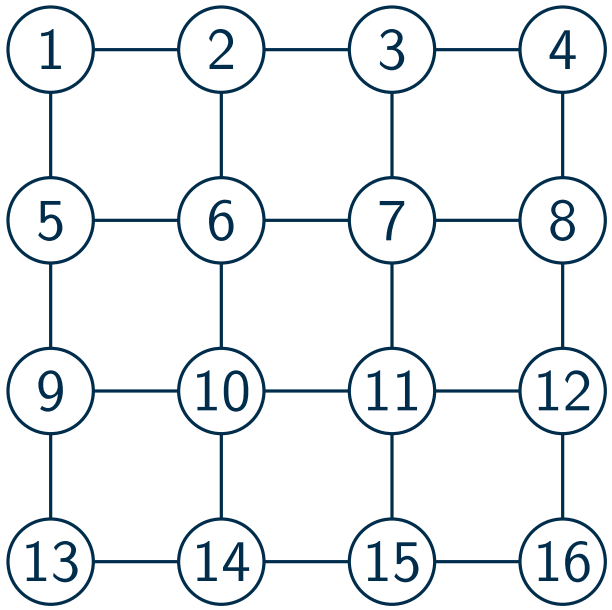
i.e. construct tree decomposition of width k





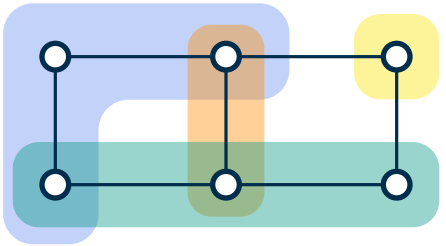
Treewidth of Grids

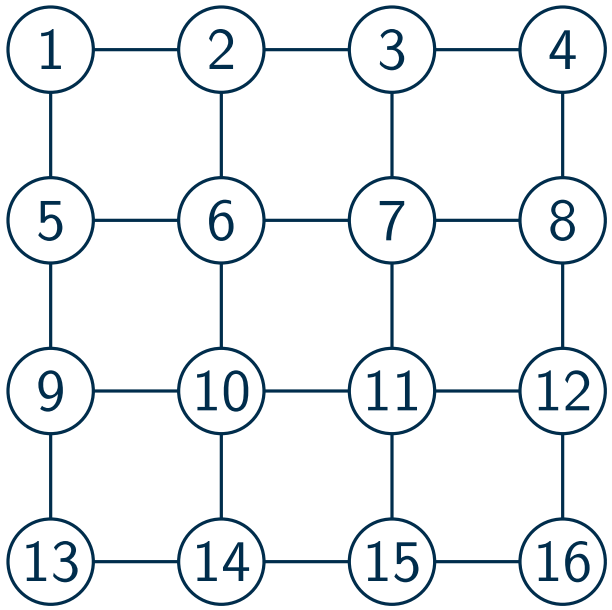
For each $k \in \mathbb{N}$, the $k \times k$-grid has treewidth at least k.



Treewidth of Grids

For each $k \in \mathbb{N}$, the $k \times k$-grid has treewidth at least k.

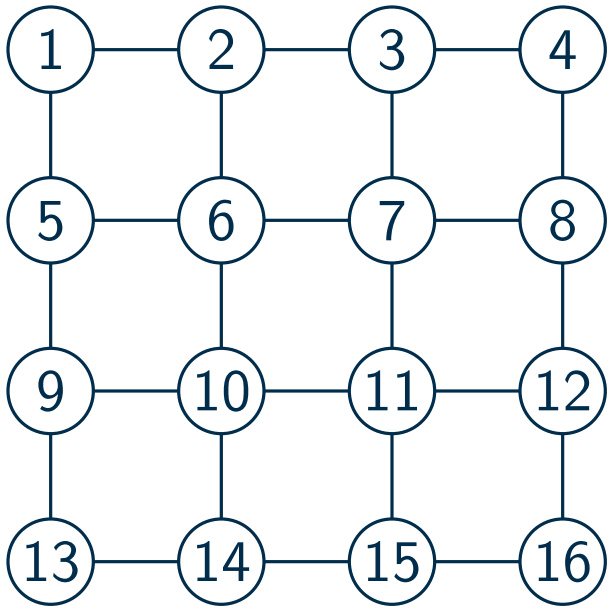


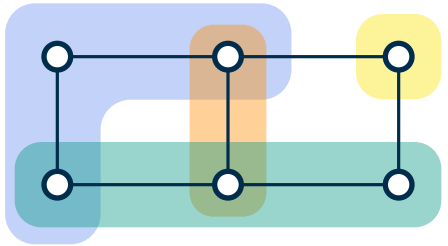


bramble: family of connected subgraphs that are pairwise adjacent

Treewidth of Grids

For each $k \in \mathbb{N}$, the $k \times k$-grid has treewidth at least k.



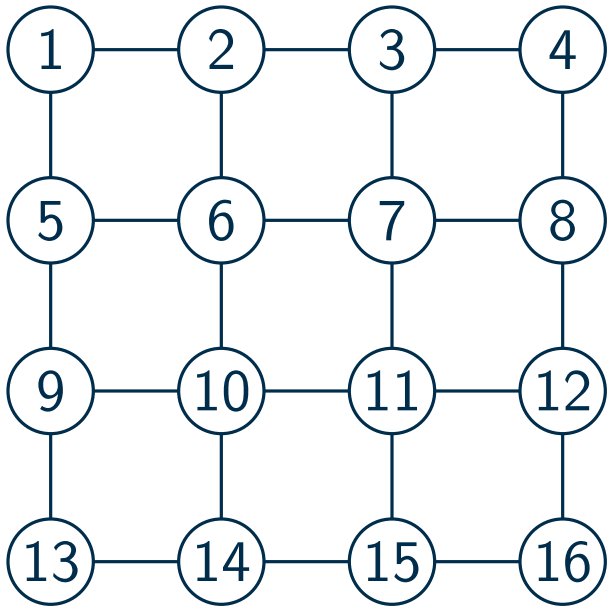


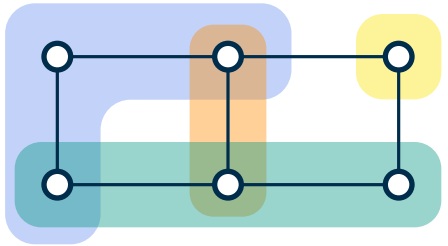
bramble: family of connected subgraphs that are pairwise adjacent

order of a bramble: size of its smallest hitting set

Treewidth of Grids

For each $k \in \mathbb{N}$, the $k \times k$-grid has treewidth at least k.





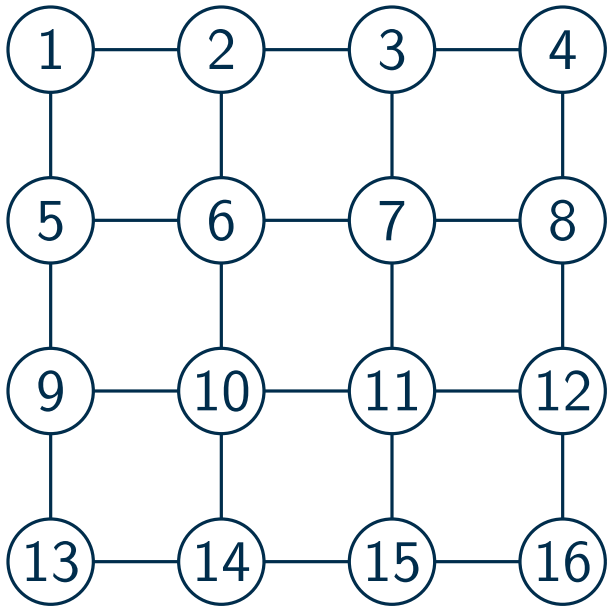
bramble: family of connected subgraphs that are pairwise adjacent

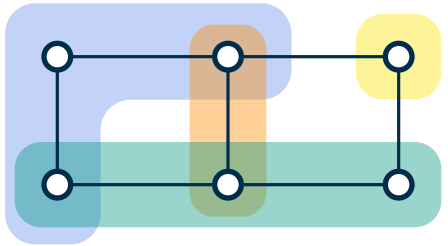
order of a bramble: size of its smallest hitting set

set S of vertices s.t. every subgraph in the bramble contains a vertex from S

Treewidth of Grids

For each $k \in \mathbb{N}$, the $k \times k$-grid has treewidth at least k.





bramble: family of connected subgraphs that are pairwise adjacent

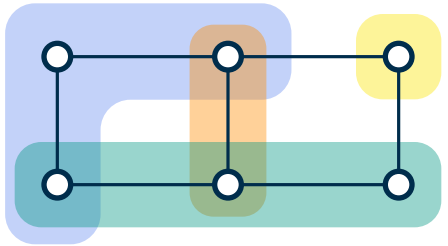
order of a bramble: size of its smallest hitting set

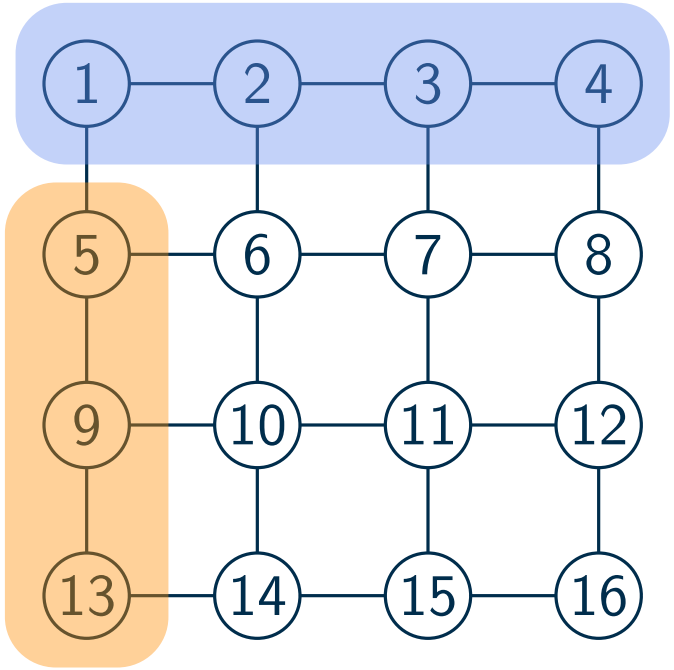
set S of vertices s.t. every subgraph in the bramble contains a vertex from S

Theorem: If a graph has a bramble of order $k+1$, the treewidth is $\geq k$.

Treewidth of Grids

For each $k \in \mathbb{N}$, the $k \times k$-grid has treewidth at least k.





bramble: family of connected subgraphs that are pairwise adjacent

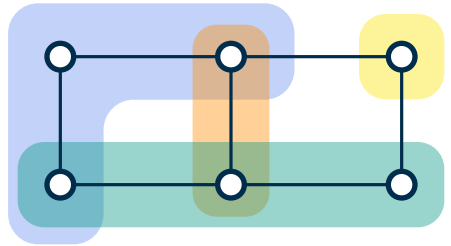
order of a bramble: size of its smallest hitting set

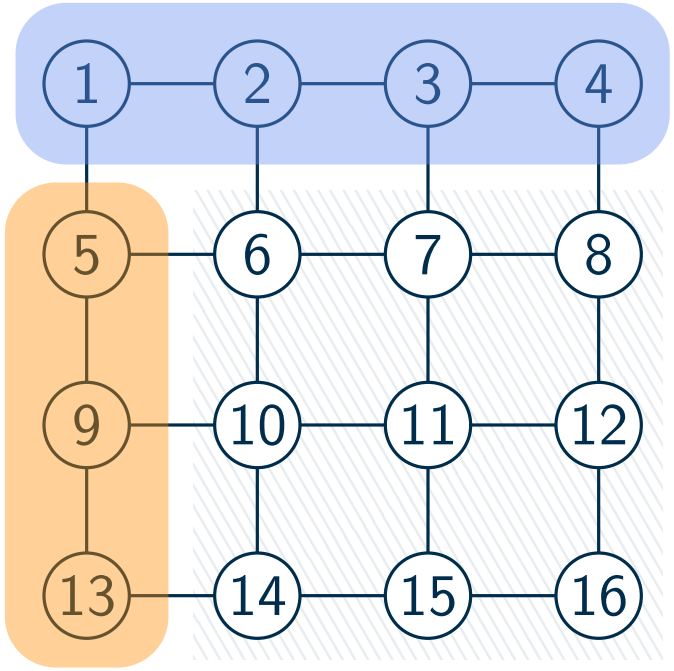
set S of vertices s.t. every subgraph in the bramble contains a vertex from S

Theorem: If a graph has a bramble of order $k+1$, the treewidth is $\geq k$.

Treewidth of Grids

For each $k \in \mathbb{N}$, the $k \times k$-grid has treewidth at least k.





all possible "crosses"

bramble: family of connected subgraphs that are pairwise adjacent

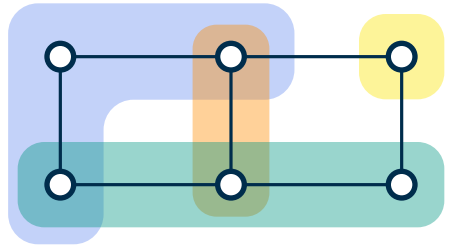
order of a bramble: size of its smallest hitting set

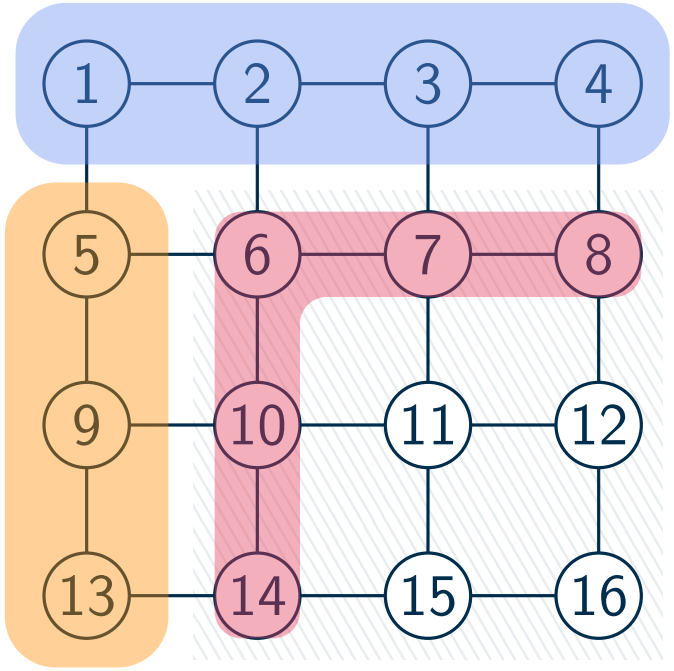
set S of vertices s.t. every subgraph in the bramble contains a vertex from S

Theorem: If a graph has a bramble of order $k+1$, the treewidth is $\geq k$.

Treewidth of Grids

For each $k \in \mathbb{N}$, the $k \times k$-grid has treewidth at least k.





all possible "crosses"

bramble: family of connected subgraphs that are pairwise adjacent

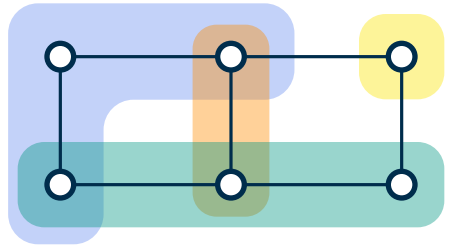
order of a bramble: size of its smallest hitting set

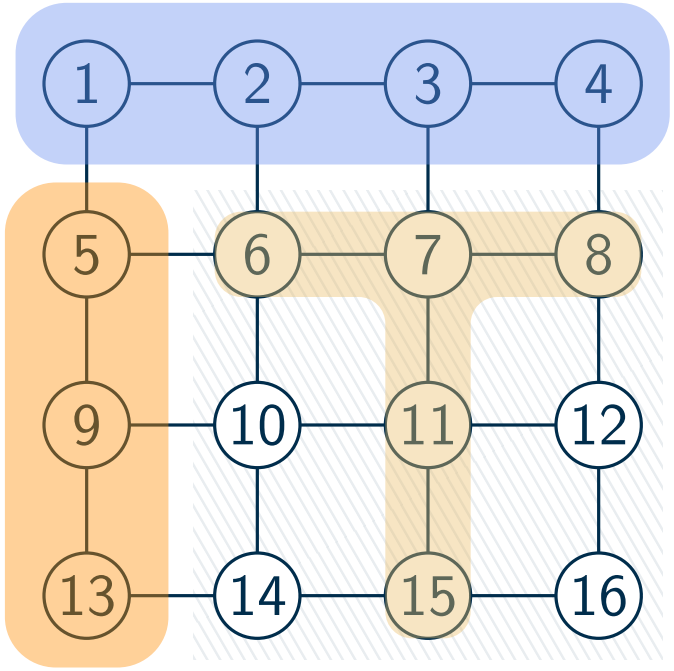
set S of vertices s.t. every subgraph in the bramble contains a vertex from S

Theorem: If a graph has a bramble of order $k+1$, the treewidth is $\geq k$.

Treewidth of Grids

For each $k \in \mathbb{N}$, the $k \times k$-grid has treewidth at least k.





all possible "crosses"

bramble: family of connected subgraphs that are pairwise adjacent

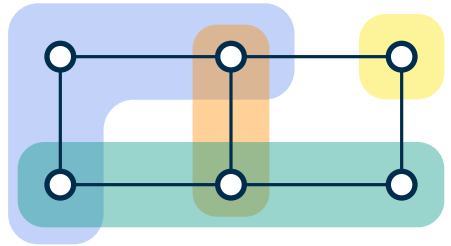
order of a bramble: size of its smallest hitting set

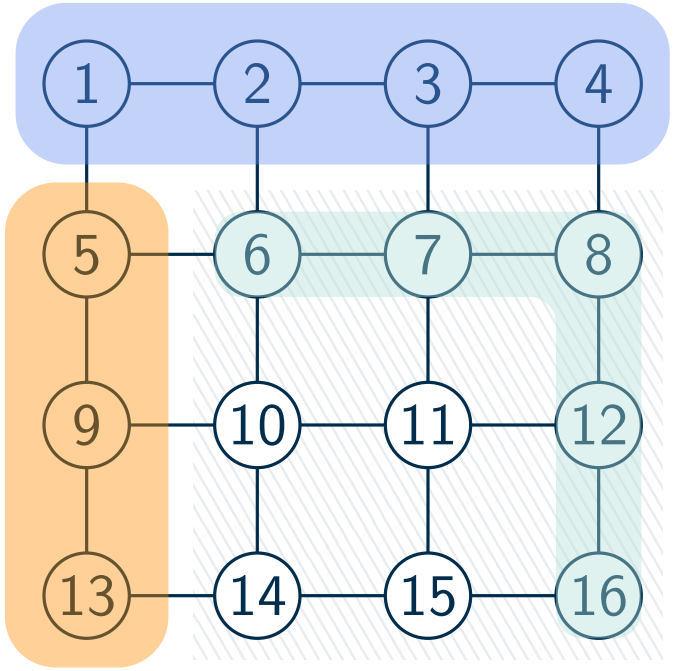
set S of vertices s.t. every subgraph in the bramble contains a vertex from S

Theorem: If a graph has a bramble of order $k+1$, the treewidth is $\geq k$.

Treewidth of Grids

For each $k \in \mathbb{N}$, the $k \times k$-grid has treewidth at least k.





all possible "crosses"

bramble: family of connected subgraphs that are pairwise adjacent

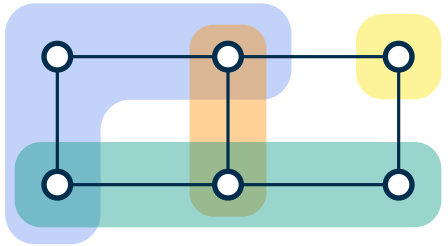
order of a bramble: size of its smallest hitting set

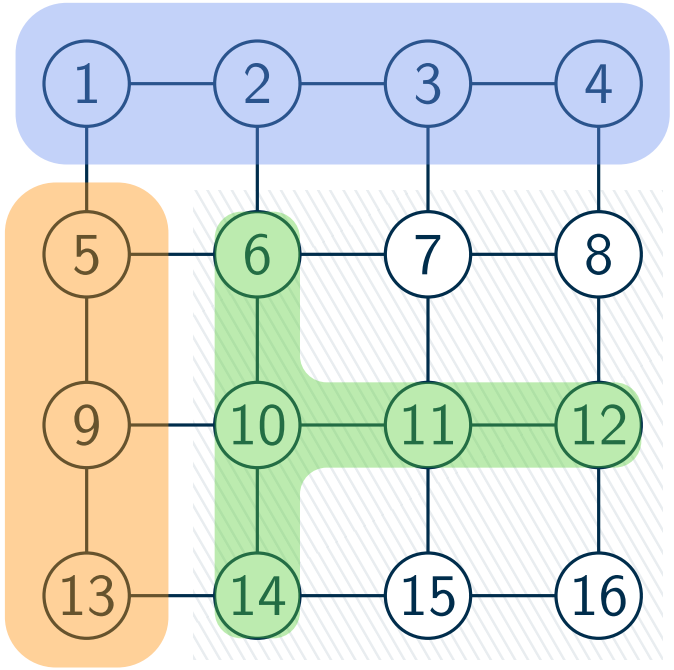
set S of vertices s.t. every subgraph in the bramble contains a vertex from S

Theorem: If a graph has a bramble of order $k+1$, the treewidth is $\geq k$.

Treewidth of Grids

For each $k \in \mathbb{N}$, the $k \times k$-grid has treewidth at least k.





all possible "crosses"

bramble: family of connected subgraphs that are pairwise adjacent

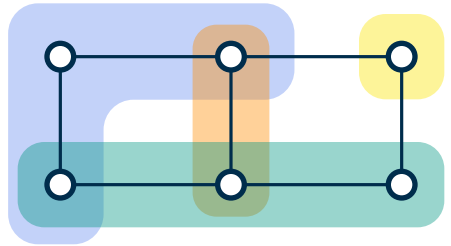
order of a bramble: size of its smallest hitting set

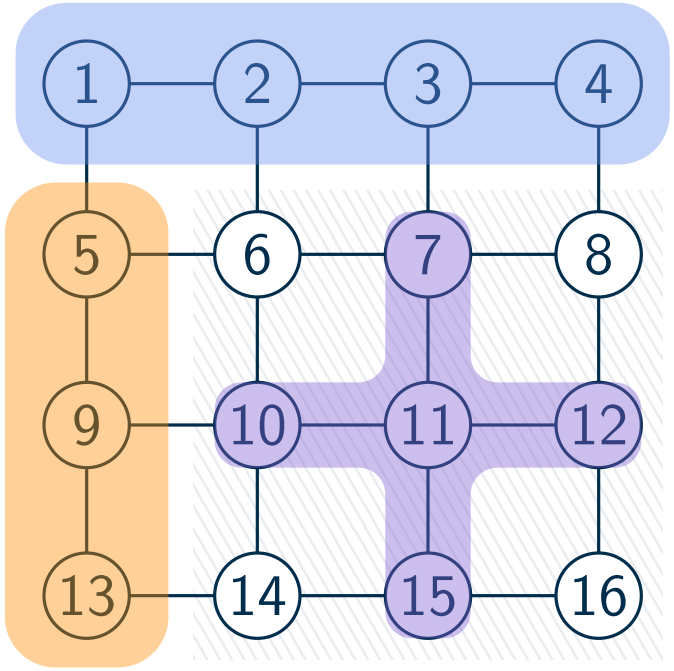
set S of vertices s.t. every subgraph in the bramble contains a vertex from S

Theorem: If a graph has a bramble of order $k+1$, the treewidth is $\geq k$.

Treewidth of Grids

For each $k \in \mathbb{N}$, the $k \times k$-grid has treewidth at least k.





all possible "crosses"

bramble: family of connected subgraphs that are pairwise adjacent

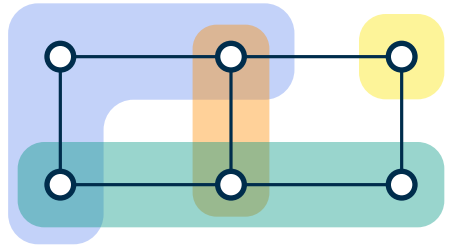
order of a bramble: size of its smallest hitting set

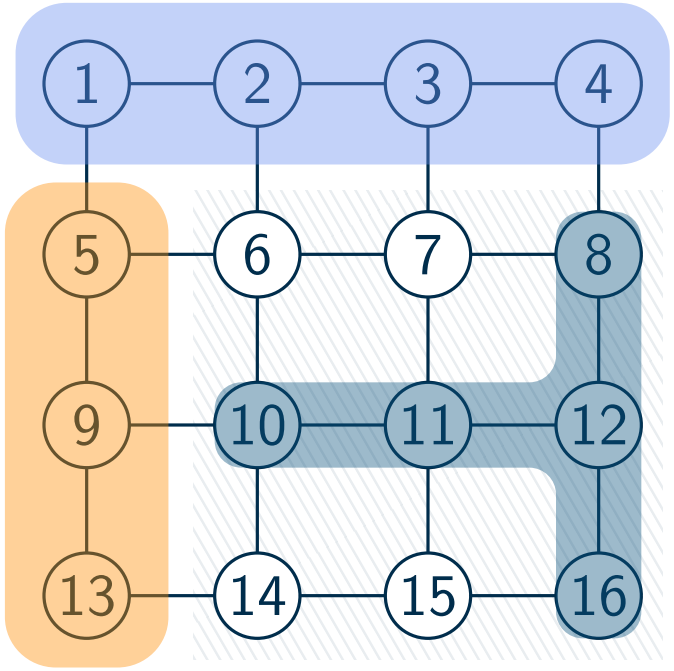
set S of vertices s.t. every subgraph in the bramble contains a vertex from S

Theorem: If a graph has a bramble of order $k+1$, the treewidth is $\geq k$.

Treewidth of Grids

For each $k \in \mathbb{N}$, the $k \times k$-grid has treewidth at least k.





all possible "crosses"

bramble: family of connected subgraphs that are pairwise adjacent

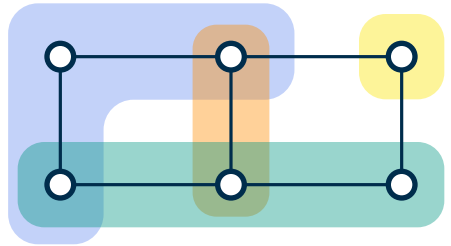
order of a bramble: size of its smallest hitting set

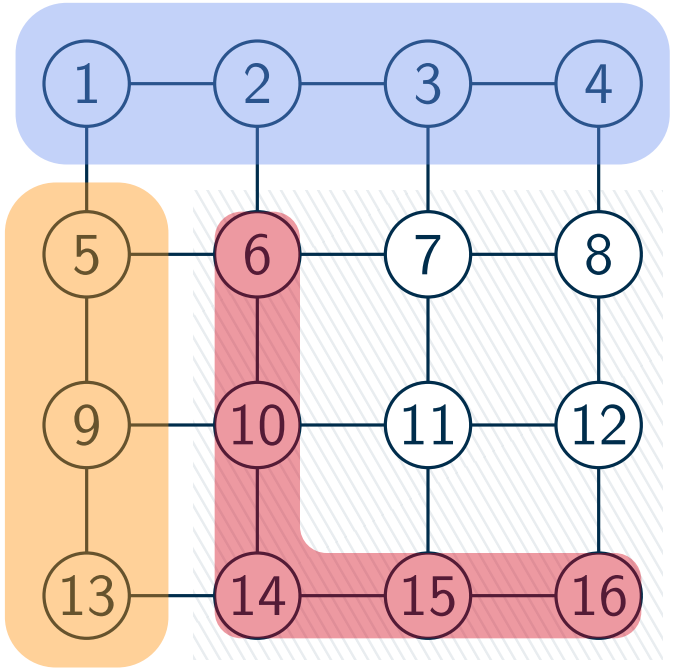
set S of vertices s.t. every subgraph in the bramble contains a vertex from S

Theorem: If a graph has a bramble of order $k+1$, the treewidth is $\geq k$.

Treewidth of Grids

For each $k \in \mathbb{N}$, the $k \times k$-grid has treewidth at least k.





all possible "crosses"

bramble: family of connected subgraphs that are pairwise adjacent

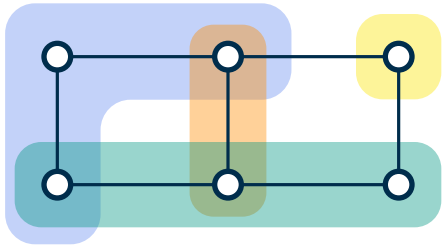
order of a bramble: size of its smallest hitting set

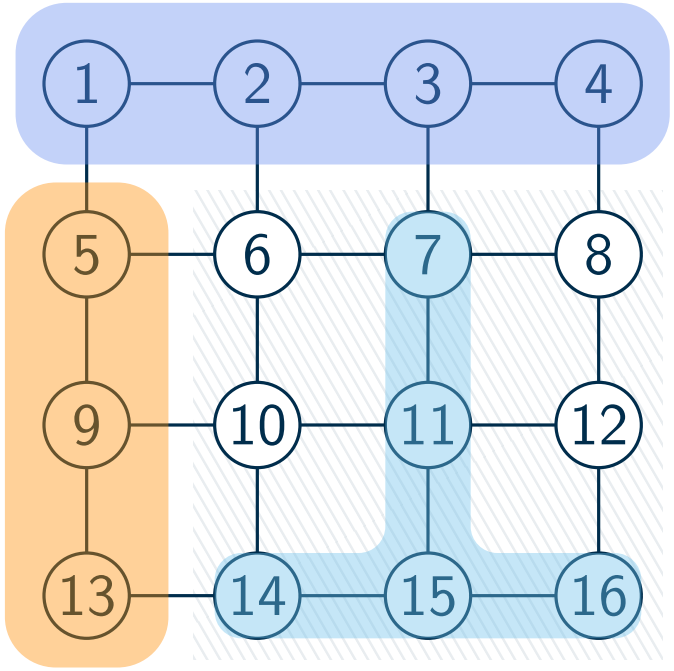
set S of vertices s.t. every subgraph in the bramble contains a vertex from S

Theorem: If a graph has a bramble of order $k+1$, the treewidth is $\geq k$.

Treewidth of Grids

For each $k \in \mathbb{N}$, the $k \times k$-grid has treewidth at least k.





all possible "crosses"

bramble: family of connected subgraphs that are pairwise adjacent

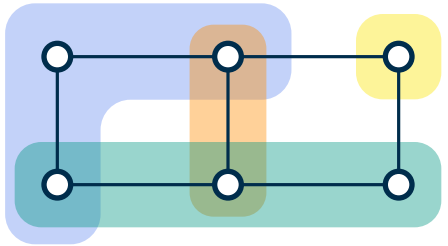
order of a bramble: size of its smallest hitting set

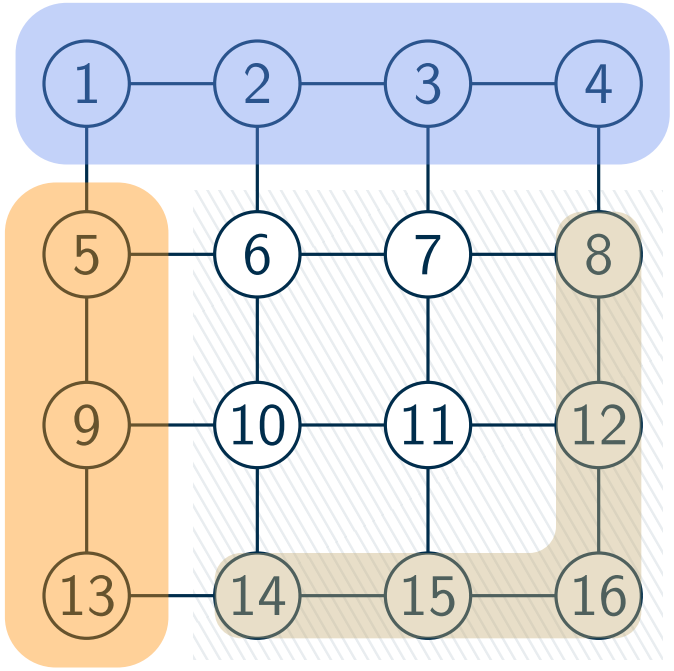
set S of vertices s.t. every subgraph in the bramble contains a vertex from S

Theorem: If a graph has a bramble of order $k+1$, the treewidth is $\geq k$.

Treewidth of Grids

For each $k \in \mathbb{N}$, the $k \times k$-grid has treewidth at least k.





all possible "crosses"

bramble: family of connected subgraphs that are pairwise adjacent

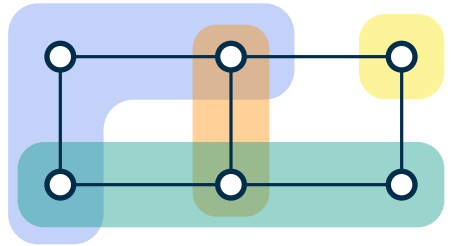
order of a bramble: size of its smallest hitting set

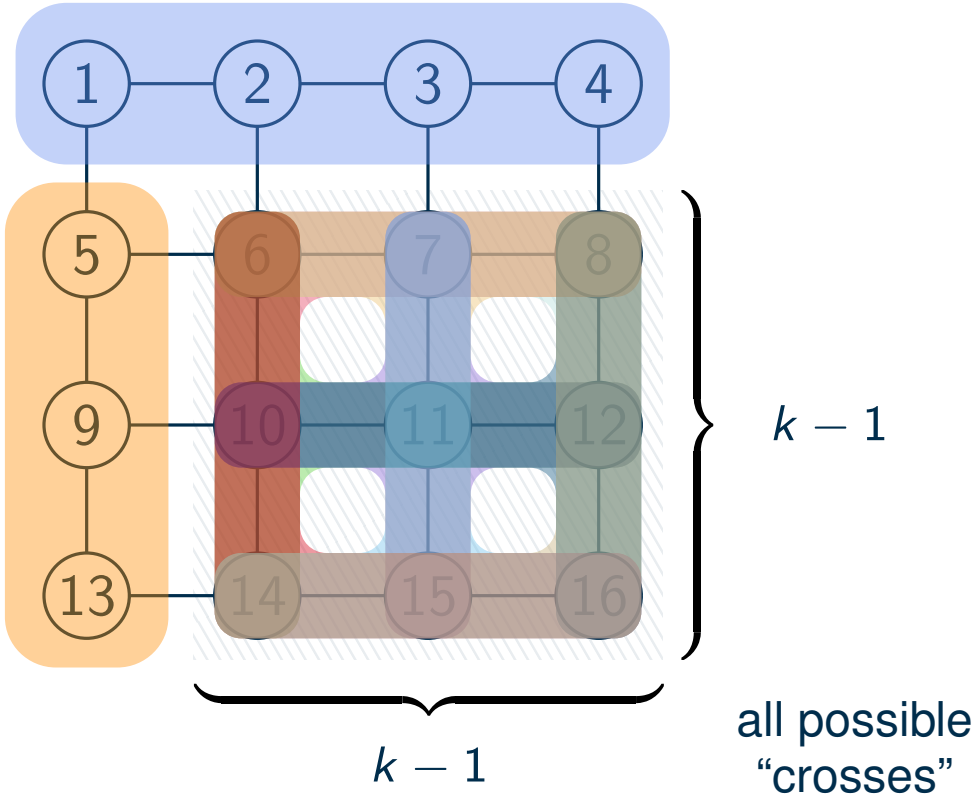
set S of vertices s.t. every subgraph in the bramble contains a vertex from S

Theorem: If a graph has a bramble of order $k+1$, the treewidth is $\geq k$.

Treewidth of Grids

For each $k \in \mathbb{N}$, the $k \times k$-grid has treewidth at least k.





all possible "crosses"

bramble: family of connected subgraphs that are pairwise adjacent

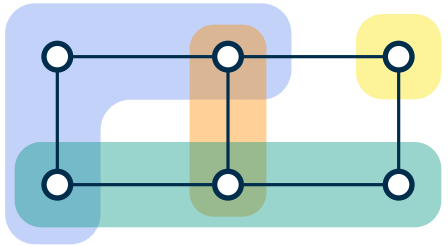
order of a bramble: size of its smallest hitting set

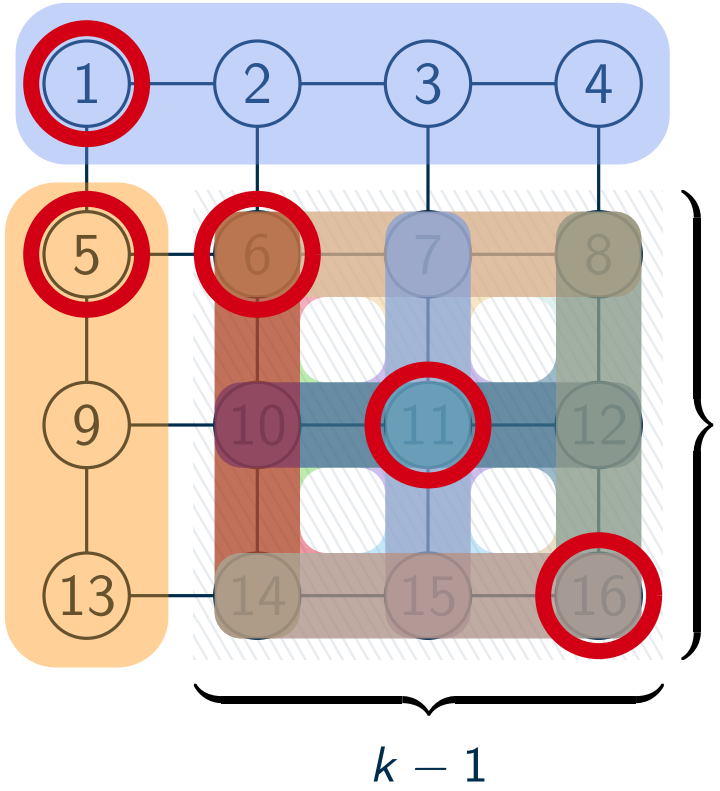
set S of vertices s.t. every subgraph in the bramble contains a vertex from S

Theorem: If a graph has a bramble of order $k+1$, the treewidth is $\geq k$.

Treewidth of Grids

For each $k \in \mathbb{N}$, the $k \times k$-grid has treewidth at least k.





bramble: family of connected subgraphs that are pairwise adjacent

order of a bramble: size of its smallest hitting set

set S of vertices s.t. every subgraph in the bramble contains a vertex from S

Theorem: If a graph has a bramble of order $k+1$, the treewidth is $\geq k$.

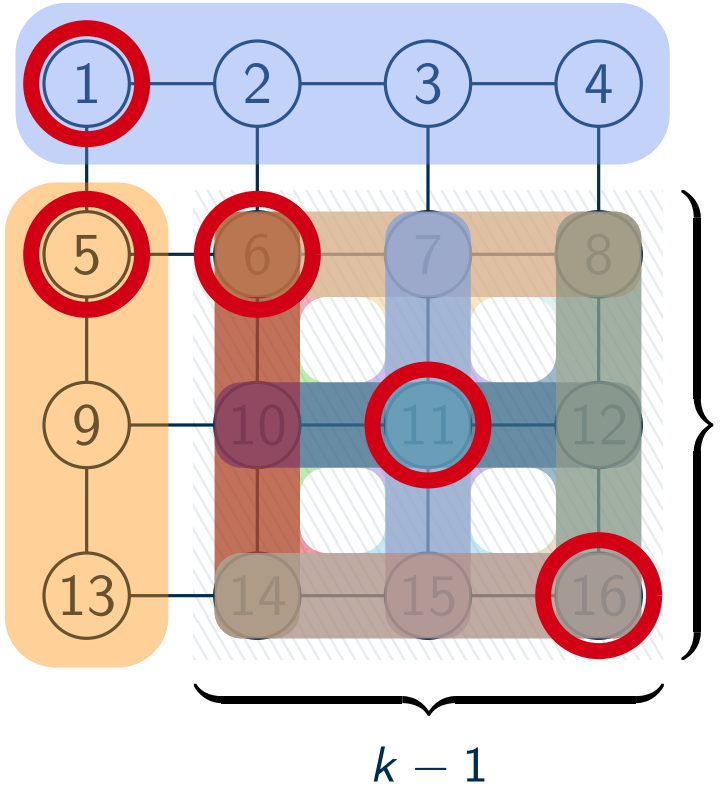
Treewidth of Grids

Further reading:

Treewidth Lower Bounds with Brambles (Bodlaender, Grigoriev, Koster)
https://link.springer.com/article/10.1007/s00453-007-9056-z

Graph Theory (Diestel) – c.f. Theorem 12.4.3 (In Fifth Edition)
https://daiwz.net/course/disc_math/2023/Diestel_Graph_Theory.pdf

For each $k \in \mathbb{N}$, the $k \times k$-grid has treewidth at least k.



bramble: family of connected subgraphs that are pairwise adjacent

order of a bramble: size of its smallest hitting set

set S of vertices s.t. every subgraph in the bramble contains a vertex from S

Theorem: If a graph has a bramble of order $k+1$, the treewidth is $\geq k$.

Every Clique Lives in a Bag

For each clique C in a graph G, every tree decomposition of G has a bag X_i with $C \subseteq X_i$

Every Clique Lives in a Bag

For each clique C in a graph G, every tree decomposition of G has a bag X_i with $C \subseteq X_i$

Proof: Induction over $|C|$

Every Clique Lives in a Bag

For each clique C in a graph G, every tree decomposition of G has a bag X_i with $C \subseteq X_i$

Proof: Induction over $|C|$

- $|C| = 2$: C is an edge, so claim follows from definition of tree decomposition

Every Clique Lives in a Bag

For each clique C in a graph G, every tree decomposition of G has a bag X_i with $C \subseteq X_i$

Proof: Induction over $|C|$

- $|C| = 2$: C is an edge, so claim follows from definition of tree decomposition
- assume we find a bag for all cliques smaller than $|C|$

Every Clique Lives in a Bag

For each clique C in a graph G, every tree decomposition of G has a bag X_i with $C \subseteq X_i$

Proof: Induction over $|C|$

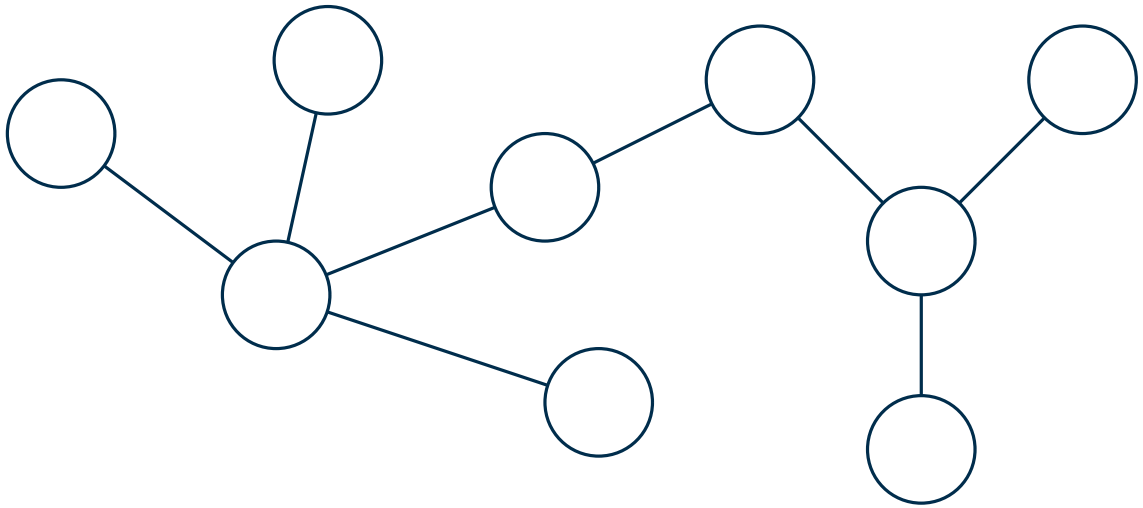
- $|C| = 2$: C is an edge, so claim follows from definition of tree decomposition
- assume we find a bag for all cliques smaller than $|C|$
- let $C' = C \setminus \{v\}$ with $v \in C$

Every Clique Lives in a Bag

For each clique C in a graph G, every tree decomposition of G has a bag X_i with $C \subseteq X_i$

Proof: Induction over $|C|$

- $|C| = 2$: C is an edge, so claim follows from definition of tree decomposition
- assume we find a bag for all cliques smaller than $|C|$
- let $C' = C \setminus \{v\}$ with $v \in C$



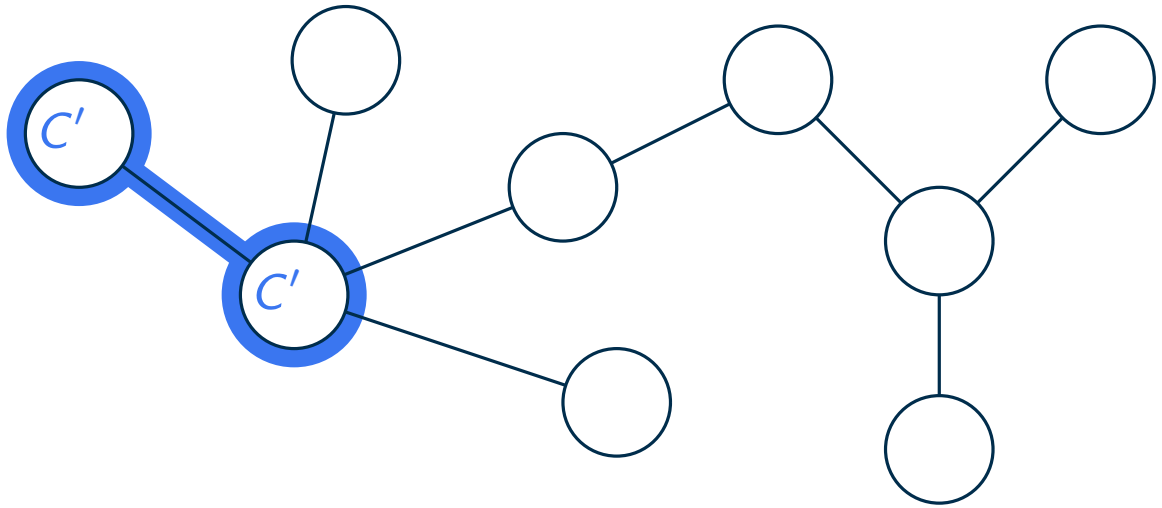
tree decomposition

Every Clique Lives in a Bag

For each clique C in a graph G, every tree decomposition of G has a bag X_i with $C \subseteq X_i$

Proof: Induction over $|C|$

- $|C| = 2$: C is an edge, so claim follows from definition of tree decomposition
- assume we find a bag for all cliques smaller than $|C|$
- let $C' = C \setminus \{v\}$ with $v \in C$
- consider subtree $T_{C'}$ of bags that contain C'



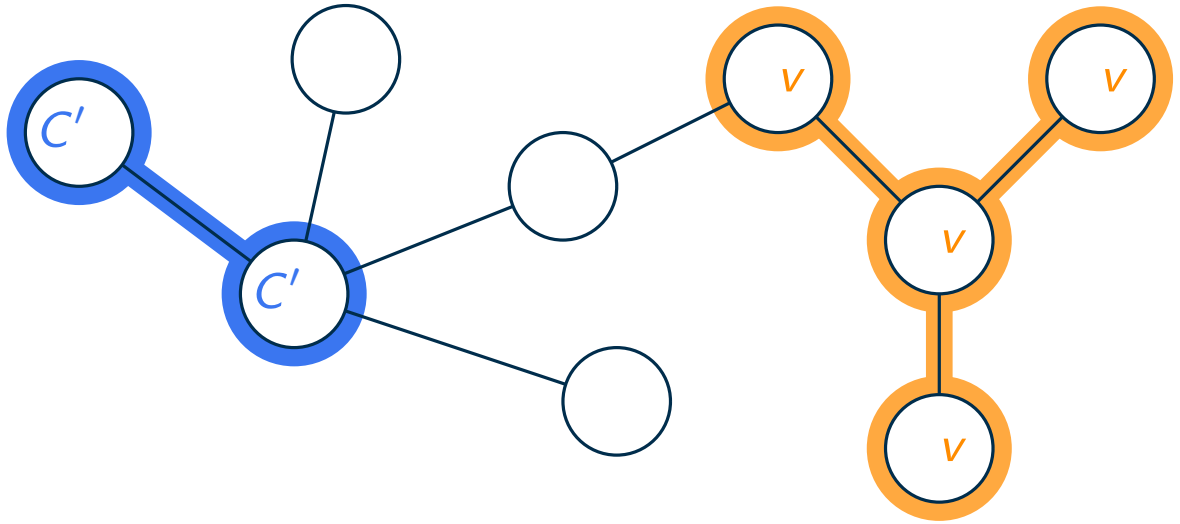
tree decomposition

Every Clique Lives in a Bag

For each clique C in a graph G, every tree decomposition of G has a bag X_i with $C \subseteq X_i$

Proof: Induction over $|C|$

- $|C| = 2$: C is an edge, so claim follows from definition of tree decomposition
- assume we find a bag for all cliques smaller than $|C|$
- let $C' = C \setminus \{v\}$ with $v \in C$
- consider subtree $T_{C'}$ of bags that contain C' and subtree T_v of bags that contain v



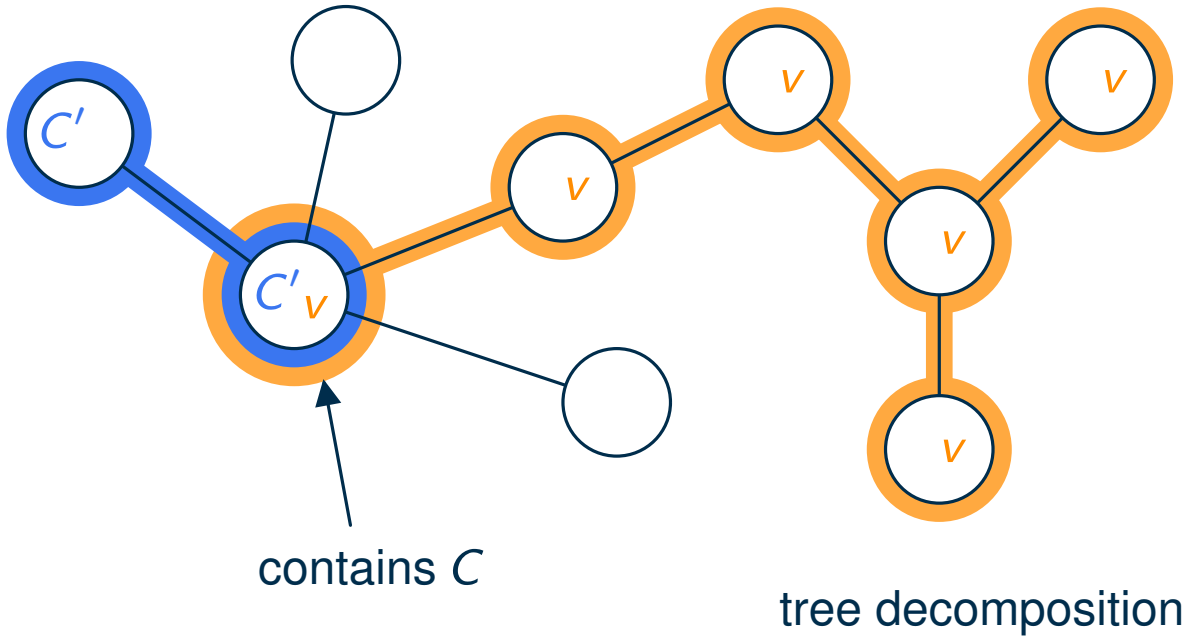
tree decomposition

Every Clique Lives in a Bag

For each clique C in a graph G, every tree decomposition of G has a bag X_i with $C \subseteq X_i$

Proof: Induction over $|C|$

- $|C| = 2$: C is an edge, so claim follows from definition of tree decomposition
- assume we find a bag for all cliques smaller than $|C|$
- let $C' = C \setminus \{v\}$ with $v \in C$
- consider subtree $T_{C'}$ of bags that contain C' and subtree T_v of bags that contain v
- if $T_{C'} \cap T_v \neq \emptyset$: done ✓

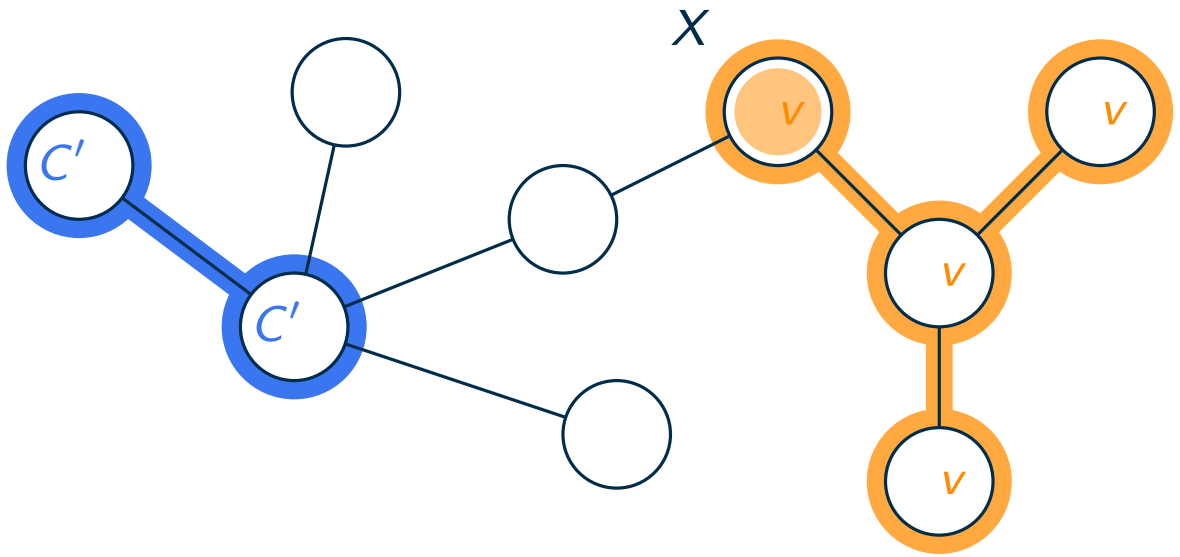


Every Clique Lives in a Bag

For each clique C in a graph G, every tree decomposition of G has a bag X_i with $C \subseteq X_i$

Proof: Induction over $|C|$

- $|C| = 2$: C is an edge, so claim follows from definition of tree decomposition
- assume we find a bag for all cliques smaller than $|C|$
- let $C' = C \setminus \{v\}$ with $v \in C$
- consider subtree $T_{C'}$ of bags that contain C' and subtree T_v of bags that contain v
- if $T_{C'} \cap T_v \neq \emptyset$: done ✓
- else: consider the bag X of T_v that is closest to $T_{C'}$



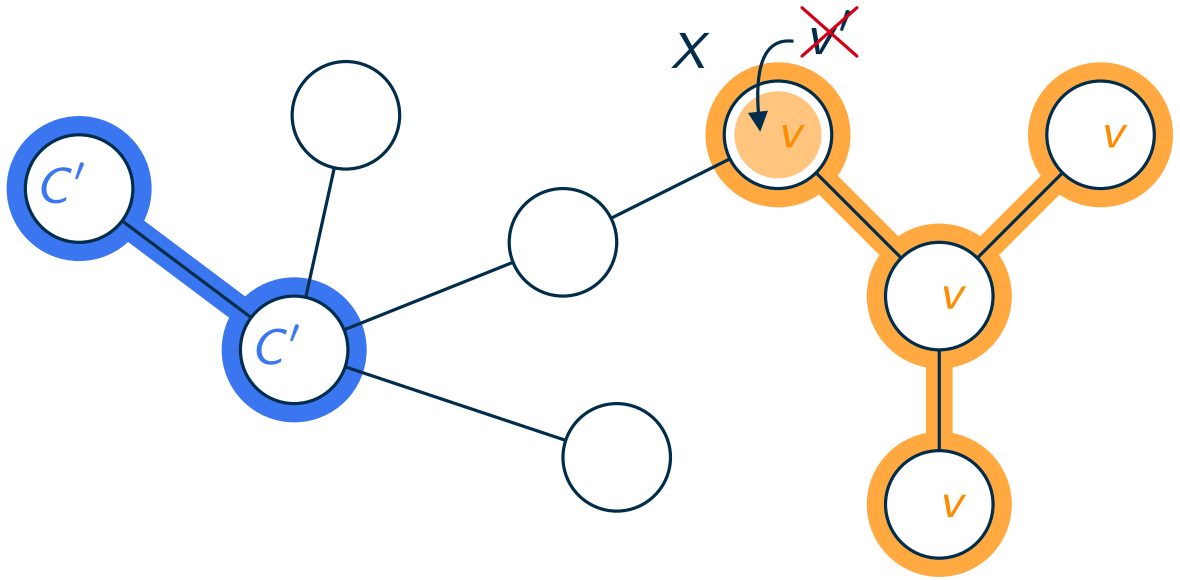
tree decomposition

Every Clique Lives in a Bag

For each clique C in a graph G, every tree decomposition of G has a bag X_i with $C \subseteq X_i$

Proof: Induction over $|C|$

- $|C| = 2$: C is an edge, so claim follows from definition of tree decomposition
- assume we find a bag for all cliques smaller than $|C|$
- let $C' = C \setminus \{v\}$ with $v \in C$
- consider subtree $T_{C'}$ of bags that contain C' and subtree T_v of bags that contain v
- if $T_{C'} \cap T_v \neq \emptyset$: done ✓
- else: consider the bag X of T_v that is closest to $T_{C'}$
- there is vertex $v' \in C'$ that is not contained in X



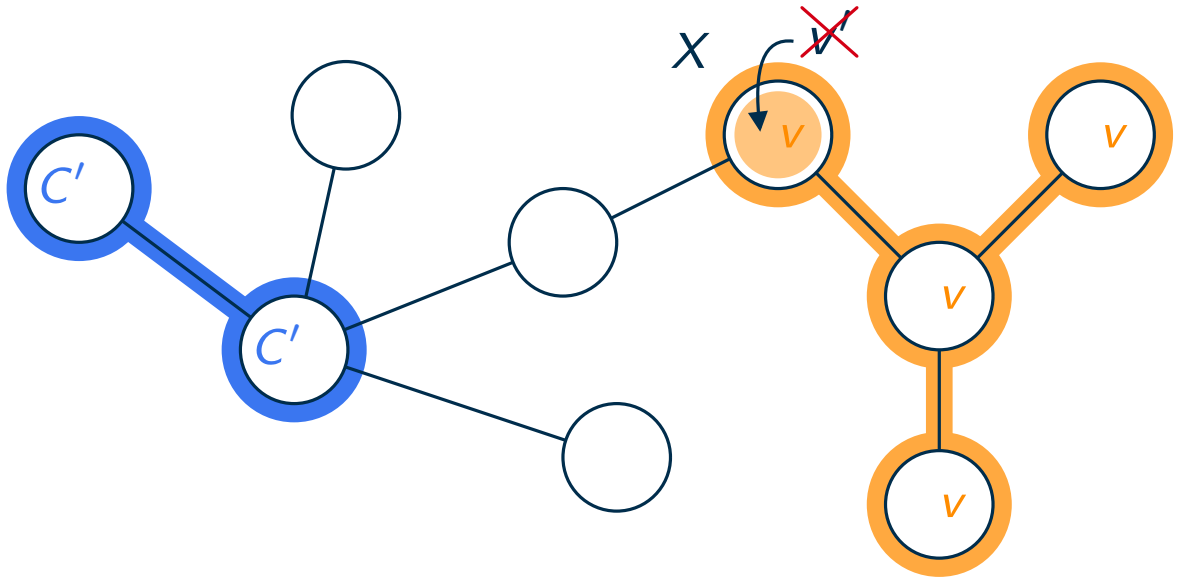
tree decomposition

Every Clique Lives in a Bag

For each clique C in a graph G, every tree decomposition of G has a bag X_i with $C \subseteq X_i$

Proof: Induction over $|C|$

- $|C| = 2$: C is an edge, so claim follows from definition of tree decomposition
- assume we find a bag for all cliques smaller than $|C|$
- let $C' = C \setminus \{v\}$ with $v \in C$
- consider subtree $T_{C'}$ of bags that contain C' and subtree T_v of bags that contain v
- if $T_{C'} \cap T_v \neq \emptyset$: done ✓
- else: consider the bag X of T_v that is closest to $T_{C'}$
- there is vertex $v' \in C'$ that is not contained in X
  $\Longrightarrow$ v' is not contained in T_v since $T_{v'}$ is connected



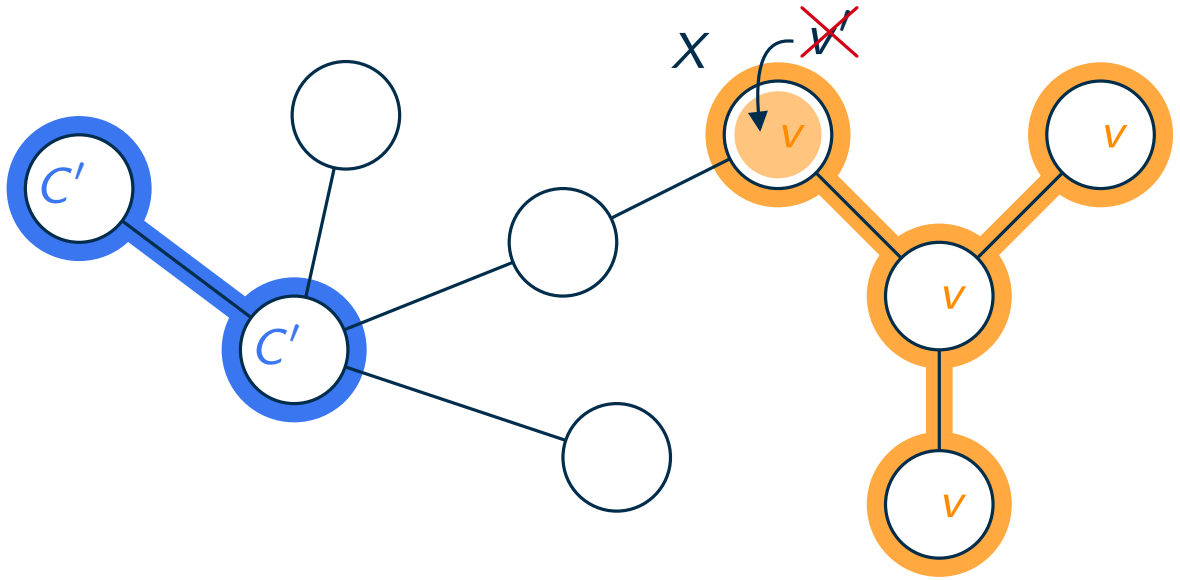
tree decomposition

Every Clique Lives in a Bag

For each clique C in a graph G, every tree decomposition of G has a bag X_i with $C \subseteq X_i$

Proof: Induction over $|C|$

- $|C| = 2$: C is an edge, so claim follows from definition of tree decomposition
- assume we find a bag for all cliques smaller than $|C|$
- let $C' = C \setminus \{v\}$ with $v \in C$
- consider subtree $T_{C'}$ of bags that contain C' and subtree T_v of bags that contain v
- if $T_{C'} \cap T_v \neq \emptyset$: done ✓
- else: consider the bag X of T_v that is closest to $T_{C'}$
- there is vertex $v' \in C'$ that is not contained in X
  $\Longrightarrow$ v' is not contained in T_v since $T_{v'}$ is connected
- edge v, v' is not contained together in any bag!



tree decomposition

Cops and Robber

In a connected graph with treewidth $\leq k$, there is a winning strategy for $k+1$ cops.

Start: cops choose starting vertices, then robber chooses vertex

In each round:
- one cop announces new position $v \in V$
- robber may move arbitrarily far, but not through cops
- cop moves to position v

Game Over: robber loses if colocated with cop at round end

Cops and Robber

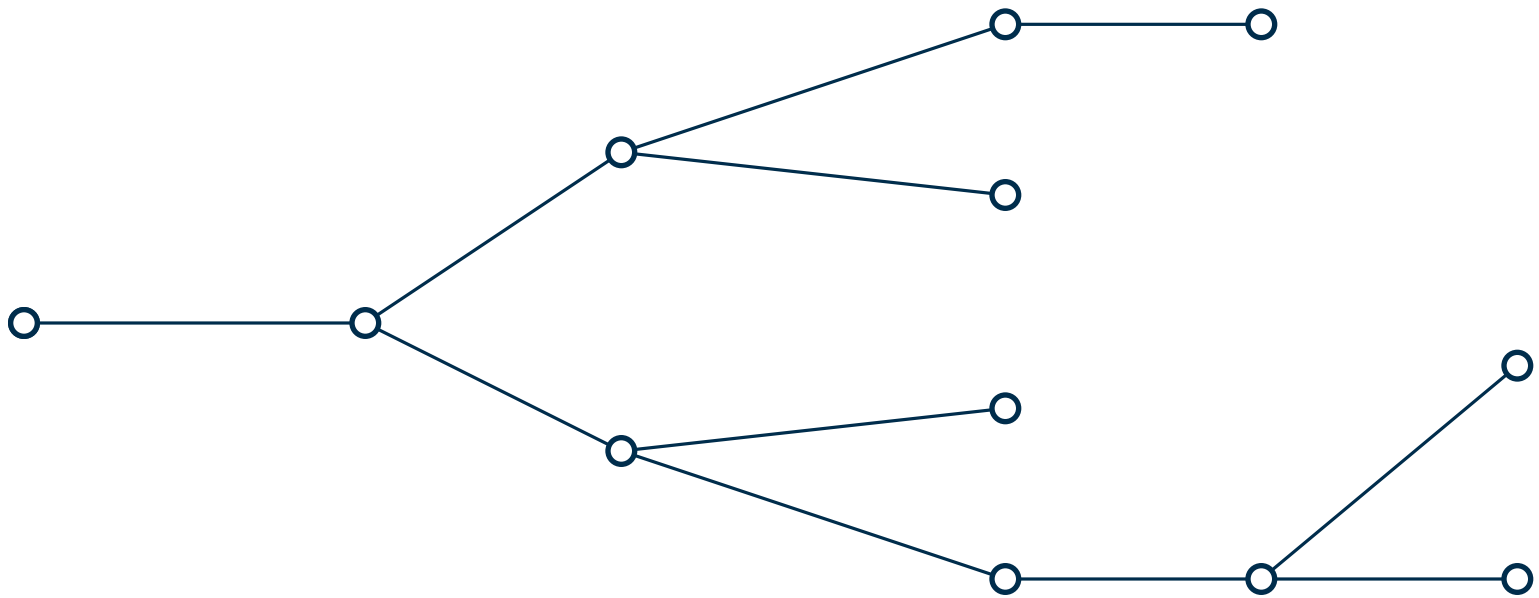
In a connected graph with treewidth $\leq k$, there is a winning strategy for $k+1$ cops.

Start: cops choose starting vertices, then robber chooses vertex

In each round:
- one cop announces new position $v \in V$
- robber may move arbitrarily far, but not through cops
- cop moves to position v

Game Over: robber loses if colocated with cop at round end

$\rightarrow$ on trees: two cops can win



Cops and Robber

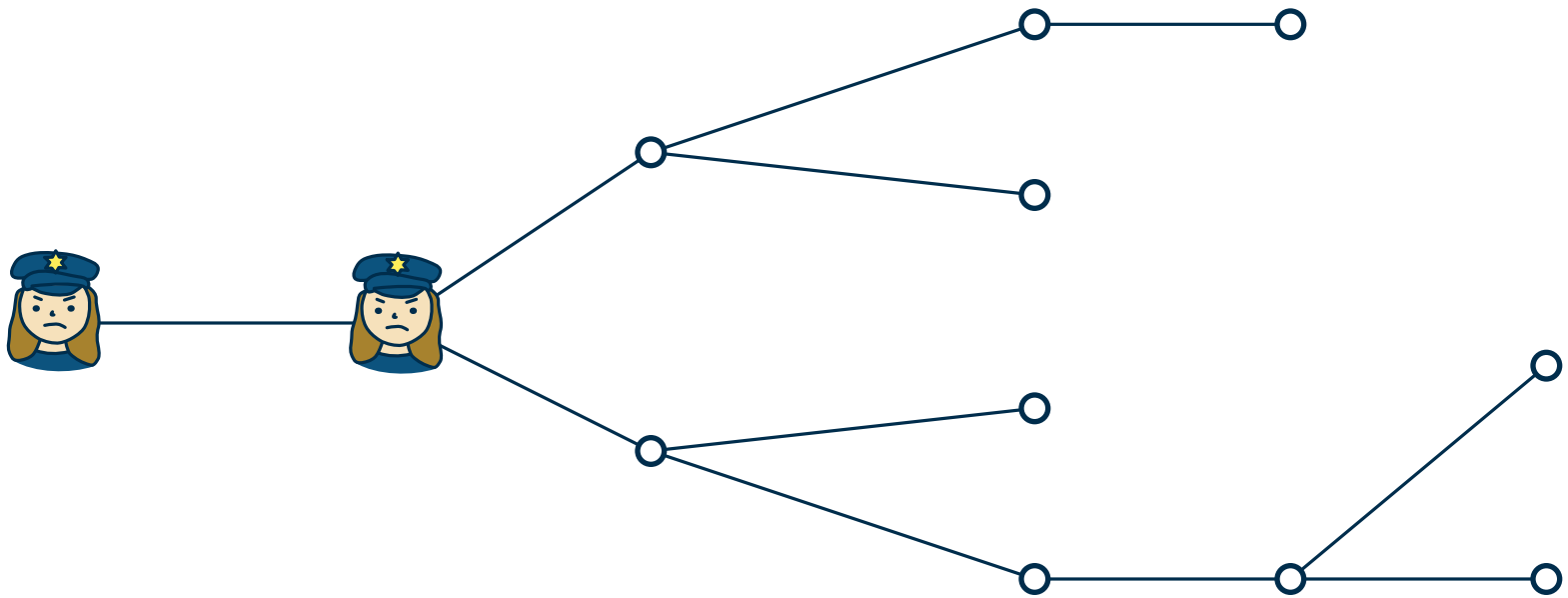
In a connected graph with treewidth $\leq k$, there is a winning strategy for $k+1$ cops.

Start: cops choose starting vertices, then robber chooses vertex

In each round:
- one cop announces new position $v \in V$
- robber may move arbitrarily far, but not through cops
- cop moves to position v

Game Over: robber loses if colocated with cop at round end

$\rightarrow$ on trees: two cops can win



Cops and Robber

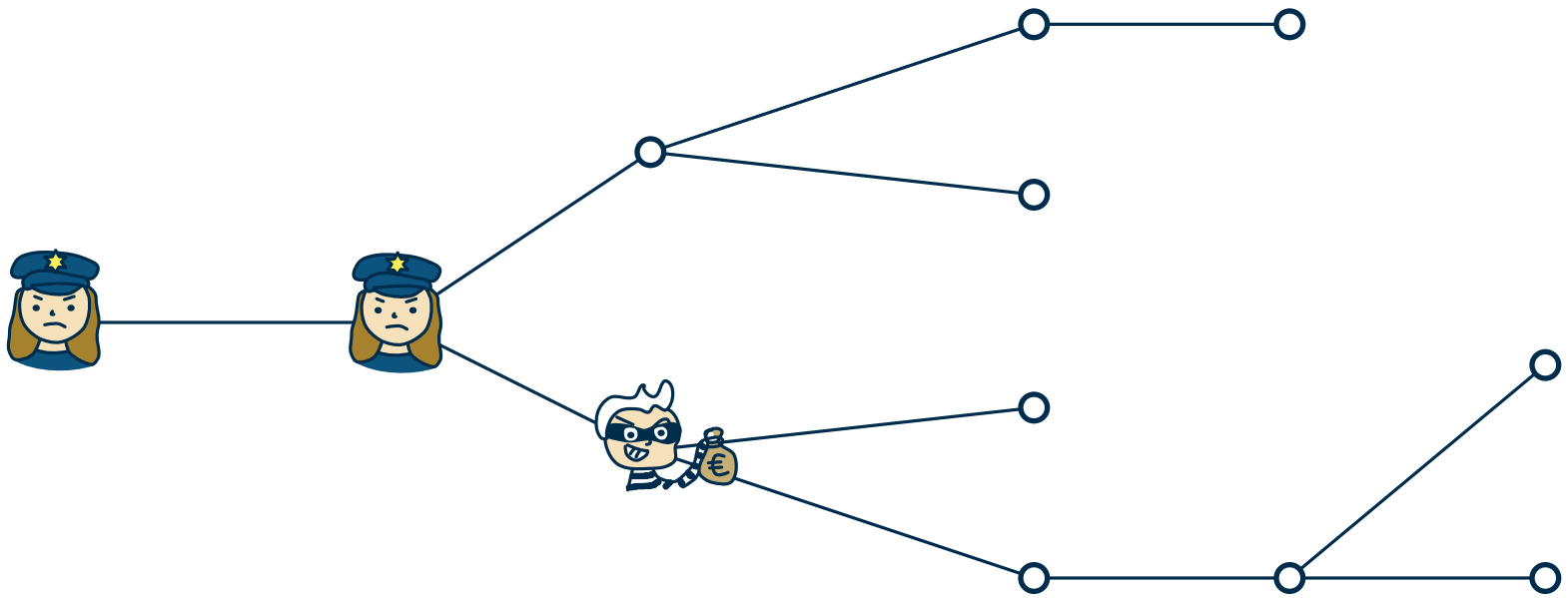
In a connected graph with treewidth $\leq k$, there is a winning strategy for $k+1$ cops.

Start: cops choose starting vertices, then robber chooses vertex

In each round:
- one cop announces new position $v \in V$
- robber may move arbitrarily far, but not through cops
- cop moves to position v

Game Over: robber loses if colocated with cop at round end

$\rightarrow$ on trees: two cops can win



Cops and Robber

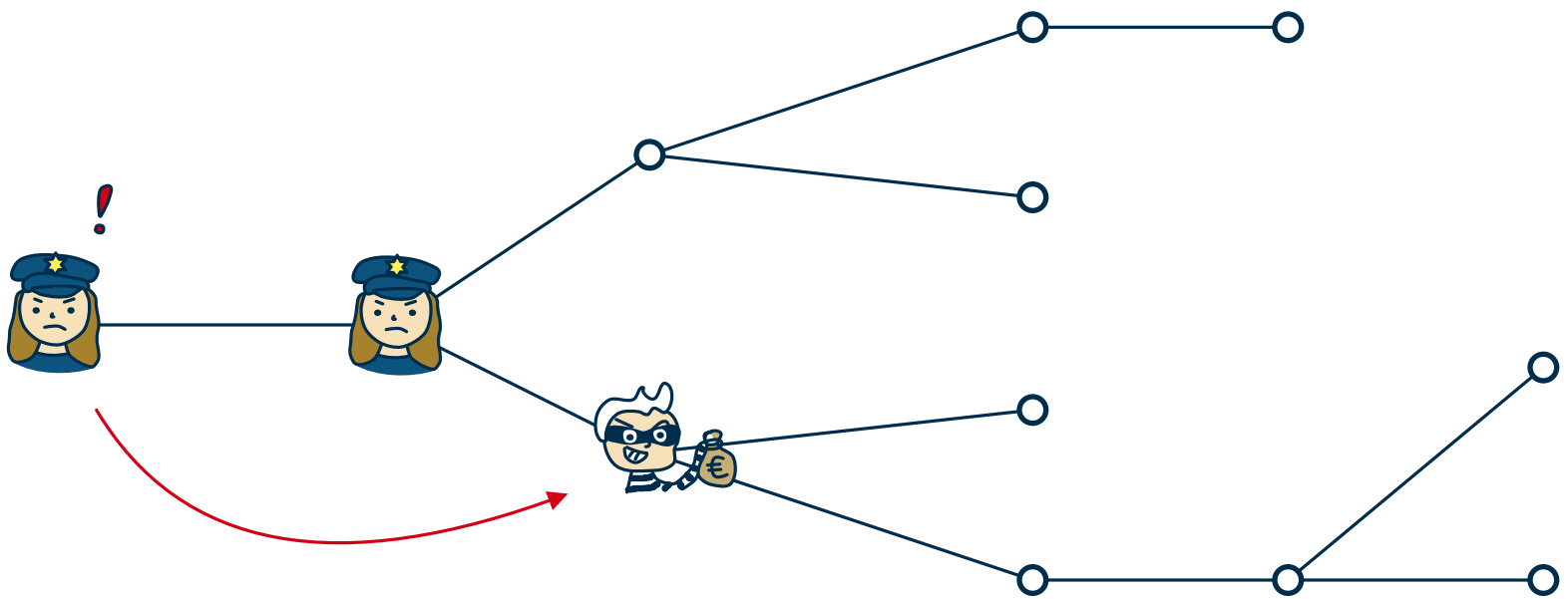
In a connected graph with treewidth $\leq k$, there is a winning strategy for $k+1$ cops.

Start: cops choose starting vertices, then robber chooses vertex

In each round:
- one cop announces new position $v \in V$
- robber may move arbitrarily far, but not through cops
- cop moves to position v

Game Over: robber loses if colocated with cop at round end

$\rightarrow$ on trees: two cops can win



Cops and Robber

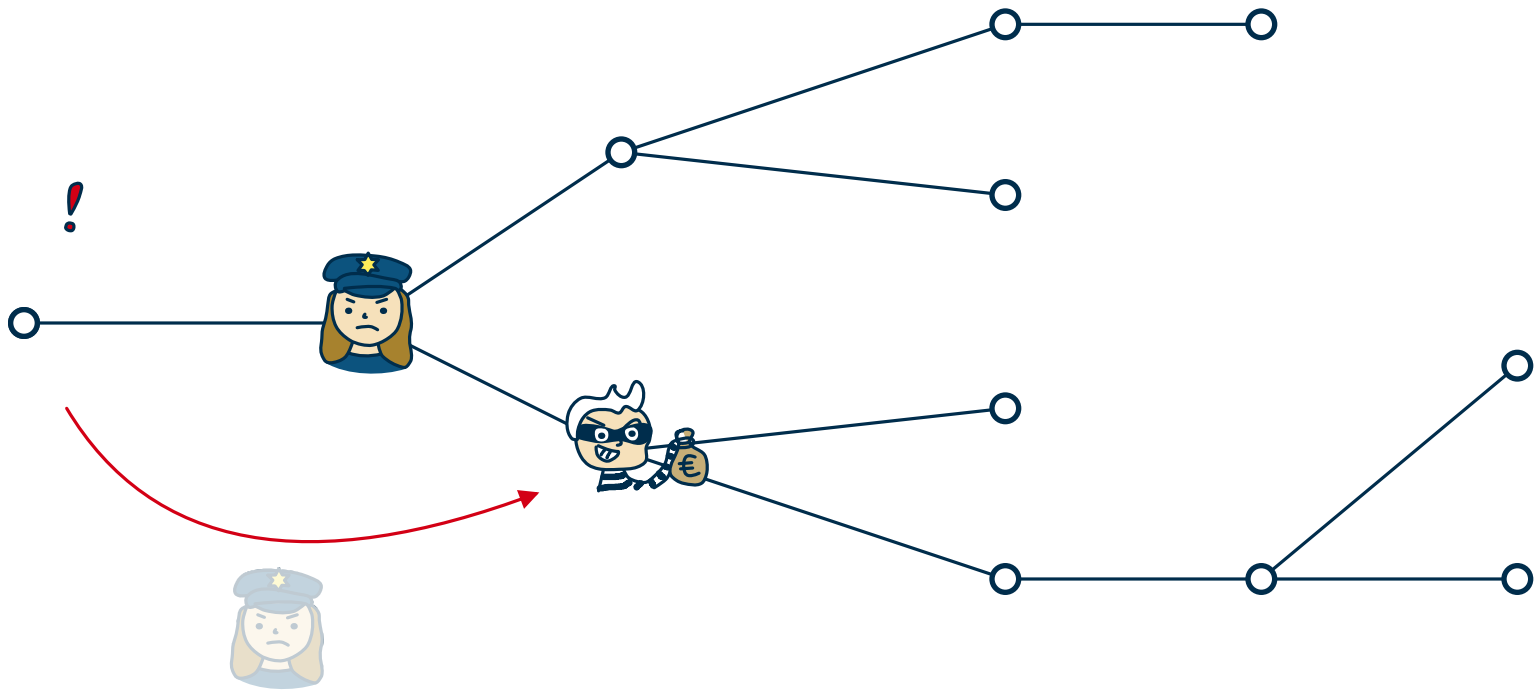
In a connected graph with treewidth $\leq k$, there is a winning strategy for $k+1$ cops.

Start: cops choose starting vertices, then robber chooses vertex

In each round:
- one cop announces new position $v \in V$
- robber may move arbitrarily far, but not through cops
- cop moves to position v

Game Over: robber loses if colocated with cop at round end

$\rightarrow$ on trees: two cops can win



Cops and Robber

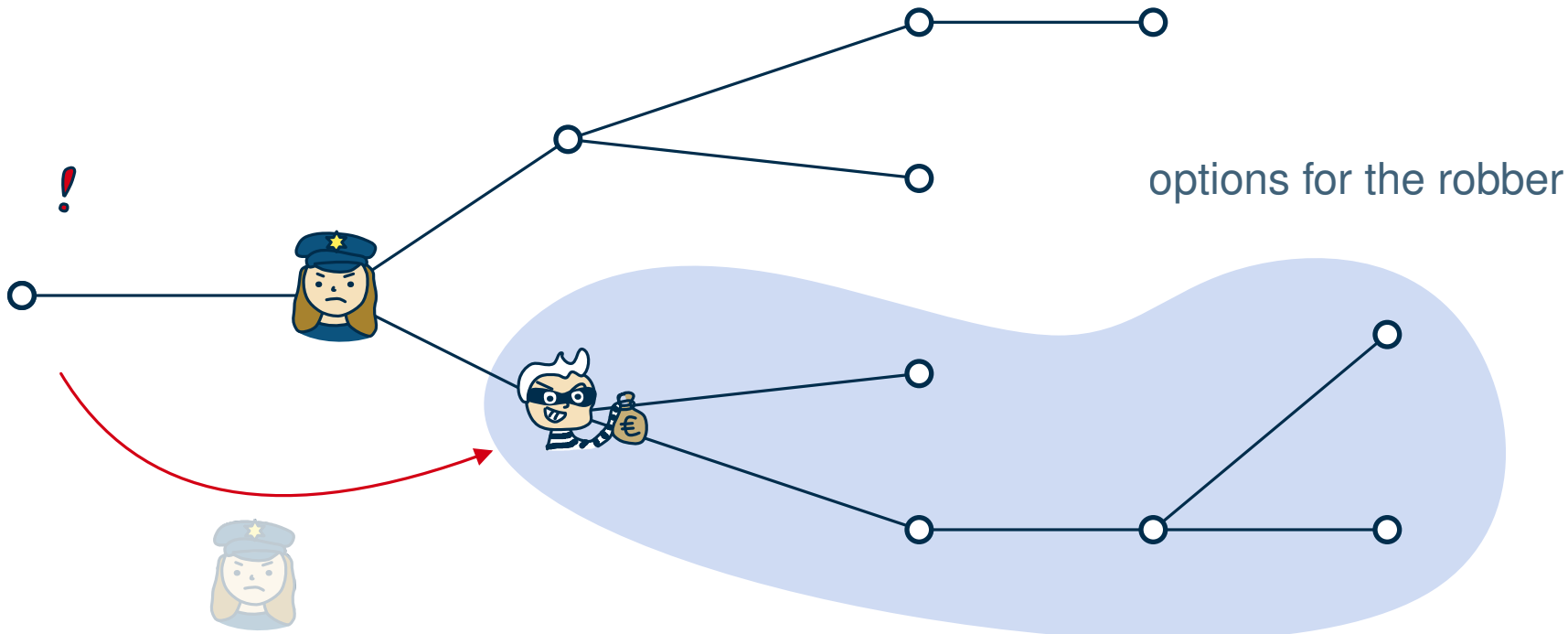
In a connected graph with treewidth $\leq k$, there is a winning strategy for $k+1$ cops.

Start: cops choose starting vertices, then robber chooses vertex

In each round:
- one cop announces new position $v \in V$
- robber may move arbitrarily far, but not through cops
- cop moves to position v

Game Over: robber loses if colocated with cop at round end

$\rightarrow$ on trees: two cops can win



Cops and Robber

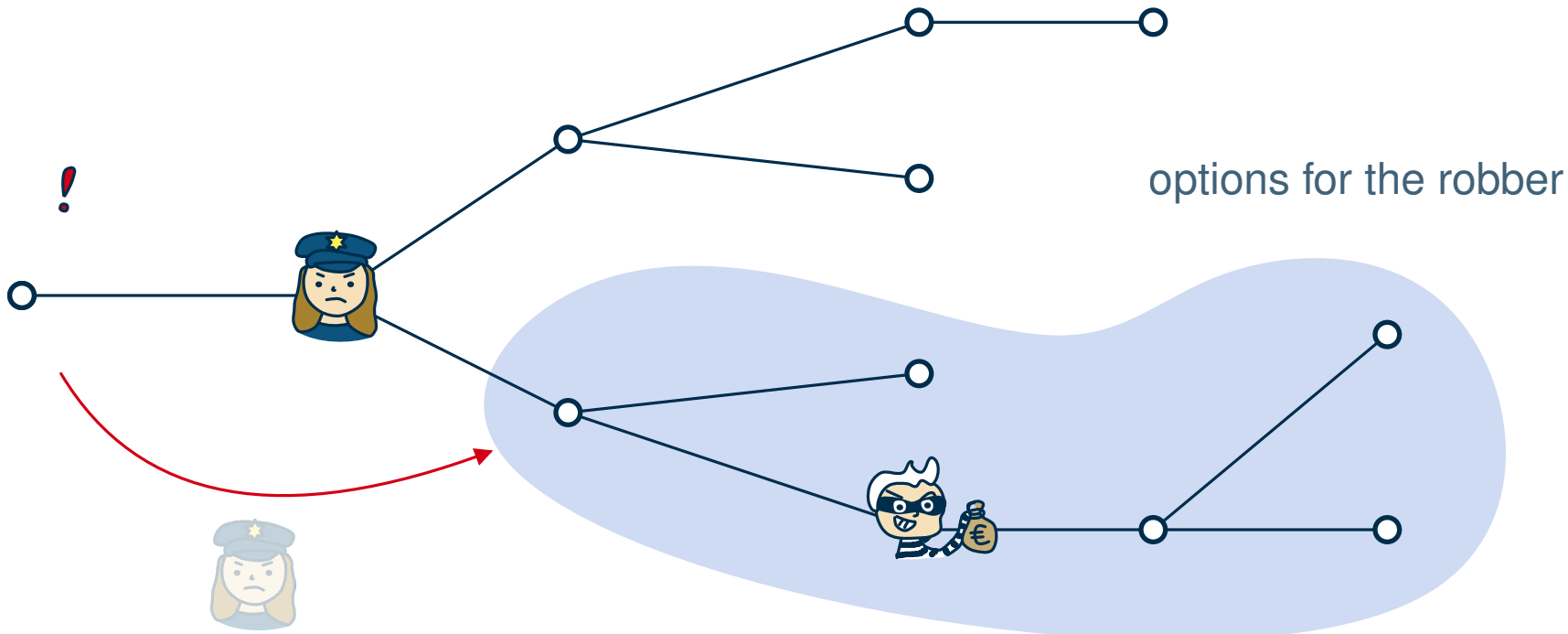
In a connected graph with treewidth $\leq k$, there is a winning strategy for $k+1$ cops.

Start: cops choose starting vertices, then robber chooses vertex

In each round:
- one cop announces new position $v \in V$
- robber may move arbitrarily far, but not through cops
- cop moves to position v

Game Over: robber loses if colocated with cop at round end

$\rightarrow$ on trees: two cops can win



Cops and Robber

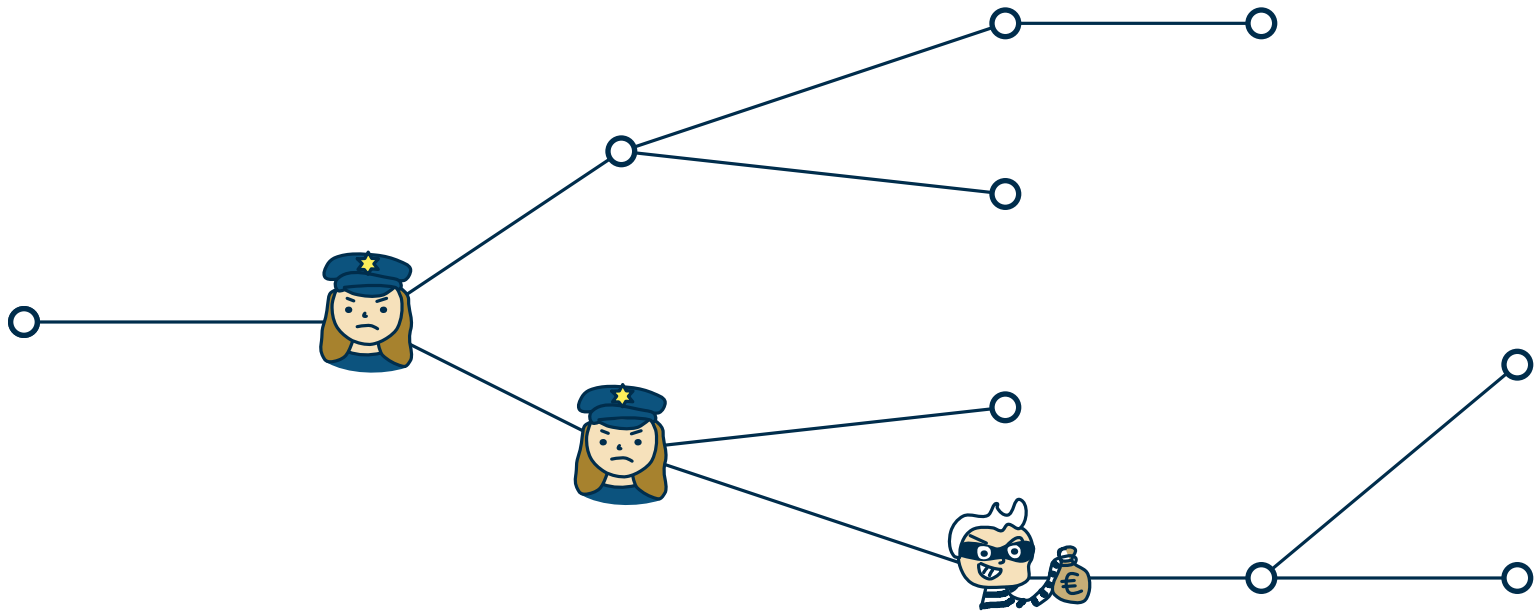
In a connected graph with treewidth $\leq k$, there is a winning strategy for $k+1$ cops.

Start: cops choose starting vertices, then robber chooses vertex

In each round:
- one cop announces new position $v \in V$
- robber may move arbitrarily far, but not through cops
- cop moves to position v

Game Over: robber loses if colocated with cop at round end

$\rightarrow$ on trees: two cops can win



Cops and Robber

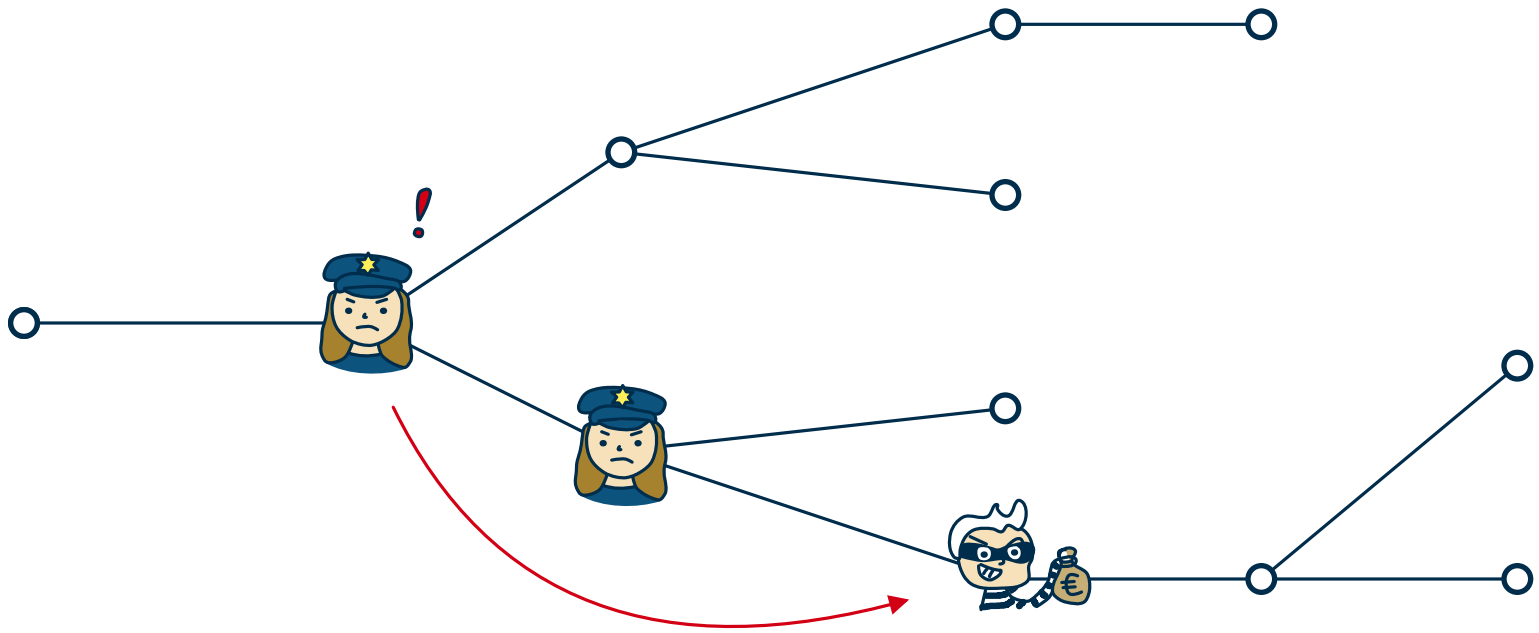
In a connected graph with treewidth $\leq k$, there is a winning strategy for $k+1$ cops.

Start: cops choose starting vertices, then robber chooses vertex

In each round:
- one cop announces new position $v \in V$
- robber may move arbitrarily far, but not through cops
- cop moves to position v

Game Over: robber loses if colocated with cop at round end

$\rightarrow$ on trees: two cops can win



Cops and Robber

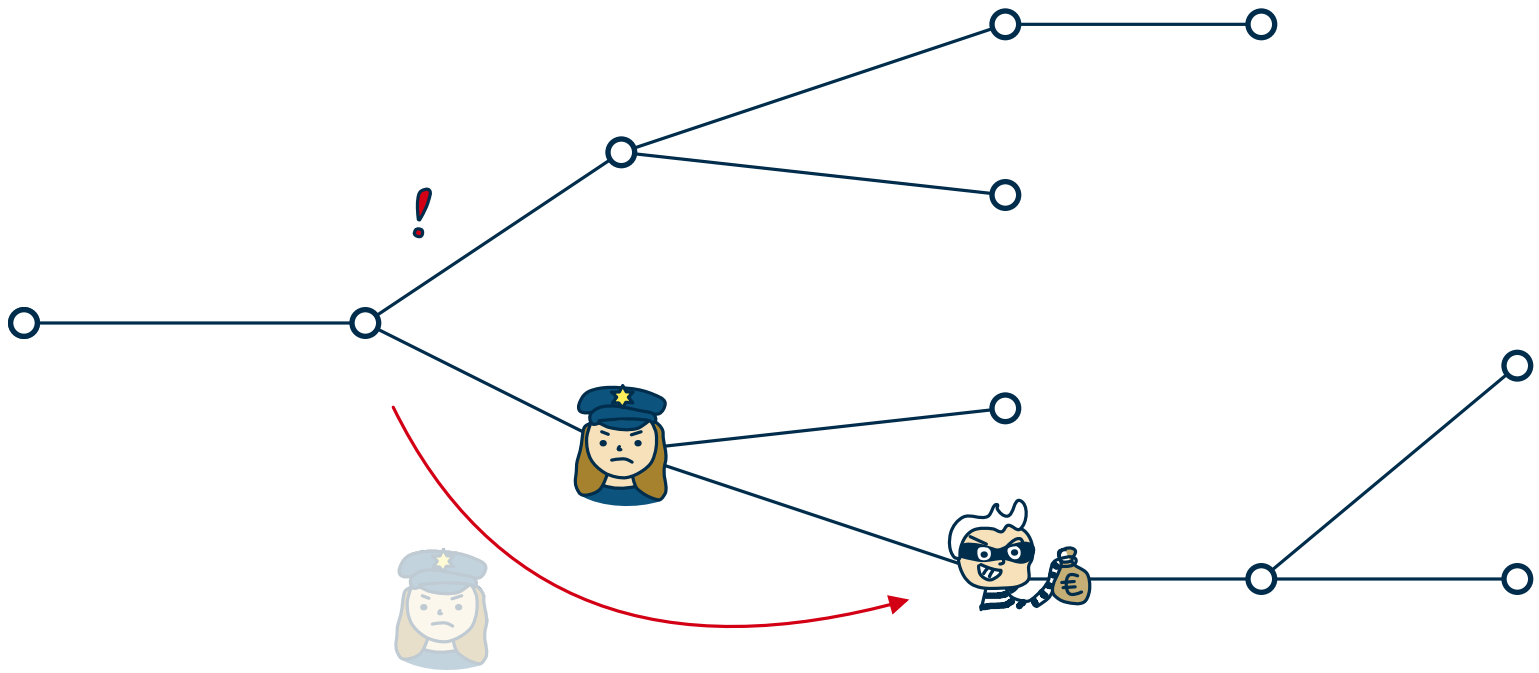
In a connected graph with treewidth $\leq k$, there is a winning strategy for $k+1$ cops.

Start: cops choose starting vertices, then robber chooses vertex

In each round:
- one cop announces new position $v \in V$
- robber may move arbitrarily far, but not through cops
- cop moves to position v

Game Over: robber loses if colocated with cop at round end

$\rightarrow$ on trees: two cops can win



Cops and Robber

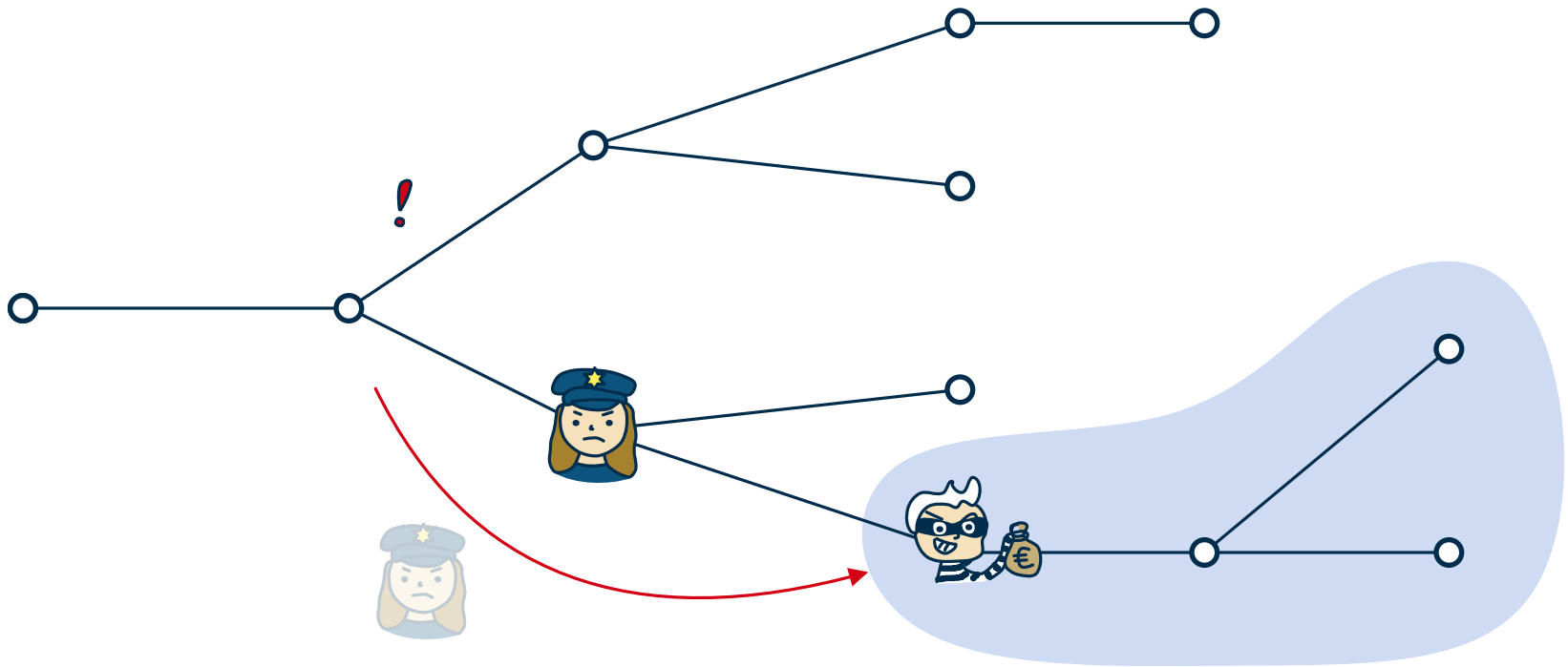
In a connected graph with treewidth $\leq k$, there is a winning strategy for $k+1$ cops.

Start: cops choose starting vertices, then robber chooses vertex

In each round:
- one cop announces new position $v \in V$
- robber may move arbitrarily far, but not through cops
- cop moves to position v

Game Over: robber loses if colocated with cop at round end

$\rightarrow$ on trees: two cops can win



Cops and Robber

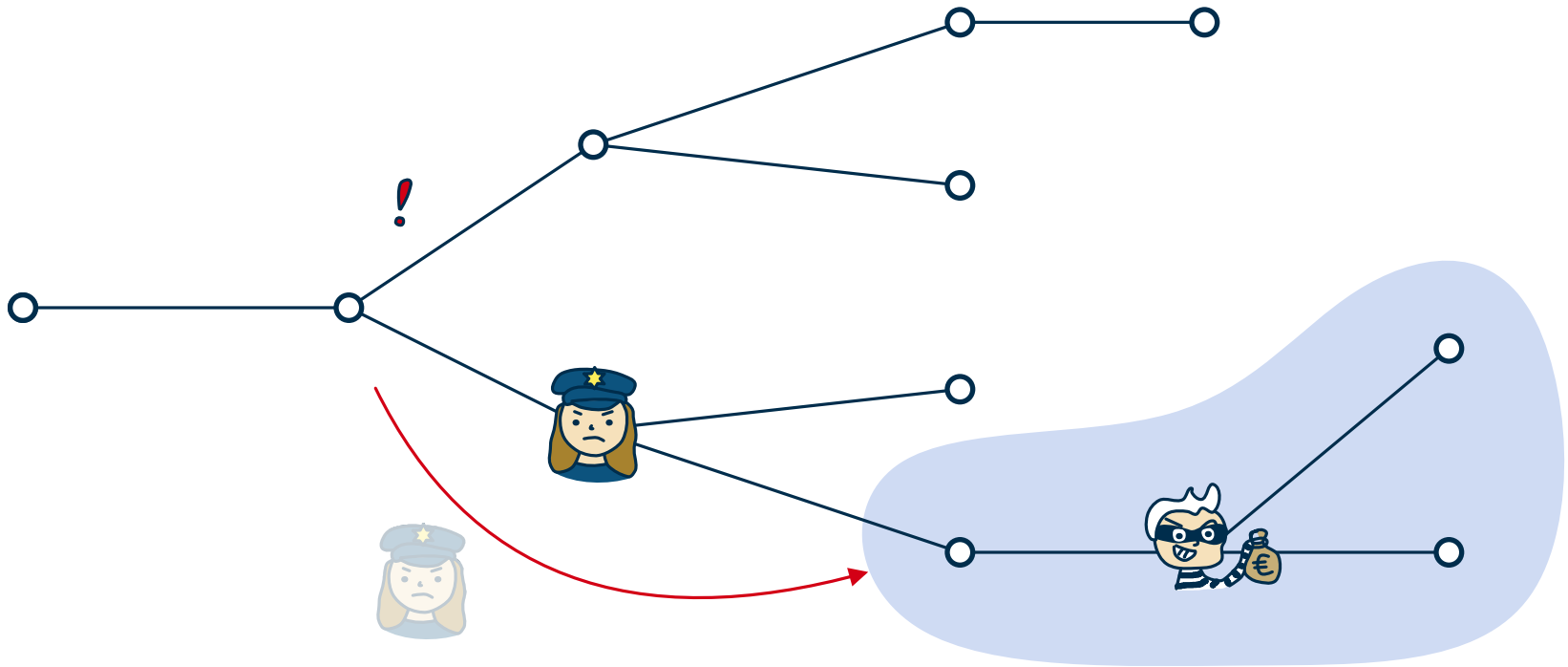
In a connected graph with treewidth $\leq k$, there is a winning strategy for $k+1$ cops.

Start: cops choose starting vertices, then robber chooses vertex

In each round:
- one cop announces new position $v \in V$
- robber may move arbitrarily far, but not through cops
- cop moves to position v

Game Over: robber loses if colocated with cop at round end

$\rightarrow$ on trees: two cops can win



Cops and Robber

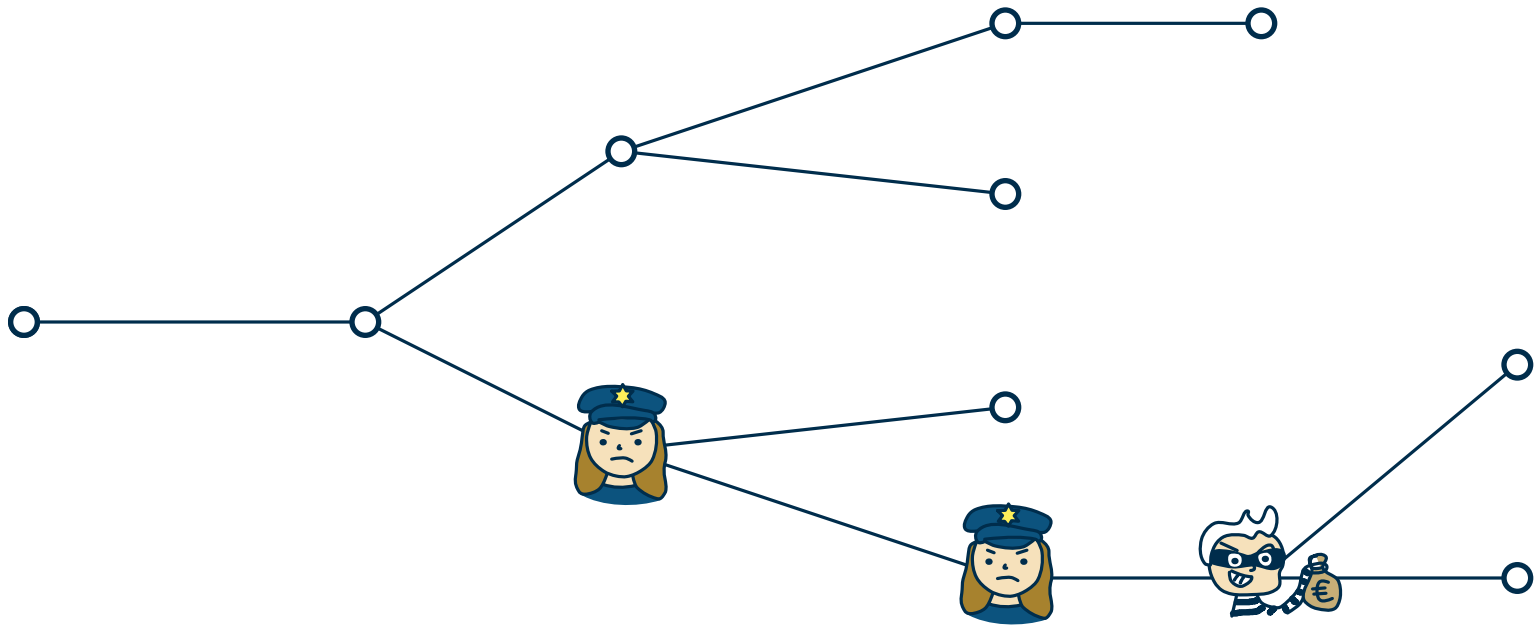
In a connected graph with treewidth $\leq k$, there is a winning strategy for $k+1$ cops.

Start: cops choose starting vertices, then robber chooses vertex

In each round:
- one cop announces new position $v \in V$
- robber may move arbitrarily far, but not through cops
- cop moves to position v

Game Over: robber loses if colocated with cop at round end

$\rightarrow$ on trees: two cops can win



Cops and Robber

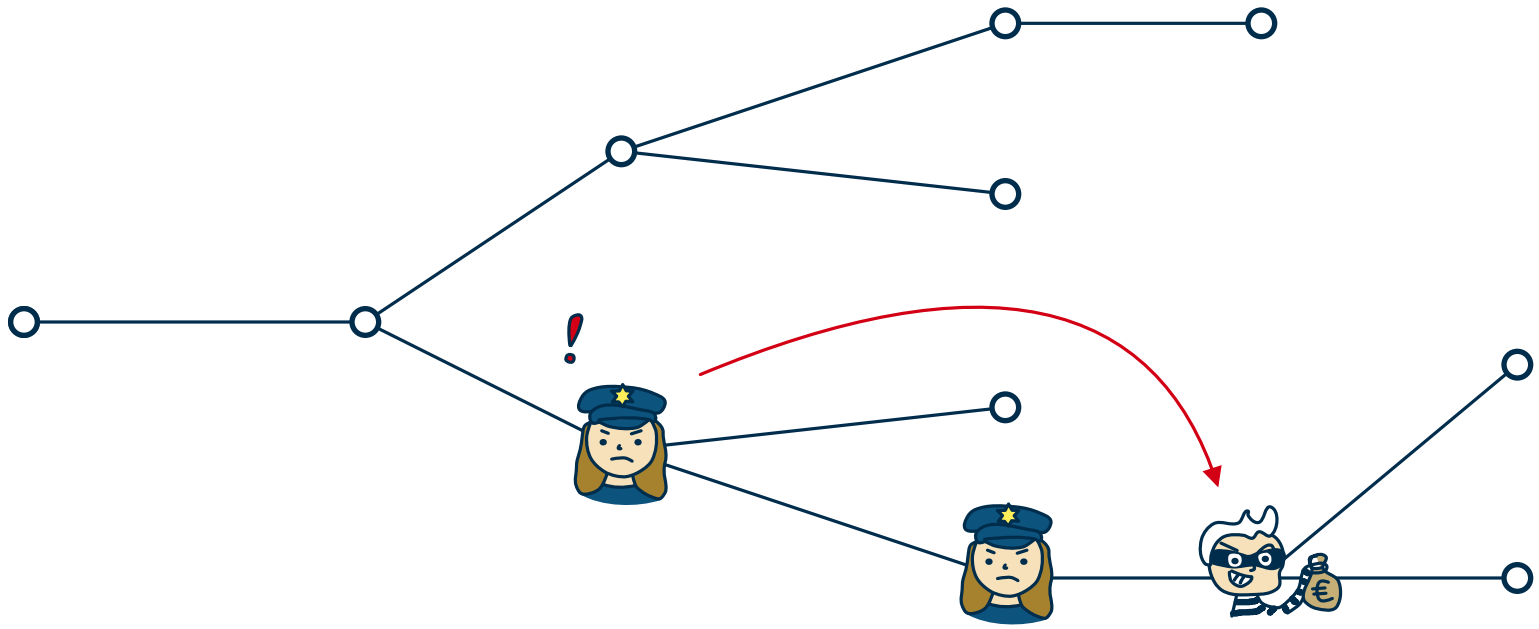
In a connected graph with treewidth $\leq k$, there is a winning strategy for $k+1$ cops.

Start: cops choose starting vertices, then robber chooses vertex

In each round:
- one cop announces new position $v \in V$
- robber may move arbitrarily far, but not through cops
- cop moves to position v

Game Over: robber loses if colocated with cop at round end

$\rightarrow$ on trees: two cops can win



Cops and Robber

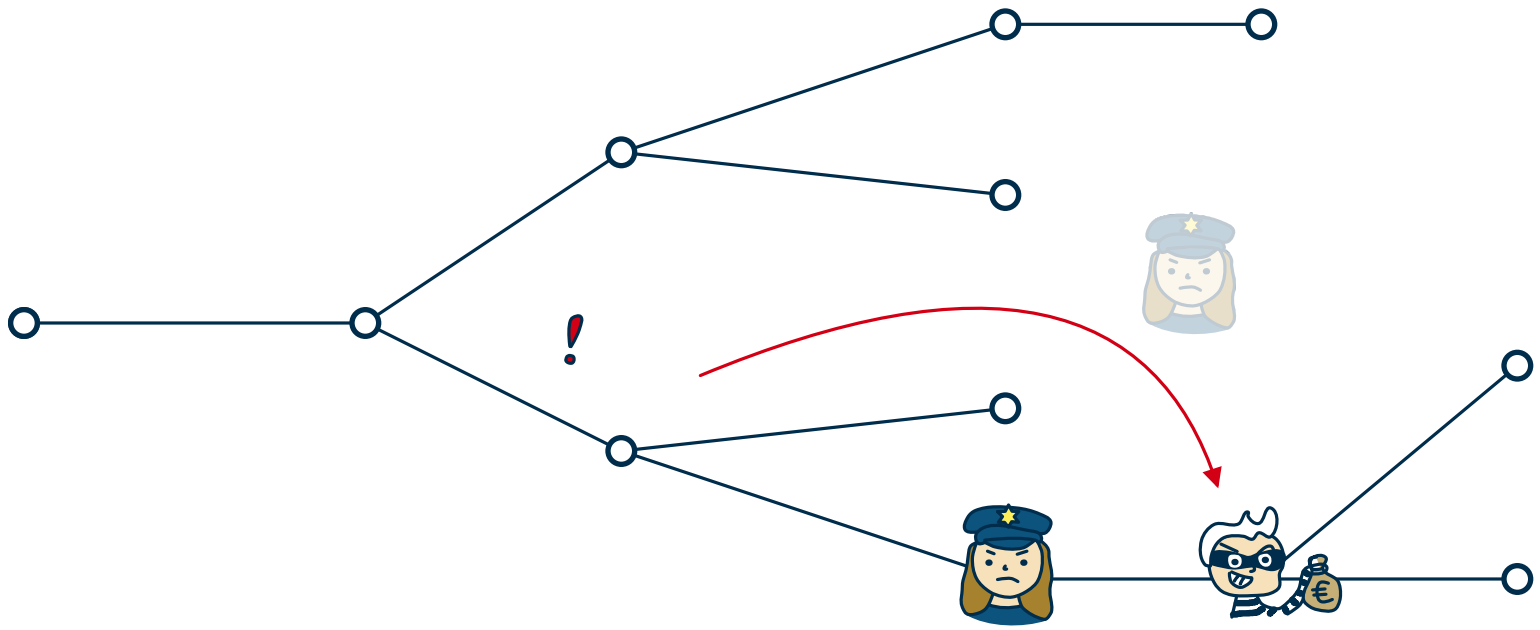
In a connected graph with treewidth $\leq k$, there is a winning strategy for $k+1$ cops.

Start: cops choose starting vertices, then robber chooses vertex

In each round:
- one cop announces new position $v \in V$
- robber may move arbitrarily far, but not through cops
- cop moves to position v

Game Over: robber loses if colocated with cop at round end

$\rightarrow$ on trees: two cops can win



Cops and Robber

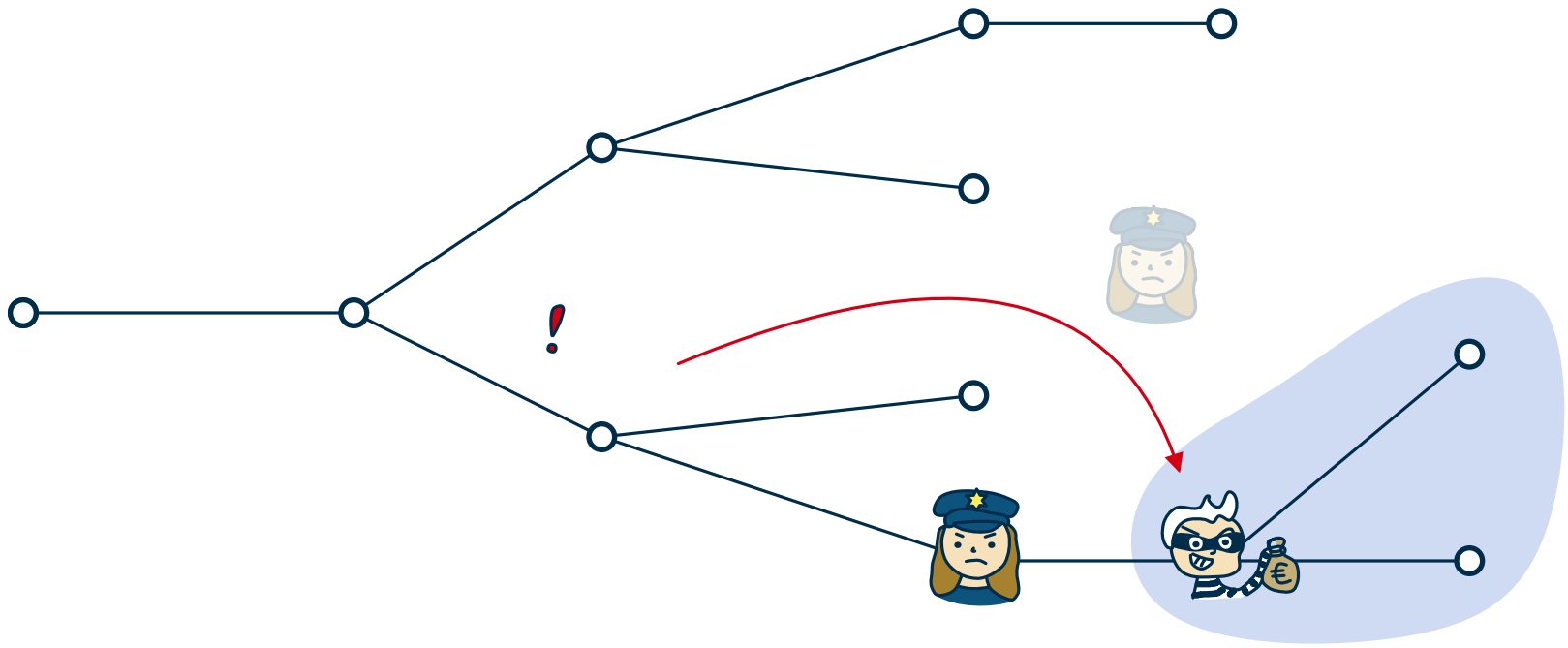
In a connected graph with treewidth $\leq k$, there is a winning strategy for $k+1$ cops.

Start: cops choose starting vertices, then robber chooses vertex

In each round:
- one cop announces new position $v \in V$
- robber may move arbitrarily far, but not through cops
- cop moves to position v

Game Over: robber loses if colocated with cop at round end

$\rightarrow$ on trees: two cops can win



Cops and Robber

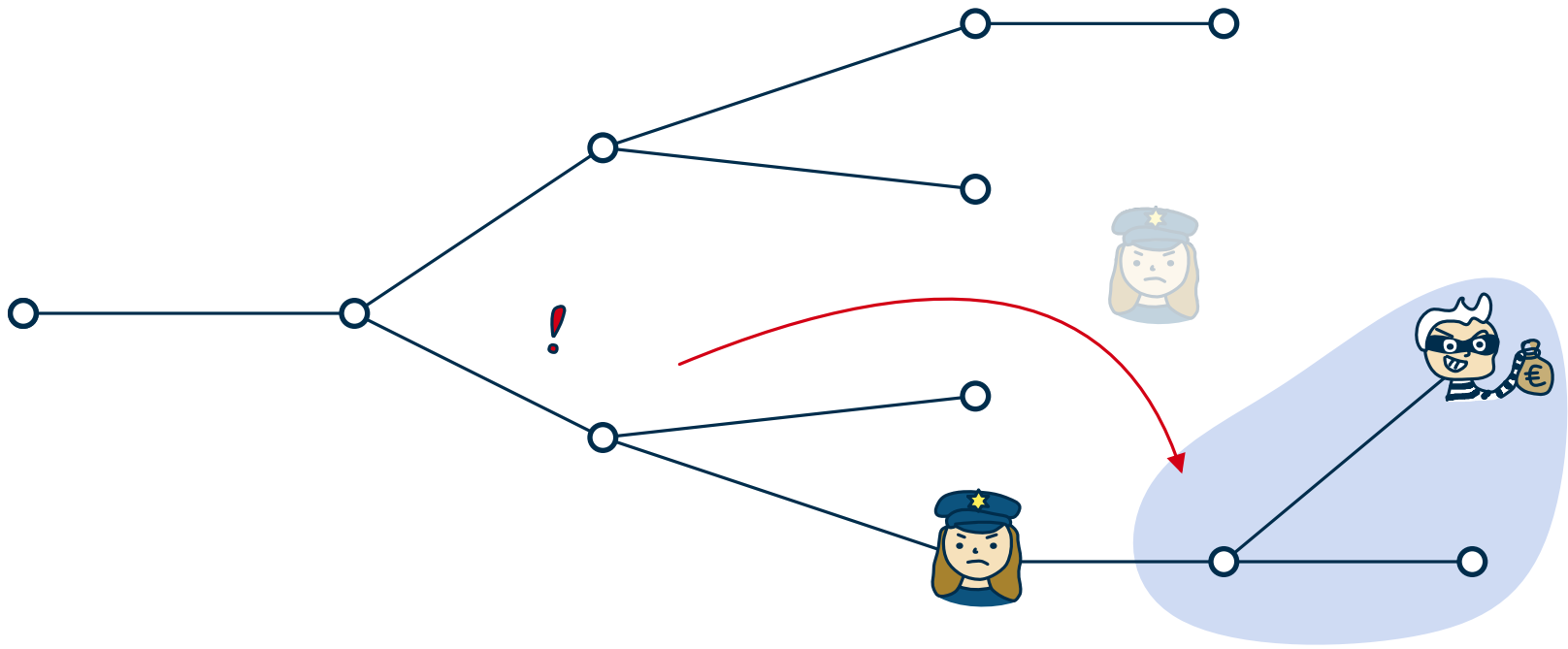
In a connected graph with treewidth $\leq k$, there is a winning strategy for $k+1$ cops.

Start: cops choose starting vertices, then robber chooses vertex

In each round:
- one cop announces new position $v \in V$
- robber may move arbitrarily far, but not through cops
- cop moves to position v

Game Over: robber loses if colocated with cop at round end

$\rightarrow$ on trees: two cops can win



Cops and Robber

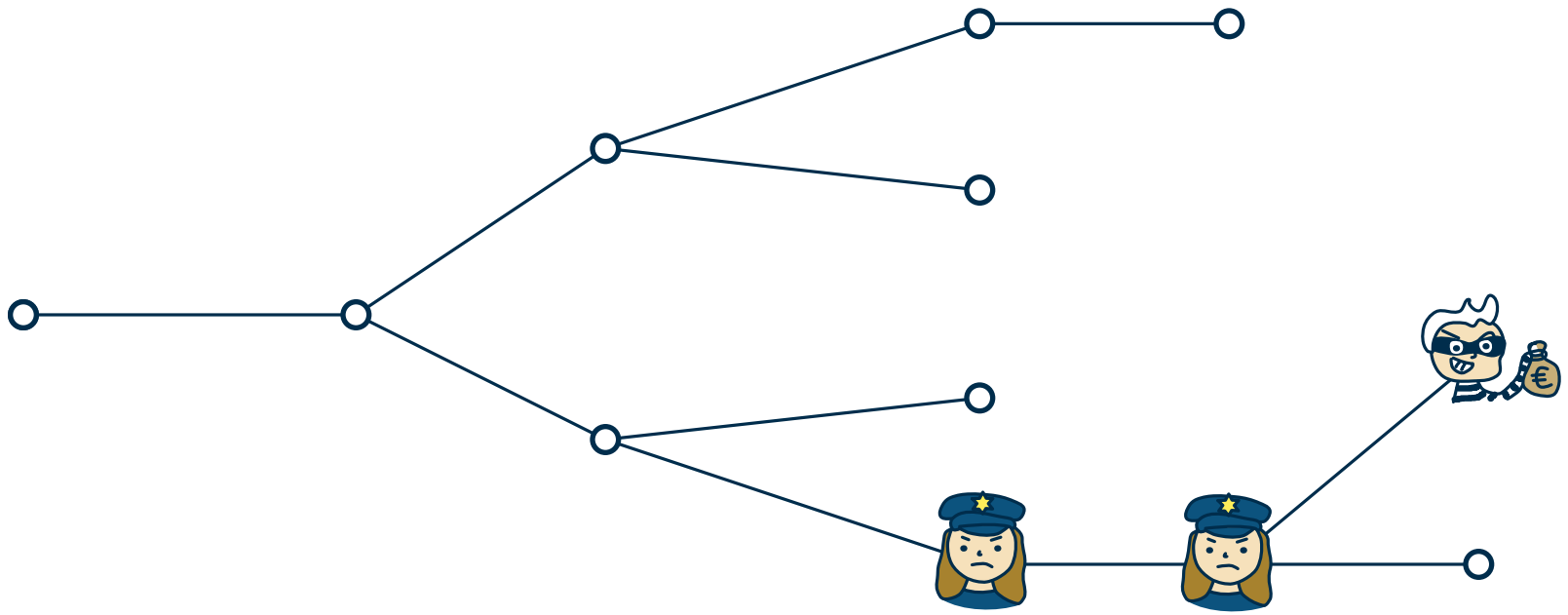
In a connected graph with treewidth $\leq k$, there is a winning strategy for $k+1$ cops.

Start: cops choose starting vertices, then robber chooses vertex

In each round:
- one cop announces new position $v \in V$
- robber may move arbitrarily far, but not through cops
- cop moves to position v

Game Over: robber loses if colocated with cop at round end

$\rightarrow$ on trees: two cops can win



Cops and Robber

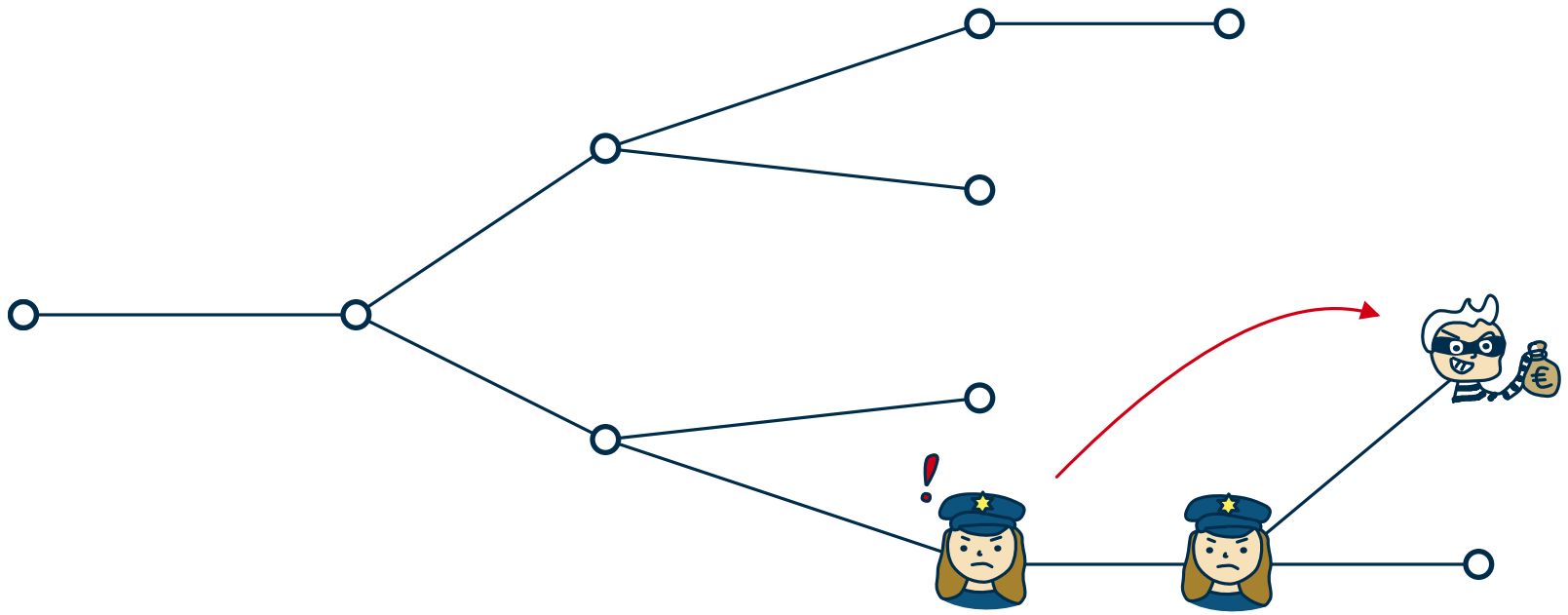
In a connected graph with treewidth $\leq k$, there is a winning strategy for $k+1$ cops.

Start: cops choose starting vertices, then robber chooses vertex

In each round:
- one cop announces new position $v \in V$
- robber may move arbitrarily far, but not through cops
- cop moves to position v

Game Over: robber loses if colocated with cop at round end

$\rightarrow$ on trees: two cops can win



Cops and Robber

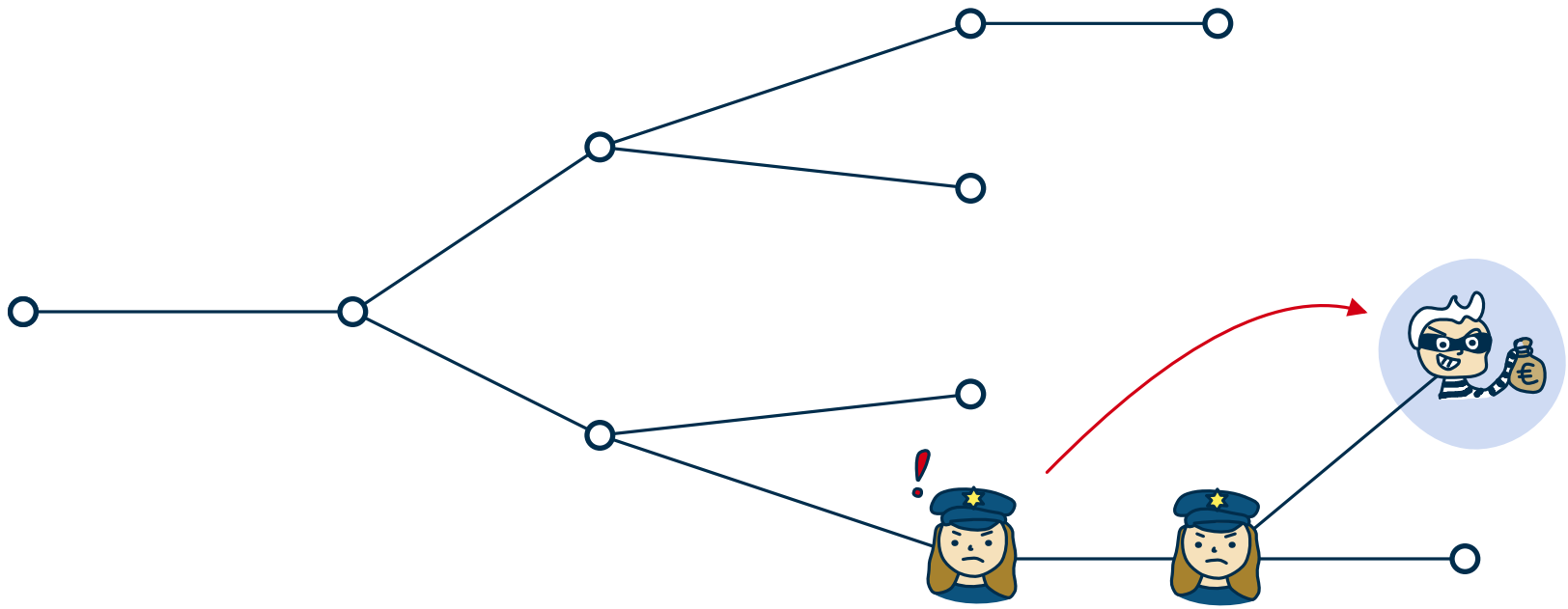
In a connected graph with treewidth $\leq k$, there is a winning strategy for $k+1$ cops.

Start: cops choose starting vertices, then robber chooses vertex

In each round:
- one cop announces new position $v \in V$
- robber may move arbitrarily far, but not through cops
- cop moves to position v

Game Over: robber loses if colocated with cop at round end

$\rightarrow$ on trees: two cops can win



Cops and Robber

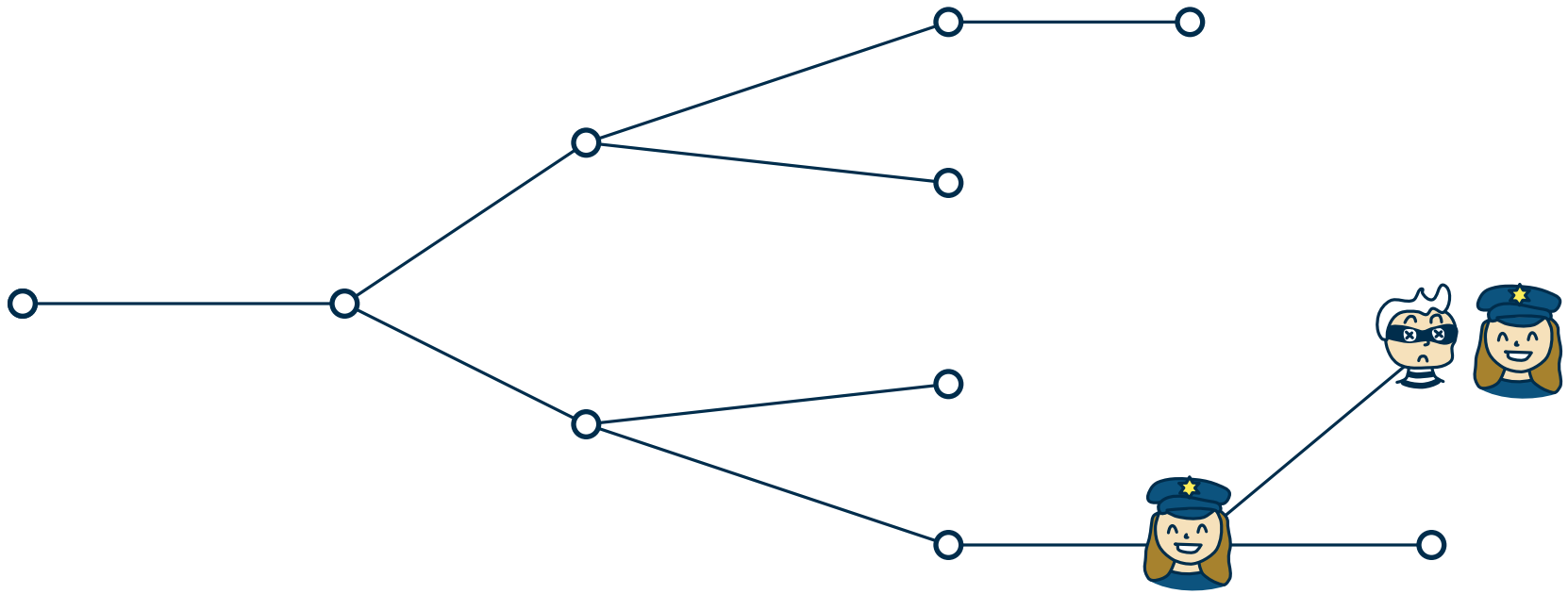
In a connected graph with treewidth $\leq k$, there is a winning strategy for $k+1$ cops.

Start: cops choose starting vertices, then robber chooses vertex

In each round:
- one cop announces new position $v \in V$
- robber may move arbitrarily far, but not through cops
- cop moves to position v

Game Over: robber loses if colocated with cop at round end

$\rightarrow$ on trees: two cops can win



Cops and Robber

In a connected graph with treewidth $\leq k$, there is a winning strategy for $k+1$ cops.

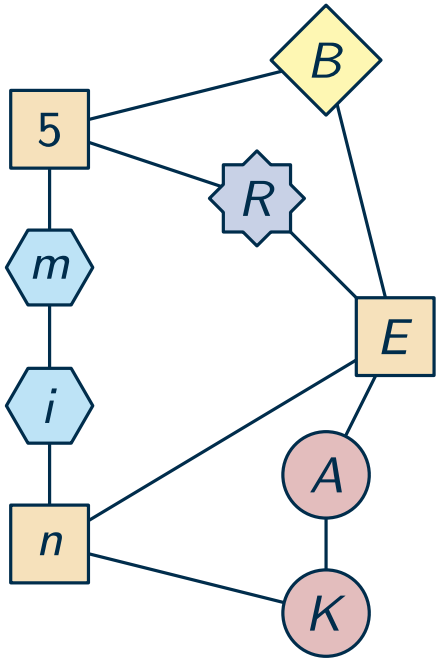
Start: cops choose starting vertices, then robber chooses vertex

In each round:
- one cop announces new position $v \in V$
- robber may move arbitrarily far, but not through cops
- cop moves to position v

Game Over: robber loses if colocated with cop at round end

graph G with treewidth $\leq k$

G



Cops and Robber

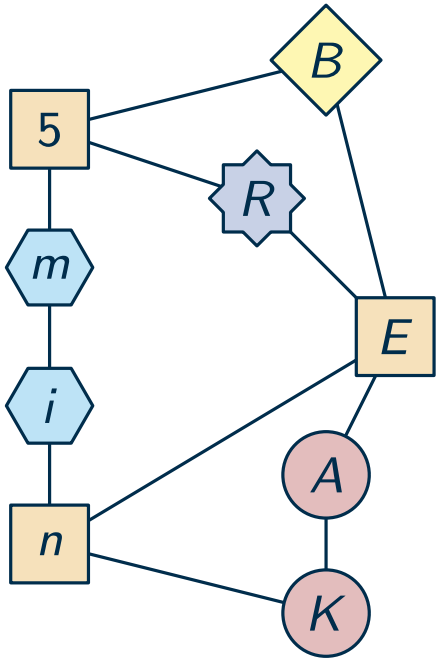
In a connected graph with treewidth $\leq k$, there is a winning strategy for $k+1$ cops.

graph G with treewidth $\leq k$

$\Longrightarrow$ nice tree decomposition of width $\leq k$

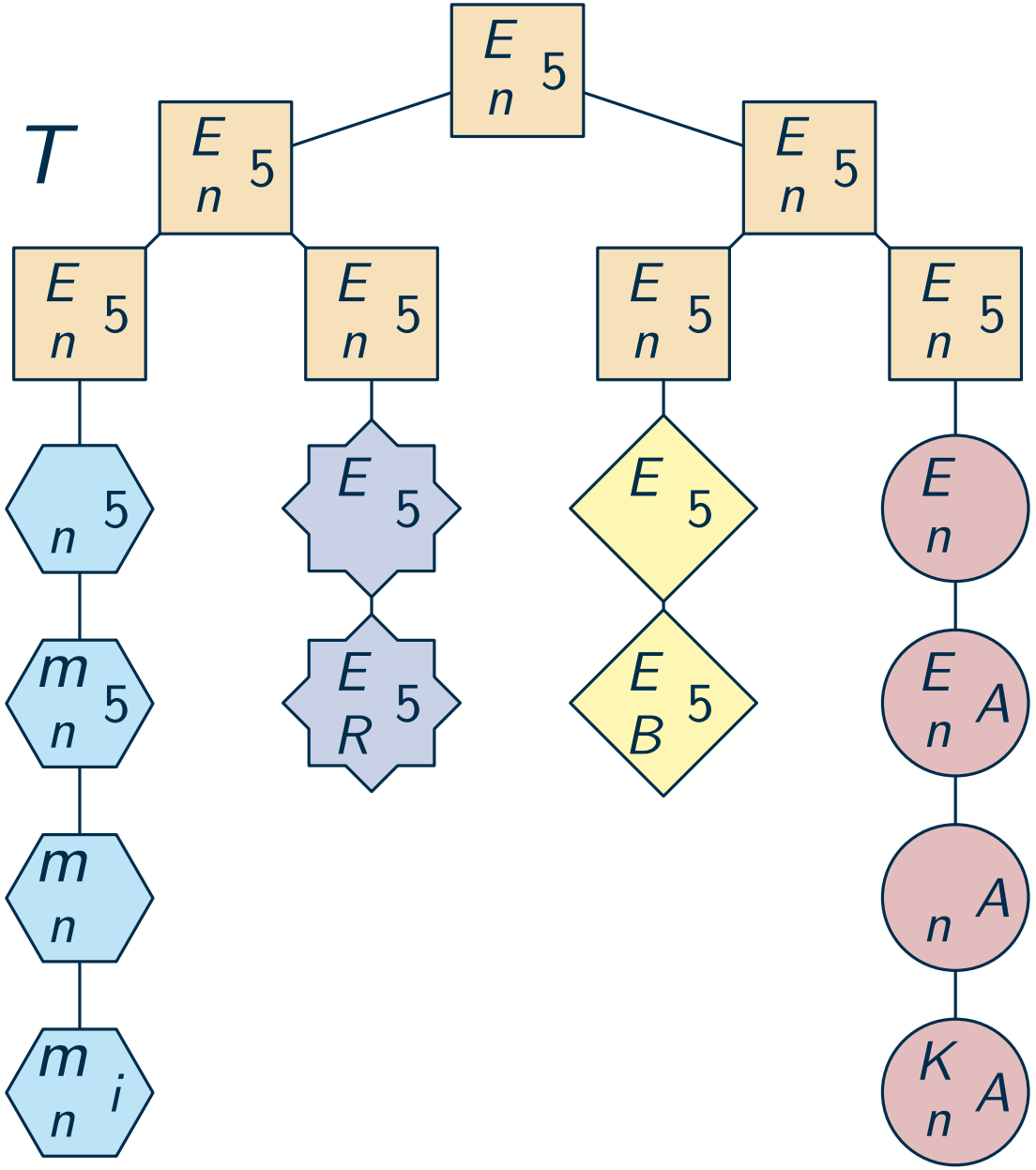
(bag size $\leq k+1$)

G



$k = 2$

T



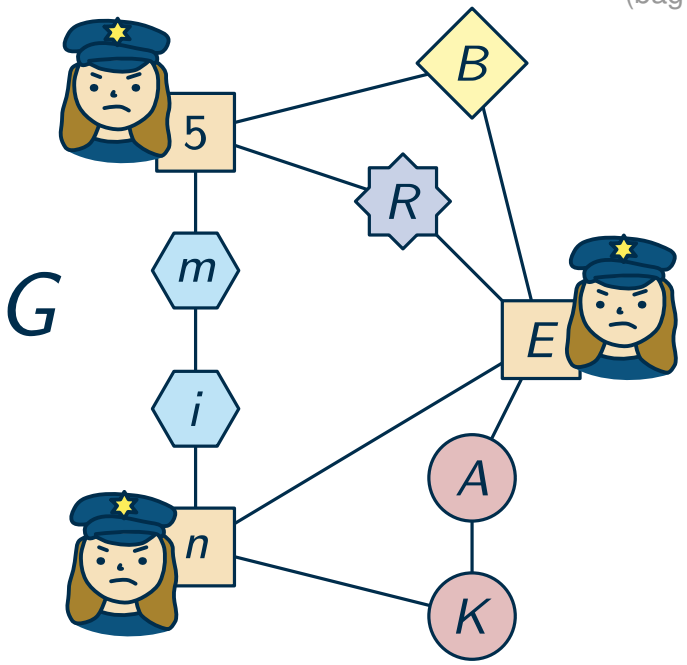
Cops and Robber

In a connected graph with treewidth $\leq k$, there is a winning strategy for $k+1$ cops.

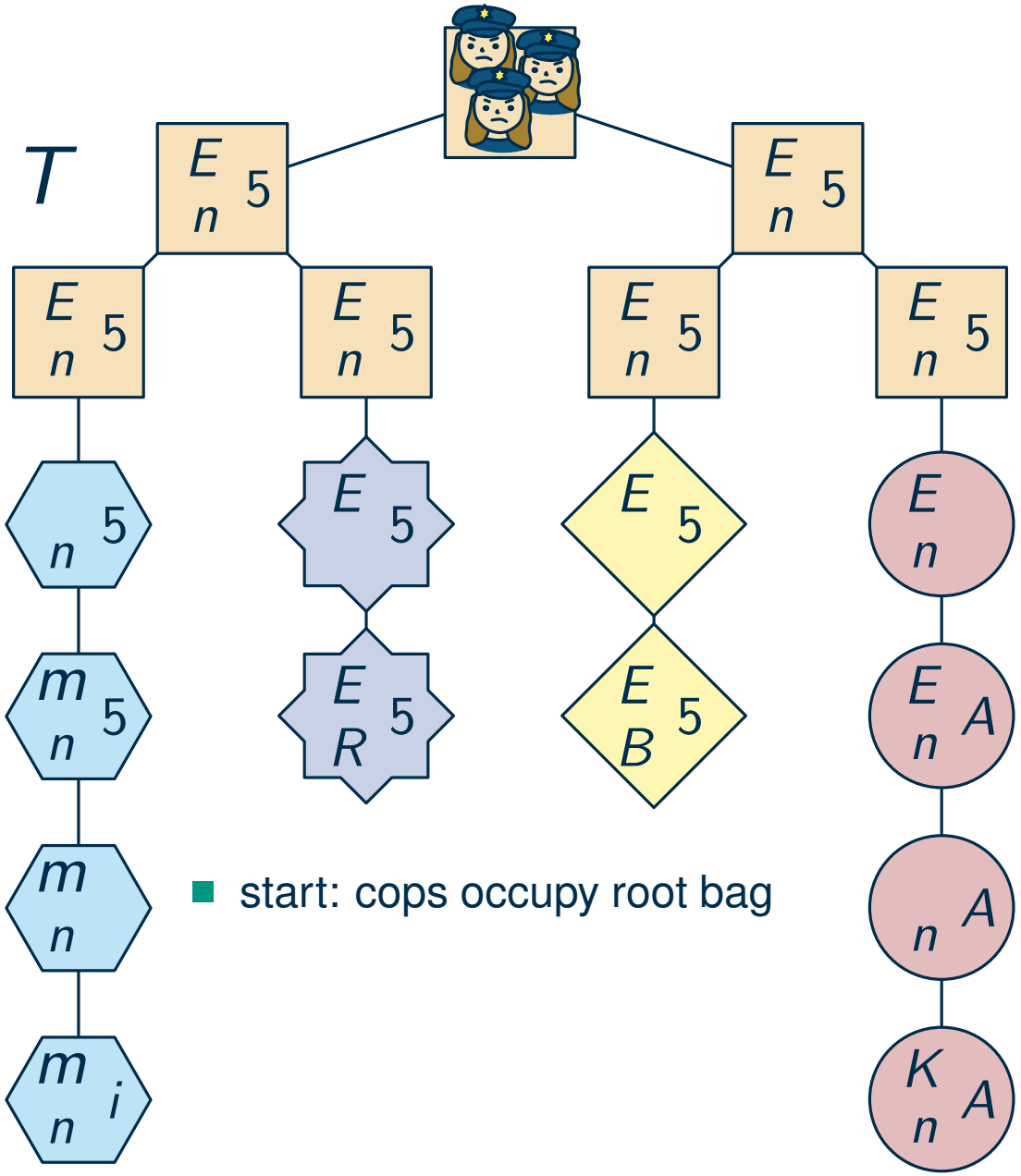
graph G with treewidth $\leq k$

$\Longrightarrow$ nice tree decomposition of width $\leq k$

(bag size $\leq k+1$)



$k = 2$



- start: cops occupy root bag

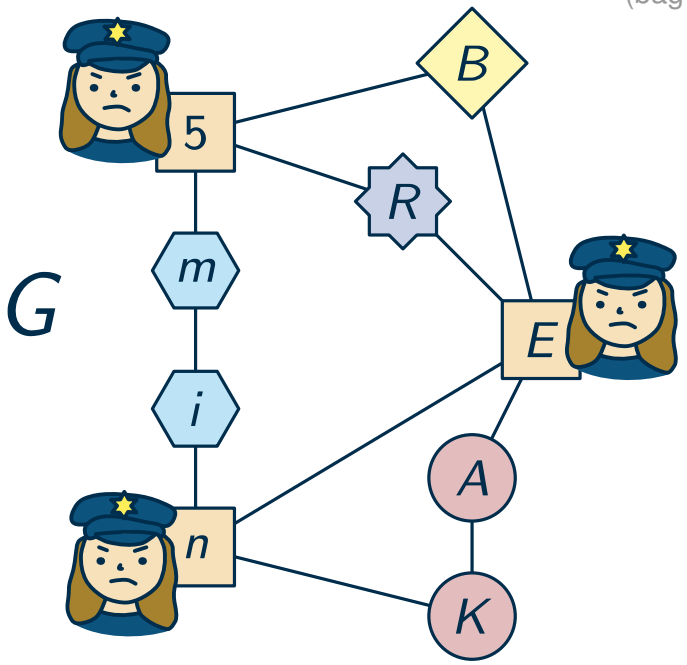
Cops and Robber

In a connected graph with treewidth $\leq k$, there is a winning strategy for $k+1$ cops.

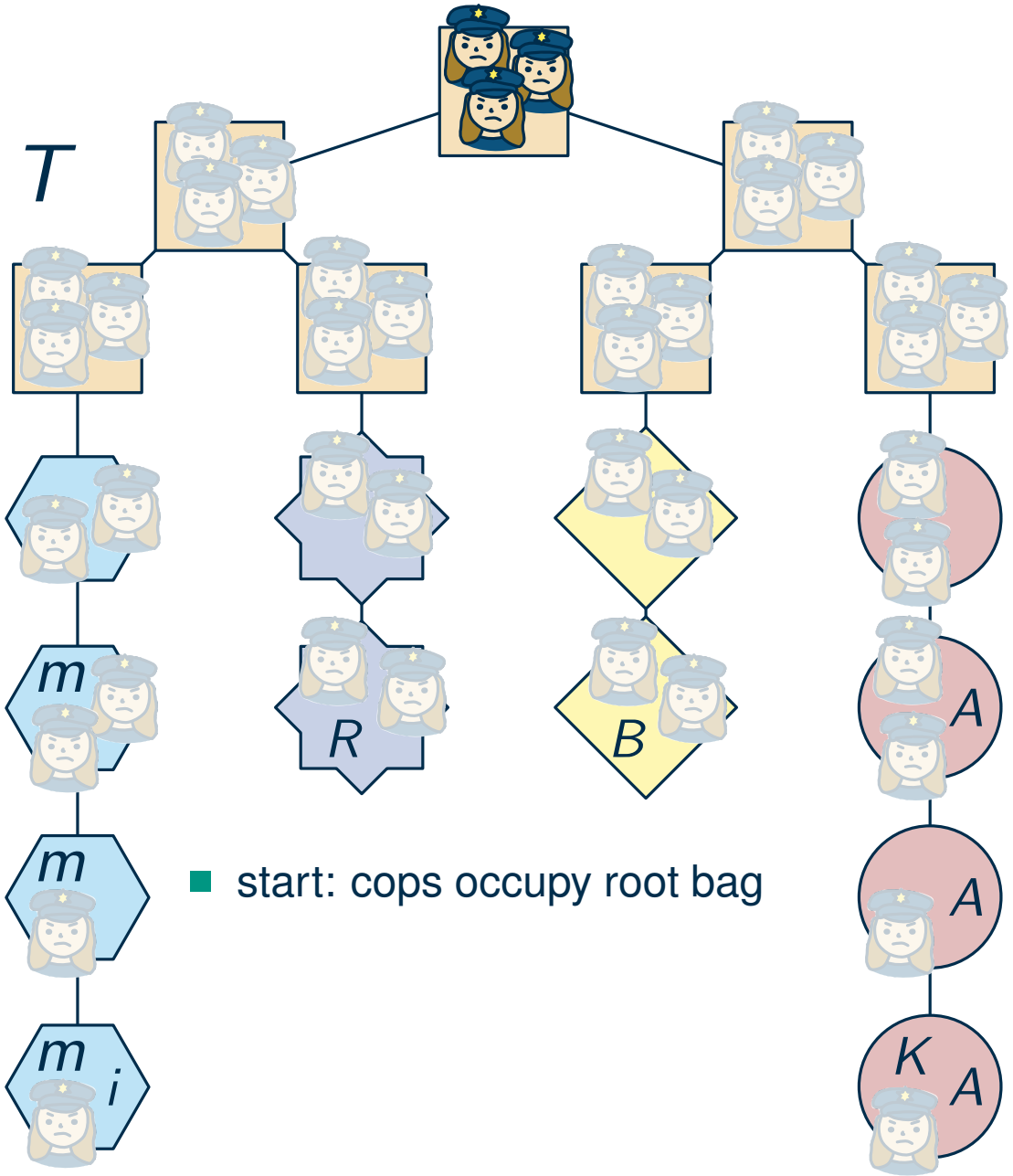
graph G with treewidth $\leq k$

$\Longrightarrow$ nice tree decomposition of width $\leq k$

(bag size $\leq k+1$)



$k = 2$



- start: cops occupy root bag

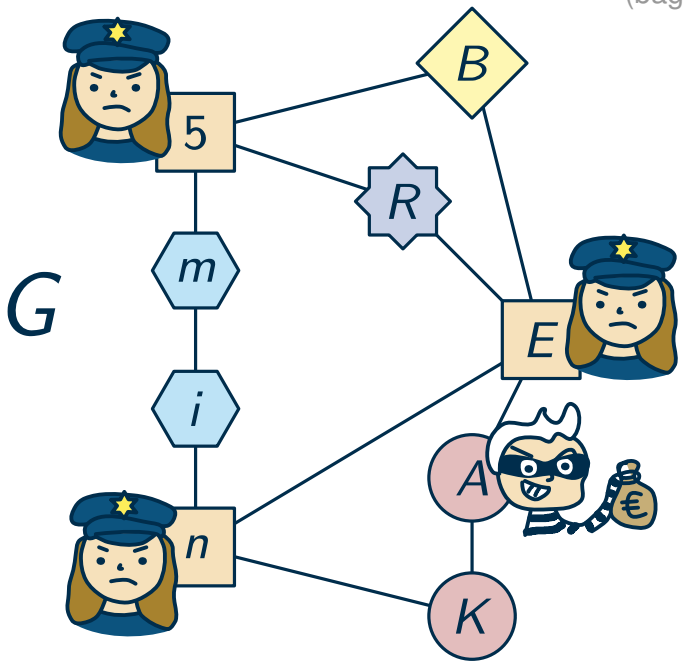
Cops and Robber

In a connected graph with treewidth $\leq k$, there is a winning strategy for $k+1$ cops.

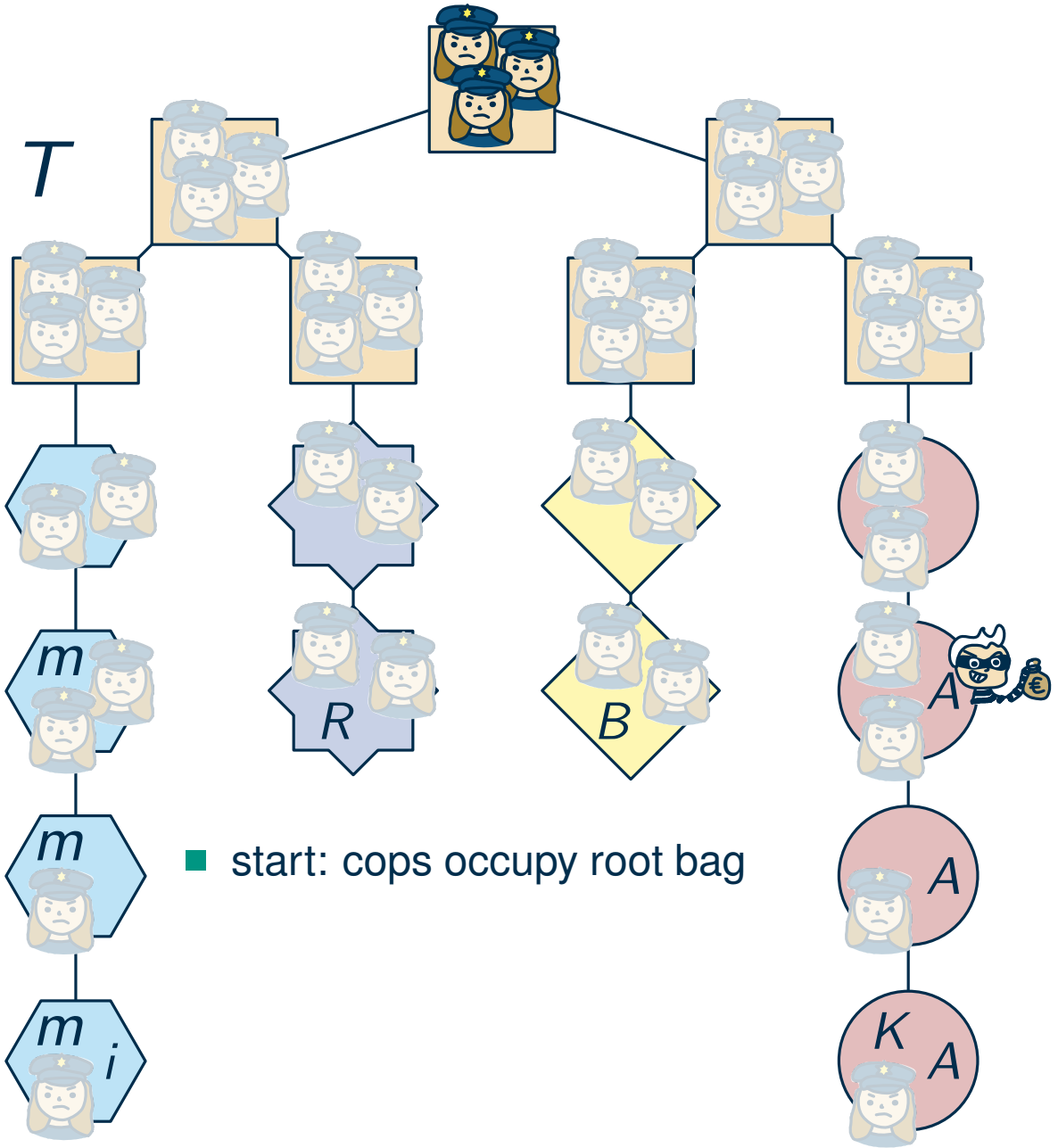
graph G with treewidth $\leq k$

$\Longrightarrow$ nice tree decomposition of width $\leq k$

(bag size $\leq k+1$)



$k = 2$



- start: cops occupy root bag

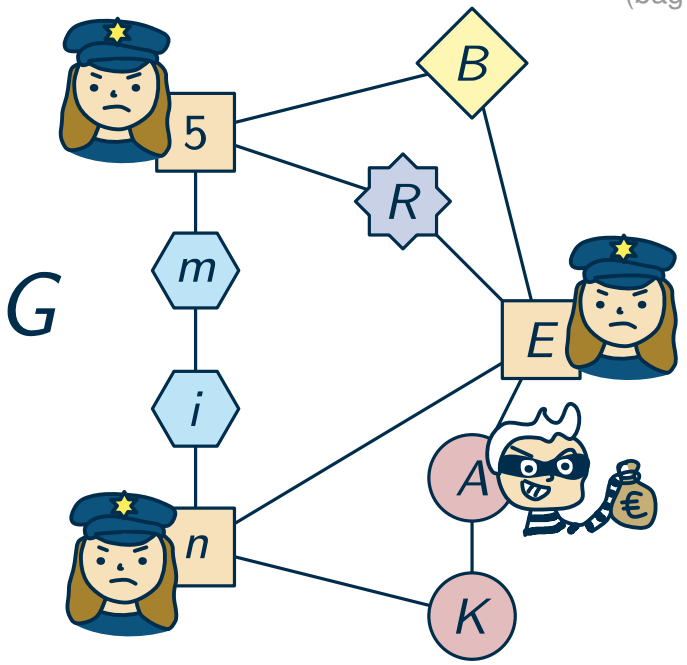
Cops and Robber

In a connected graph with treewidth $\leq k$, there is a winning strategy for $k+1$ cops.

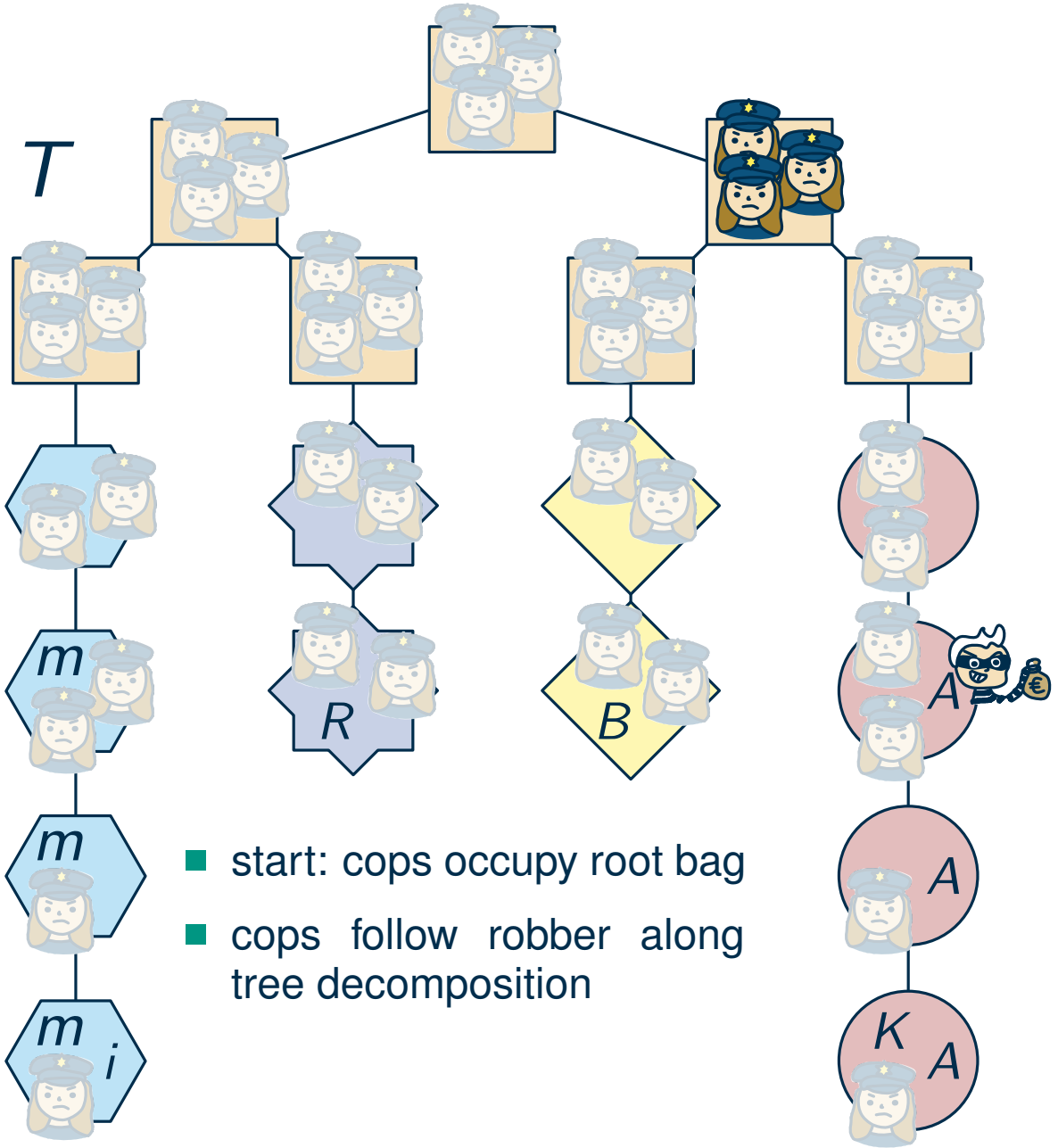
graph G with treewidth $\leq k$

$\Longrightarrow$ nice tree decomposition of width $\leq k$

(bag size $\leq k+1$)



$k = 2$



- start: cops occupy root bag
- cops follow robber along tree decomposition

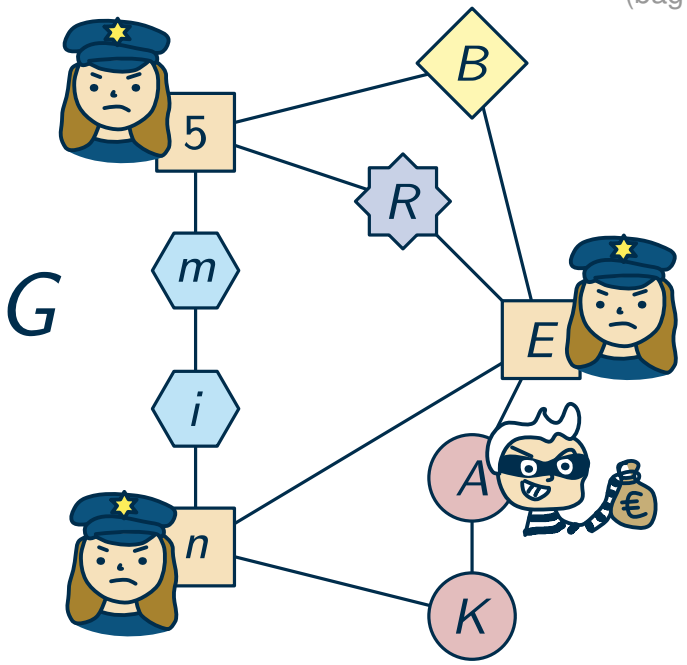
Cops and Robber

In a connected graph with treewidth $\leq k$, there is a winning strategy for $k+1$ cops.

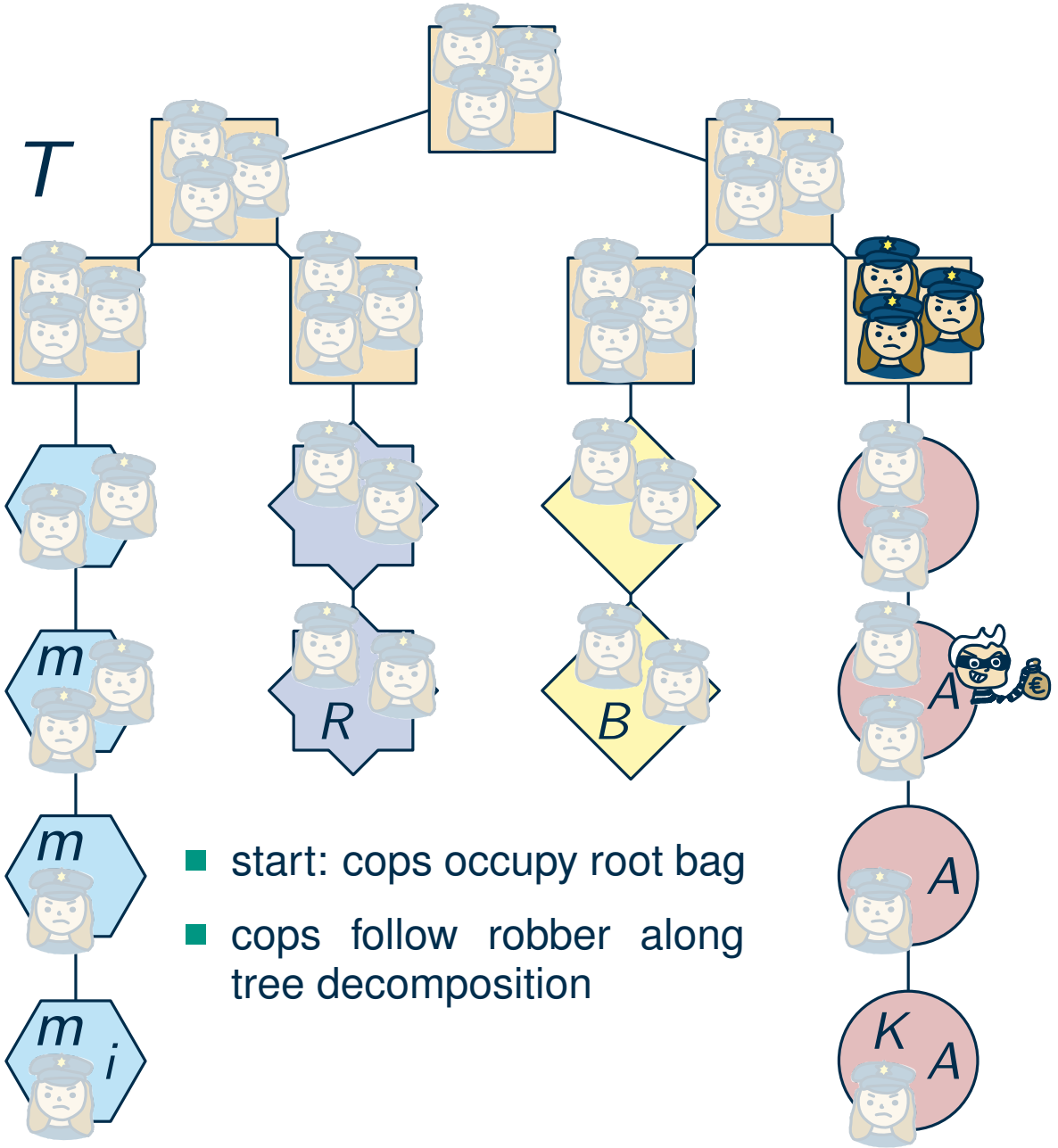
graph G with treewidth $\leq k$

$\Longrightarrow$ nice tree decomposition of width $\leq k$

(bag size $\leq k+1$)



$k = 2$



- start: cops occupy root bag
- cops follow robber along tree decomposition

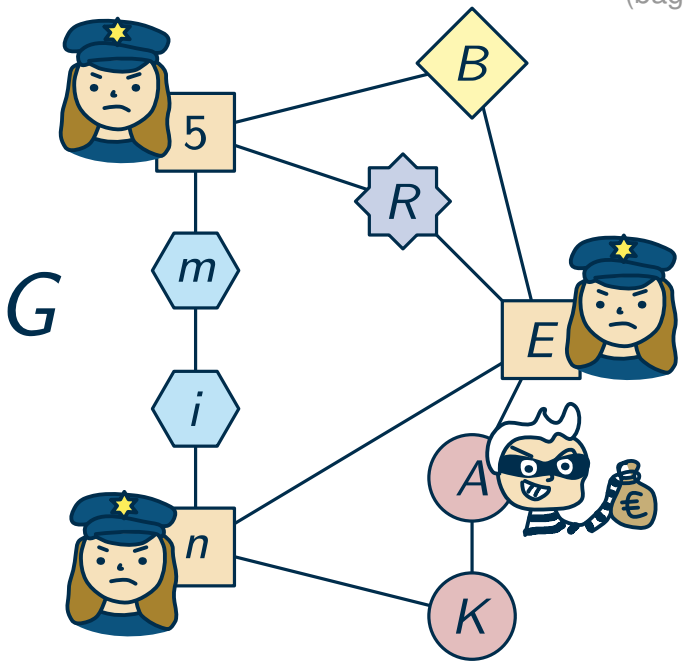
Cops and Robber

In a connected graph with treewidth $\leq k$, there is a winning strategy for $k+1$ cops.

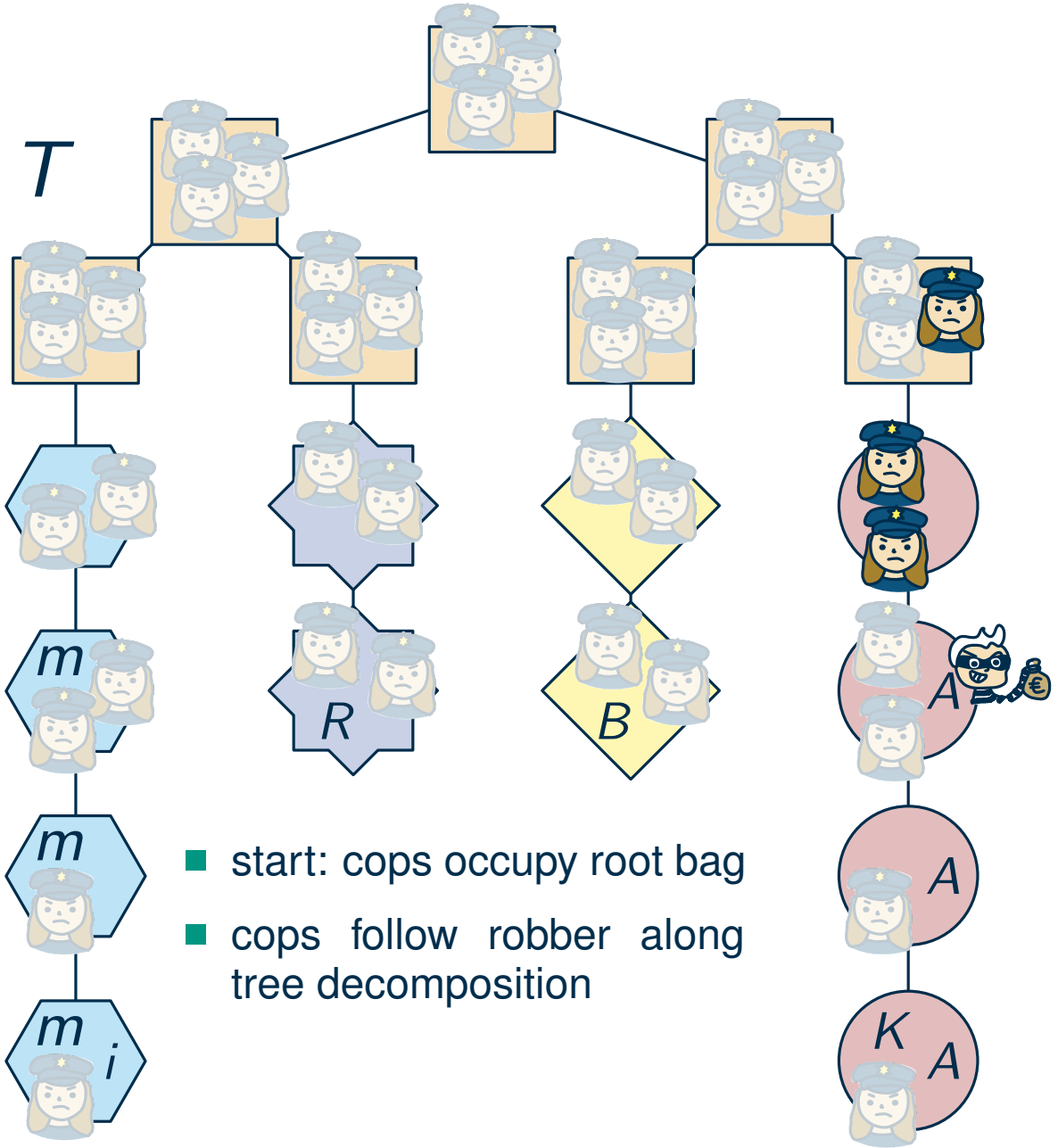
graph G with treewidth $\leq k$

$\Longrightarrow$ nice tree decomposition of width $\leq k$

(bag size $\leq k+1$)



$k = 2$



- start: cops occupy root bag
- cops follow robber along tree decomposition

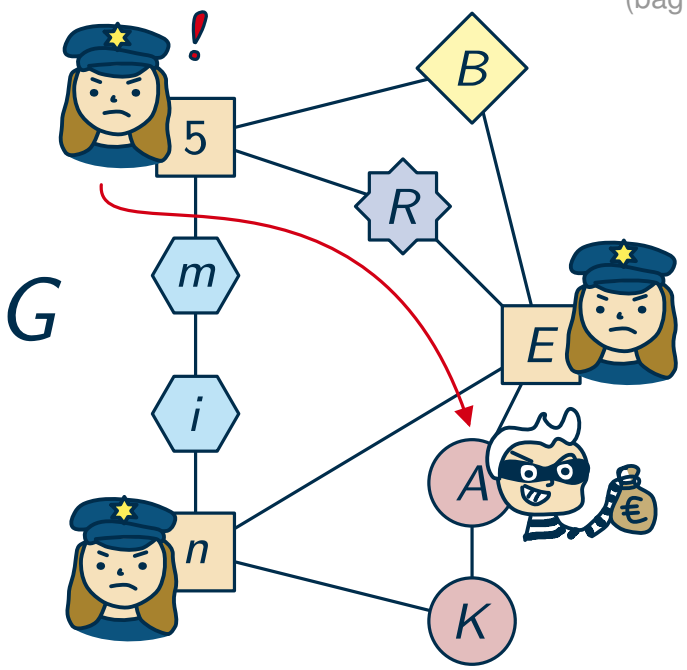
Cops and Robber

In a connected graph with treewidth $\leq k$, there is a winning strategy for $k+1$ cops.

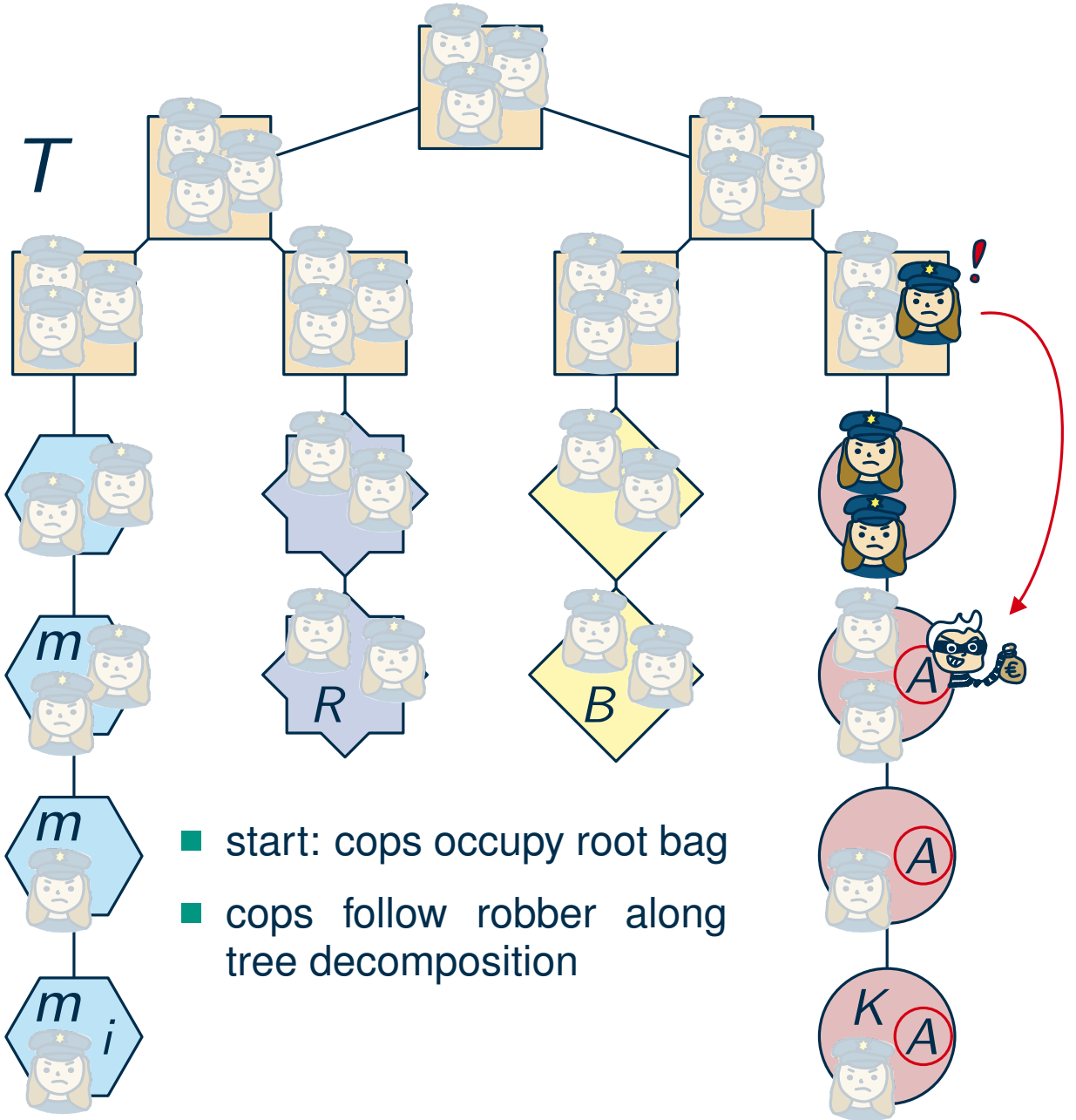
graph G with treewidth $\leq k$

$\Longrightarrow$ nice tree decomposition of width $\leq k$

(bag size $\leq k+1$)



$k = 2$



- start: cops occupy root bag
- cops follow robber along tree decomposition

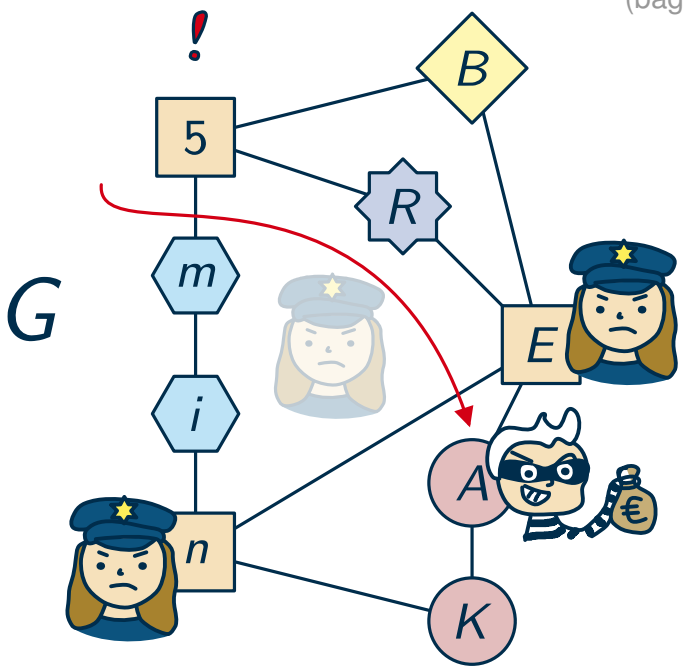
Cops and Robber

In a connected graph with treewidth $\leq k$, there is a winning strategy for $k+1$ cops.

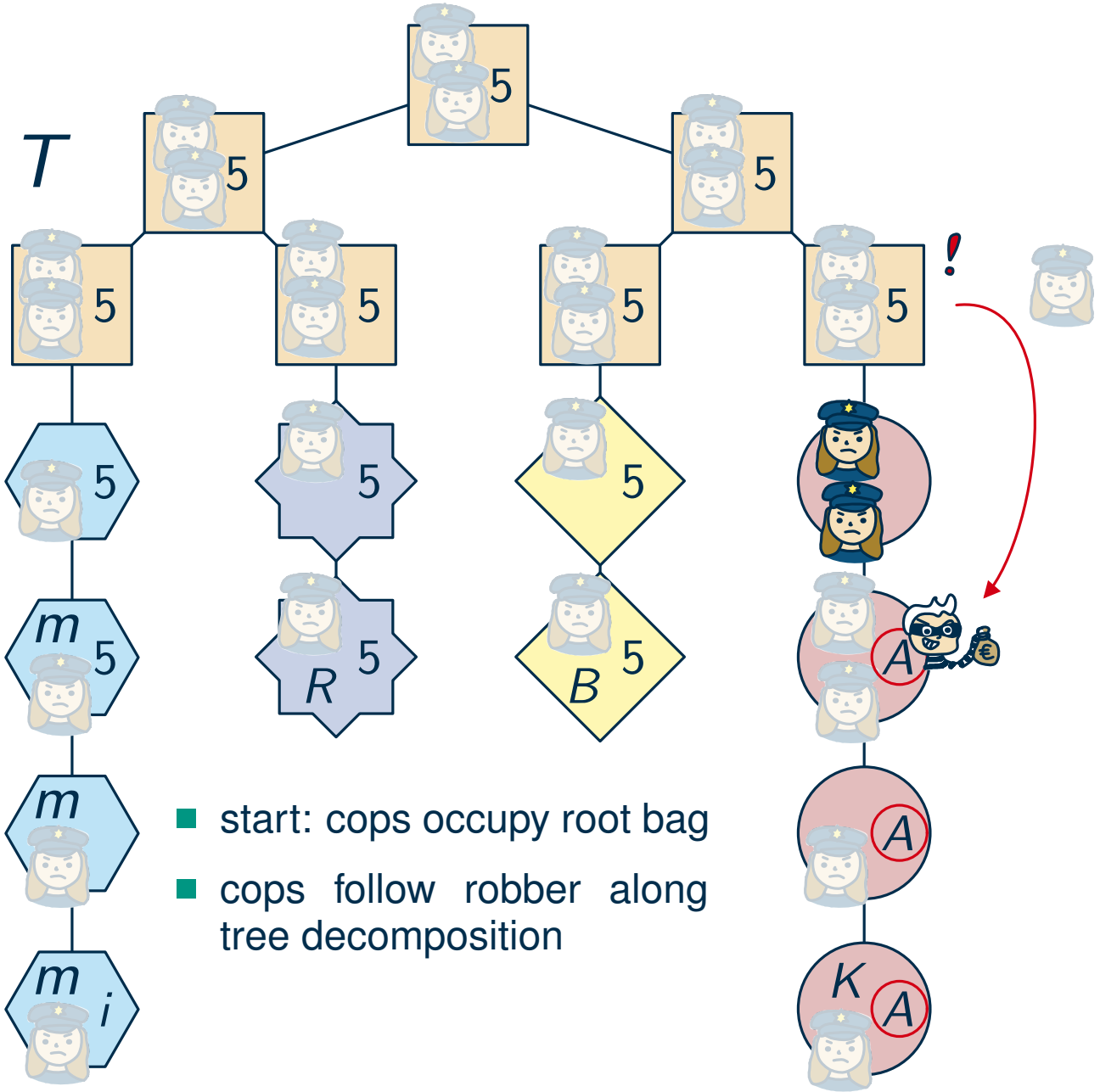
graph G with treewidth $\leq k$

$\Longrightarrow$ nice tree decomposition of width $\leq k$

(bag size $\leq k+1$)



$k = 2$



- start: cops occupy root bag
- cops follow robber along tree decomposition

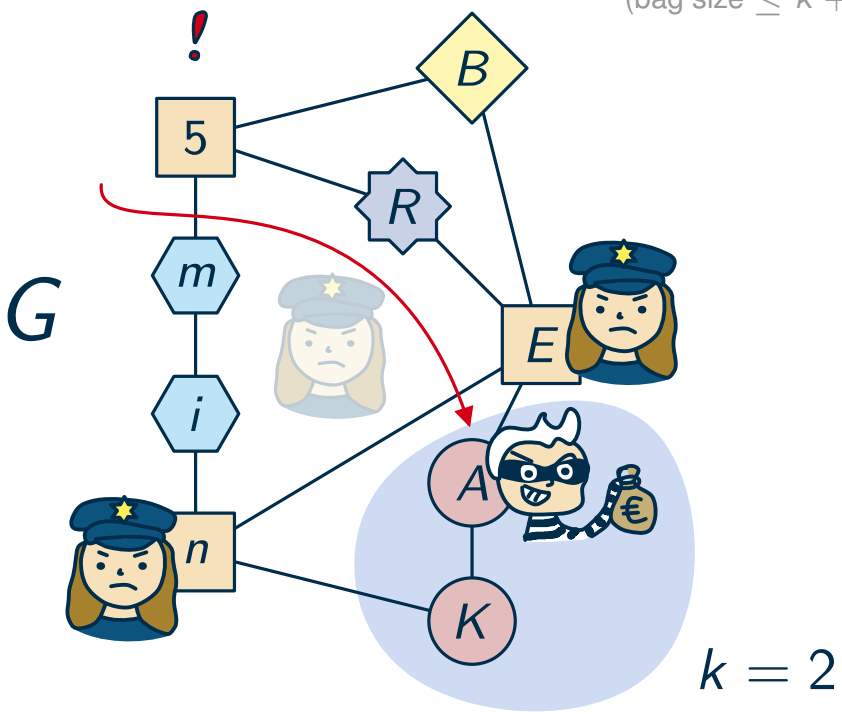
Cops and Robber

In a connected graph with treewidth $\leq k$, there is a winning strategy for $k+1$ cops.

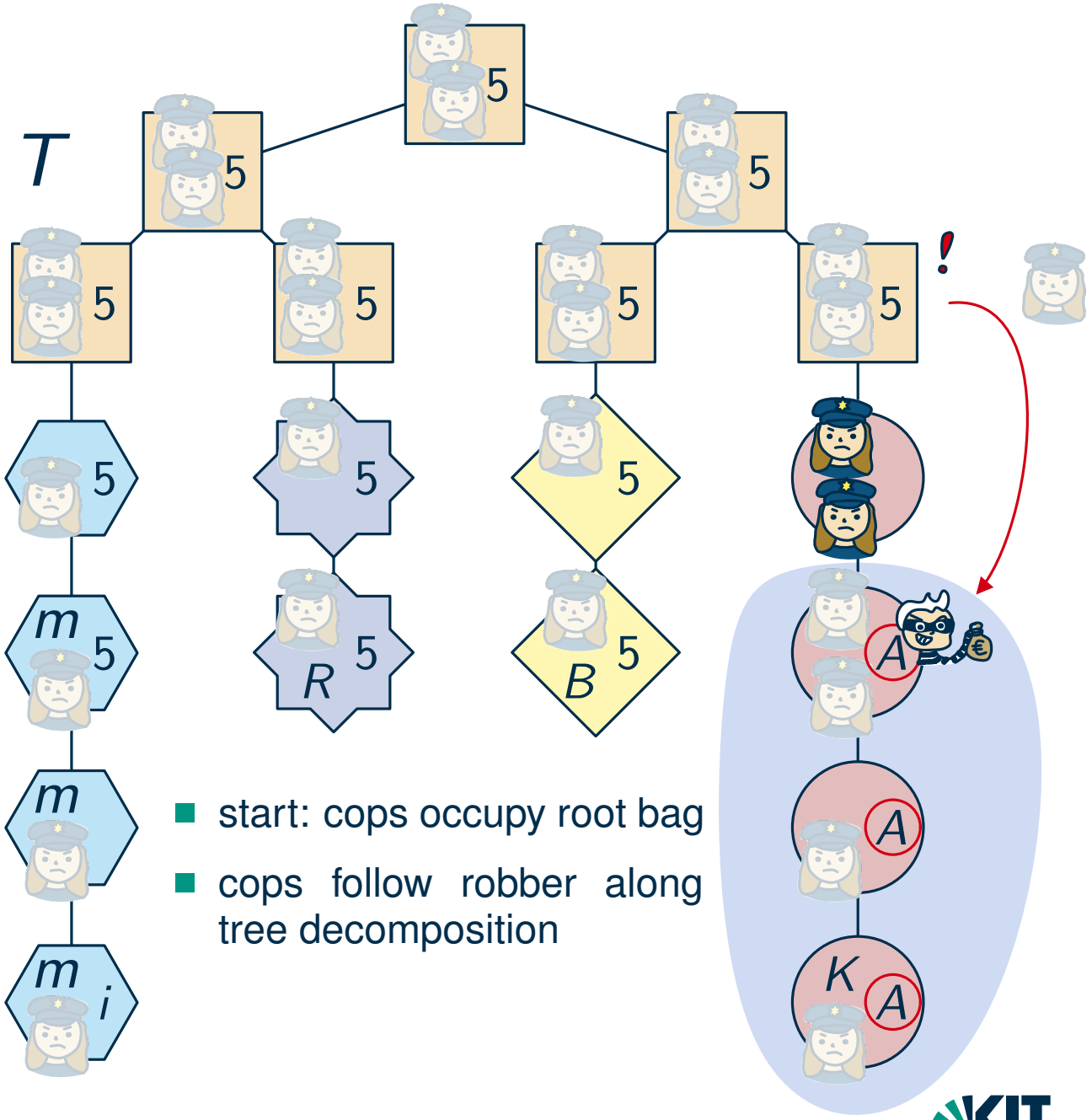
graph G with treewidth $\leq k$

$\Longrightarrow$ nice tree decomposition of width $\leq k$

(bag size $\leq k+1$)



$k = 2$



- start: cops occupy root bag
- cops follow robber along tree decomposition

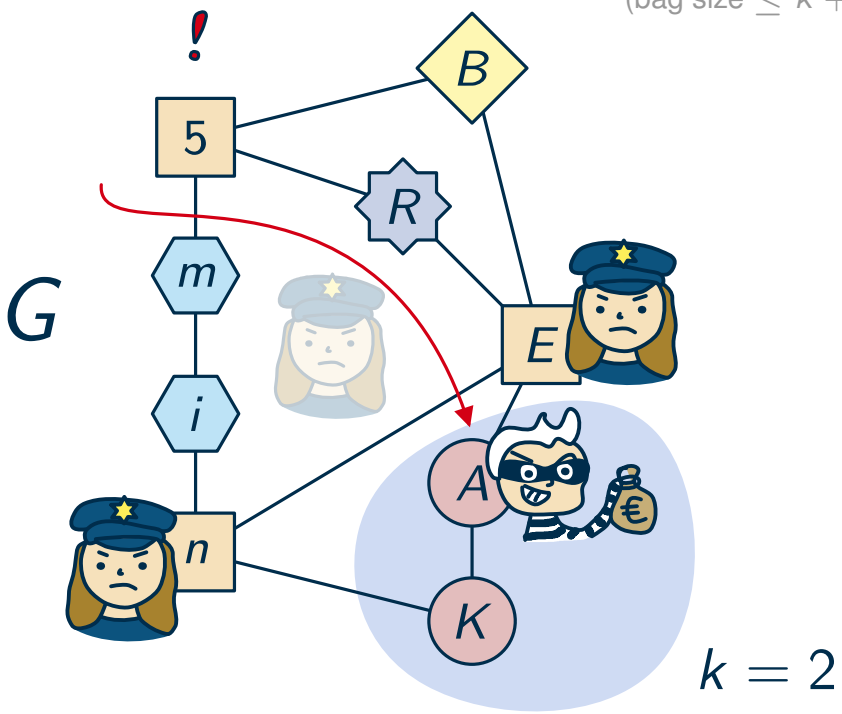
Cops and Robber

In a connected graph with treewidth $\leq k$, there is a winning strategy for $k+1$ cops.

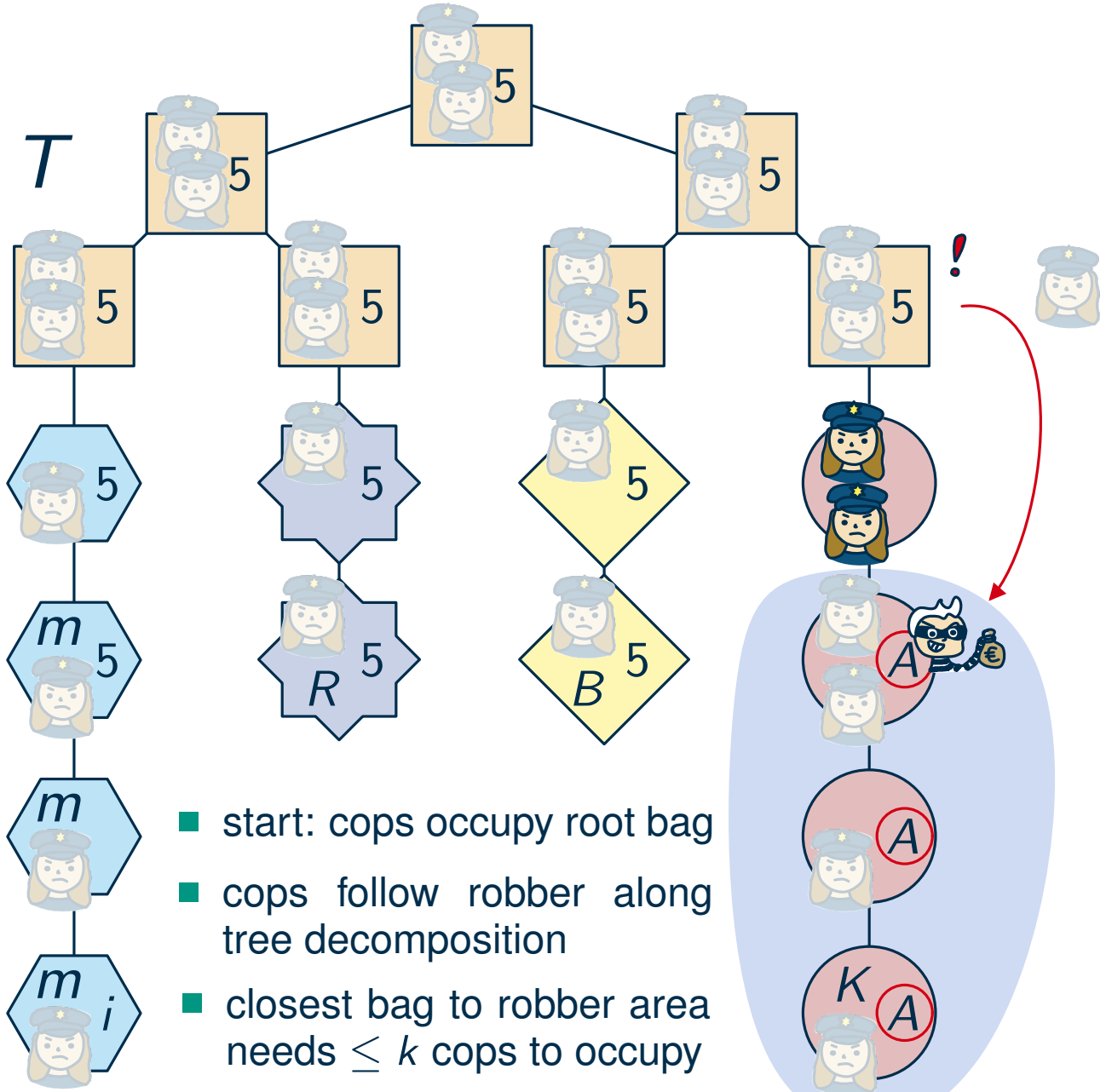
graph G with treewidth $\leq k$

$\Longrightarrow$ nice tree decomposition of width $\leq k$

(bag size $\leq k+1$)



$k = 2$



- start: cops occupy root bag
- cops follow robber along tree decomposition
- closest bag to robber area needs $\leq k$ cops to occupy

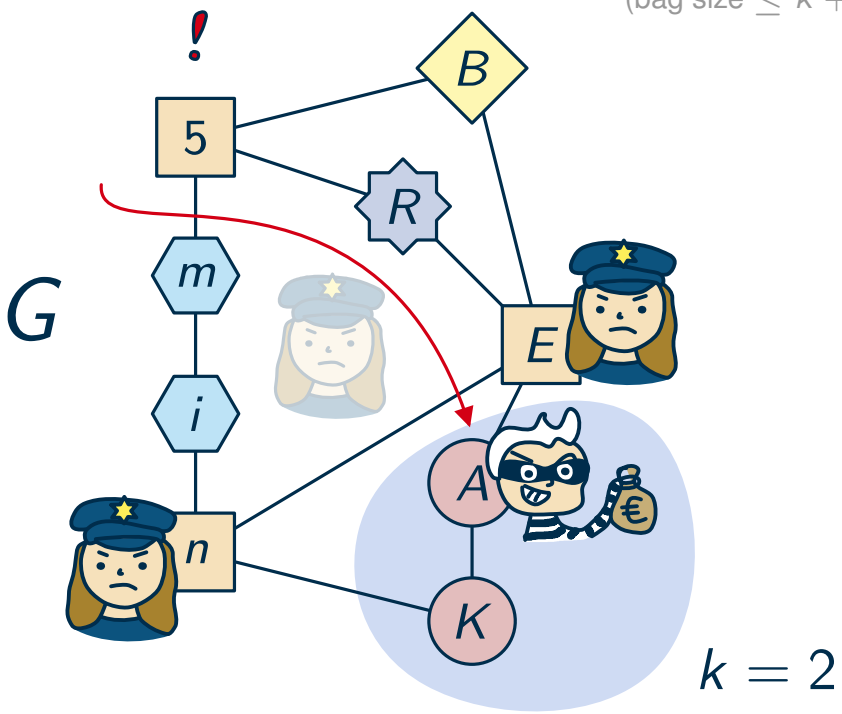
Cops and Robber

In a connected graph with treewidth $\leq k$, there is a winning strategy for $k+1$ cops.

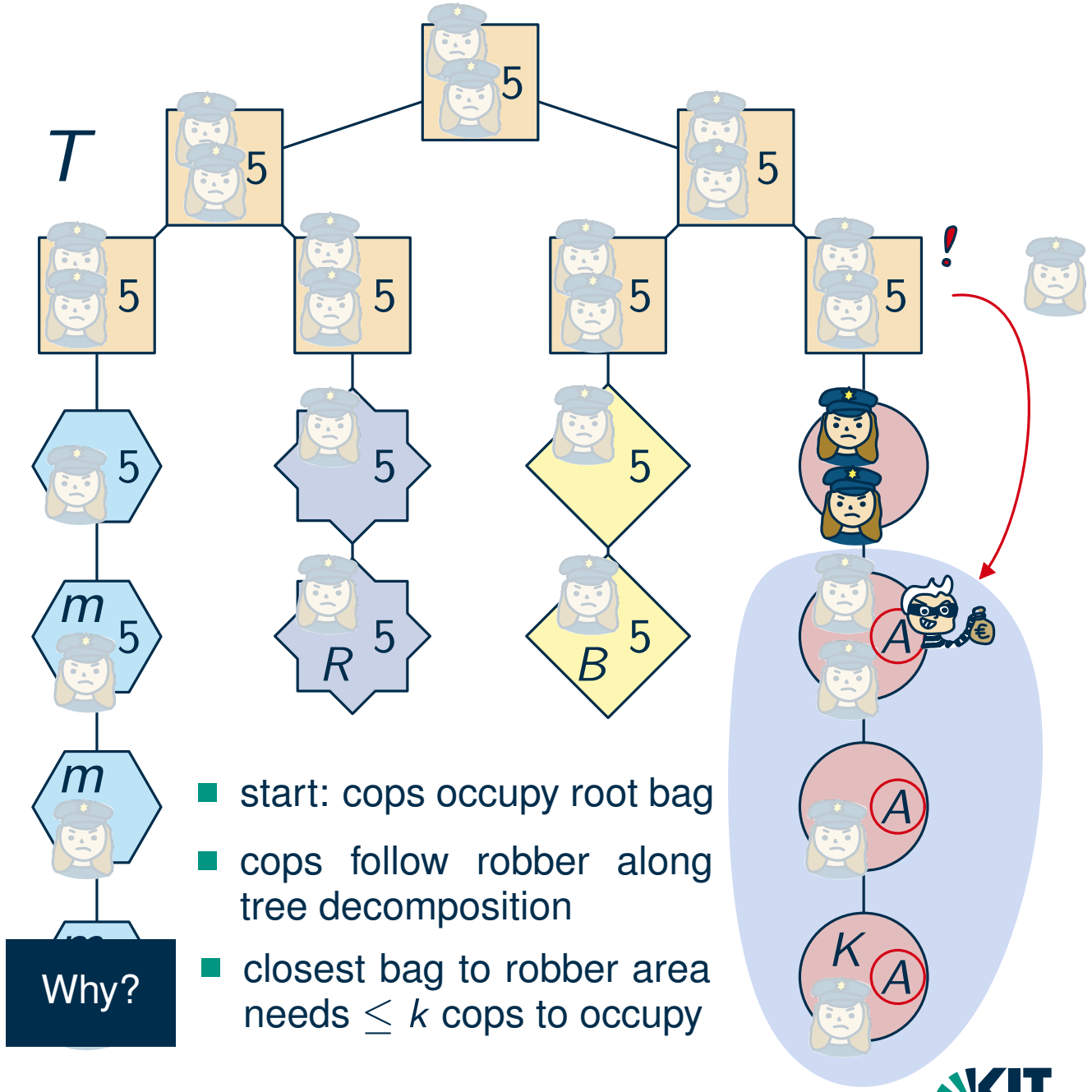
graph G with treewidth $\leq k$

$\Longrightarrow$ nice tree decomposition of width $\leq k$

(bag size $\leq k+1$)



$k = 2$



- start: cops occupy root bag
- cops follow robber along tree decomposition
- closest bag to robber area needs $\leq k$ cops to occupy

Why?

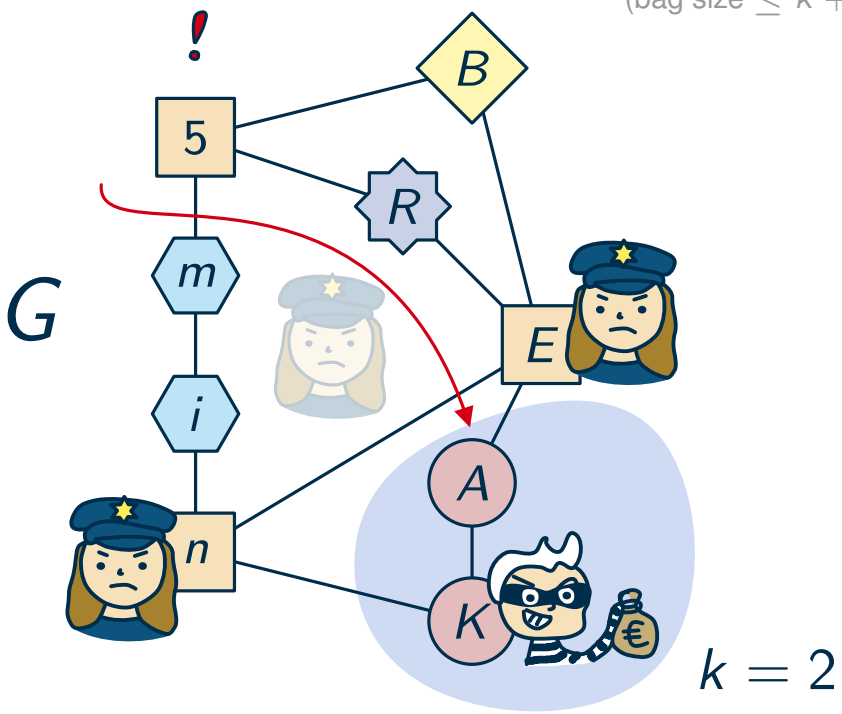
Cops and Robber

In a connected graph with treewidth $\leq k$, there is a winning strategy for $k+1$ cops.

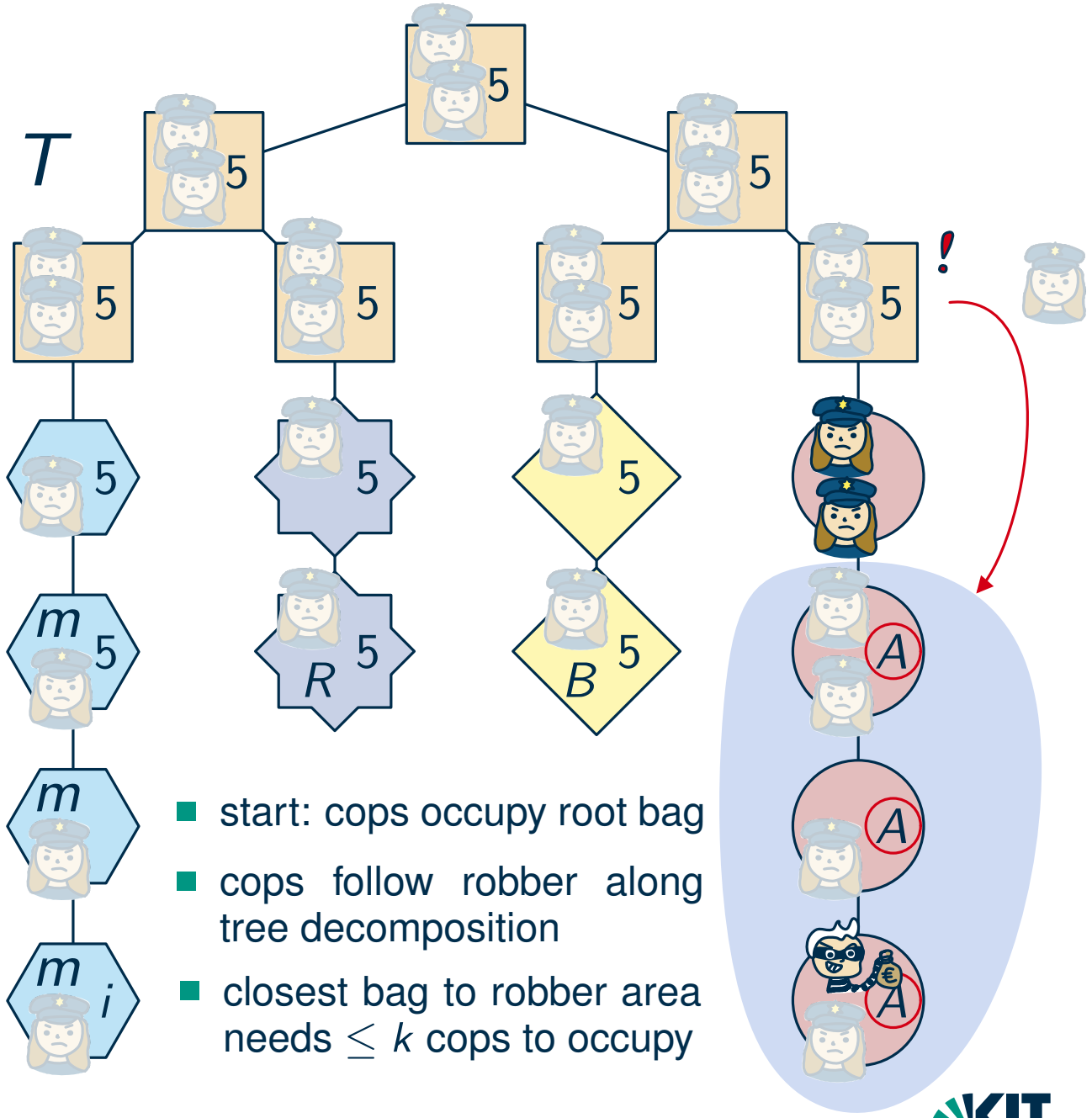
graph G with treewidth $\leq k$

$\Longrightarrow$ nice tree decomposition of width $\leq k$

(bag size $\leq k+1$)



$k = 2$



- start: cops occupy root bag
- cops follow robber along tree decomposition
- closest bag to robber area needs $\leq k$ cops to occupy

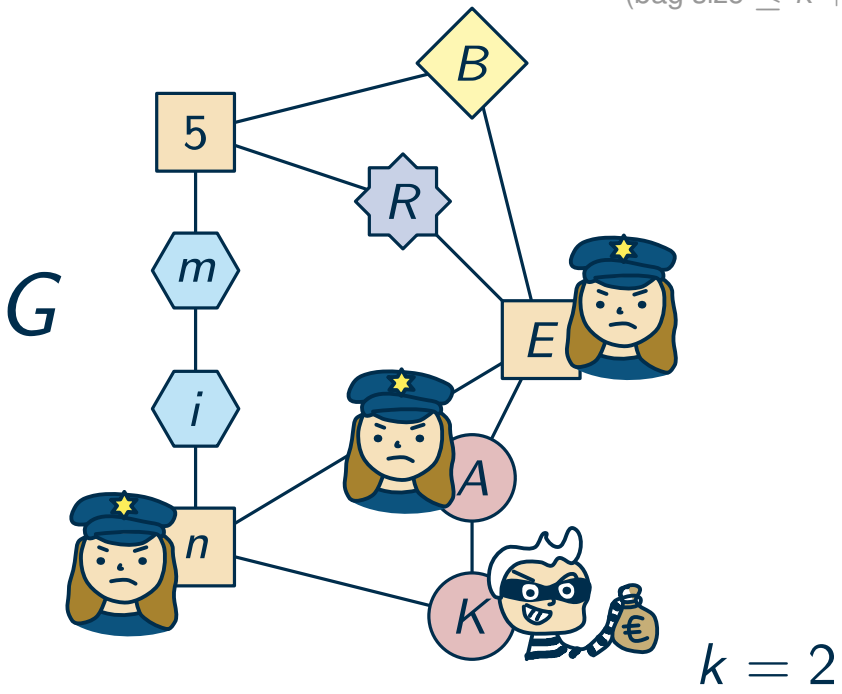
Cops and Robber

In a connected graph with treewidth $\leq k$, there is a winning strategy for $k+1$ cops.

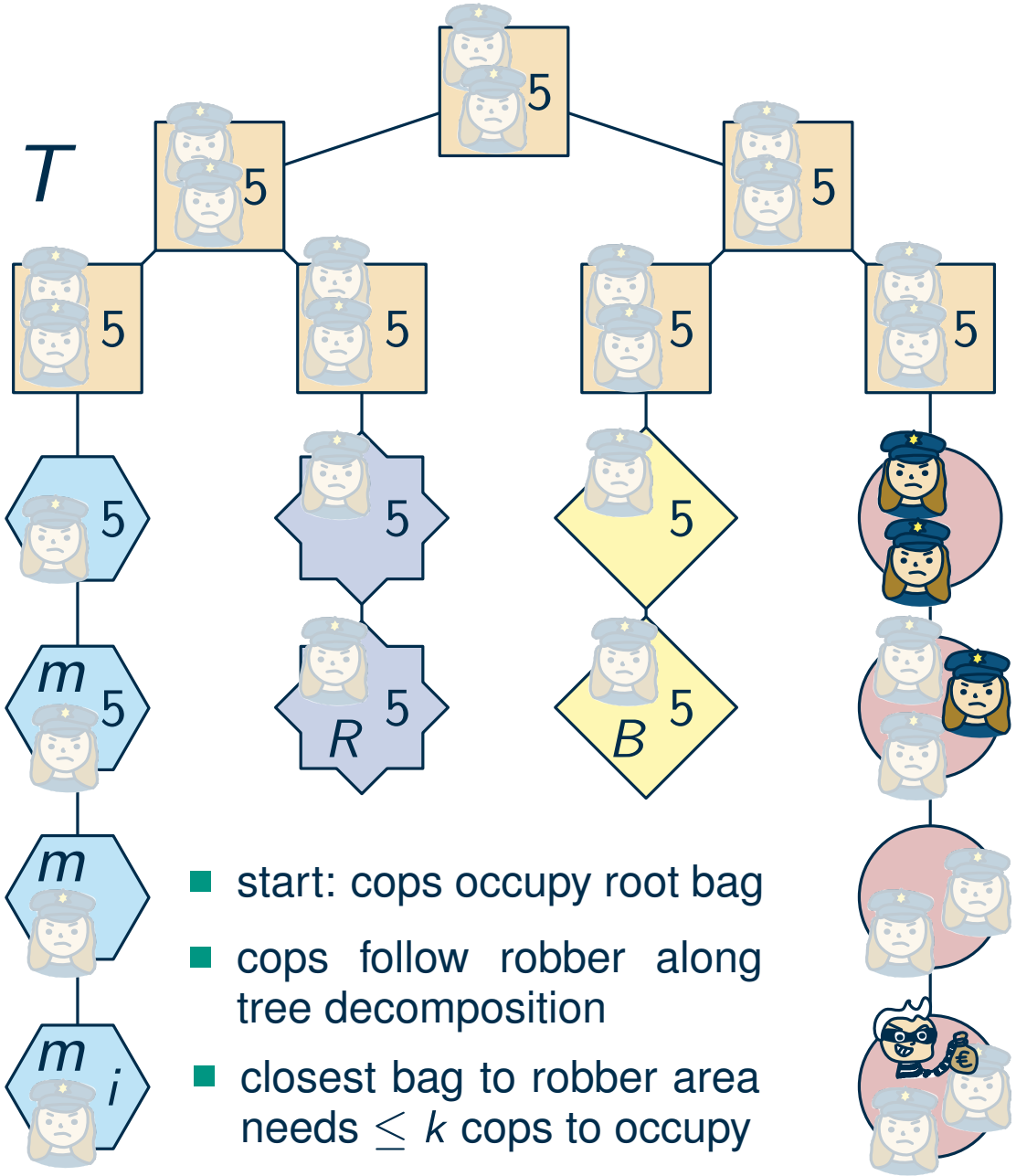
graph G with treewidth $\leq k$

$\Longrightarrow$ nice tree decomposition of width $\leq k$

(bag size $\leq k+1$)



$k = 2$



- start: cops occupy root bag
- cops follow robber along tree decomposition
- closest bag to robber area needs $\leq k$ cops to occupy

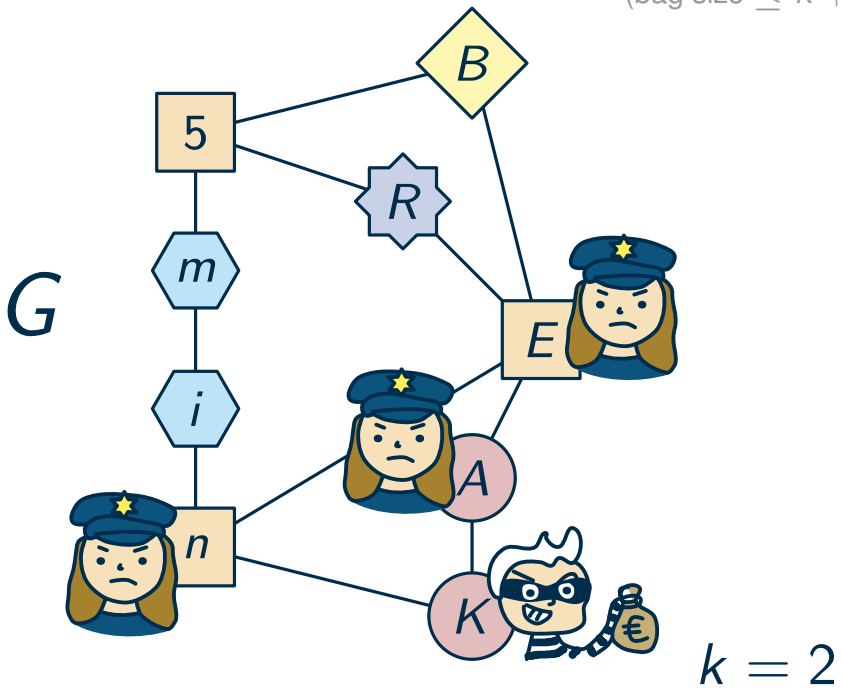
Cops and Robber

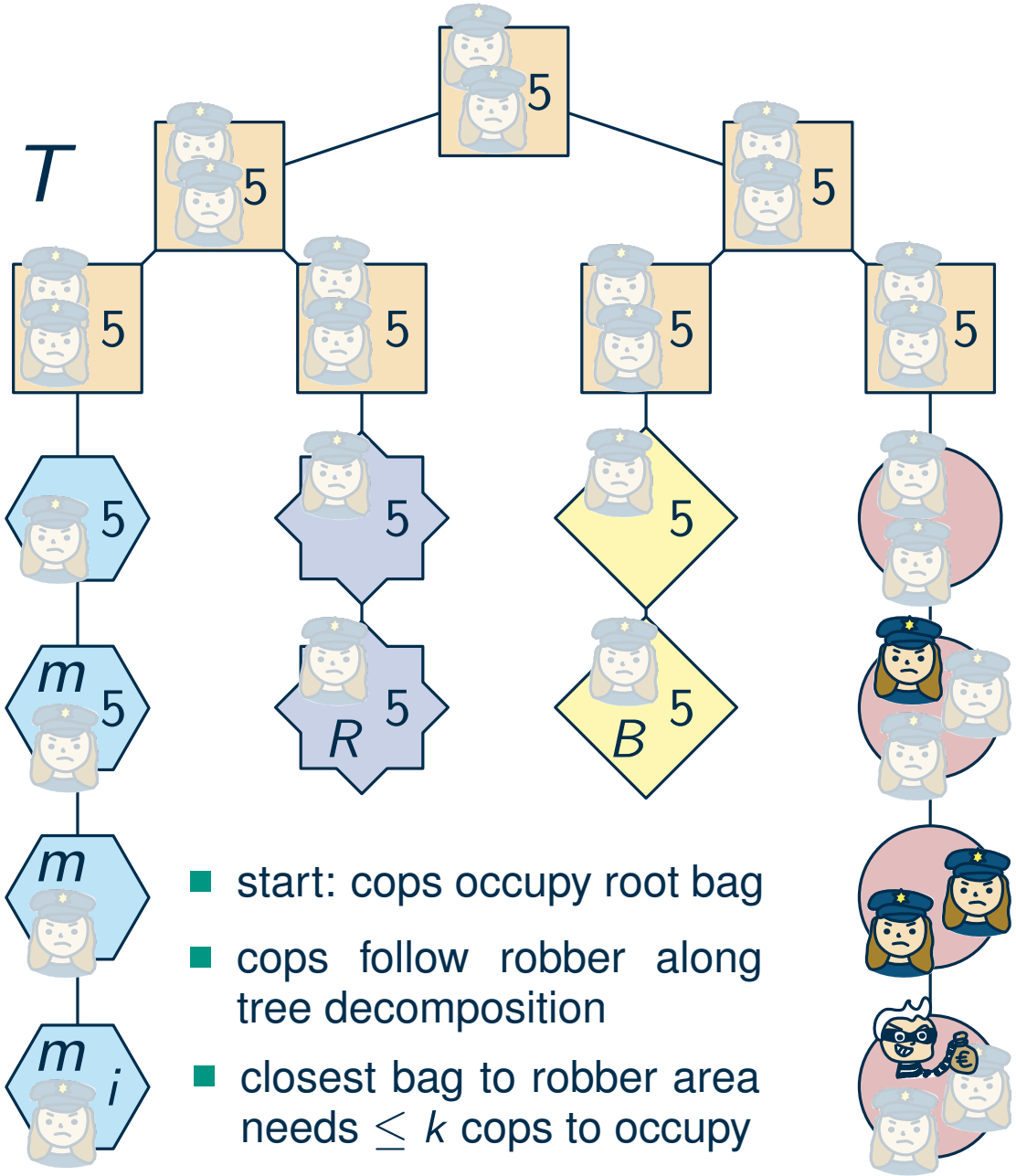
In a connected graph with treewidth $\leq k$, there is a winning strategy for $k+1$ cops.

graph G with treewidth $\leq k$

$\Longrightarrow$ nice tree decomposition of width $\leq k$

(bag size $\leq k+1$)





- start: cops occupy root bag
- cops follow robber along tree decomposition
- closest bag to robber area needs $\leq k$ cops to occupy

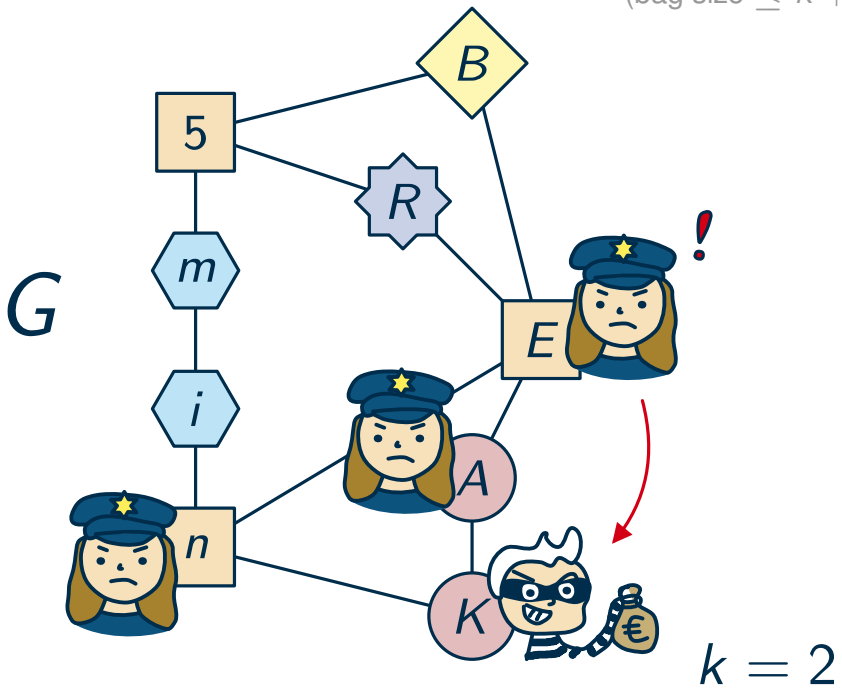
Cops and Robber

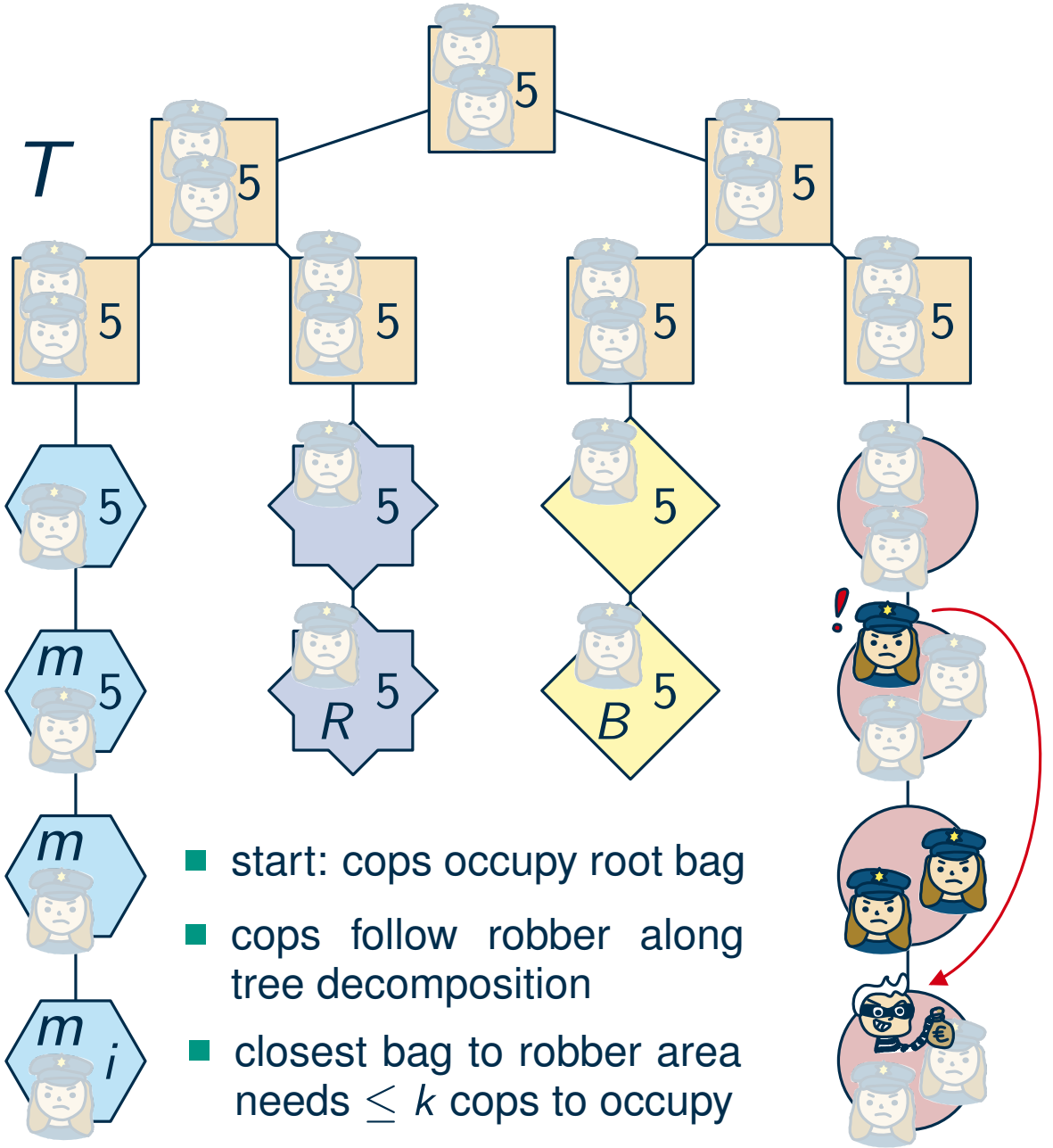
In a connected graph with treewidth $\leq k$, there is a winning strategy for $k+1$ cops.

graph G with treewidth $\leq k$

$\Longrightarrow$ nice tree decomposition of width $\leq k$

(bag size $\leq k+1$)





- start: cops occupy root bag
- cops follow robber along tree decomposition
- closest bag to robber area needs $\leq k$ cops to occupy

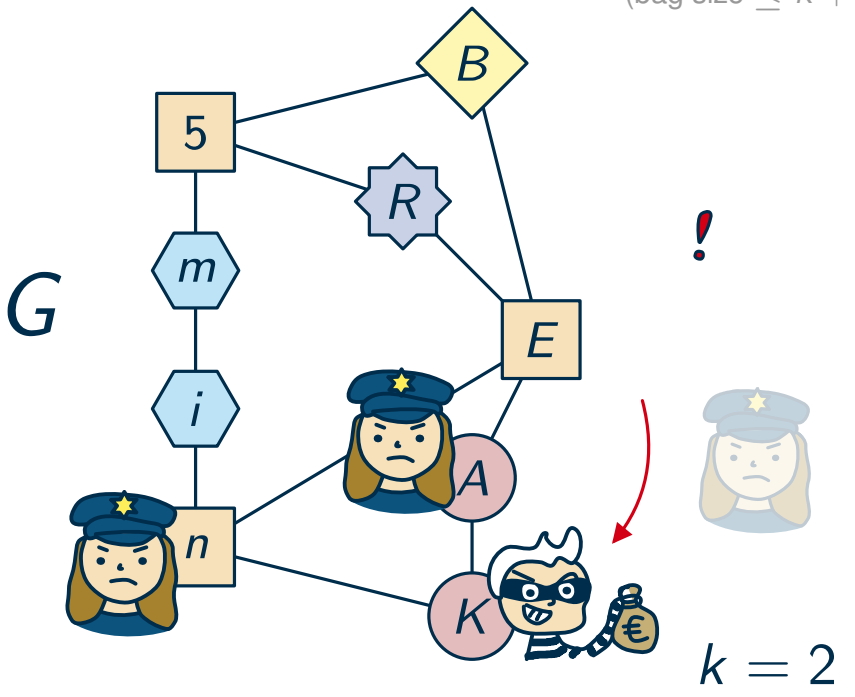
Cops and Robber

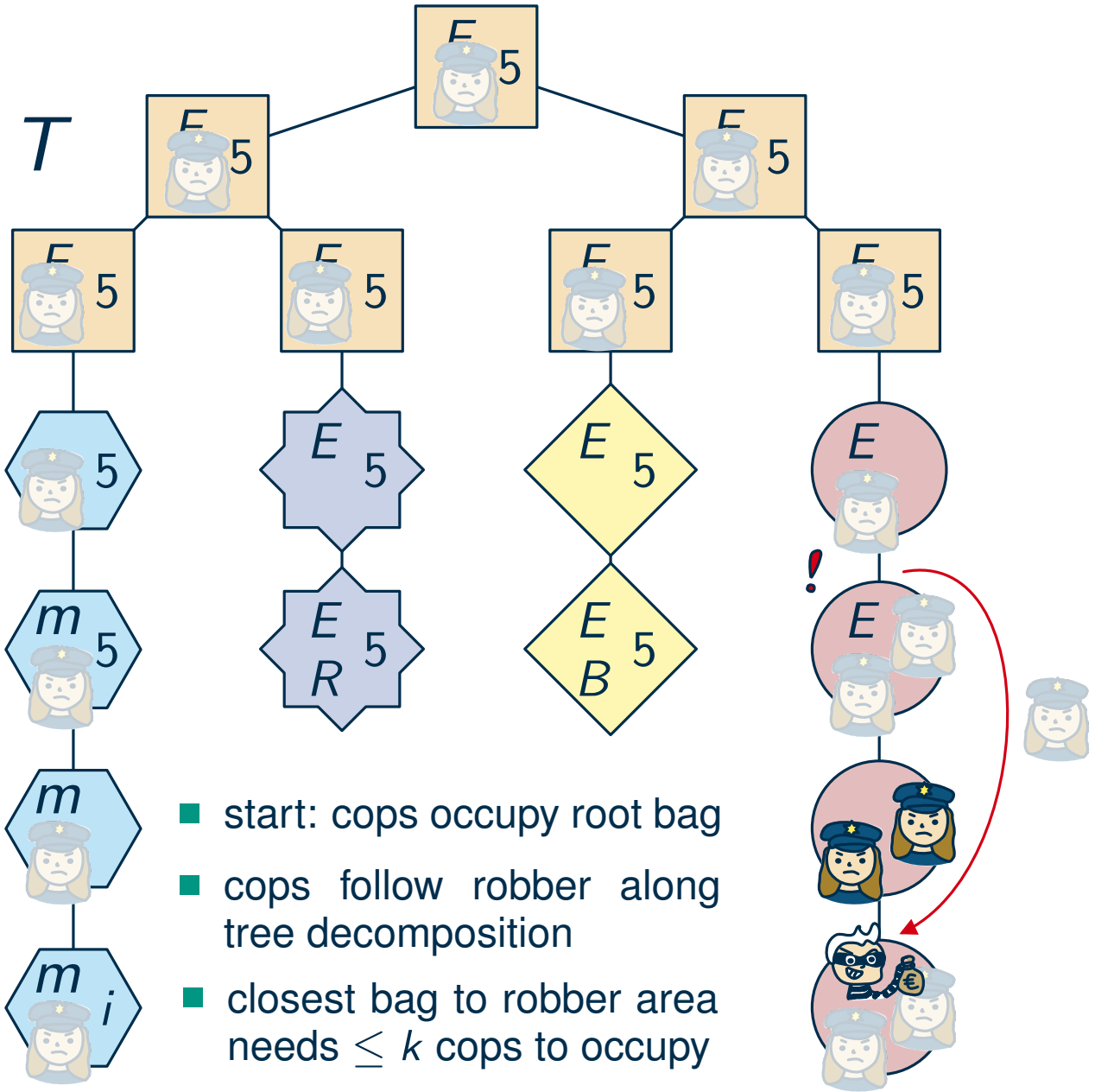
In a connected graph with treewidth $\leq k$, there is a winning strategy for $k+1$ cops.

graph G with treewidth $\leq k$

$\Longrightarrow$ nice tree decomposition of width $\leq k$

(bag size $\leq k+1$)





- start: cops occupy root bag
- cops follow robber along tree decomposition
- closest bag to robber area needs $\leq k$ cops to occupy

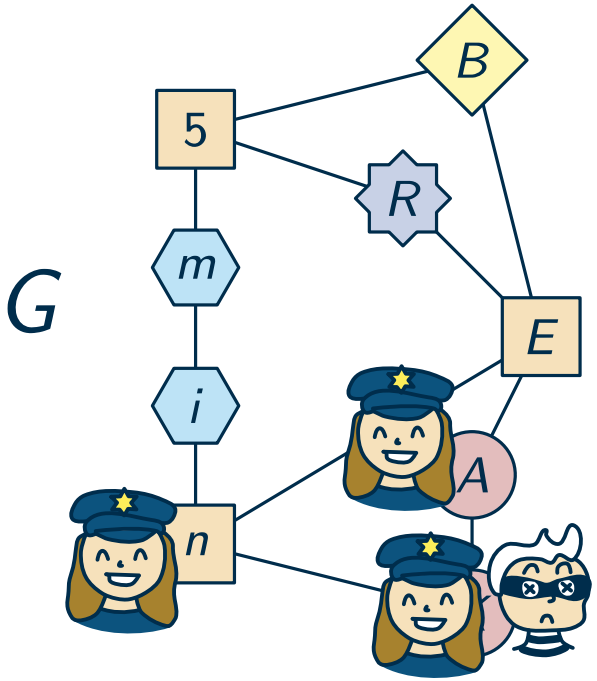
Cops and Robber

In a connected graph with treewidth $\leq k$, there is a winning strategy for $k+1$ cops.

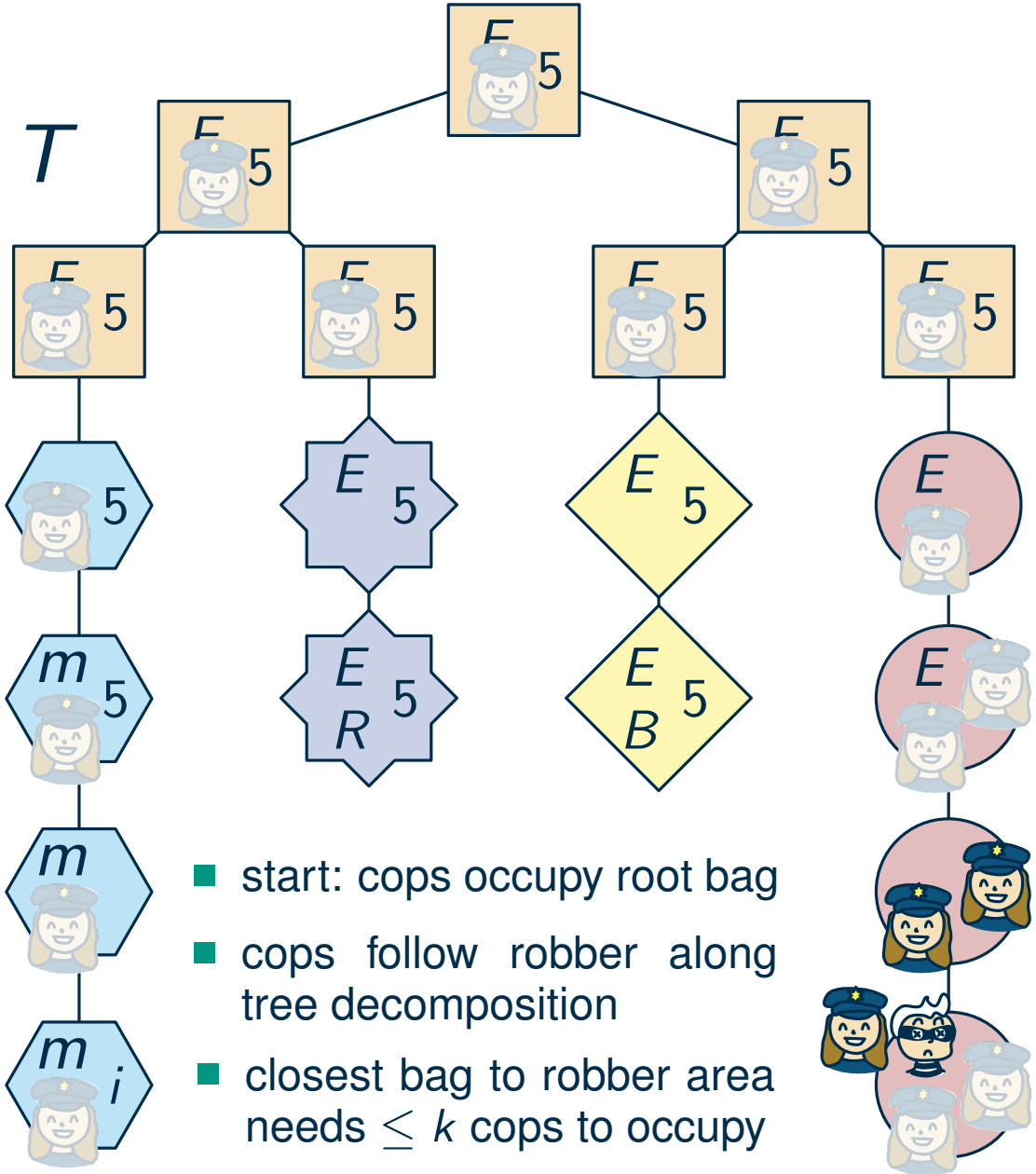
graph G with treewidth $\leq k$

$\Longrightarrow$ nice tree decomposition of width $\leq k$

(bag size $\leq k+1$)



$k = 2$



- start: cops occupy root bag
- cops follow robber along tree decomposition
- closest bag to robber area needs $\leq k$ cops to occupy

Cops and Robber

other direction?

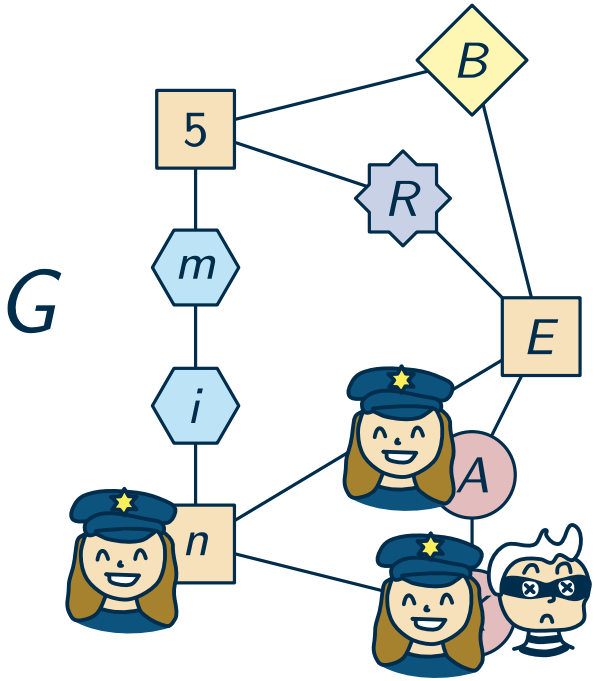
Graph Searching and a Min-Max Theorem for Tree-Width (Seymour & Thomas)
https://www.sciencedirect.com/science/article/pii/S0095895683710270

In a connected graph with treewidth $\leq k$, there is a winning strategy for $k+1$ cops.

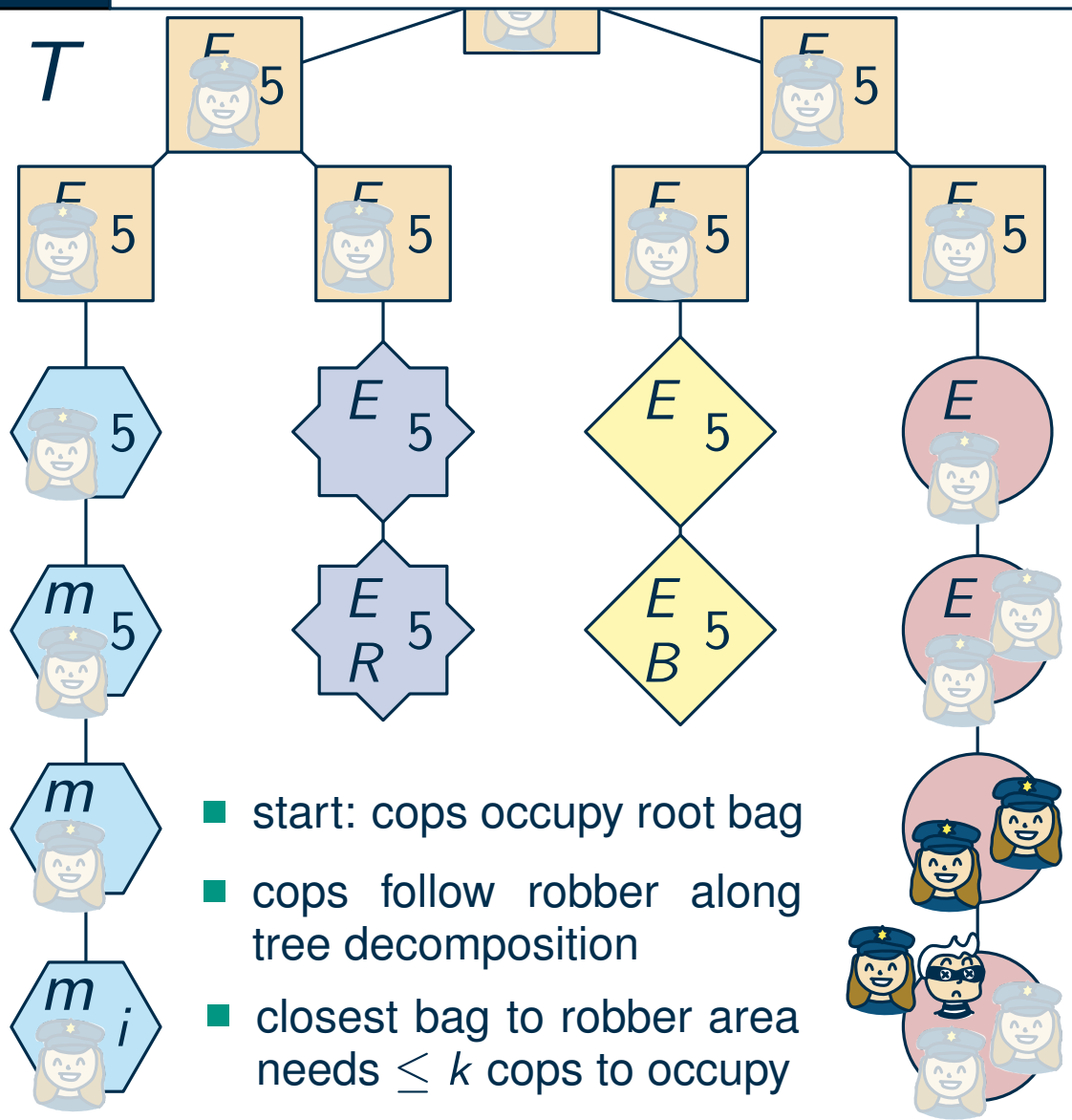
graph G with treewidth $\leq k$

$\Longrightarrow$ nice tree decomposition of width $\leq k$

(bag size $\leq k+1$)



$k = 2$



- start: cops occupy root bag
- cops follow robber along tree decomposition
- closest bag to robber area needs $\leq k$ cops to occupy