Parameterized Algorithms

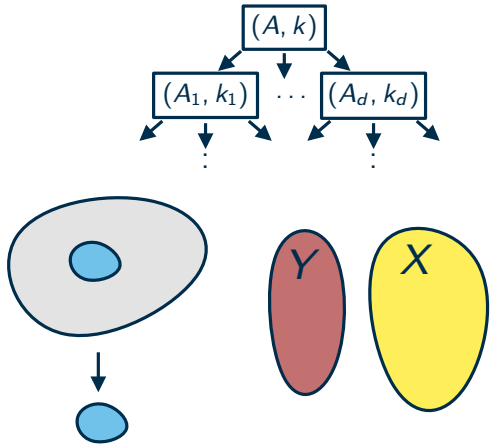
Color Coding

Thomas Bläsius

Content

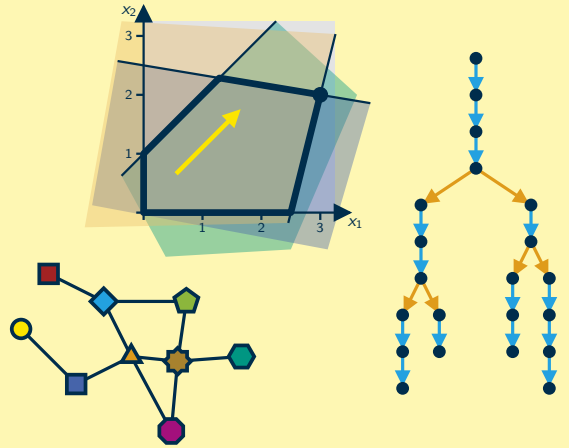
Basic toolbox

- bounded search trees
- kernelization
- iterative compression



Extended toolbox

- linear programs
- branch-and-reduce
- color coding

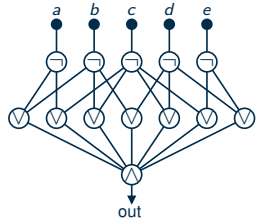


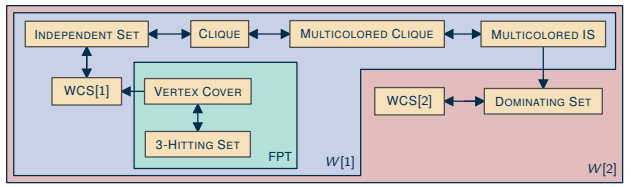
Tree width

- dynamic programming
- chordal and planar graphs
- Courcelle's theorem

Lower bonds

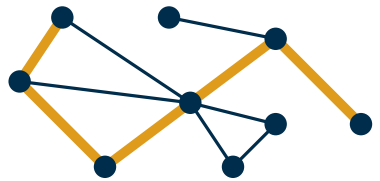
- kernel lower bounds
- parameterized reductions
- circuits and the W-hierarchy
- ETH and SETH





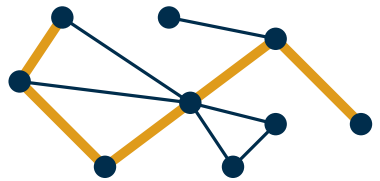
Best detours

Problem: LONGEST PATH
Given a graph G and a parameter k. Does G have a k-vertex path?



Best detours

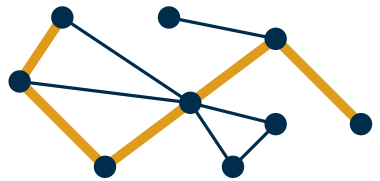
Problem: LONGEST PATH
Given a graph G and a parameter k. Does G have a k-vertex path?



Complexity: NP-hard: reduction from HAMILTONIAN PATH $\rightarrow$ today: FPT

Best detours

Problem: LONGEST PATH
Given a graph G and a parameter k. Does G have a k-vertex path?



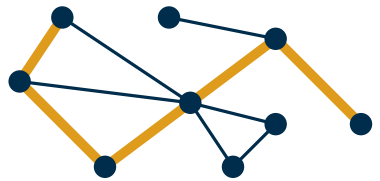
Complexity: NP-hard: reduction from HAMILTONIAN PATH $\rightarrow$ today: FPT

For directed acyclic graphs (DAGs)

Can we do this in polynomial time?

Best detours

Problem: LONGEST PATH
Given a graph G and a parameter k. Does G have a k-vertex path?



Complexity: NP-hard: reduction from HAMILTONIAN PATH $\rightarrow$ today: FPT

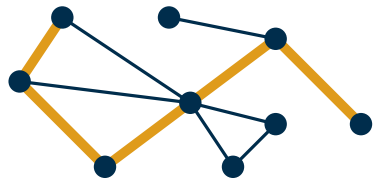
For directed acyclic graphs (DAGs)

- dynamic program

Can we do this in polynomial time?

Best detours

Problem: LONGEST PATH
Given a graph G and a parameter k. Does G have a k-vertex path?

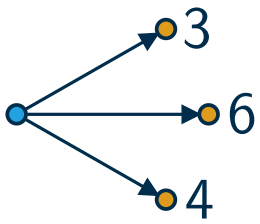


Complexity: NP-hard: reduction from HAMILTONIAN PATH $\rightarrow$ today: FPT

For directed acyclic graphs (DAGs)

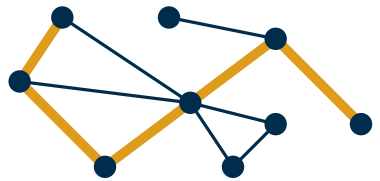
- dynamic program
  - for every vertex v: longest path starting at v
  - iterate vertices according to topological ordering

Can we do this in polynomial time?



Best detours

Problem: LONGEST PATH
Given a graph G and a parameter k. Does G have a k-vertex path?

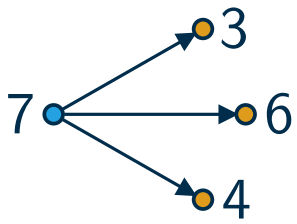


Complexity: NP-hard: reduction from HAMILTONIAN PATH $\rightarrow$ today: FPT

For directed acyclic graphs (DAGs)

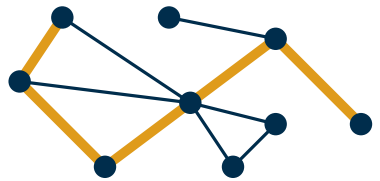
- dynamic program
  - for every vertex v: longest path starting at v
  - iterate vertices according to topological ordering

Can we do this in polynomial time?



Best detours

Problem: LONGEST PATH
Given a graph G and a parameter k. Does G have a k-vertex path?

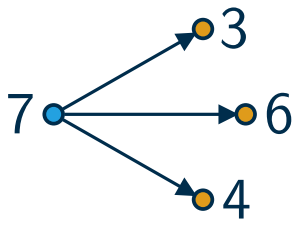


Complexity: NP-hard: reduction from HAMILTONIAN PATH $\rightarrow$ today: FPT

For directed acyclic graphs (DAGs)

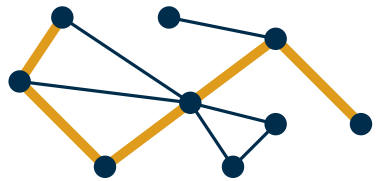
- dynamic program
  - for every vertex v: longest path starting at v
  - iterate vertices according to topological ordering
- alternative: matrix multiplication

Can we do this in polynomial time?



Best detours

Problem: LONGEST PATH
Given a graph G and a parameter k. Does G have a k-vertex path?

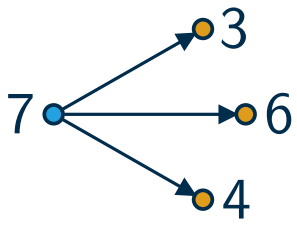


Complexity: NP-hard: reduction from HAMILTONIAN PATH $\rightarrow$ today: FPT

Can we do this in polynomial time?

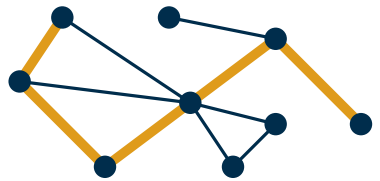
For directed acyclic graphs (DAGs)

- dynamic program
  - for every vertex v: longest path starting at v
  - iterate vertices according to topological ordering
- alternative: matrix multiplication
  - let A be the adjacency matrix and consider A^k
  - entry (u, v) equals the number of paths of length k from u to v



Best detours

Problem: LONGEST PATH
Given a graph G and a parameter k. Does G have a k-vertex path?

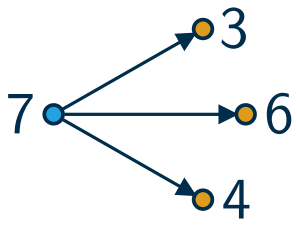


Complexity: NP-hard: reduction from HAMILTONIAN PATH $\rightarrow$ today: FPT

For directed acyclic graphs (DAGs)

- dynamic program
  - for every vertex v: longest path starting at v
  - iterate vertices according to topological ordering
- alternative: matrix multiplication
  - let A be the adjacency matrix and consider A^k
  - entry (u, v) equals the number of paths of length k from u to v

Can we do this in polynomial time?

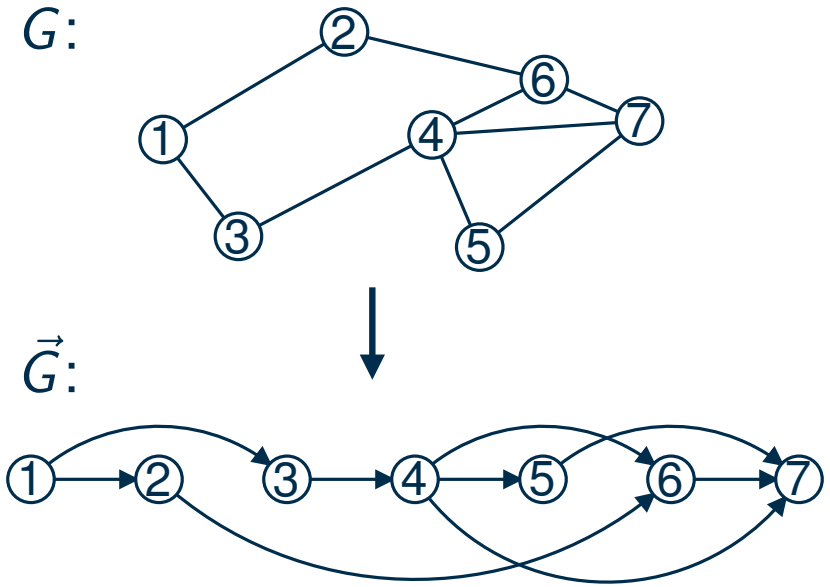


Why does this not work for undirected graphs?

Preventing cycles

Idea for undirected graphs

- transform the undirected graph G into a DAG $\vec{G}$
- find a longest path in the DAG $\vec{G}$

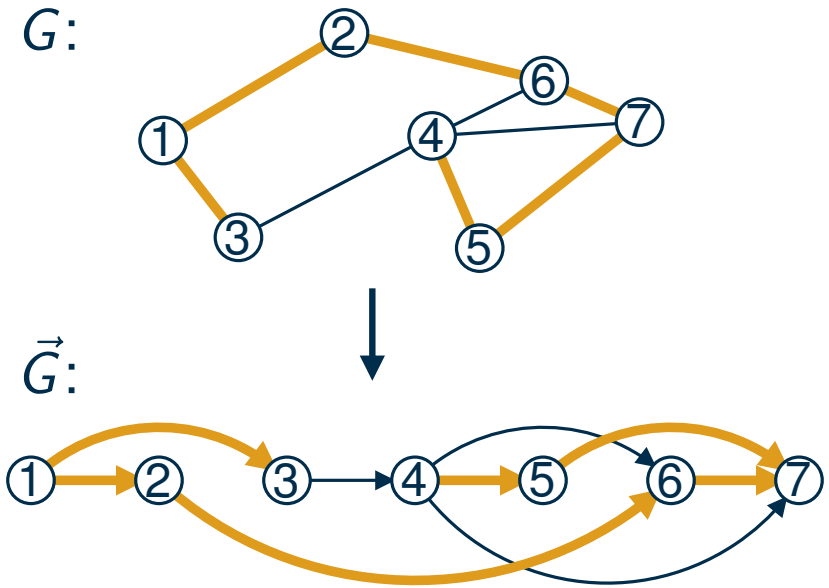


Preventing cycles

Idea for undirected graphs

- transform the undirected graph G into a DAG $\vec{G}$
- find a longest path in the DAG $\vec{G}$

Problem: not all paths in G are directed paths in $\vec{G}$



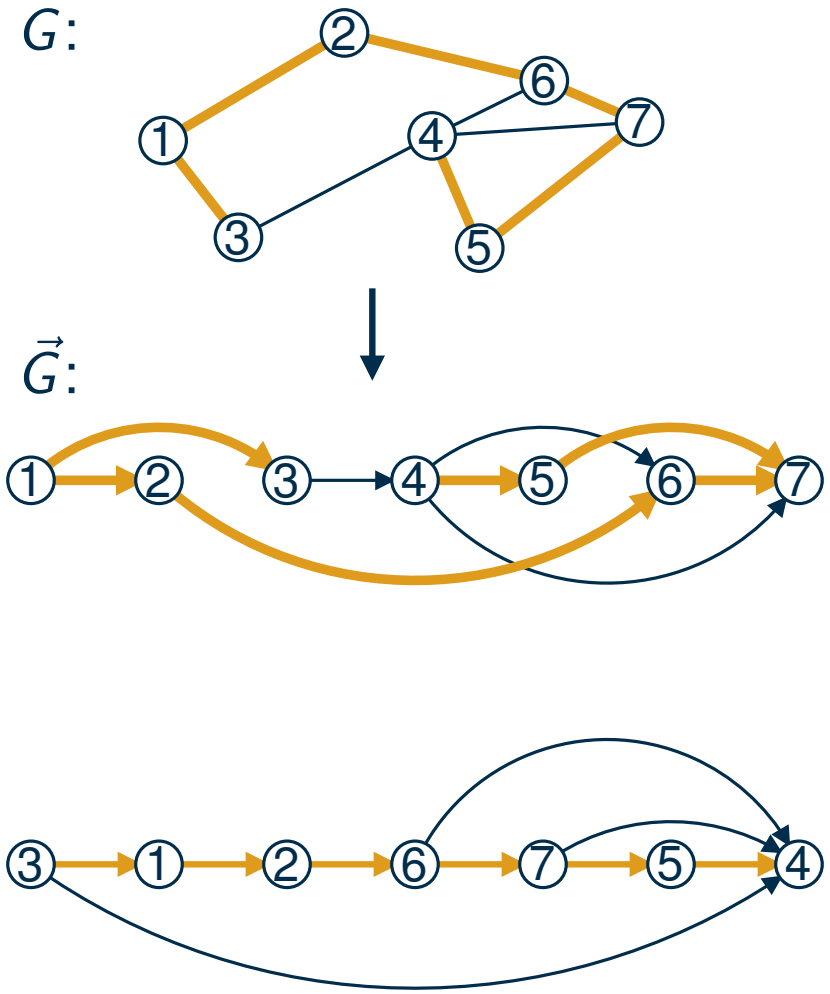
Preventing cycles

Idea for undirected graphs

- transform the undirected graph G into a DAG $\vec{G}$
- find a longest path in the DAG $\vec{G}$

Problem: not all paths in G are directed paths in $\vec{G}$

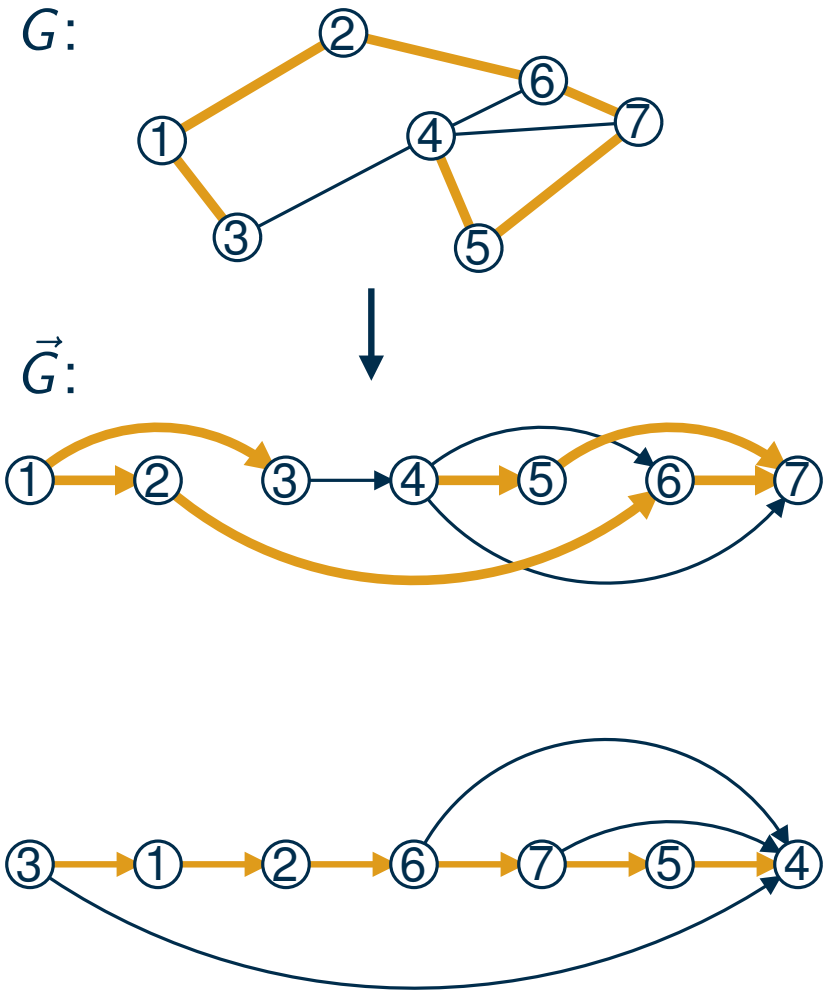
But: there is always a good orientation



Preventing cycles

Idea for undirected graphs

- transform the undirected graph G into a DAG $\vec{G}$
- find a longest path in the DAG $\vec{G}$



Problem: not all paths in G are directed paths in $\vec{G}$

But: there is always a good orientation

How do we find a good orientation?

Randomization

Construction of the DAG $\vec{G}$

- choose random order for the vertices of G
- direct all edges in G from front to back

Randomization

Construction of the DAG $\vec{G}$

- choose random order for the vertices of G
- direct all edges in G from front to back

Theorem
Let P be a k-vertex path k in G. Then P is a directed path in $\vec{G}$ with probability $\frac{2}{k!}$.

Randomization

Construction of the DAG $\vec{G}$

- choose random order for the vertices of G
- direct all edges in G from front to back

Proof

- let $P = v_1, \ldots, v_k$

Theorem
Let P be a k-vertex path k in G. Then P is a directed path in $\vec{G}$ with probability $\frac{2}{k!}$.

Randomization

Construction of the DAG $\vec{G}$

- choose random order for the vertices of G
- direct all edges in G from front to back

Theorem
Let P be a k-vertex path k in G. Then P is a directed path in $\vec{G}$ with probability $\frac{2}{k!}$.

Proof

- let $P = v_1, \ldots, v_k$
- every order on $\{v_1, \ldots, v_k\}$ has the same probability

Randomization

Construction of the DAG $\vec{G}$

- choose random order for the vertices of G
- direct all edges in G from front to back

Theorem
Let P be a k-vertex path k in G. Then P is a directed path in $\vec{G}$ with probability $\frac{2}{k!}$.

Proof

- let $P = v_1, \ldots, v_k$
- every order on $\{v_1, \ldots, v_k\}$ has the same probability
- $k!$ events, each happening with the same probability $\frac{1}{k!}$

Randomization

Construction of the DAG $\vec{G}$

- choose random order for the vertices of G
- direct all edges in G from front to back

Theorem
Let P be a k-vertex path k in G. Then P is a directed path in $\vec{G}$ with probability $\frac{2}{k!}$.

Proof

- let $P = v_1, \ldots, v_k$
- every order on $\{v_1, \ldots, v_k\}$ has the same probability
- $k!$ events, each happening with the same probability $\frac{1}{k!}$
- two of these events are good: $v_1, \ldots, v_k$ and $v_k, \ldots, v_1$

Randomization

Construction of the DAG $\vec{G}$

- choose random order for the vertices of G
- direct all edges in G from front to back

Theorem
Let P be a k-vertex path k in G. Then P is a directed path in $\vec{G}$ with probability $\frac{2}{k!}$.

Proof

- let $P = v_1, \ldots, v_k$
- every order on $\{v_1, \ldots, v_k\}$ has the same probability
- $k!$ events, each happening with the same probability $\frac{1}{k!}$
- two of these events are good: $v_1, \ldots, v_k$ and $v_k, \ldots, v_1$

Comments

- success probability only depends on k

Randomization

Construction of the DAG $\vec{G}$

- choose random order for the vertices of G
- direct all edges in G from front to back

Theorem
Let P be a k-vertex path k in G. Then P is a directed path in $\vec{G}$ with probability $\frac{2}{k!}$.

Proof

- let $P = v_1, \ldots, v_k$
- every order on $\{v_1, \ldots, v_k\}$ has the same probability
- $k!$ events, each happening with the same probability $\frac{1}{k!}$
- two of these events are good: $v_1, \ldots, v_k$ and $v_k, \ldots, v_1$

Comments

- success probability only depends on k
- probability can be boosted by repeating the process
- to get constant probability: number of repetitions only depends on k

Randomized FPT algorithm

Theorem
With $k!$ independent trials of orienting $\vec{G}$, the probability to correctly solve LONGEST PATH for (G, k) is larger than $\frac{1}{2}$.

Theorem
Let P be a k-vertex path k in G. Then P is a directed path in $\vec{G}$ with probability $\frac{2}{k!}$.

Randomized FPT algorithm

Theorem
With $k!$ independent trials of orienting $\vec{G}$, the probability to correctly solve LONGEST PATH for (G, k) is larger than $\frac{1}{2}$.

Theorem
Let P be a k-vertex path k in G. Then P is a directed path in $\vec{G}$ with probability $\frac{2}{k!}$.

Proof

- `no`-instances are always correctly recognized

Randomized FPT algorithm

Theorem
With $k!$ independent trials of orienting $\vec{G}$, the probability to correctly solve LONGEST PATH for (G, k) is larger than $\frac{1}{2}$.

Theorem
Let P be a k-vertex path k in G. Then P is a directed path in $\vec{G}$ with probability $\frac{2}{k!}$.

Proof

- `no`-instances are always correctly recognized
- the algorithm fails on `yes`-instances with probability:

Randomized FPT algorithm

Theorem
With $k!$ independent trials of orienting $\vec{G}$, the probability to correctly solve LONGEST PATH for (G, k) is larger than $\frac{1}{2}$.

Theorem
Let P be a k-vertex path k in G. Then P is a directed path in $\vec{G}$ with probability $\frac{2}{k!}$.

Proof

- `no`-instances are always correctly recognized
- the algorithm fails on `yes`-instances with probability:
  - $1 - \frac{2}{k!}$ for an individual trial
  - $\left(1 - \frac{2}{k!}\right)^{k!}$ for $k!$ independent trials

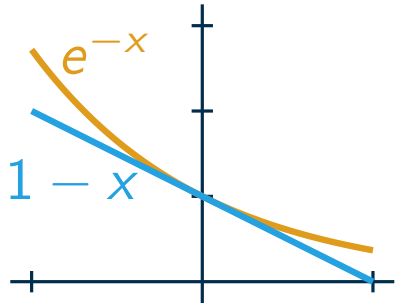
Randomized FPT algorithm

Theorem
With $k!$ independent trials of orienting $\vec{G}$, the probability to correctly solve LONGEST PATH for (G, k) is larger than $\frac{1}{2}$.

Theorem
Let P be a k-vertex path k in G. Then P is a directed path in $\vec{G}$ with probability $\frac{2}{k!}$.

Proof

- `no`-instances are always correctly recognized
- the algorithm fails on `yes`-instances with probability:
  - $1 - \frac{2}{k!}$ for an individual trial
  - $\left(1 - \frac{2}{k!}\right)^{k!}$ for $k!$ independent trials



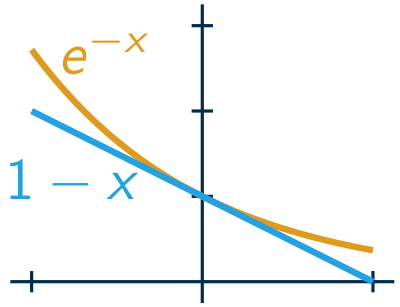
Randomized FPT algorithm

Theorem
With $k!$ independent trials of orienting $\vec{G}$, the probability to correctly solve LONGEST PATH for (G, k) is larger than $\frac{1}{2}$.

Theorem
Let P be a k-vertex path k in G. Then P is a directed path in $\vec{G}$ with probability $\frac{2}{k!}$.

Proof

- `no`-instances are always correctly recognized
- the algorithm fails on `yes`-instances with probability:
  - $1 - \frac{2}{k!}$ for an individual trial
  - $\left(1 - \frac{2}{k!}\right)^{k!}$ for $k!$ independent trials



$$\left(1 - \frac{2}{k!}\right)^{k!} \leq \left(e^{-\frac{2}{k!}}\right)^{k!} = e^{-2} < \frac{1}{2}$$

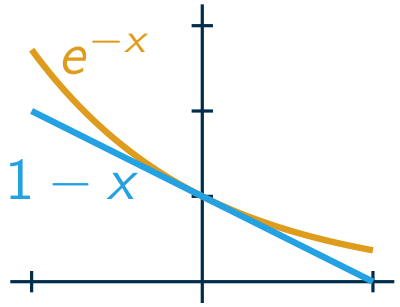
Randomized FPT algorithm

Theorem
With $k!$ independent trials of orienting $\vec{G}$, the probability to correctly solve LONGEST PATH for (G, k) is larger than $\frac{1}{2}$.

Theorem
Let P be a k-vertex path k in G. Then P is a directed path in $\vec{G}$ with probability $\frac{2}{k!}$.

Proof

- `no`-instances are always correctly recognized
- the algorithm fails on `yes`-instances with probability:
  - $1 - \frac{2}{k!}$ for an individual trial
  - $\left(1 - \frac{2}{k!}\right)^{k!}$ for $k!$ independent trials



$$\left(1 - \frac{2}{k!}\right)^{k!} \leq \left(e^{-\frac{2}{k!}}\right)^{k!} = e^{-2} < \frac{1}{2}$$

So what do we have?

- Monte Carlo algorithm with one-sided error

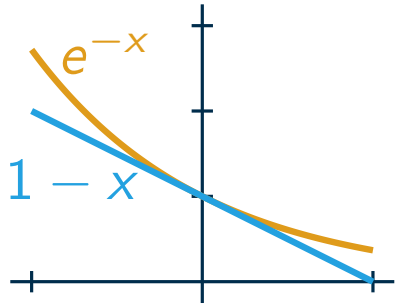
Randomized FPT algorithm

Theorem
With $k!$ independent trials of orienting $\vec{G}$, the probability to correctly solve LONGEST PATH for (G, k) is larger than $\frac{1}{2}$.

Theorem
Let P be a k-vertex path k in G. Then P is a directed path in $\vec{G}$ with probability $\frac{2}{k!}$.

Proof

- `no`-instances are always correctly recognized
- the algorithm fails on `yes`-instances with probability:
  - $1 - \frac{2}{k!}$ for an individual trial
  - $\left(1 - \frac{2}{k!}\right)^{k!}$ for $k!$ independent trials



$$\left(1 - \frac{2}{k!}\right)^{k!} \leq \left(e^{-\frac{2}{k!}}\right)^{k!} = e^{-2} < \frac{1}{2}$$

So what do we have?

- Monte Carlo algorithm with one-sided error
- ℓ repetitions $\rightarrow$ correct with probability $(1 - \frac{1}{2^\ell})$ in time $O(k!m\ell)$

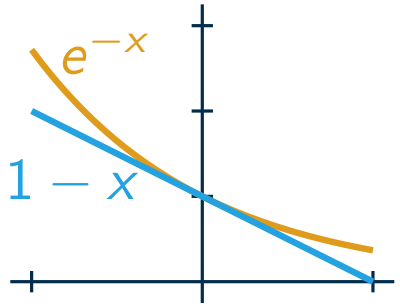
Randomized FPT algorithm

Theorem
With $k!$ independent trials of orienting $\vec{G}$, the probability to correctly solve LONGEST PATH for (G, k) is larger than $\frac{1}{2}$.

Theorem
Let P be a k-vertex path k in G. Then P is a directed path in $\vec{G}$ with probability $\frac{2}{k!}$.

Proof

- `no`-instances are always correctly recognized
- the algorithm fails on `yes`-instances with probability:
  - $1 - \frac{2}{k!}$ for an individual trial
  - $\left(1 - \frac{2}{k!}\right)^{k!}$ for $k!$ independent trials



$$\left(1 - \frac{2}{k!}\right)^{k!} \leq \left(e^{-\frac{2}{k!}}\right)^{k!} = e^{-2} < \frac{1}{2}$$

So what do we have?

- Monte Carlo algorithm with one-sided error
- ℓ repetitions $\rightarrow$ correct with probability $(1 - \frac{1}{2^\ell})$ in time $O(k!m\ell)$

("with high probability" for $\ell = \log n$)

Taking a step back

What did we just do?

- wish for some additional structure (vertex order) that makes the problem easier
- for each solution: number of choices for the structure on the solution only depends on k

Taking a step back

What did we just do?

- wish for some additional structure (vertex order) that makes the problem easier
- for each solution: number of choices for the structure on the solution only depends on k
- probability for bad choice only dependent on k
- probability-boosting: number of trials only depends on k

Taking a step back

What did we just do?

- wish for some additional structure (vertex order) that makes the problem easier
- for each solution: number of choices for the structure on the solution only depends on k
- probability for bad choice only dependent on k
- probability-boosting: number of trials only depends on k

Less order, more colors

- color the vertices
- only allow colorful solutions

Taking a step back

What did we just do?

- wish for some additional structure (vertex order) that makes the problem easier
- for each solution: number of choices for the structure on the solution only depends on k
- probability for bad choice only dependent on k
- probability-boosting: number of trials only depends on k

Less order, more colors

- color the vertices
- only allow colorful solutions

Problem: 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?
(a path is colorful if it contains at most one vertex from each V_i)

Taking a step back

What did we just do?

- wish for some additional structure (vertex order) that makes the problem easier
- for each solution: number of choices for the structure on the solution only depends on k
- probability for bad choice only dependent on k
- probability-boosting: number of trials only depends on k

Less order, more colors

- color the vertices
- only allow colorful solutions
- show that the colors help: FPT-algo for colorful problem
- random colors $\rightarrow$ error probability only depends on k

Problem: 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?
(a path is colorful if it contains at most one vertex from each V_i)

Taking a step back

What did we just do?

- wish for some additional structure (vertex order) that makes the problem easier
- for each solution: number of choices for the structure on the solution only depends on k
- probability for bad choice only dependent on k
- probability-boosting: number of trials only depends on k

Less order, more colors

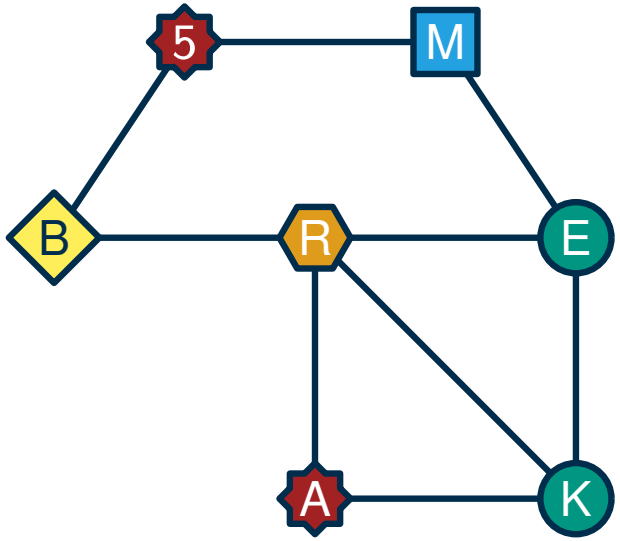
- color the vertices
- only allow colorful solutions
- show that the colors help: FPT-algo for colorful problem
- random colors $\rightarrow$ error probability only depends on k
- afterwards: derandomization

Problem: 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?
(a path is colorful if it contains at most one vertex from each V_i)

Colorful paths

Problem: 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?
(a path is colorful if it contains at most one vertex from each V_i)

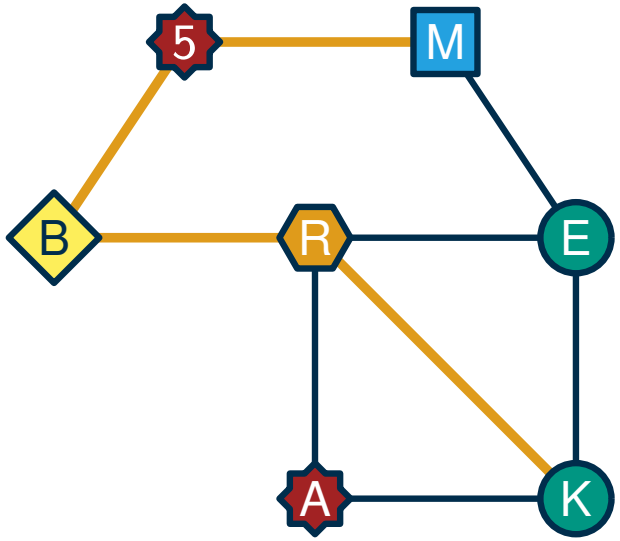
How long is the longest colorful path?



Colorful paths

Problem: 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?
(a path is colorful if it contains at most one vertex from each V_i)

How long is the longest colorful path?



How do colors help?

How do colors help?

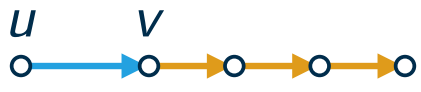
Solution for DAGs

- path of length ℓ from v and $uv \in E \Rightarrow$ path of length $\ell + 1$ from u

How do colors help?

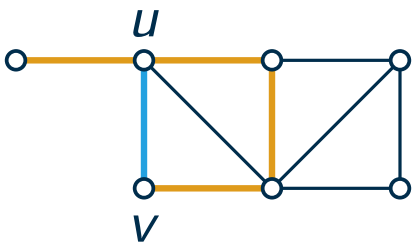
Solution for DAGs

- path of length ℓ from v and $uv \in E \Rightarrow$ path of length $\ell + 1$ from u



Undirected graphs

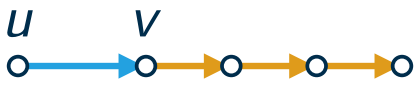
- problem: path of length ℓ from v might already contain u



How do colors help?

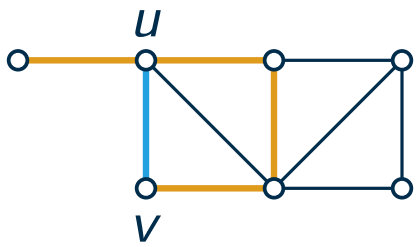
Solution for DAGs

- path of length ℓ from v and $uv \in E \Rightarrow$ path of length $\ell + 1$ from u



Undirected graphs

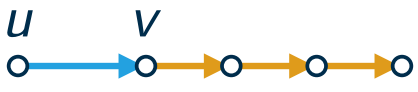
- problem: path of length ℓ from v might already contain u
- idea: remember vertices visited so far



How do colors help?

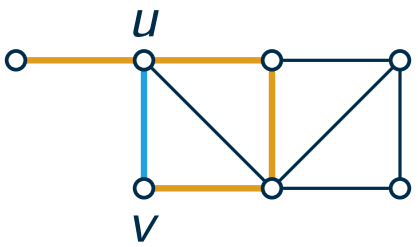
Solution for DAGs

- path of length ℓ from v and $uv \in E \Rightarrow$ path of length $\ell + 1$ from u



Undirected graphs

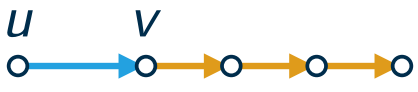
- problem: path of length ℓ from v might already contain u
- idea: remember vertices visited so far $\rightarrow n^\ell$ possibilities



How do colors help?

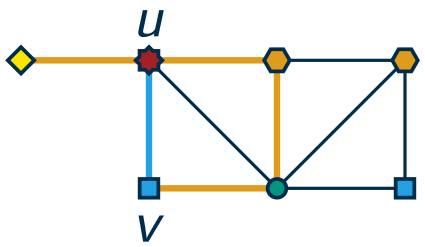
Solution for DAGs

- path of length ℓ from v and $uv \in E \Rightarrow$ path of length $\ell + 1$ from u



Undirected graphs

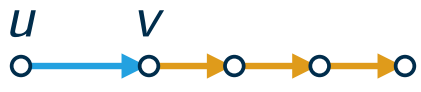
- problem: path of length ℓ from v might already contain u
- idea: remember vertices visited so far $\rightarrow n^\ell$ possibilities
- better: remember colors visited so far



How do colors help?

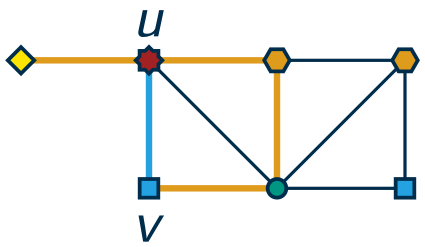
Solution for DAGs

- path of length ℓ from v and $uv \in E \Rightarrow$ path of length $\ell + 1$ from u



Undirected graphs

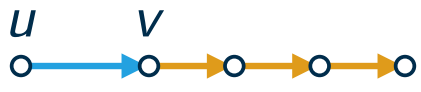
- problem: path of length ℓ from v might already contain u
- idea: remember vertices visited so far $\rightarrow n^\ell$ possibilities
- better: remember colors visited so far $\rightarrow 2^k$ possibilities



How do colors help?

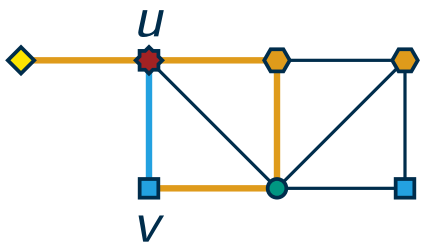
Solution for DAGs

- path of length ℓ from v and $uv \in E \Rightarrow$ path of length $\ell + 1$ from u



Undirected graphs

- problem: path of length ℓ from v might already contain u
- idea: remember vertices visited so far $\rightarrow n^\ell$ possibilities
- better: remember colors visited so far $\rightarrow 2^k$ possibilities

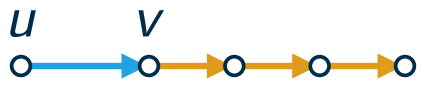


Dynamic program

How do colors help?

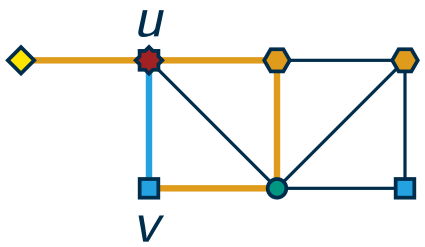
Solution for DAGs

- path of length ℓ from v and $uv \in E \Rightarrow$ path of length $\ell + 1$ from u



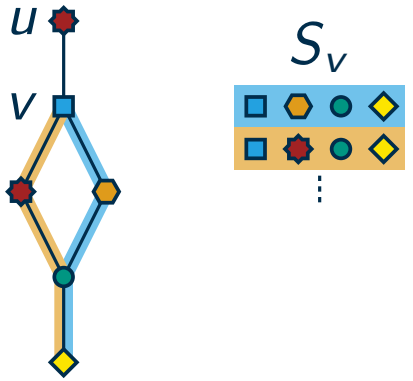
Undirected graphs

- problem: path of length ℓ from v might already contain u
- idea: remember vertices visited so far $\rightarrow n^\ell$ possibilities
- better: remember colors visited so far $\rightarrow 2^k$ possibilities



Dynamic program

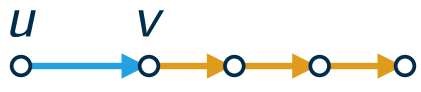
- for every vertex v: compute for which color sets $S_v \subseteq [k]$ a path with these colors starting at v exists



How do colors help?

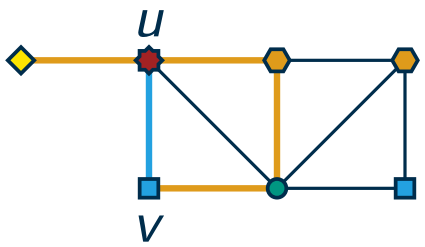
Solution for DAGs

- path of length ℓ from v and $uv \in E \Rightarrow$ path of length $\ell + 1$ from u



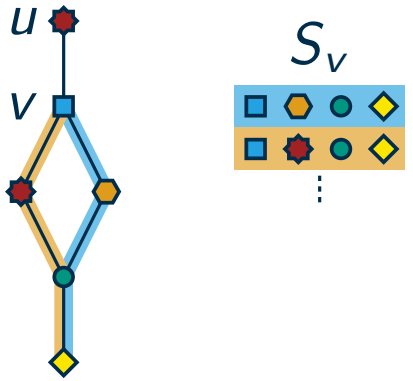
Undirected graphs

- problem: path of length ℓ from v might already contain u
- idea: remember vertices visited so far $\rightarrow n^\ell$ possibilities
- better: remember colors visited so far $\rightarrow 2^k$ possibilities



Dynamic program

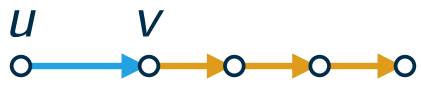
- for every vertex v: compute for which color sets $S_v \subseteq [k]$ a path with these colors starting at v exists
- consider the color sets in increasing size



How do colors help?

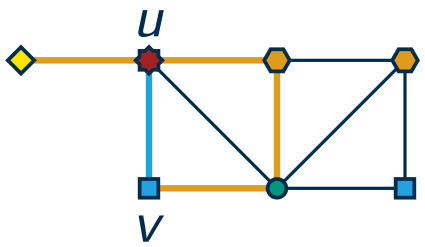
Solution for DAGs

- path of length ℓ from v and $uv \in E \Rightarrow$ path of length $\ell + 1$ from u



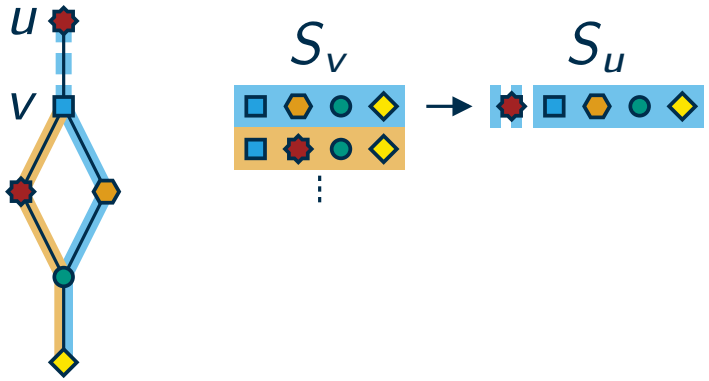
Undirected graphs

- problem: path of length ℓ from v might already contain u
- idea: remember vertices visited so far $\rightarrow n^\ell$ possibilities
- better: remember colors visited so far $\rightarrow 2^k$ possibilities

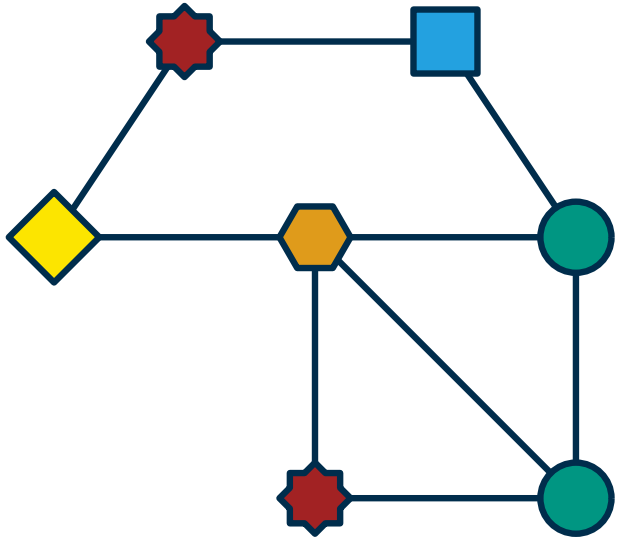


Dynamic program

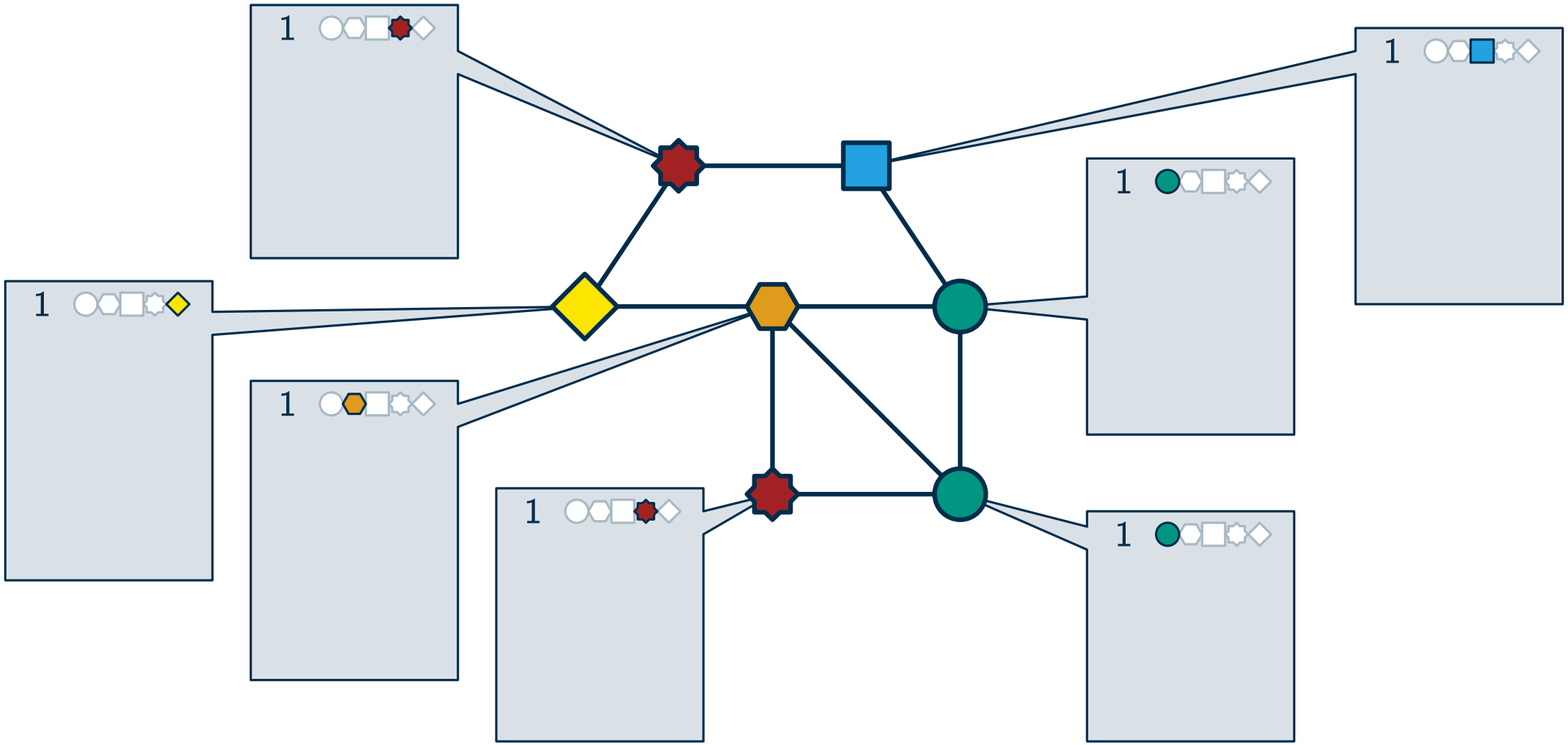
- for every vertex v: compute for which color sets $S_v \subseteq [k]$ a path with these colors starting at v exists
- consider the color sets in increasing size
- color sets of size $\ell + 1$ can be computed from color sets of size ℓ of the neighbors



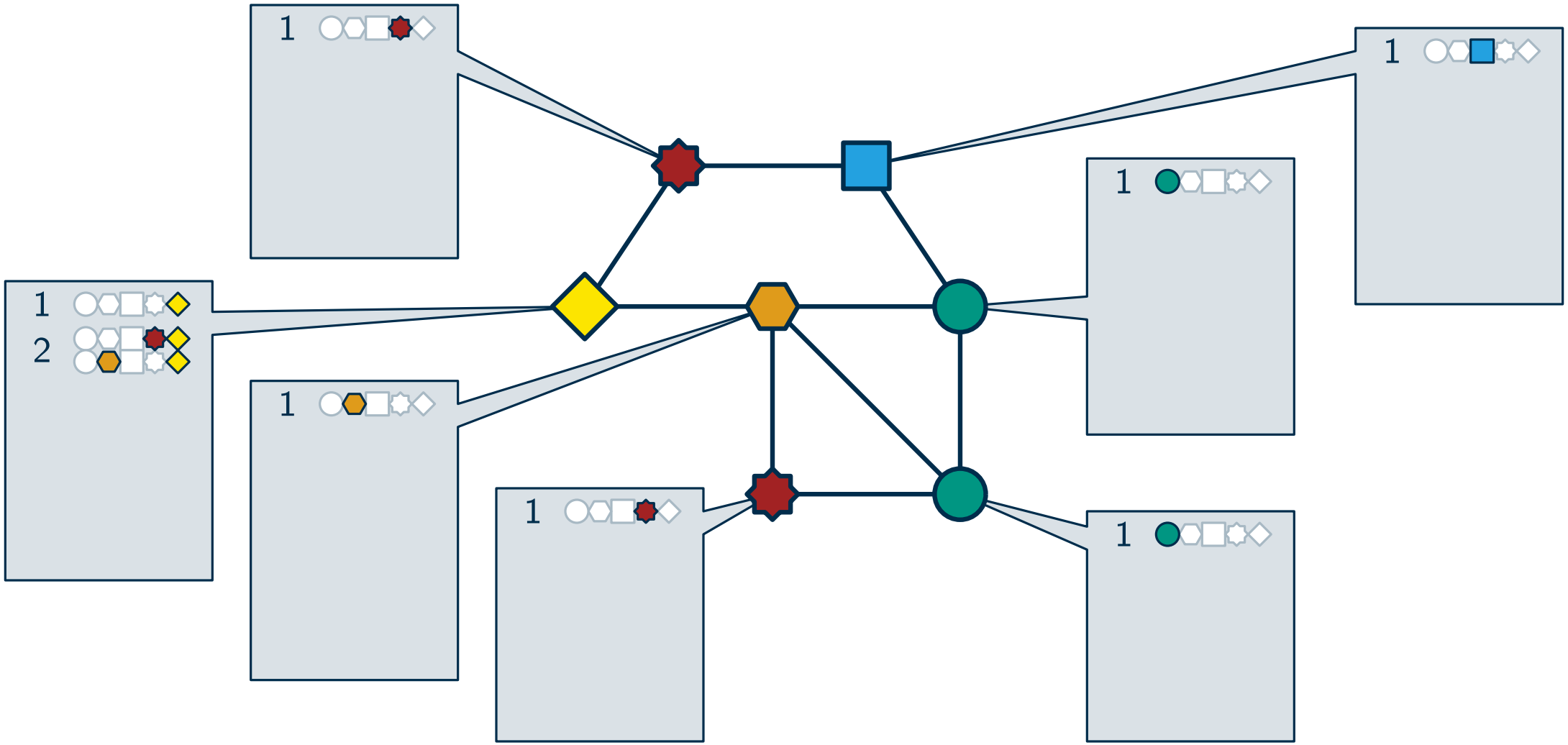
Dynamic program



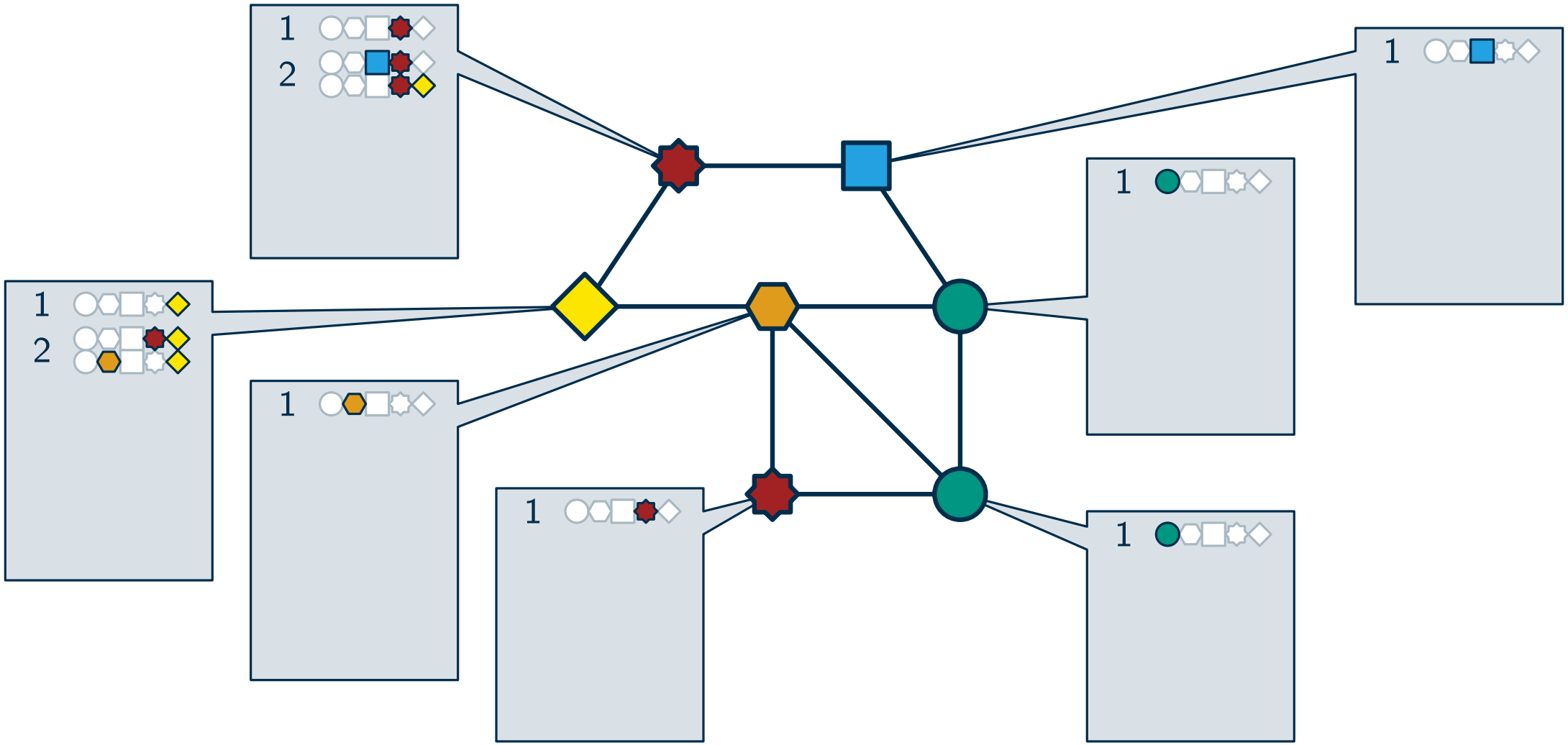
Dynamic program



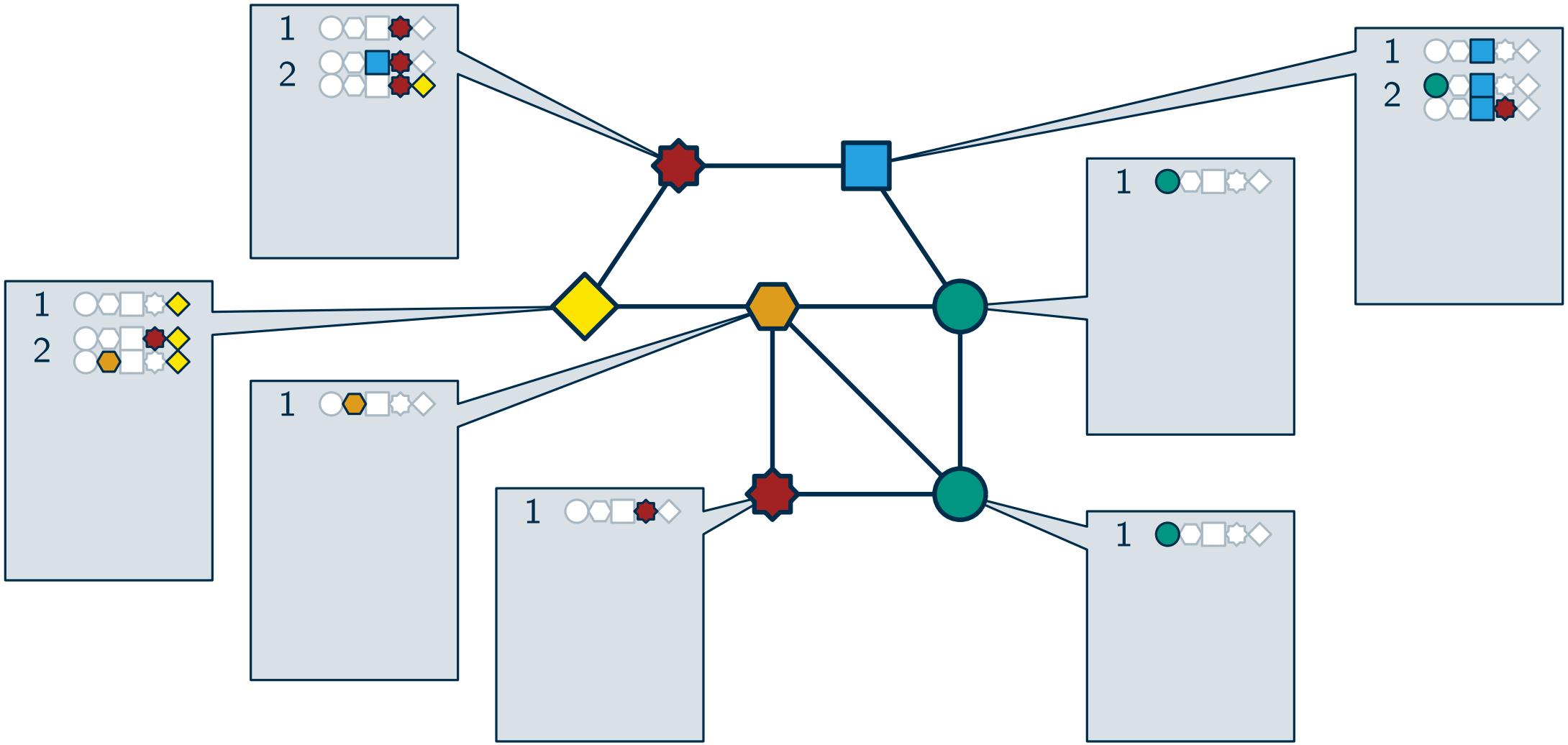
Dynamic program



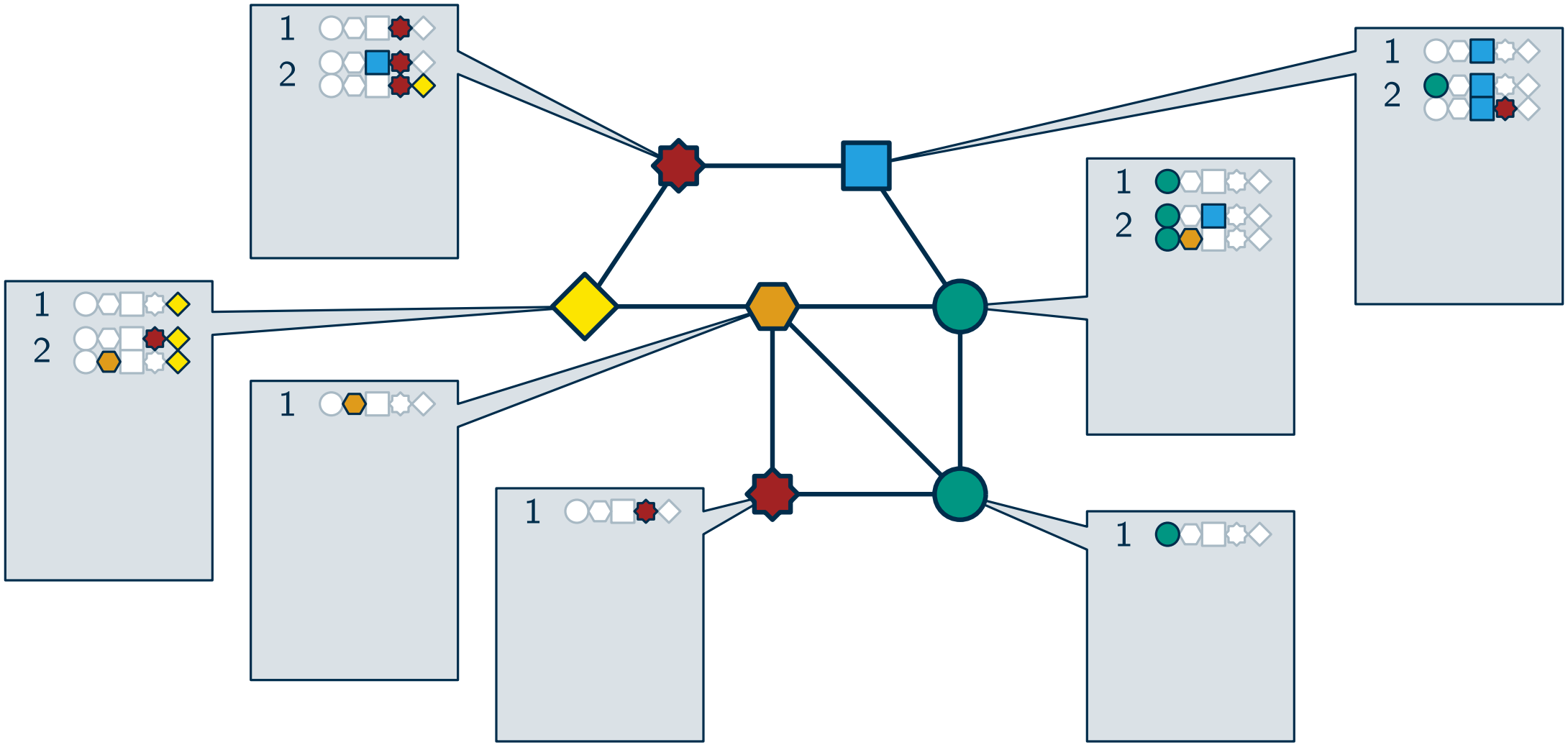
Dynamic program



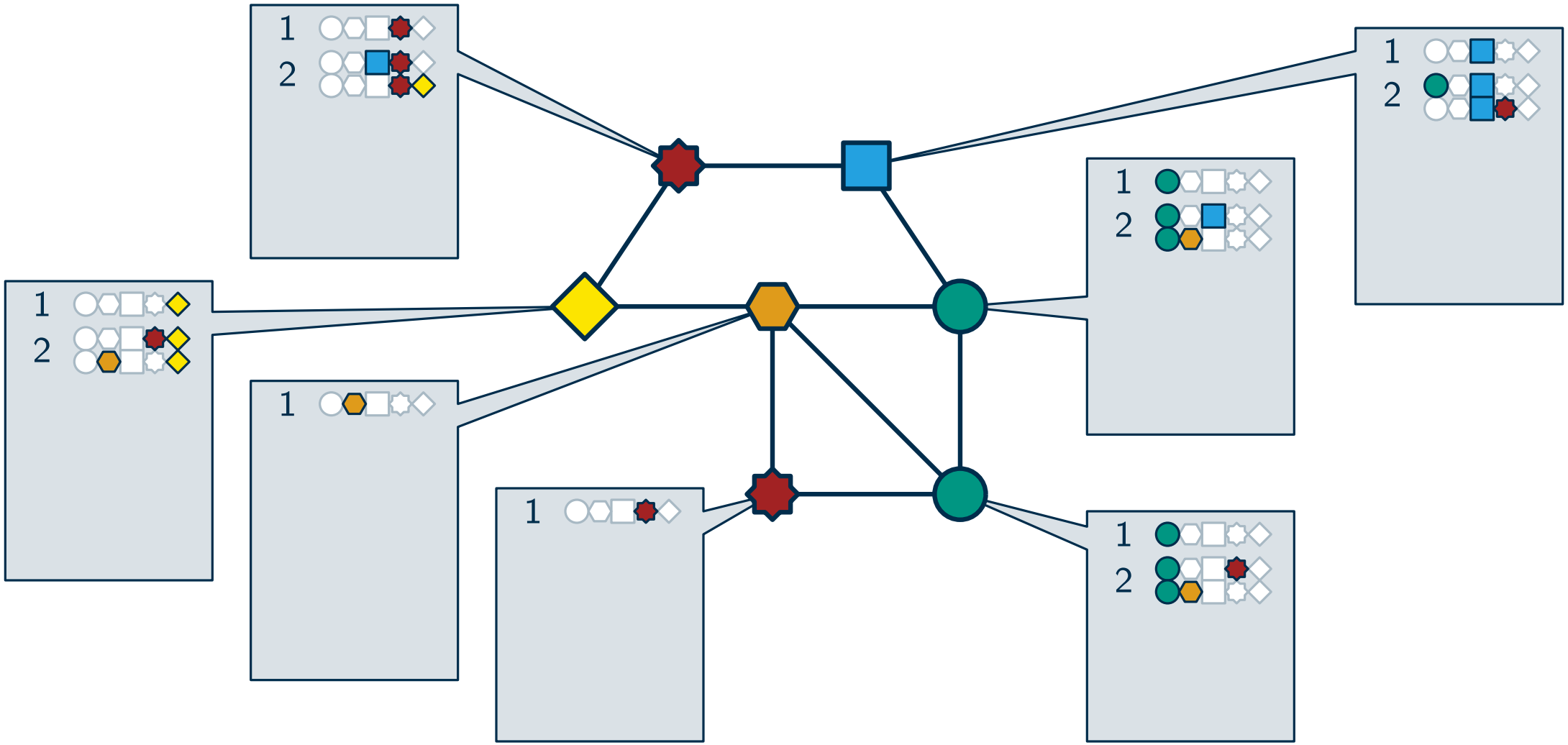
Dynamic program



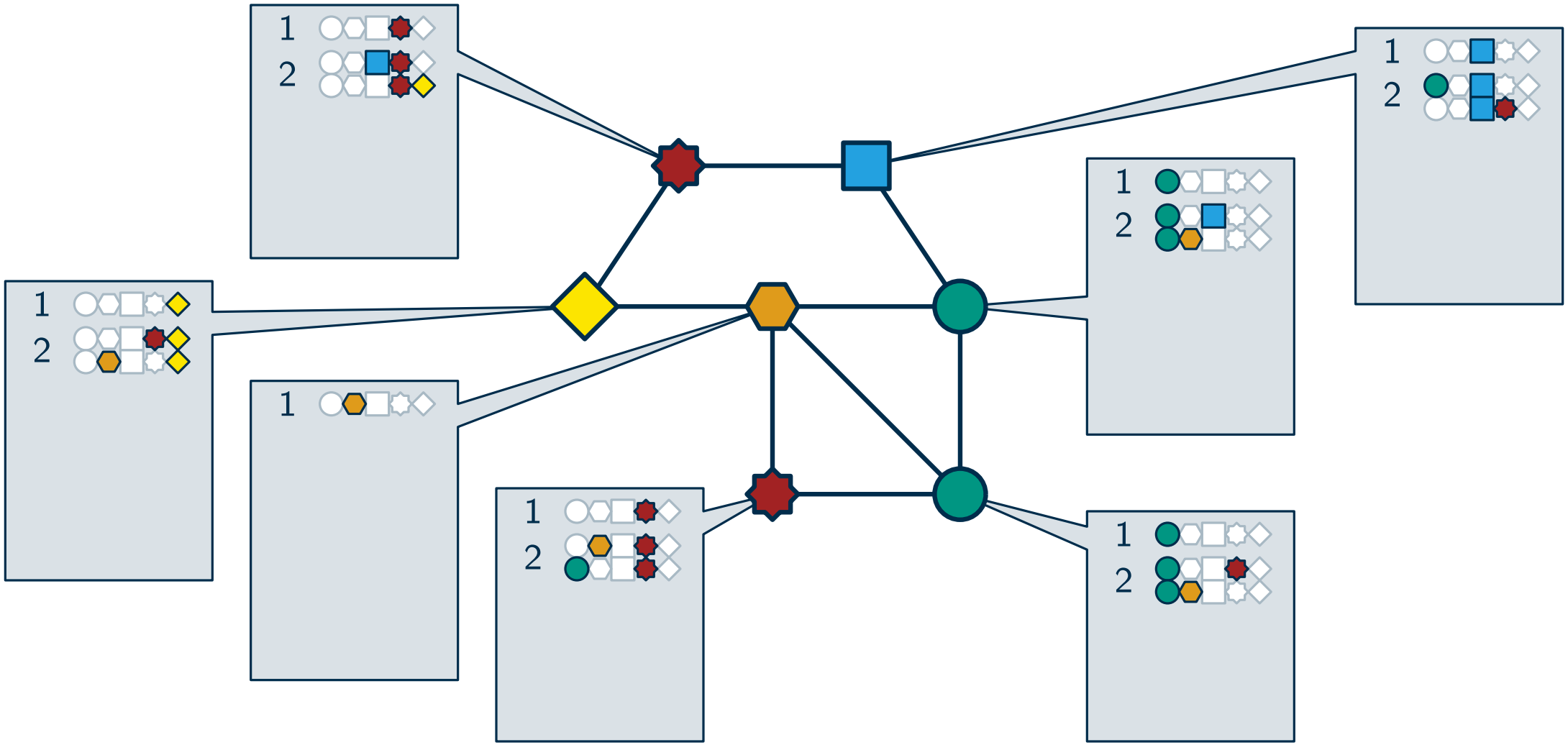
Dynamic program



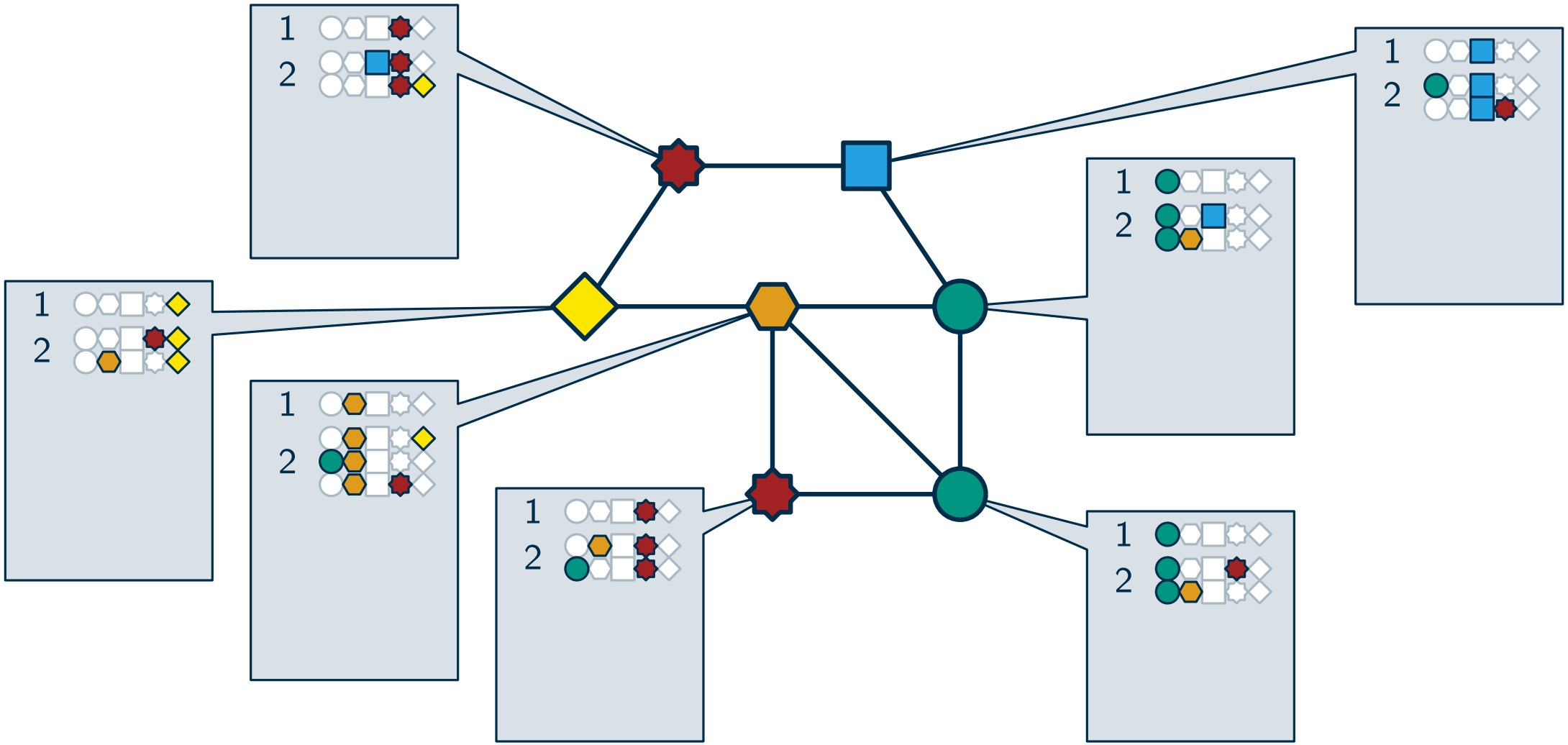
Dynamic program



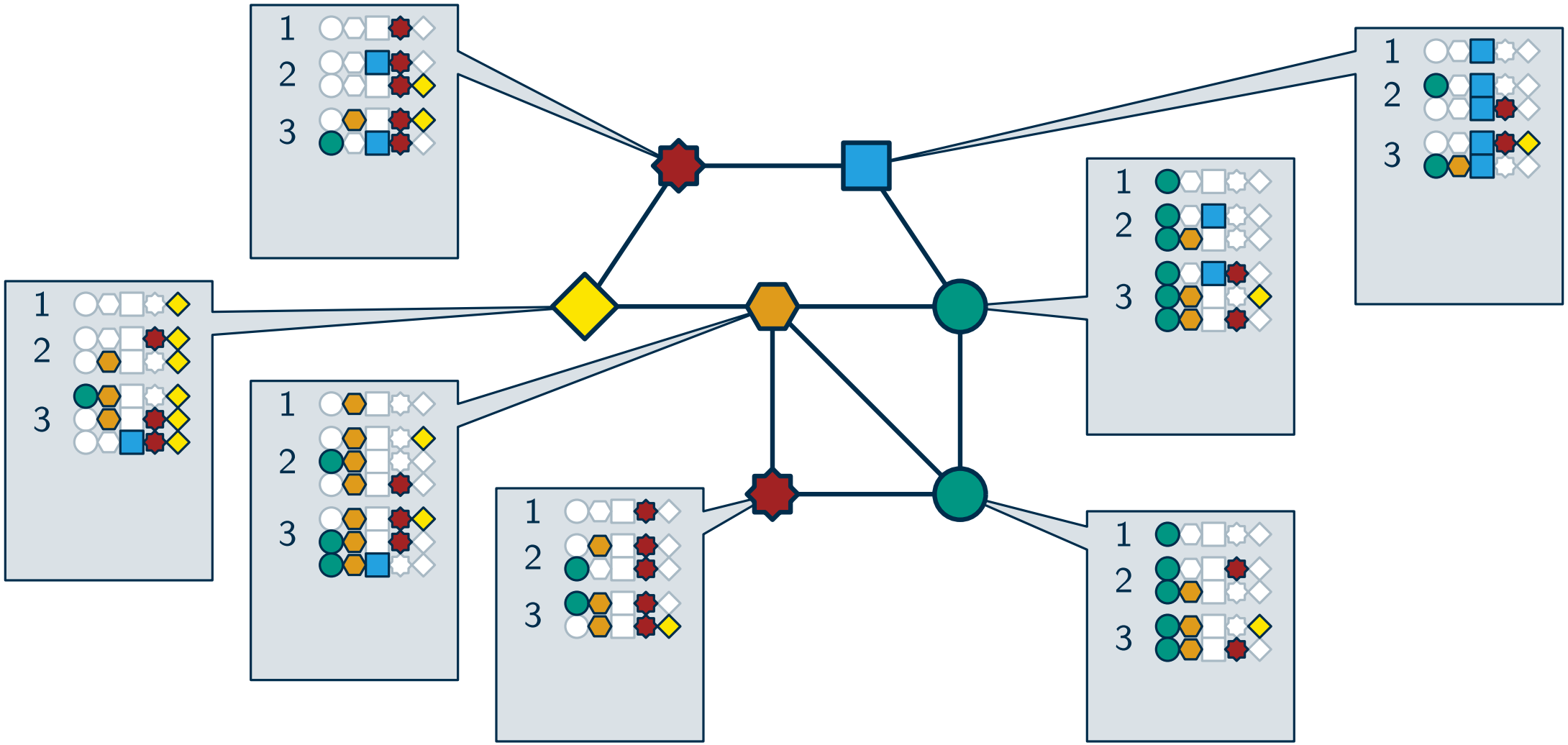
Dynamic program



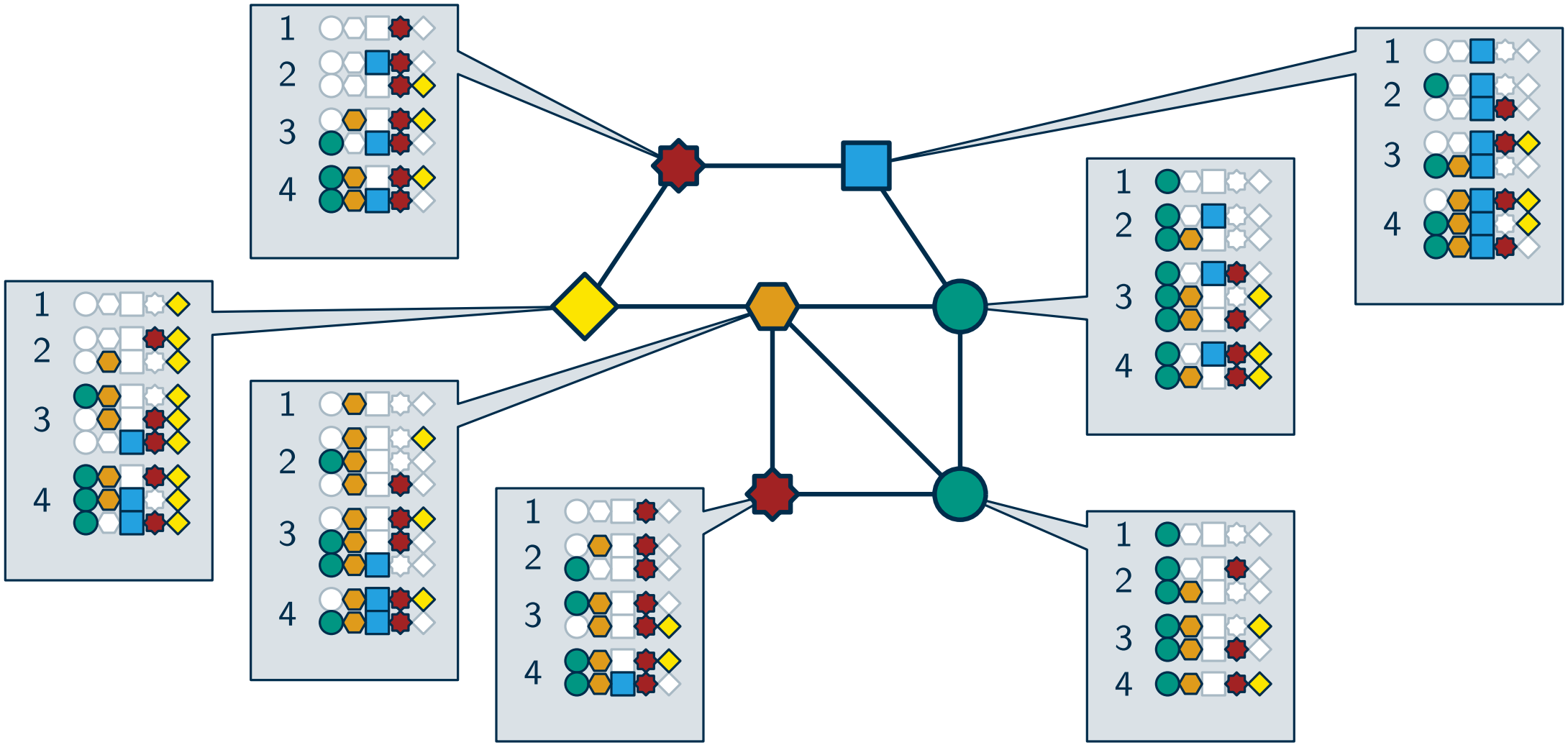
Dynamic program



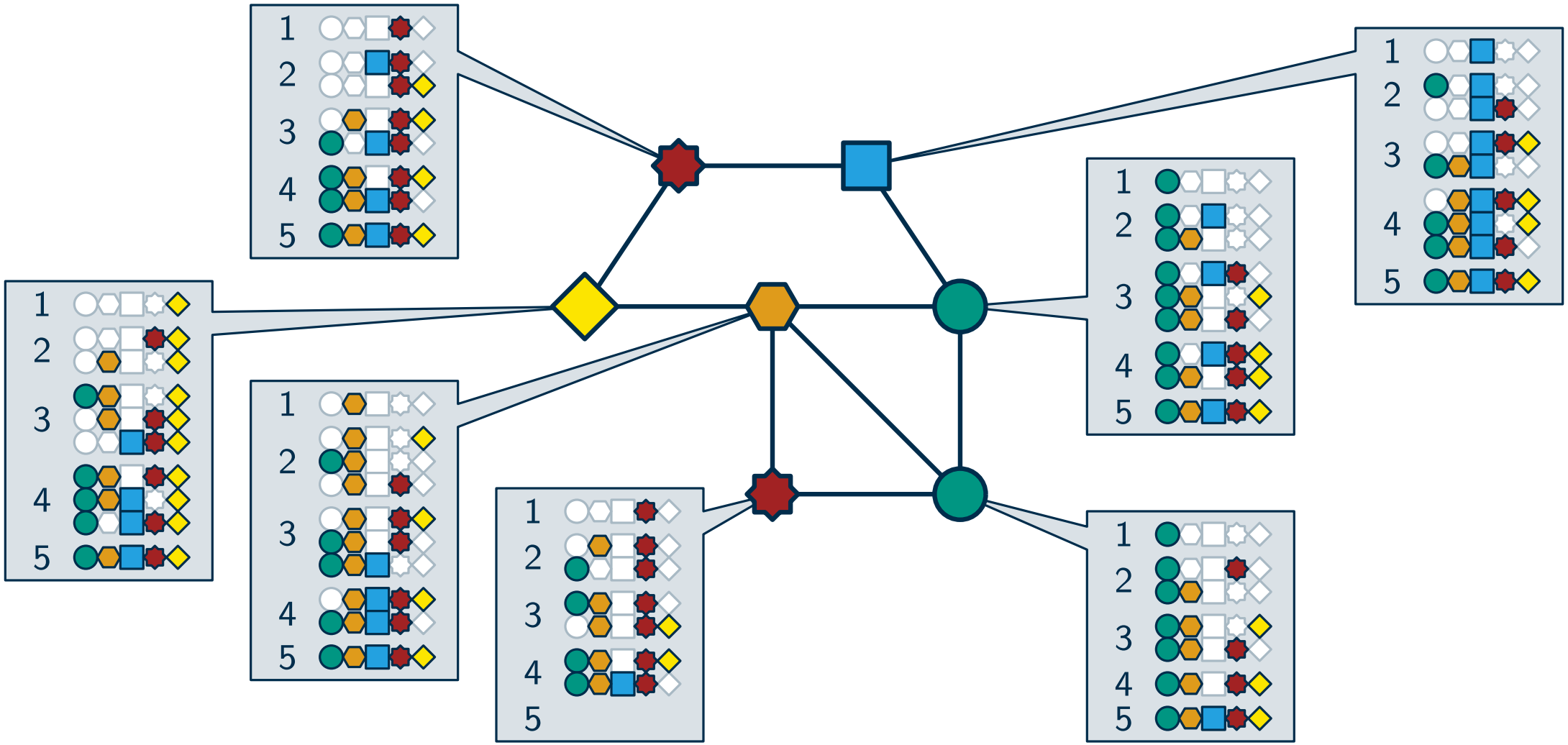
Dynamic program



Dynamic program

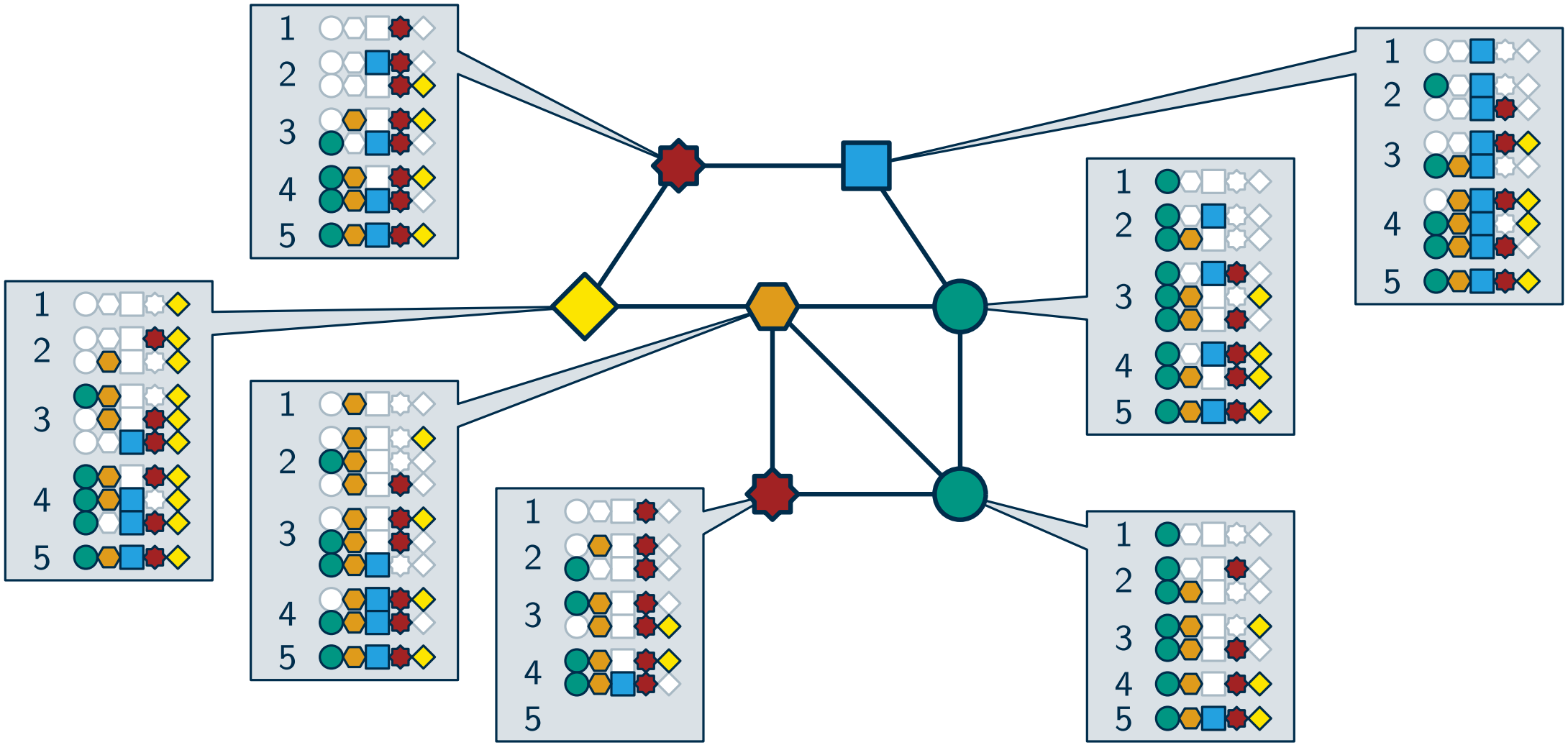


Dynamic program



Dynamic program

Theorem
COLORFUL LONGEST PATH can be solved in $O(2^k km)$ time.



Random colors

Wassily Kandinsky: Color Study
Squares with Concentric Circles

Plan

- color vertices and solve COLORFUL LONGEST PATH
- problem: a path on k vertices is maybe not colorful
- solution: for every vertex, randomly choose one of k colors (uniformly, independent)

Random colors

Wassily Kandinsky: Color Study
Squares with Concentric Circles

Plan

- color vertices and solve COLORFUL LONGEST PATH
- problem: a path on k vertices is maybe not colorful
- solution: for every vertex, randomly choose one of k colors (uniformly, independent)

Theorem
Let P be a k-vertex path. Then P is a colorful path with probability $\geq \frac{1}{e^k}$.

Random colors

Wassily Kandinsky: Color Study Squares with Concentric Circles

Plan

- color vertices and solve COLORFUL LONGEST PATH
- problem: a path on k vertices is maybe not colorful
- solution: for every vertex, randomly choose one of k colors (uniformly, independent)

Theorem
Let P be a k-vertex path. Then P is a colorful path with probability $\geq \frac{1}{e^k}$.

Proof

- let $P = v_1, \ldots, v_k$

Random colors

Wassily Kandinsky: Color Study Squares with Concentric Circles

Plan

- color vertices and solve COLORFUL LONGEST PATH
- problem: a path on k vertices is maybe not colorful
- solution: for every vertex, randomly choose one of k colors (uniformly, independent)

Theorem
Let P be a k-vertex path. Then P is a colorful path with probability $\geq \frac{1}{e^k}$.

Proof

- let $P = v_1, \ldots, v_k$
- every color combination of $v_1, \ldots, v_k$ is equally likely
- k^k events, each happening with probability $\frac{1}{k^k}$

Random colors

Wassily Kandinsky: Color Study Squares with Concentric Circles

Plan

- color vertices and solve COLORFUL LONGEST PATH
- problem: a path on k vertices is maybe not colorful
- solution: for every vertex, randomly choose one of k colors (uniformly, independent)

Theorem
Let P be a k-vertex path. Then P is a colorful path with probability $\geq \frac{1}{e^k}$.

Proof

- let $P = v_1, \ldots, v_k$
- every color combination of $v_1, \ldots, v_k$ is equally likely
- k^k events, each happening with probability $\frac{1}{k^k}$
- $k!$ of these events are good

Random colors

Wassily Kandinsky: Color Study Squares with Concentric Circles

Plan

- color vertices and solve COLORFUL LONGEST PATH
- problem: a path on k vertices is maybe not colorful
- solution: for every vertex, randomly choose one of k colors (uniformly, independent)

Theorem
Let P be a k-vertex path. Then P is a colorful path with probability $\geq \frac{1}{e^k}$.

Proof

- let $P = v_1, \ldots, v_k$
- every color combination of $v_1, \ldots, v_k$ is equally likely
- k^k events, each happening with probability $\frac{1}{k^k}$
- $k!$ of these events are good
- thus: $\frac{k!}{k^k}$

Random colors

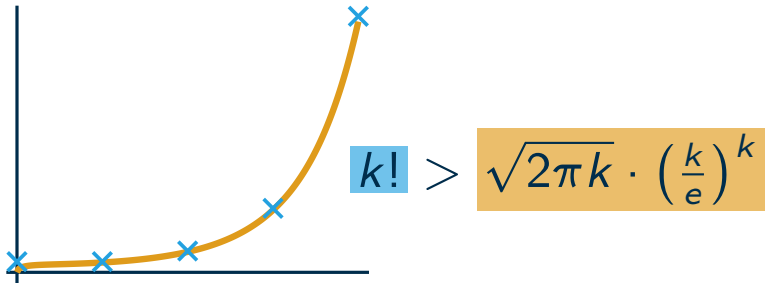
Wassily Kandinsky: Color Study
Squares with Concentric Circles

Plan

- color vertices and solve COLORFUL LONGEST PATH
- problem: a path on k vertices is maybe not colorful
- solution: for every vertex, randomly choose one of k colors (uniformly, independent)

Theorem
Let P be a k-vertex path. Then P is a colorful path with probability $\geq \frac{1}{e^k}$.

Stirling's approximation



$$k! > \sqrt{2\pi k} \cdot \left(\frac{k}{e}\right)^k$$

Proof

- let $P = v_1, \ldots, v_k$
- every color combination of $v_1, \ldots, v_k$ is equally likely
- k^k events, each happening with probability $\frac{1}{k^k}$
- $k!$ of these events are good
- thus: $\frac{k!}{k^k}$

Random colors

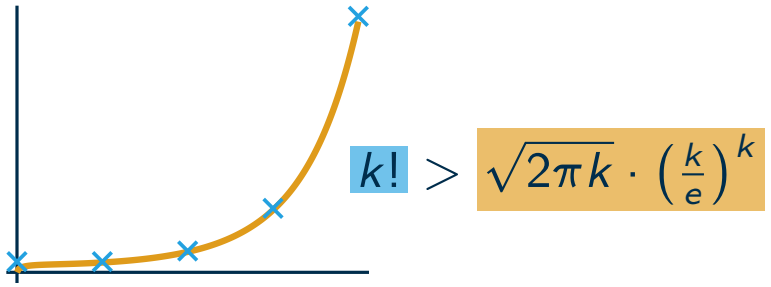
Wassily Kandinsky: Color Study Squares with Concentric Circles

Plan

- color vertices and solve COLORFUL LONGEST PATH
- problem: a path on k vertices is maybe not colorful
- solution: for every vertex, randomly choose one of k colors (uniformly, independent)

Theorem
Let P be a k-vertex path. Then P is a colorful path with probability $\geq \frac{1}{e^k}$.

Stirling's approximation



$$k! > \sqrt{2\pi k} \cdot \left(\frac{k}{e}\right)^k$$

Proof

- let $P = v_1, \ldots, v_k$
- every color combination of $v_1, \ldots, v_k$ is equally likely
- k^k events, each happening with probability $\frac{1}{k^k}$
- $k!$ of these events are good
- thus: $\frac{k!}{k^k} > \frac{1}{e^k}$

Probability boosting

- $\Theta(e^k)$ repetitions $\rightarrow$ constant success probability

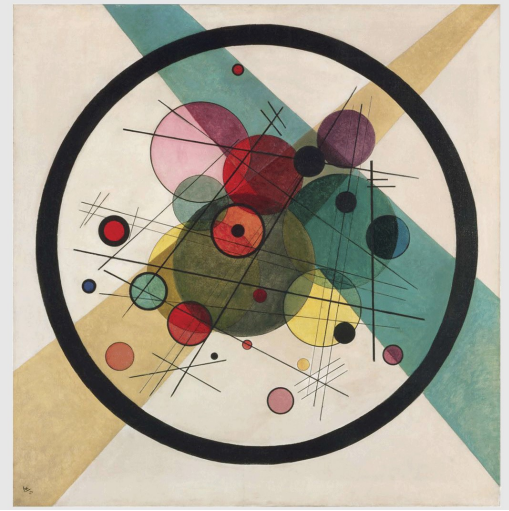
Derandomization

Random colorings

- enough random colorings
  $\rightarrow$ one is likely good

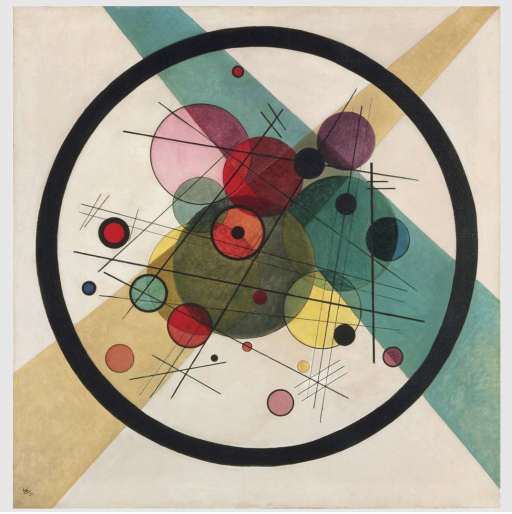
Deterministic colorings

- cleverly choose multiple colorings such that at least one is good



Wassily Kandinsky
Circles in a Circle

Derandomization



Wassily Kandinsky
Circles in a Circle

Random colorings

- enough random colorings
  $\rightarrow$ one is likely good

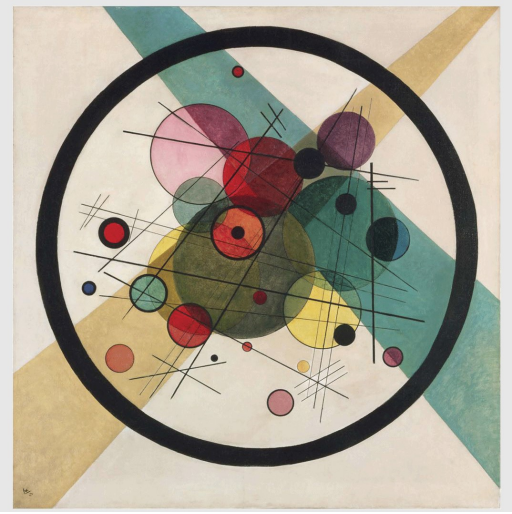
Deterministic colorings

- cleverly choose multiple colorings such that at least one is good

Make a wish

- set of colorings of n elements (vertices) with k colors
- for every k-element subset there is a coloring making it colorful

Derandomization



Wassily Kandinsky
Circles in a Circle

Random colorings

- enough random colorings
  $\rightarrow$ one is likely good

Deterministic colorings

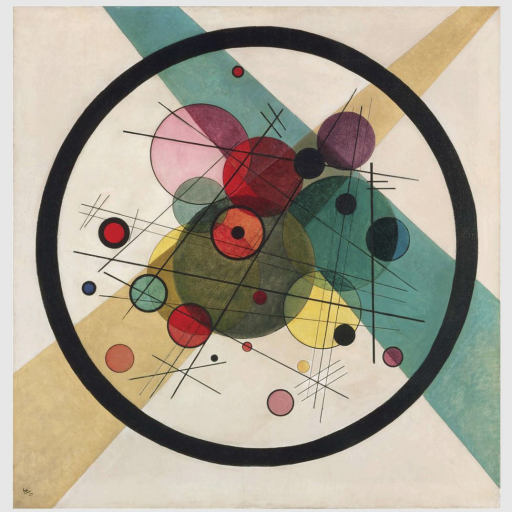
- cleverly choose multiple colorings such that at least one is good

Make a wish

- set of colorings of n elements (vertices) with k colors
- for every k-element subset there is a coloring making it colorful

$\Big\}$ (n, k)-perfect hash family

Derandomization



Wassily Kandinsky
Circles in a Circle

Random colorings

- enough random colorings
  $\rightarrow$ one is likely good

Deterministic colorings

- cleverly choose multiple colorings such that at least one is good

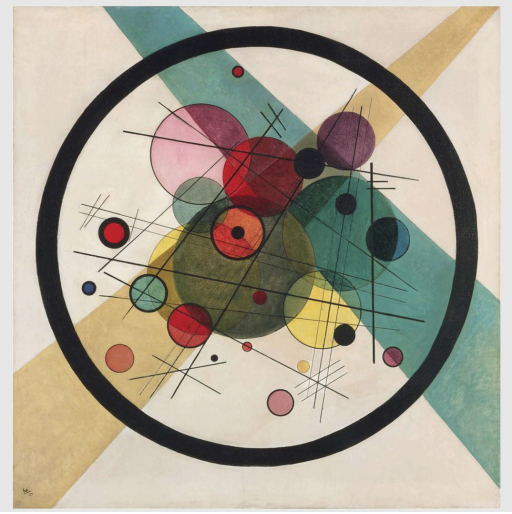
Make a wish

- set of colorings of n elements (vertices) with k colors
- for every k-element subset there is a coloring making it colorful

} (n, k)-perfect hash family

What is the relation to hashing?

Derandomization



Wassily Kandinsky
Circles in a Circle

Random colorings

- enough random colorings
  $\rightarrow$ one is likely good

Deterministic colorings

- cleverly choose multiple colorings such that at least one is good

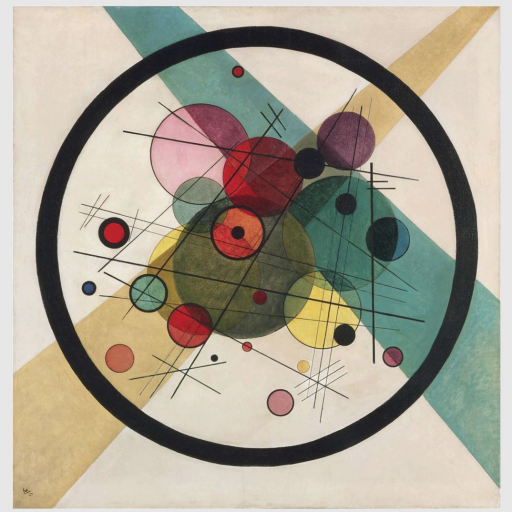
Make a wish

- set of colorings of n elements (vertices) with k colors
- for every k-element subset there is a coloring making it colorful

$\Big\}$ (n, k)-perfect hash family

Theorem (without proof)
An (n, k)-perfect hash family of size $O(6.4^k \log^2 n)$ can be constructed in $O(6.4^k\, n \log^2 n)$ time.

Derandomization



Wassily Kandinsky
Circles in a Circle

Random colorings

- enough random colorings
  $\rightarrow$ one is likely good

Deterministic colorings

- cleverly choose multiple colorings such that at least one is good

Make a wish

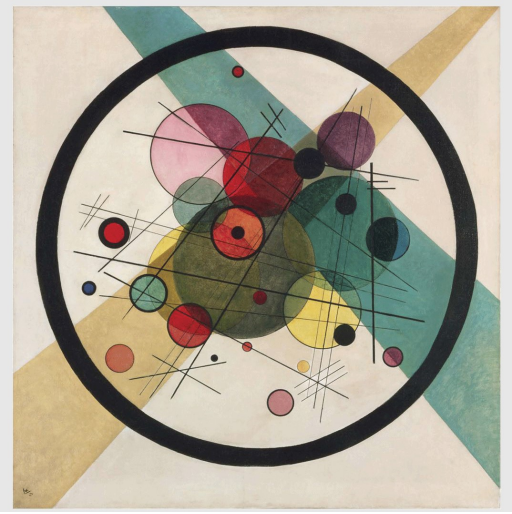
- set of colorings of n elements (vertices) with k colors
- for every k-element subset there is a coloring making it colorful

$\Big\}$ (n, k)-perfect hash family

Theorem (without proof)
An (n, k)-perfect hash family of size $O(6.4^k \log^2 n)$ can be constructed in $O(6.4^k n \log^2 n)$ time.

Theorem (DP from earlier)
COLORFUL LONGEST PATH can be solved in time $O(2^k km)$.

Derandomization



Wassily Kandinsky
Circles in a Circle

Random colorings

- enough random colorings
  $\rightarrow$ one is likely good

Deterministic colorings

- cleverly choose multiple colorings such that at least one is good

Make a wish

- set of colorings of n elements (vertices) with k colors
- for every k-element subset there is a coloring making it colorful

$\Big\}$ (n, k)-perfect hash family

Theorem (without proof)
An (n, k)-perfect hash family of size $O(6.4^k \log^2 n)$ can be constructed in $O(6.4^k n \log^2 n)$ time.

Theorem (DP from earlier)
Colorful Longest Path can be solved in time $O(2^k km)$.

Theorem (directly follows)
Longest Path can be solved in time $O(12.8^k km \log^2 n)$.

Color Coding

General Approach

- defined colored problem variant (with prefix COLORFUL or MULTICOLORED)
- design FPT-algorithm for the colored problem
- show:
  - `no`-instance becomes colored `no`-instance
  - `yes`-instance becomes colored `yes`-instance for ≥ 1 coloring from a perfect hash family

Color Coding

General Approach

- defined colored problem variant (with prefix COLORFUL or MULTICOLORED)
- design FPT-algorithm for the colored problem
- show:
  - `no`-instance becomes colored `no`-instance
  - `yes`-instance becomes colored `yes`-instance for ≥ 1 coloring from a perfect hash family

Tips for color coding

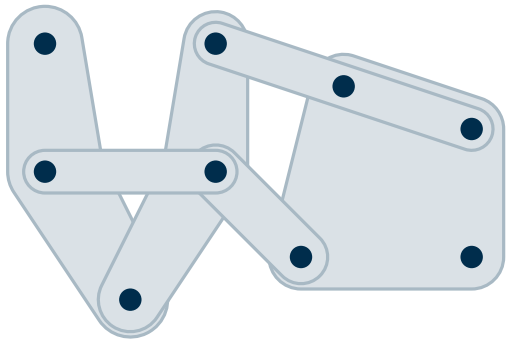
- look at a solution of an instance
- identify a witness of size only depending on k that certifies that it is a solution
- require that this witness is colorful

Set Splitting

Problem: Set Splitting (essentially Max Cut for hypergraphs)
Given a hypergraph $\mathcal{H} = (V, \mathcal{E})$ with vertices V and hyperedges $\mathcal{E} \subseteq 2^V$, and a parameter s.
Is there a set $X \subseteq V$ such that X splits at least s edges of $\mathcal{H}$?

X splits $E \in \mathcal{E}$: $E \cap X \neq \emptyset$ and $E \not\subseteq X$

Example: Is there a set X that splits all hyperedges?



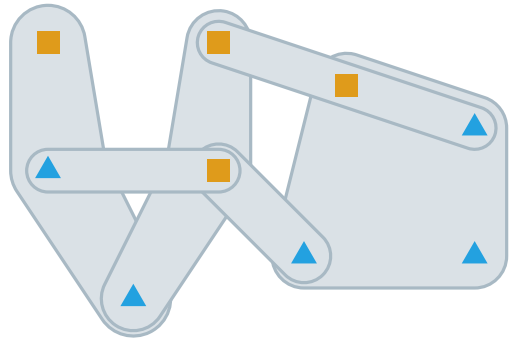
Set Splitting

Problem: Set Splitting (essentially Max Cut for hypergraphs)
Given a hypergraph $\mathcal{H} = (V, \mathcal{E})$ with vertices V and hyperedges $\mathcal{E} \subseteq 2^V$, and a parameter s.
Is there a set $X \subseteq V$ such that X splits at least s edges of $\mathcal{H}$?

X splits $E \in \mathcal{E}$: $E \cap X \neq \emptyset$ and $E \not\subseteq X$

Example: Is there a set X that splits all hyperedges?

■ : X
▲ : $V \setminus X$



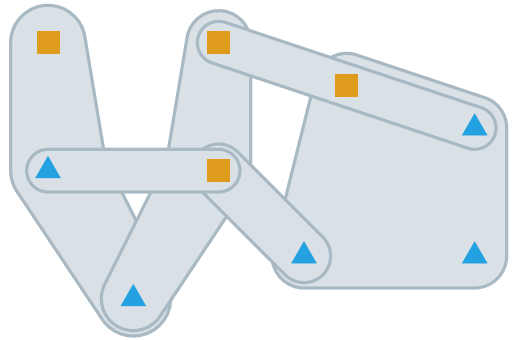
Set Splitting

Problem: Set Splitting (essentially Max Cut for hypergraphs)
Given a hypergraph $\mathcal{H} = (V, \mathcal{E})$ with vertices V and hyperedges $\mathcal{E} \subseteq 2^V$, and a parameter s.
Is there a set $X \subseteq V$ such that X splits at least s edges of $\mathcal{H}$?

X splits $E \in \mathcal{E}$: $E \cap X \neq \emptyset$ and $E \not\subseteq X$

Example: Is there a set X that splits all hyperedges?

■ : X
▲ : $V \setminus X$



First step of color coding

- identify a solution witness of size only depending on s

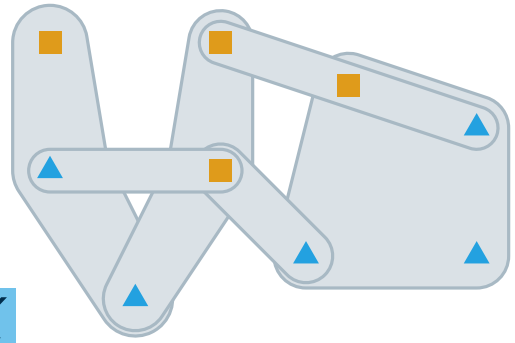
Set Splitting

Problem: Set Splitting (essentially Max Cut for hypergraphs)
Given a hypergraph $\mathcal{H} = (V, \mathcal{E})$ with vertices V and hyperedges $\mathcal{E} \subseteq 2^V$, and a parameter s.
Is there a set $X \subseteq V$ such that X splits at least s edges of $\mathcal{H}$?

X splits $E \in \mathcal{E}$: $E \cap X \neq \emptyset$ and $E \not\subseteq X$

Example: Is there a set X that splits all hyperedges?

■ : X
▲ : $V \setminus X$



First step of color coding

- identify a solution witness of size only depending on s
- here: X splits $E \in \mathcal{E} \Rightarrow E$ contains a vertex from X and one from $V \setminus X$

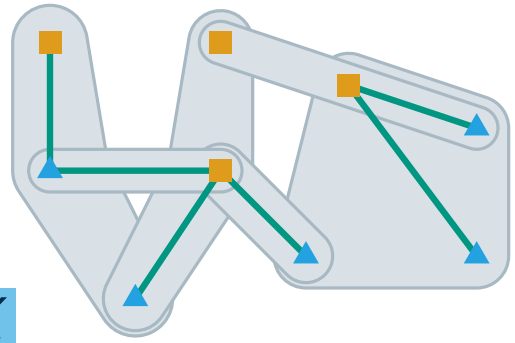
Set Splitting

Problem: Set Splitting (essentially Max Cut for hypergraphs)
Given a hypergraph $\mathcal{H} = (V, \mathcal{E})$ with vertices V and hyperedges $\mathcal{E} \subseteq 2^V$, and a parameter s.
Is there a set $X \subseteq V$ such that X splits at least s edges of $\mathcal{H}$?

X splits $E \in \mathcal{E}$: $E \cap X \neq \emptyset$ and $E \not\subseteq X$

Example: Is there a set X that splits all hyperedges?

■ : X
▲ : $V \setminus X$



First step of color coding

- identify a solution witness of size only depending on s
- here: X splits $E \in \mathcal{E} \Rightarrow E$ contains a vertex from X and one from $V \setminus X$
- there is a bi-colored pair for each split hyperedge $\rightarrow$ witness of size $\leq 2s$

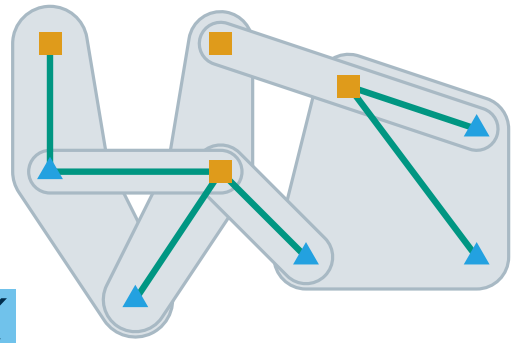
Set Splitting

Problem: Set Splitting (essentially Max Cut for hypergraphs)
Given a hypergraph $\mathcal{H} = (V, \mathcal{E})$ with vertices V and hyperedges $\mathcal{E} \subseteq 2^V$, and a parameter s.
Is there a set $X \subseteq V$ such that X splits at least s edges of $\mathcal{H}$?

X splits $E \in \mathcal{E}$: $E \cap X \neq \emptyset$ and $E \not\subseteq X$

Example: Is there a set X that splits all hyperedges?

■ : X
▲ : $V \setminus X$



First step of color coding

- identify a solution witness of size only depending on s
- here: X splits $E \in \mathcal{E} \Rightarrow E$ contains a vertex from X and one from $V \setminus X$
- there is a bi-colored pair for each split hyperedge $\rightarrow$ witness of size $\leq 2s$

Same but different

- randomly assign one of two colors to each vertex
- probability of bi-colored witness pairs:

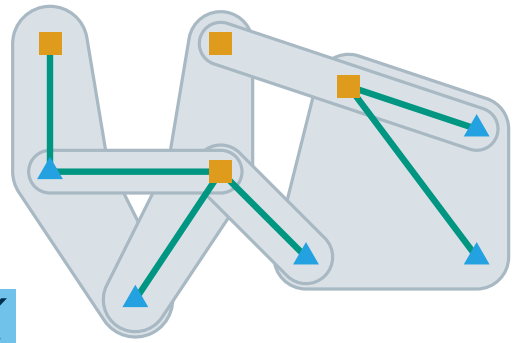
Set Splitting

Problem: Set Splitting (essentially Max Cut for hypergraphs)
Given a hypergraph $\mathcal{H} = (V, \mathcal{E})$ with vertices V and hyperedges $\mathcal{E} \subseteq 2^V$, and a parameter s.
Is there a set $X \subseteq V$ such that X splits at least s edges of $\mathcal{H}$?

X splits $E \in \mathcal{E}$: $E \cap X \neq \emptyset$ and $E \not\subseteq X$

Example: Is there a set X that splits all hyperedges?

■ : X
▲ : $V \setminus X$



First step of color coding

- identify a solution witness of size only depending on s
- here: X splits $E \in \mathcal{E} \Rightarrow E$ contains a vertex from X and one from $V \setminus X$
- there is a bi-colored pair for each split hyperedge $\rightarrow$ witness of size $\leq 2s$

Same but different

- randomly assign one of two colors to each vertex
- probability of bi-colored witness pairs: $\approx \frac{1}{2^s}$

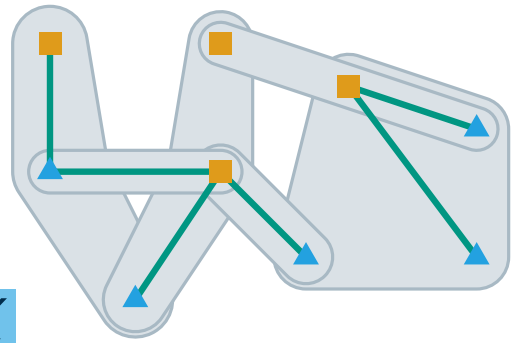
Set Splitting

Problem: Set Splitting (essentially Max Cut for hypergraphs)
Given a hypergraph $\mathcal{H} = (V, \mathcal{E})$ with vertices V and hyperedges $\mathcal{E} \subseteq 2^V$, and a parameter s.
Is there a set $X \subseteq V$ such that X splits at least s edges of $\mathcal{H}$?

X splits $E \in \mathcal{E}$: $E \cap X \neq \emptyset$ and $E \not\subseteq X$

Example: Is there a set X that splits all hyperedges?

■ : X
▲ : $V \setminus X$



First step of color coding

- identify a solution witness of size only depending on s
- here: X splits $E \in \mathcal{E} \Rightarrow E$ contains a vertex from X and one from $V \setminus X$
- there is a bi-colored pair for each split hyperedge $\rightarrow$ witness of size $\leq 2s$

Same but different

- randomly assign one of two colors to each vertex
- probability of bi-colored witness pairs: $\approx \frac{1}{2^s}$

} $\Theta(2^s)$ trials $\rightarrow$ constant success probability

Derandomization: Universal sets

Random 2-colorings

- enough random 2-colorings
  $\rightarrow$ one is likely good

Derandomization: Universal sets

Random 2-colorings

- enough random 2-colorings $\rightarrow$ one is likely good

Deterministic 2-colorings

- cleverly choose multiple 2-colorings such that at least one is good

Derandomization: Universal sets

Random 2-colorings

- enough random 2-colorings
  $\rightarrow$ one is likely good

Deterministic 2-colorings

- cleverly choose multiple 2-colorings such that at least one is good

Make a wish

- set of colorings of n elements (vertices) with two colors
- every k-element subset is colored in every possible way

Derandomization: Universal sets

Random 2-colorings

- enough random 2-colorings
  $\rightarrow$ one is likely good

Deterministic 2-colorings

- cleverly choose multiple 2-colorings such that at least one is good

Make a wish

- set of colorings of n elements (vertices) with two colors
- every k-element subset is colored in every possible way

$\Big\}$ (n, k)-univeral set

Derandomization: Universal sets

Random 2-colorings

- enough random 2-colorings
  $\rightarrow$ one is likely good

Deterministic 2-colorings

- cleverly choose multiple 2-colorings such that at least one is good

Make a wish

- set of colorings of n elements (vertices) with two colors
- every k-element subset is colored in every possible way

$\Big\}$ (n, k)-univeral set

Theorem (without proof)
An (n, k)-universal set can be computed on $O(2^{k+o(k)} \cdot n)$ time.

Derandomization: Universal sets

Random 2-colorings

- enough random 2-colorings
  $\rightarrow$ one is likely good

Deterministic 2-colorings

- cleverly choose multiple 2-colorings such that at least one is good

Make a wish

- set of colorings of n elements (vertices) with two colors
- every k-element subset is colored in every possible way

$\Big\}$ (n, k)-univeral set

Theorem (without proof)
An (n, k)-universal set can be computed on $O(2^{k+o(k)} \cdot n)$ time.

Theorem
SET SPLITTING can be solved in $4^{s+o(s)} \cdot n^{O(1)}$ time.

Note: witness has size $2s$
$\rightarrow$ choose $k = 2s$

Wrap-Up

Randomization

- guessing additional structure enables new solution approaches
- combines well with FPT if it suffices to be lucky on a substructure of size $f(k)$

Wrap-Up

Randomization

- guessing additional structure enables new solution approaches
- combines well with FPT if it suffices to be lucky on a substructure of size $f(k)$
- randomization is only an intermediate step $\rightarrow$ we can often derandomize

Wrap-Up

Randomization

- guessing additional structure enables new solution approaches
- combines well with FPT if it suffices to be lucky on a substructure of size $f(k)$
- randomization is only an intermediate step $\rightarrow$ we can often derandomize

Color coding

- coloring or partitioning yields useful additional structure

Wrap-Up

Randomization

- guessing additional structure enables new solution approaches
- combines well with FPT if it suffices to be lucky on a substructure of size $f(k)$
- randomization is only an intermediate step $\rightarrow$ we can often derandomize

Color coding

- coloring or partitioning yields useful additional structure
- central tools: perfect hash families and universal sets

Wrap-Up

Randomization

- guessing additional structure enables new solution approaches
- combines well with FPT if it suffices to be lucky on a substructure of size $f(k)$
- randomization is only an intermediate step $\rightarrow$ we can often derandomize

Color coding

- coloring or partitioning yields useful additional structure
- central tools: perfect hash families and universal sets
- formally, color coding is some kind of Turing reduction