Guarding Polyominoes Under k-Hop Visibility

Christiane Schmidt

Malmö University, December 5, 2024

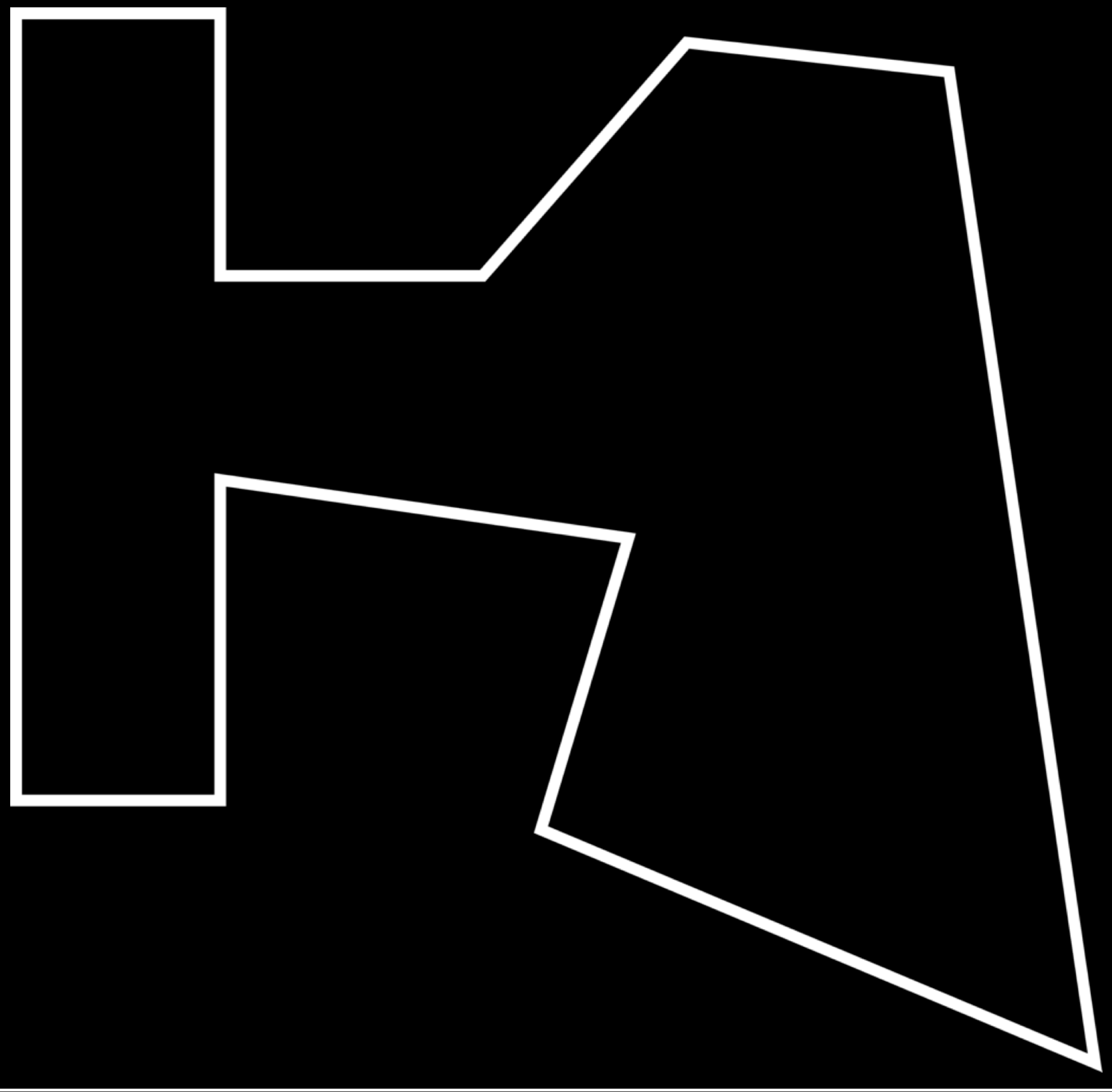
Based on joint work with

Omrit Filtser, Erik Krohn, Bengt J. Nilsson, and Christian Rieck

Agenda

- The Art Gallery Problem and Its Variants
- *k*-Hop Visibility
- What Do We Know About Guarding Polyominoes + Thin Polygons and About the Minimum k-Hop Dominating Set Problem? Aka: Related Work
- Some More Definitions
- VC Dimension
- Computational Complexity: NP-Completeness for 1-Thin Polyominoes with Holes
- A Linear-Time 4-Approximation for Simple 2-Thin Polyominoes
- Outlook

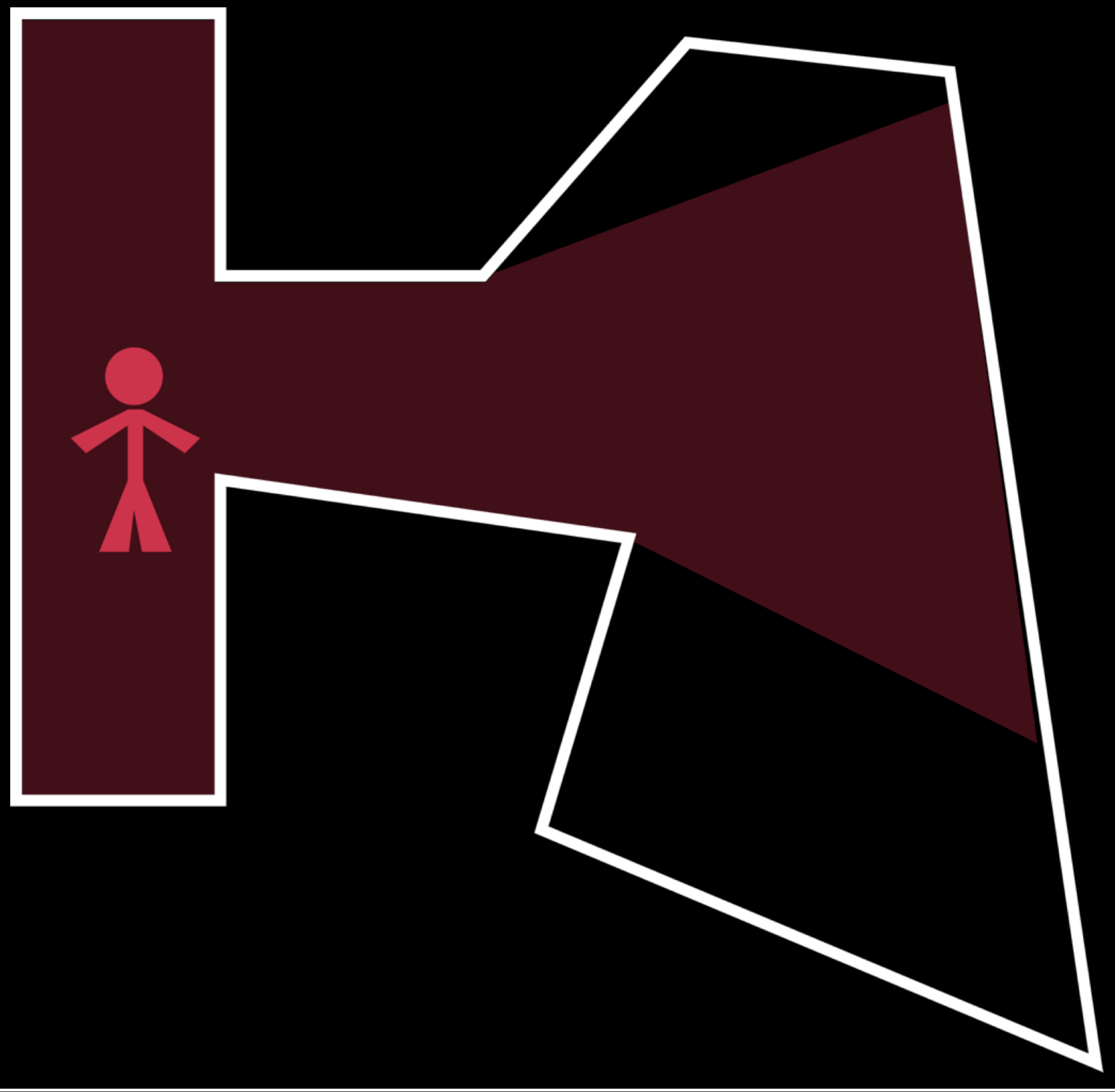
The Art Gallery Problem (AGP)



Given: Polygon P

How many guards do we need to monitor P?

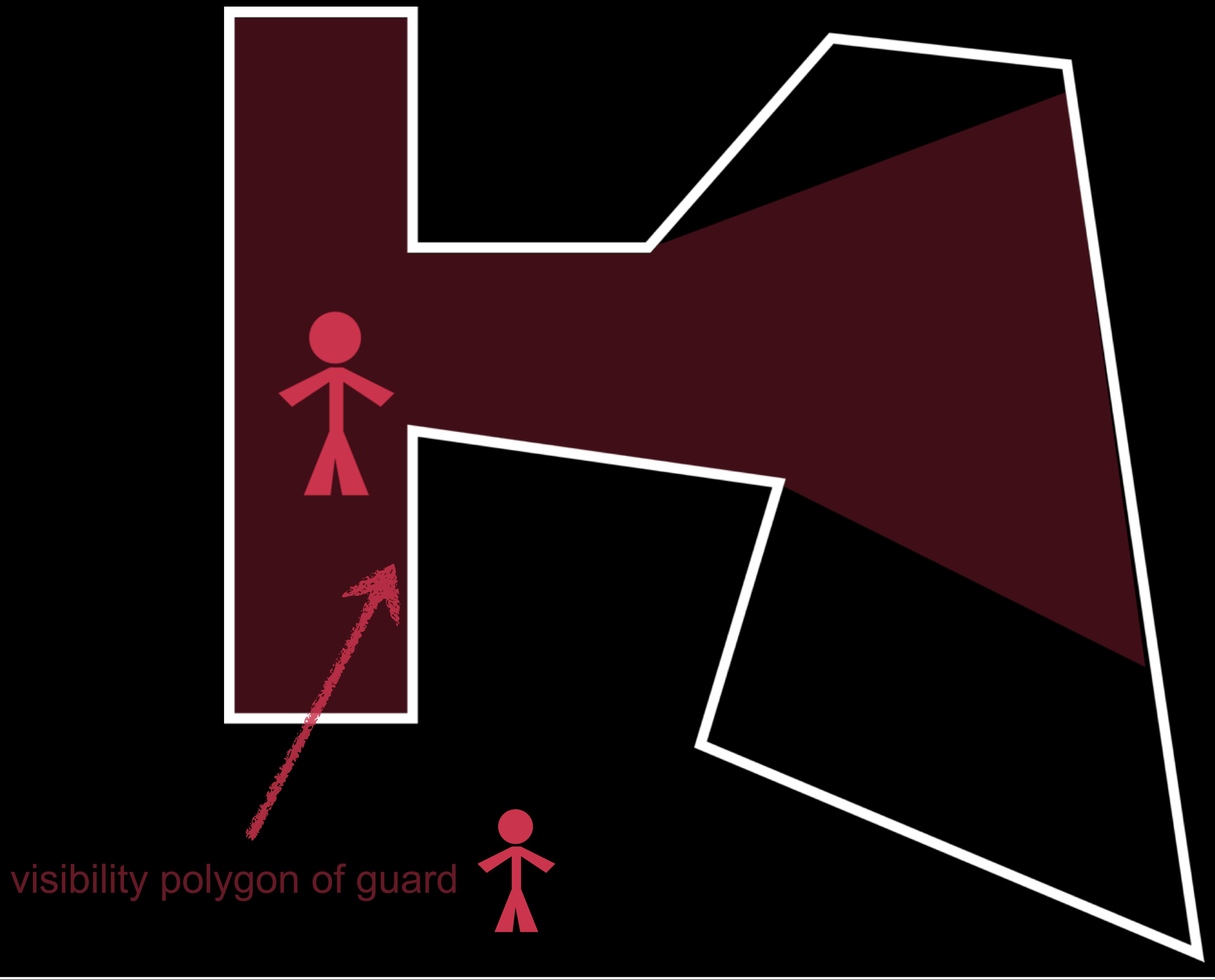
The Art Gallery Problem (AGP)



Given: Polygon P

How many guards do we need to monitor P?

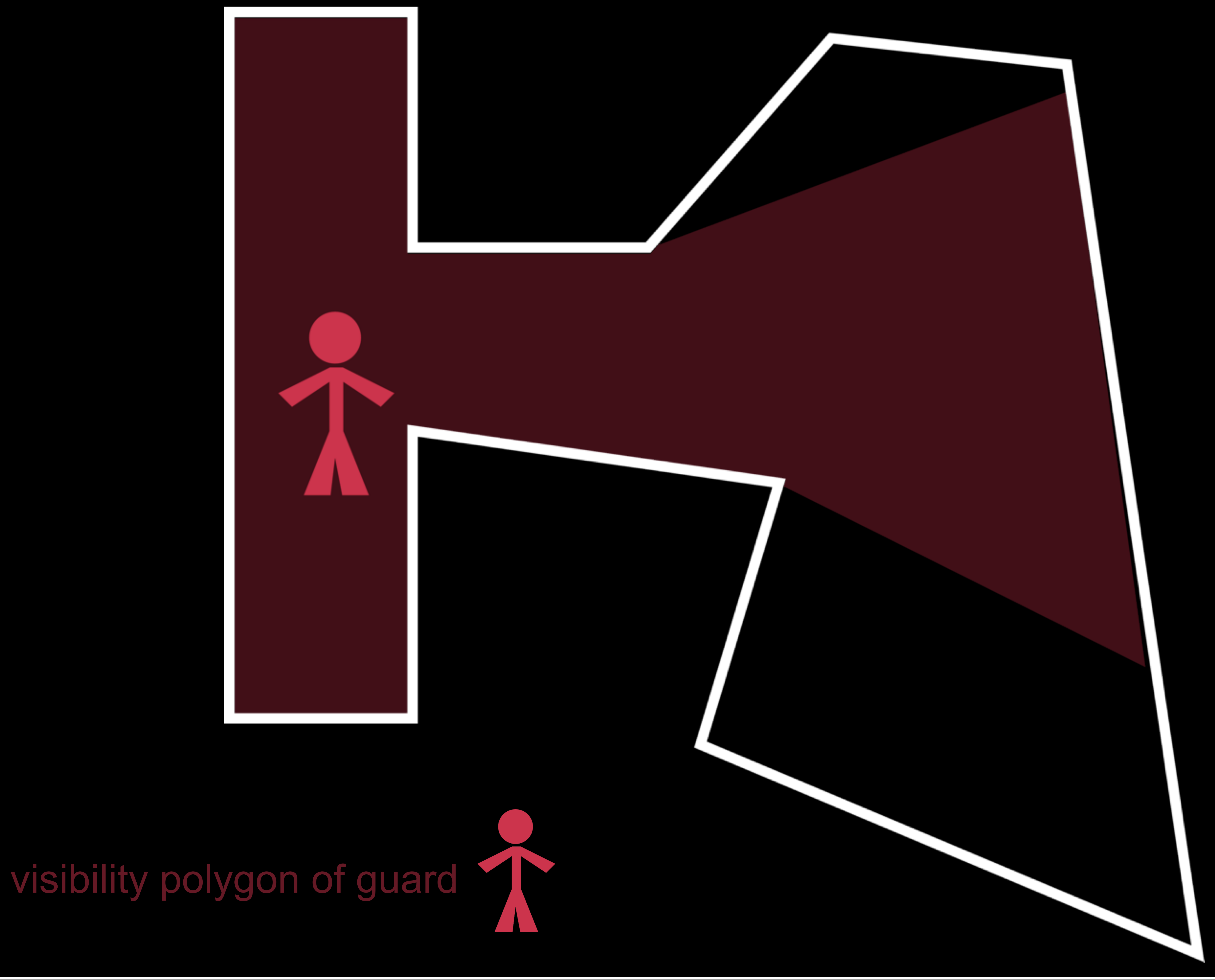
The Art Gallery Problem (AGP)



Given: Polygon P

How many guards do we need to monitor P?

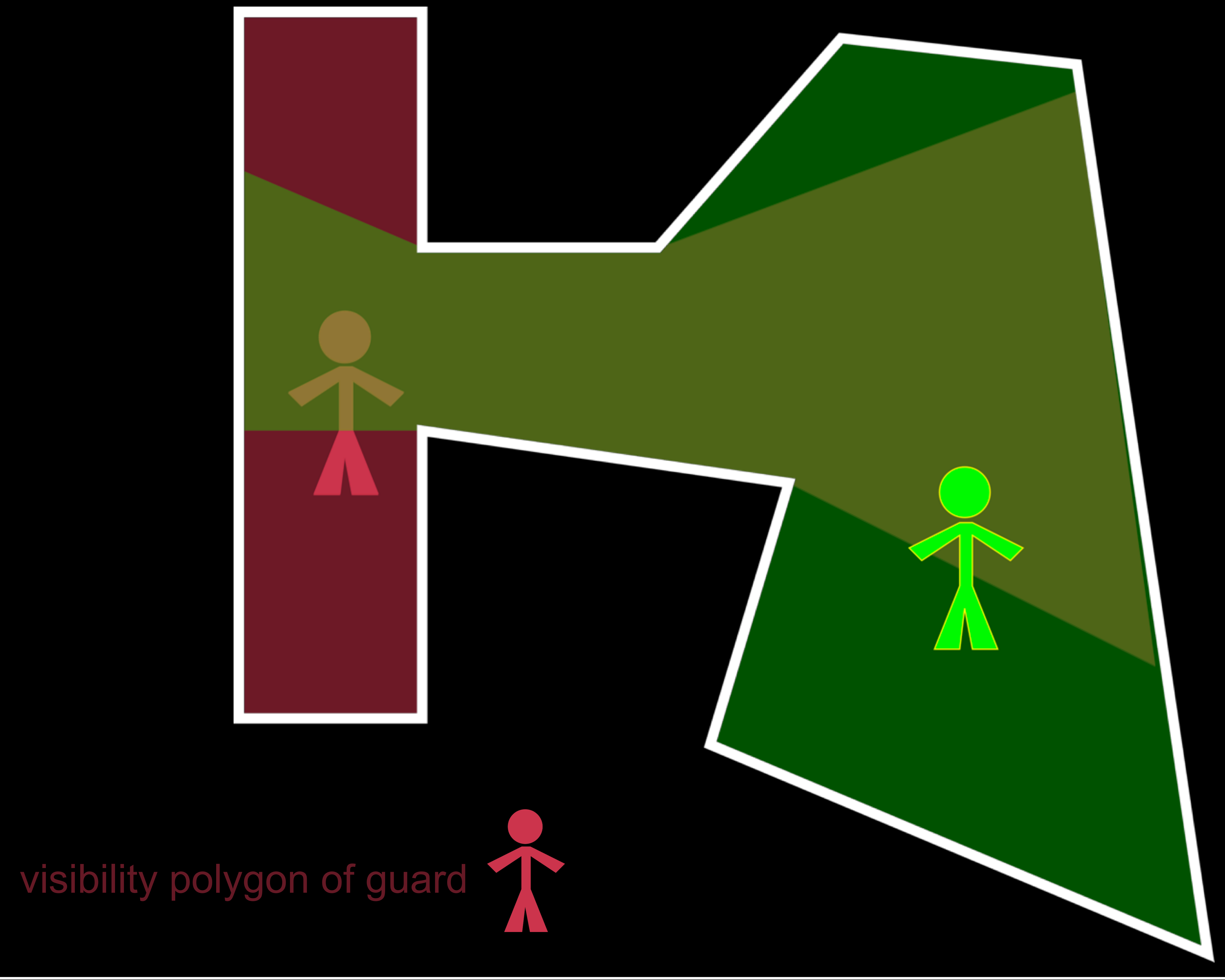
The Art Gallery Problem (AGP)



Given: Polygon P

How many guards do we need to monitor P?

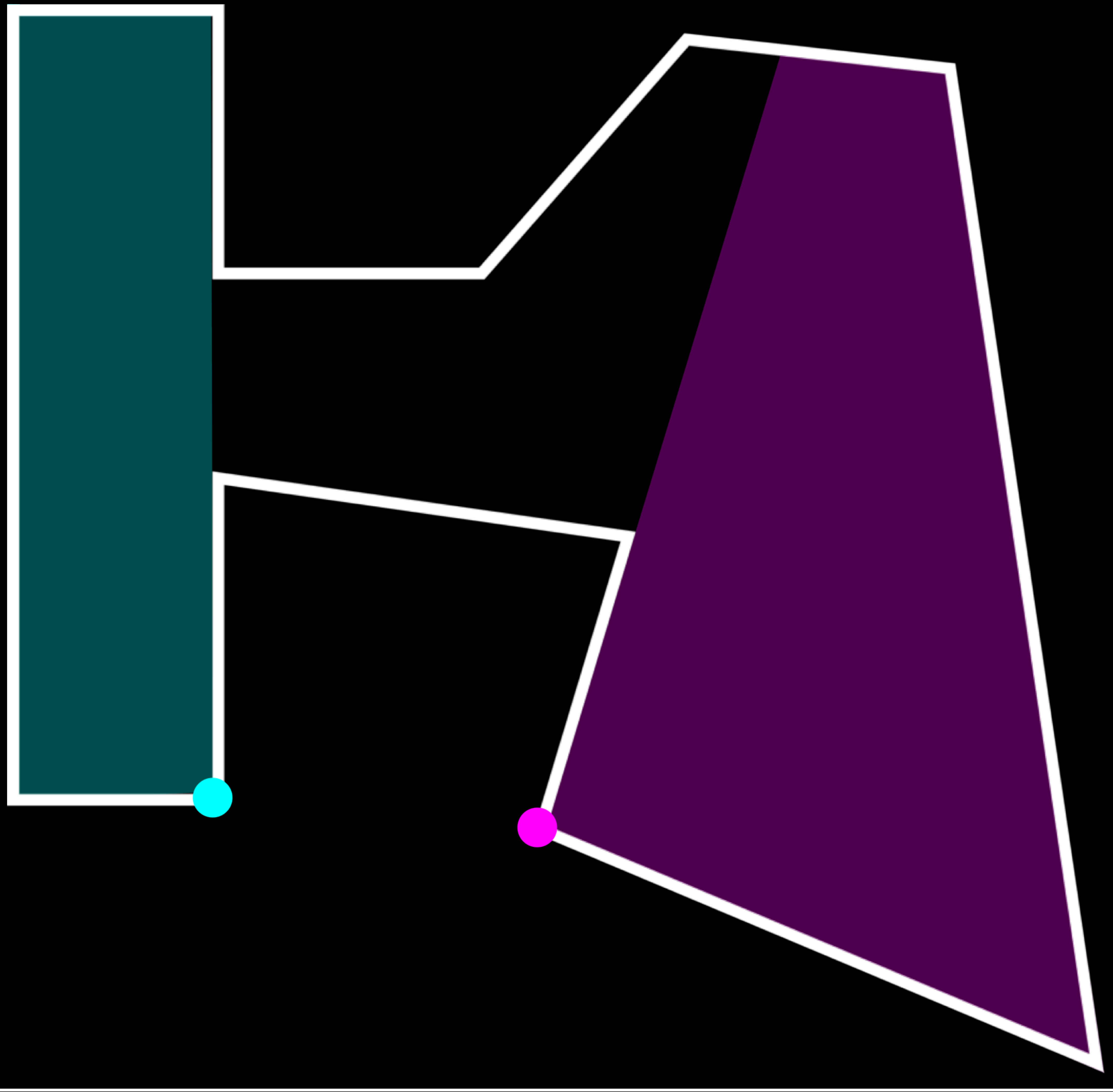
The Art Gallery Problem (AGP)



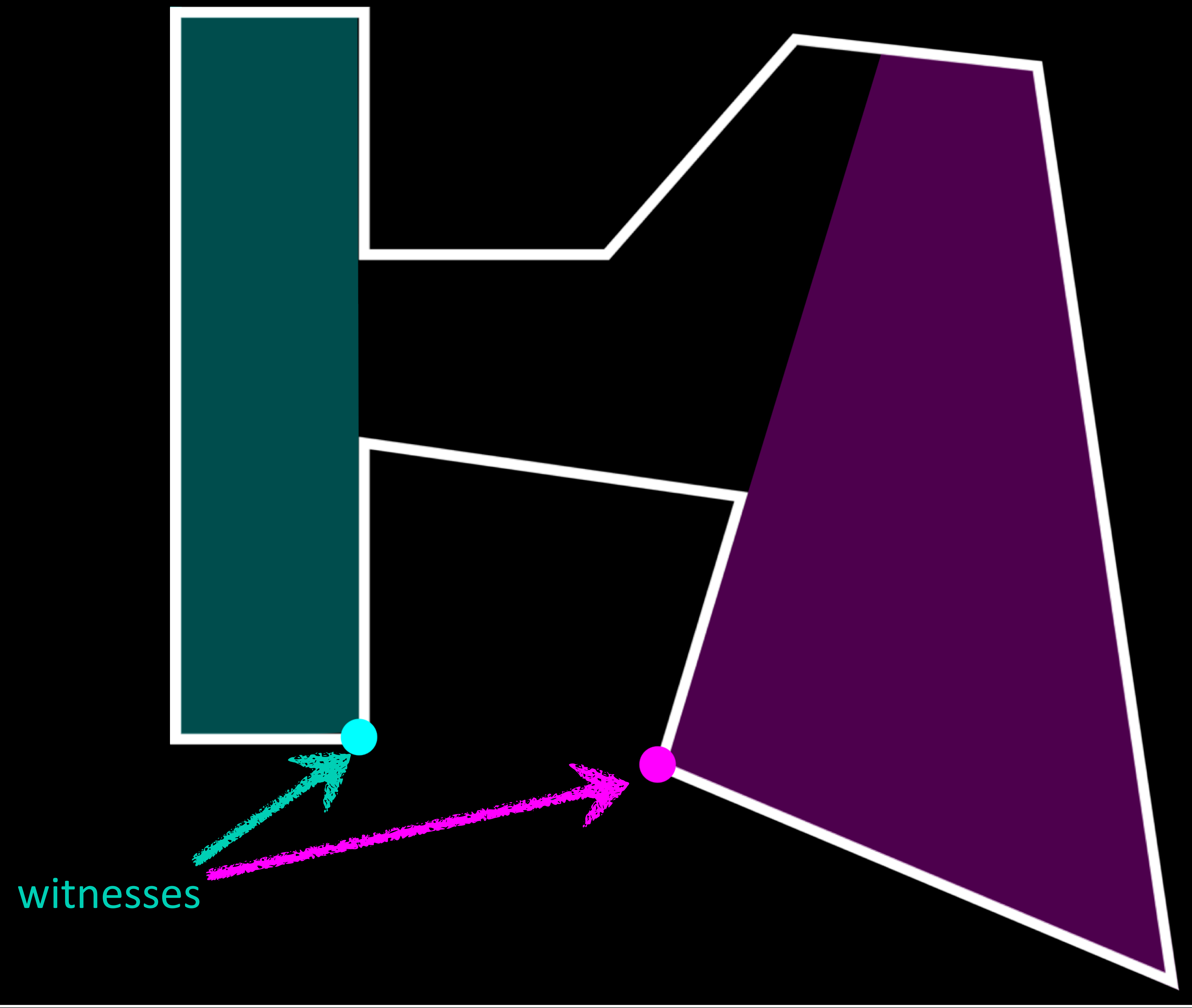
Given: Polygon P

How many guards do we need to monitor P?

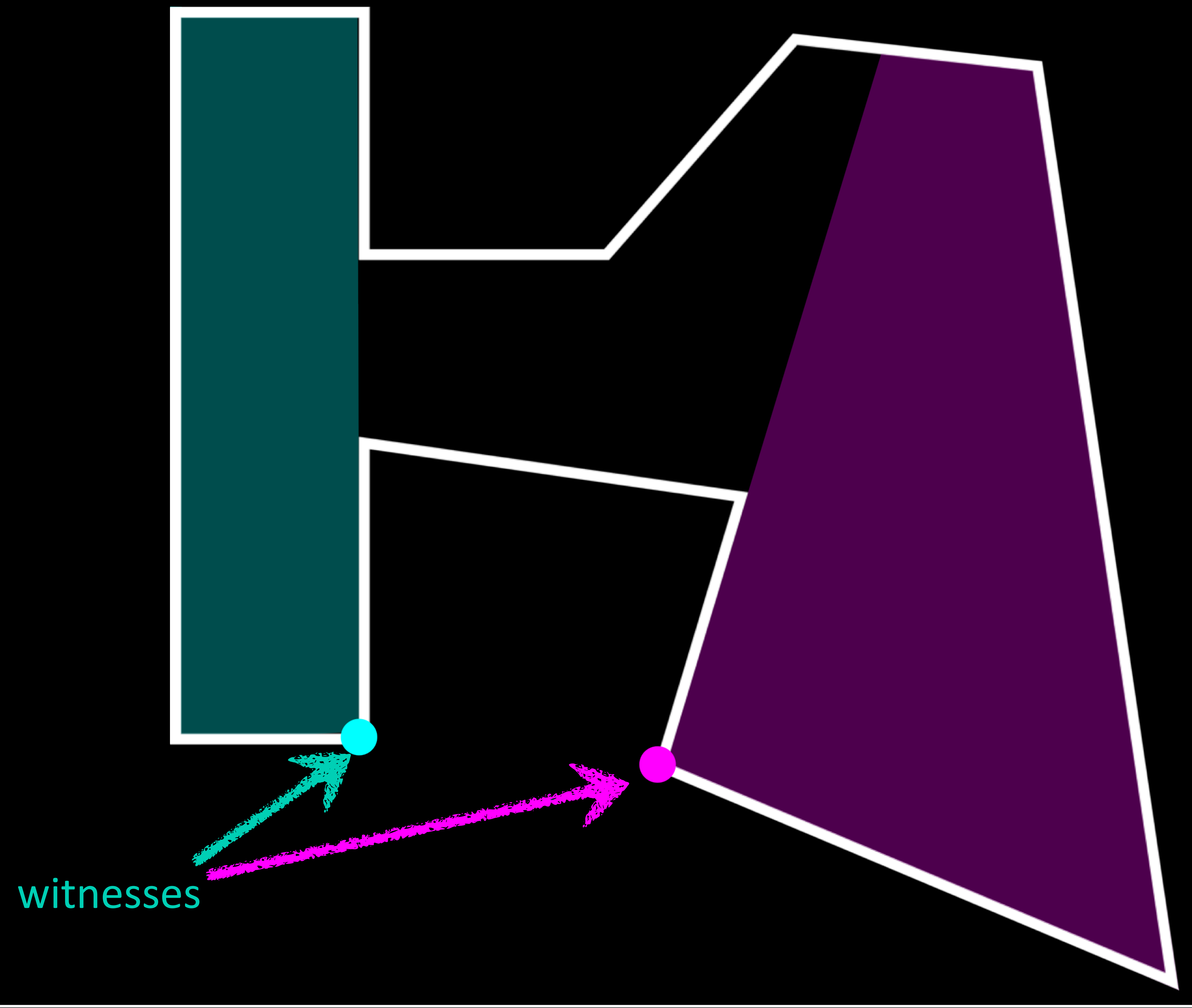
The Art Gallery Problem (AGP)



The Art Gallery Problem (AGP)

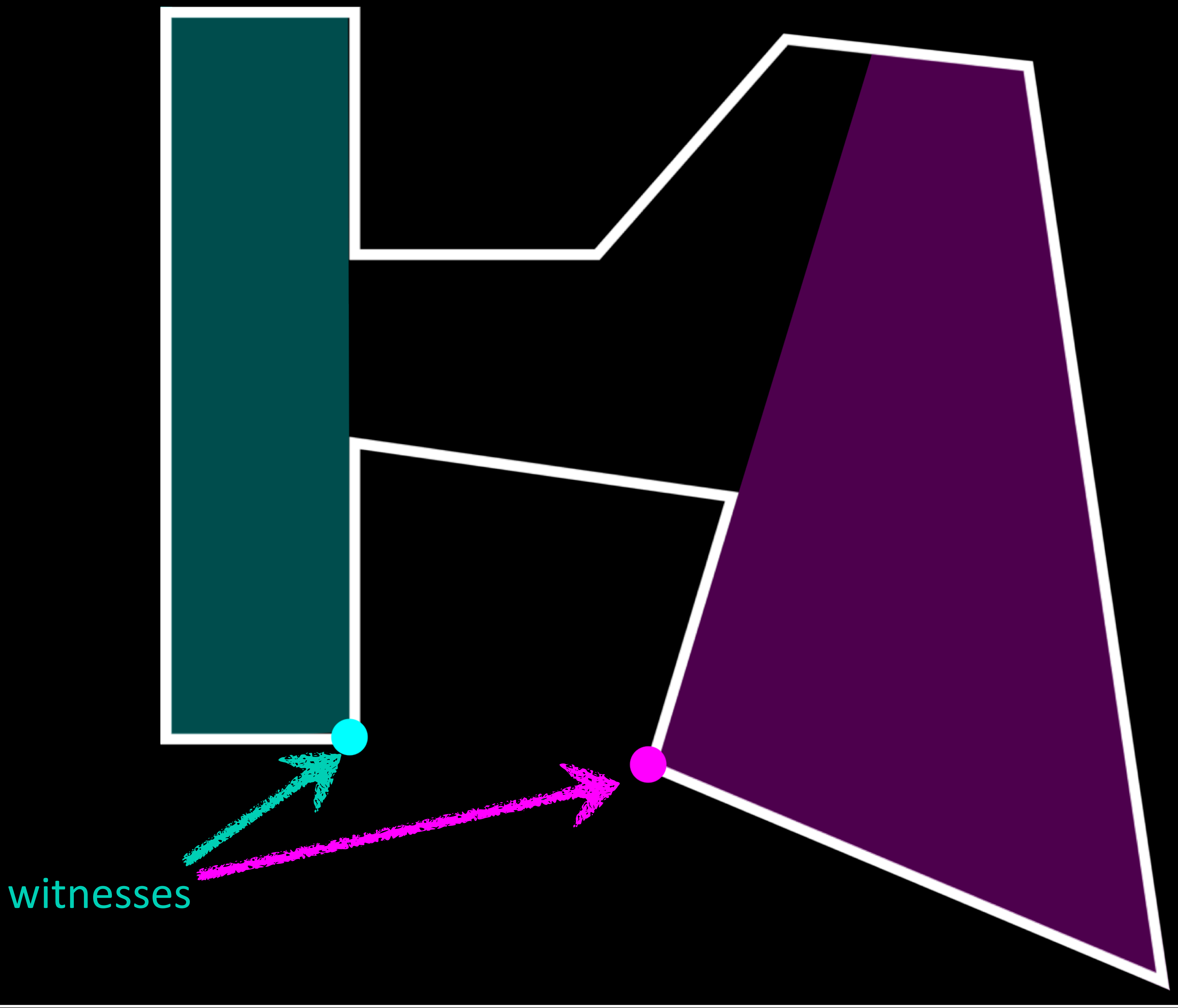


The Art Gallery Problem (AGP)



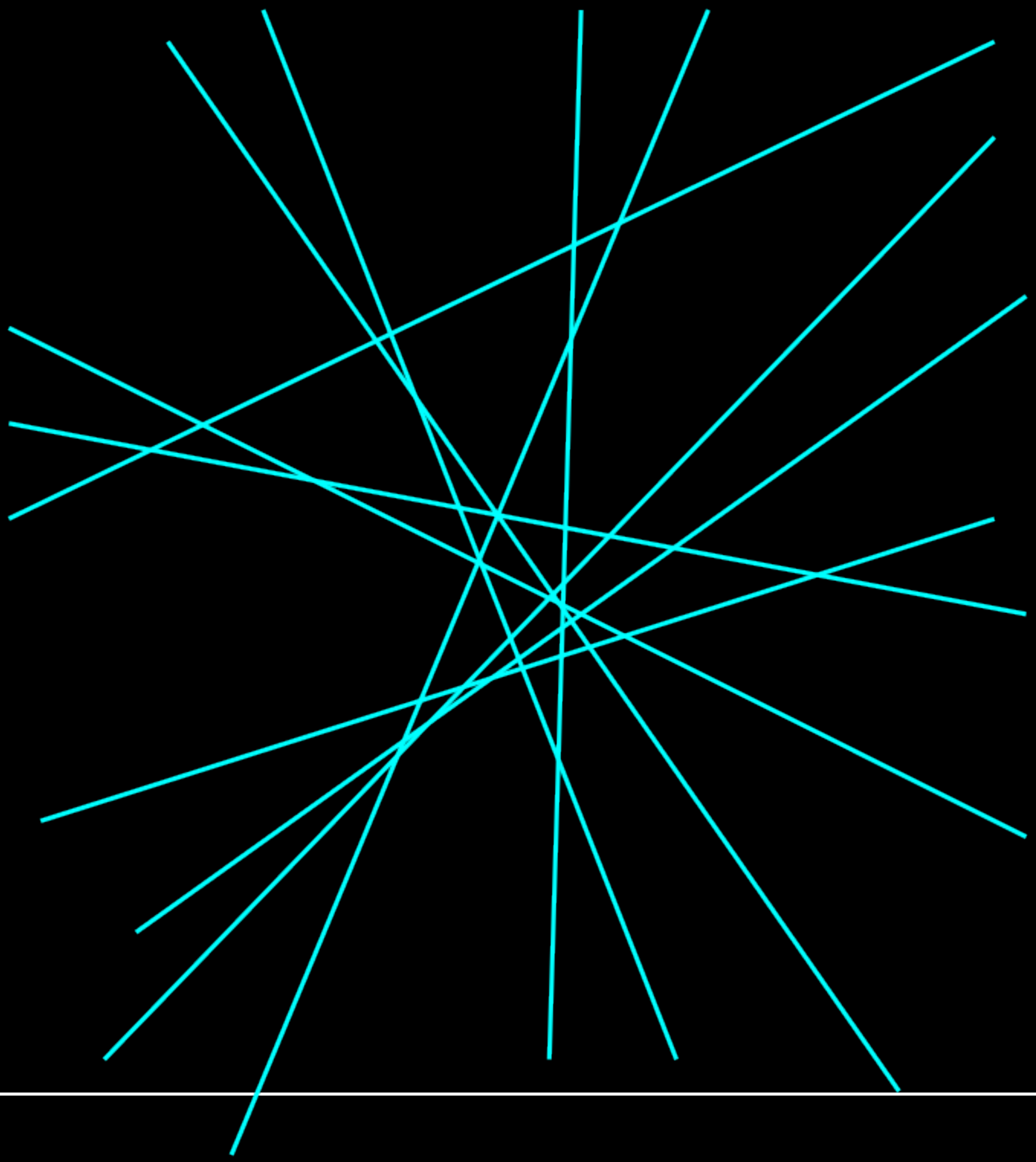
➜ Lower bound of 2
However, generally, the ratio between minimum number of guards and maximum number of witnesses can be arbitrarily bad:

The Art Gallery Problem (AGP)



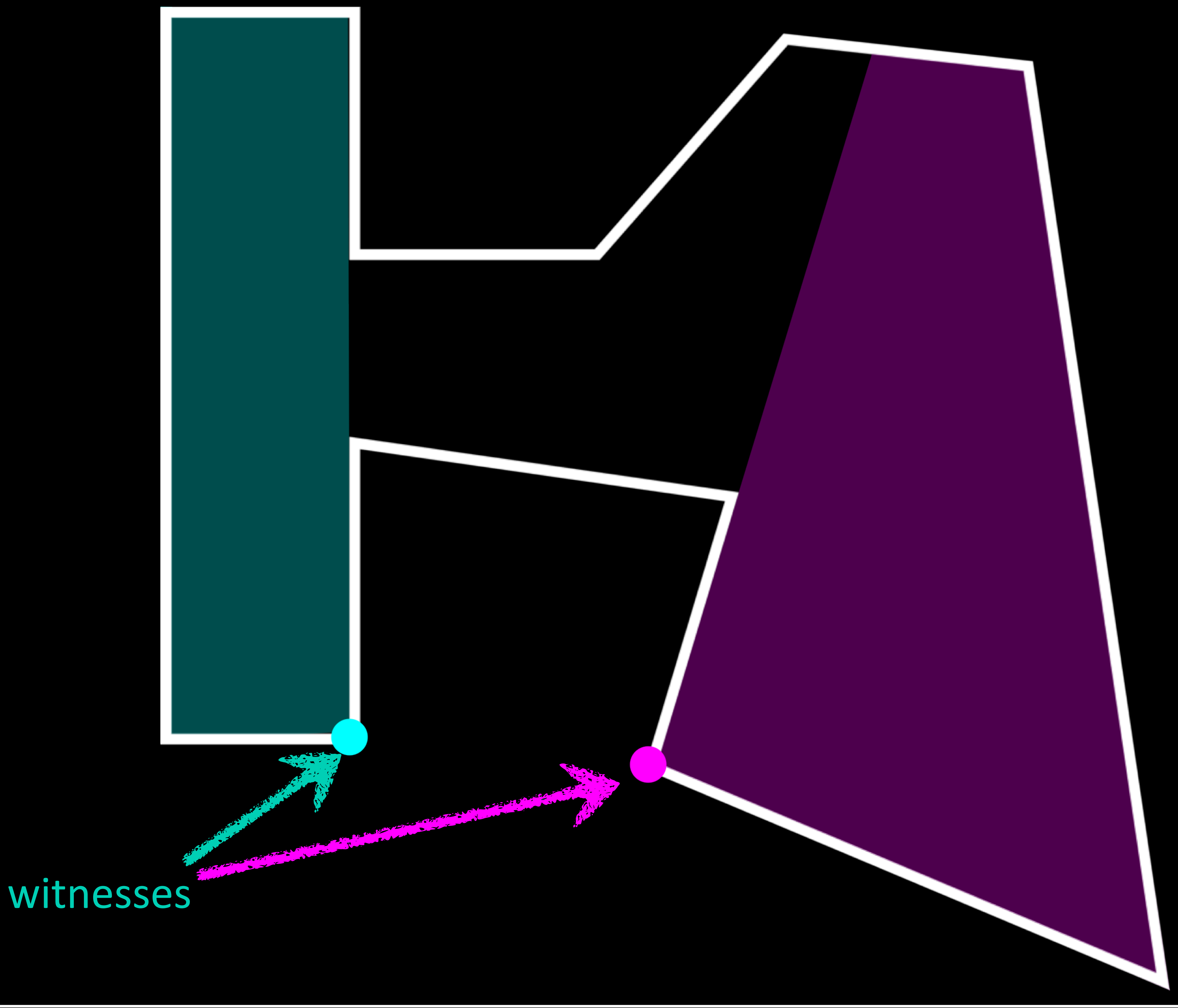
➜ Lower bound of 2

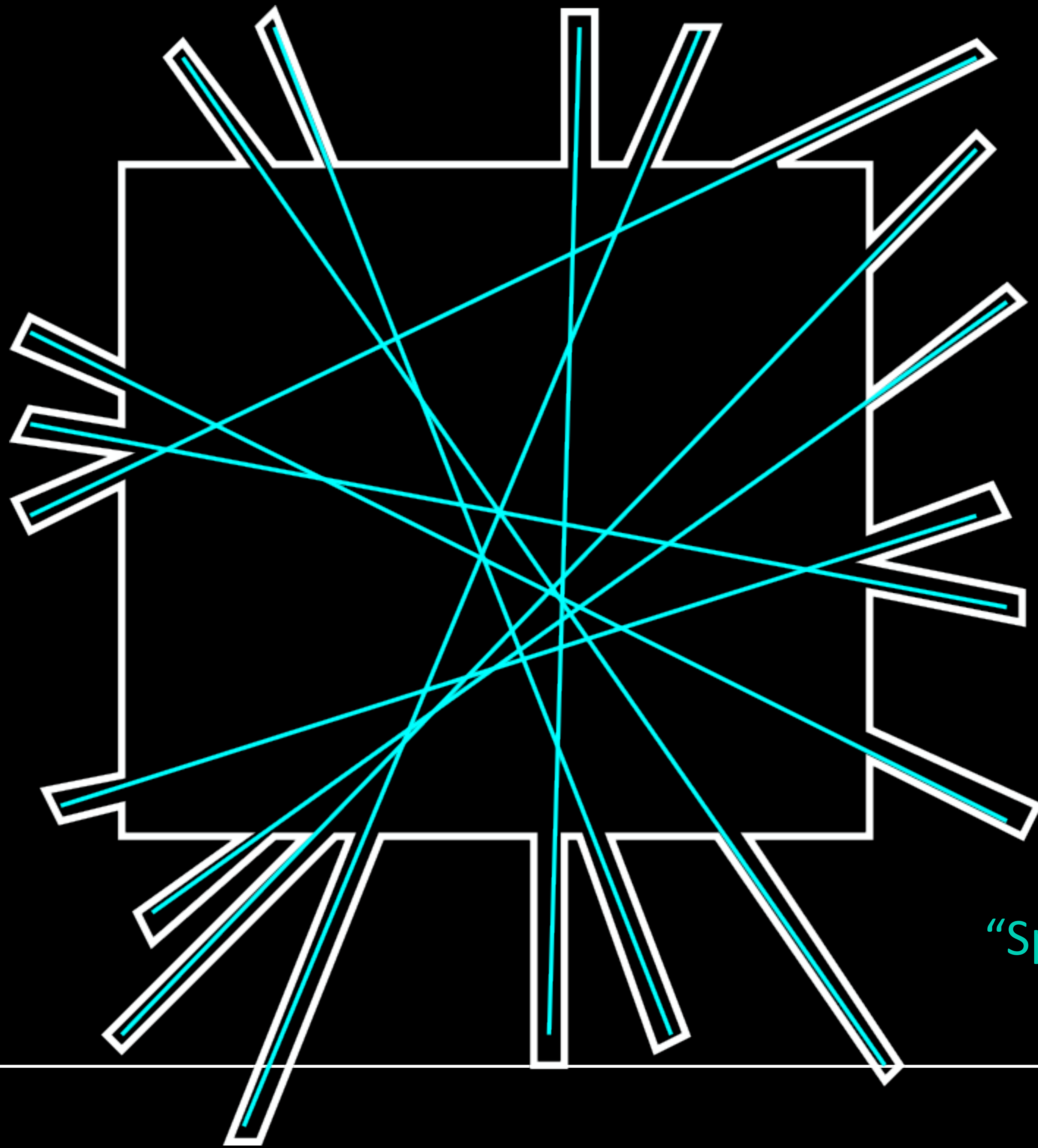
However, generally, the ratio between minimum number of guards and maximum number of witnesses can be arbitrarily bad:



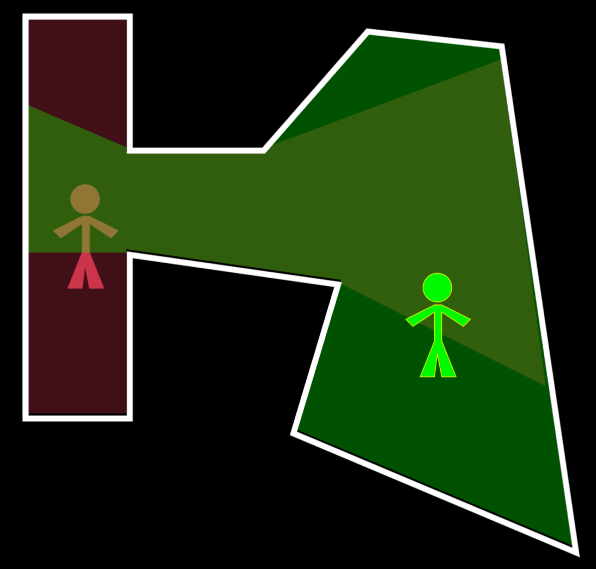
The Art Gallery Problem (AGP)

➜ Lower bound of 2
However, generally, the ratio between minimum number of guards and maximum number of witnesses can be arbitrarily bad:

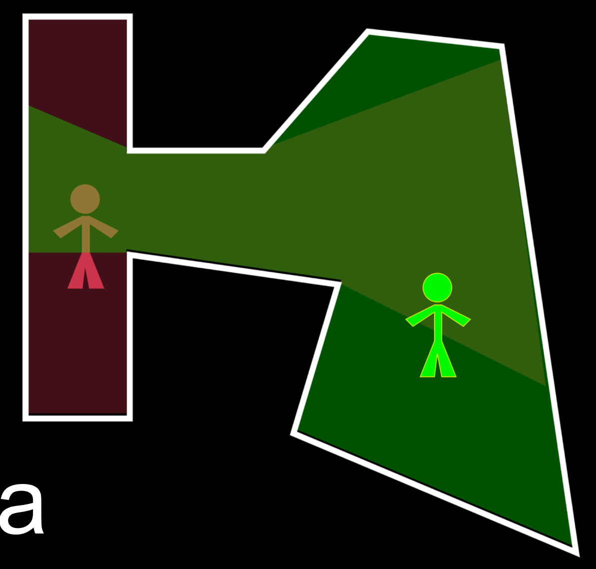




The Art Gallery Problem (AGP)

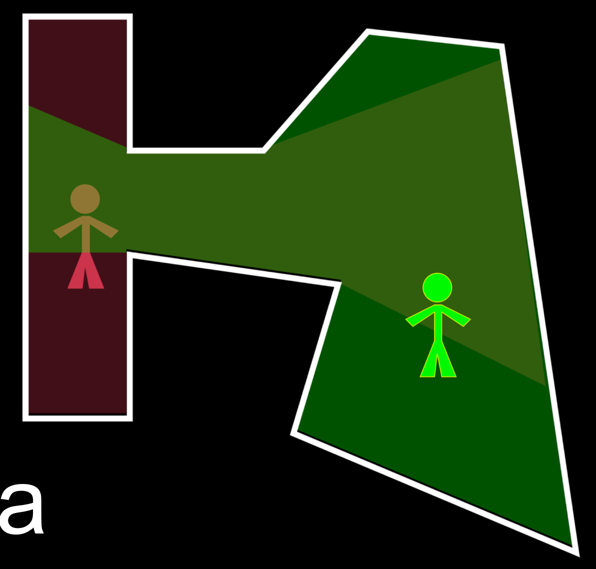


The Art Gallery Problem (AGP)



So-called “Art Gallery Theorems”: x guards are always sufficient and sometimes necessary to guard a polygon with n vertices (polygon from a specific class)

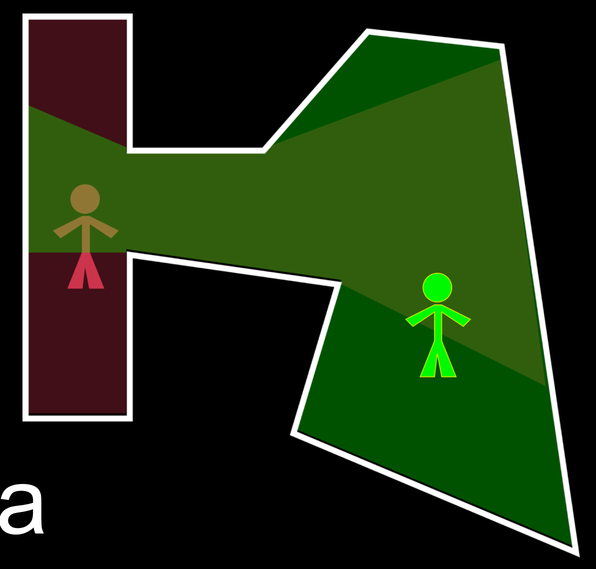
The Art Gallery Problem (AGP)



So-called “Art Gallery Theorems”: x guards are always sufficient and sometimes necessary to guard a polygon with n vertices (polygon from a specific class)

- Simple polygon with *n* vertices: $\lfloor \frac{n}{3} \rfloor$ are sometimes necessary and always sufficient. [Chvátal ’75]

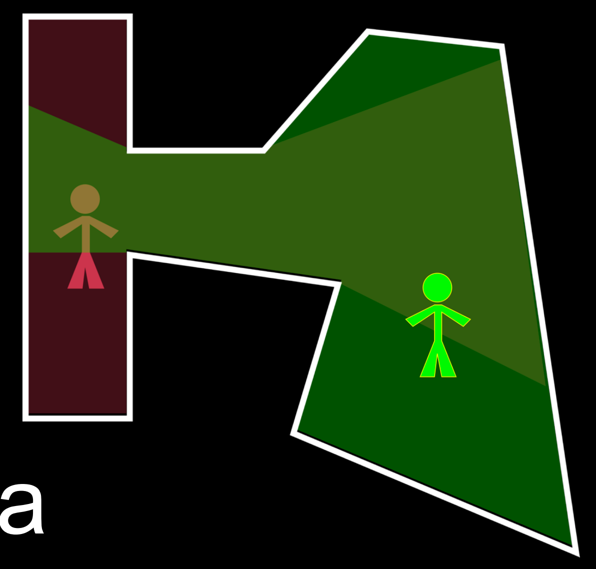
The Art Gallery Problem (AGP)



So-called “Art Gallery Theorems”: x guards are always sufficient and sometimes necessary to guard a polygon with n vertices (polygon from a specific class)

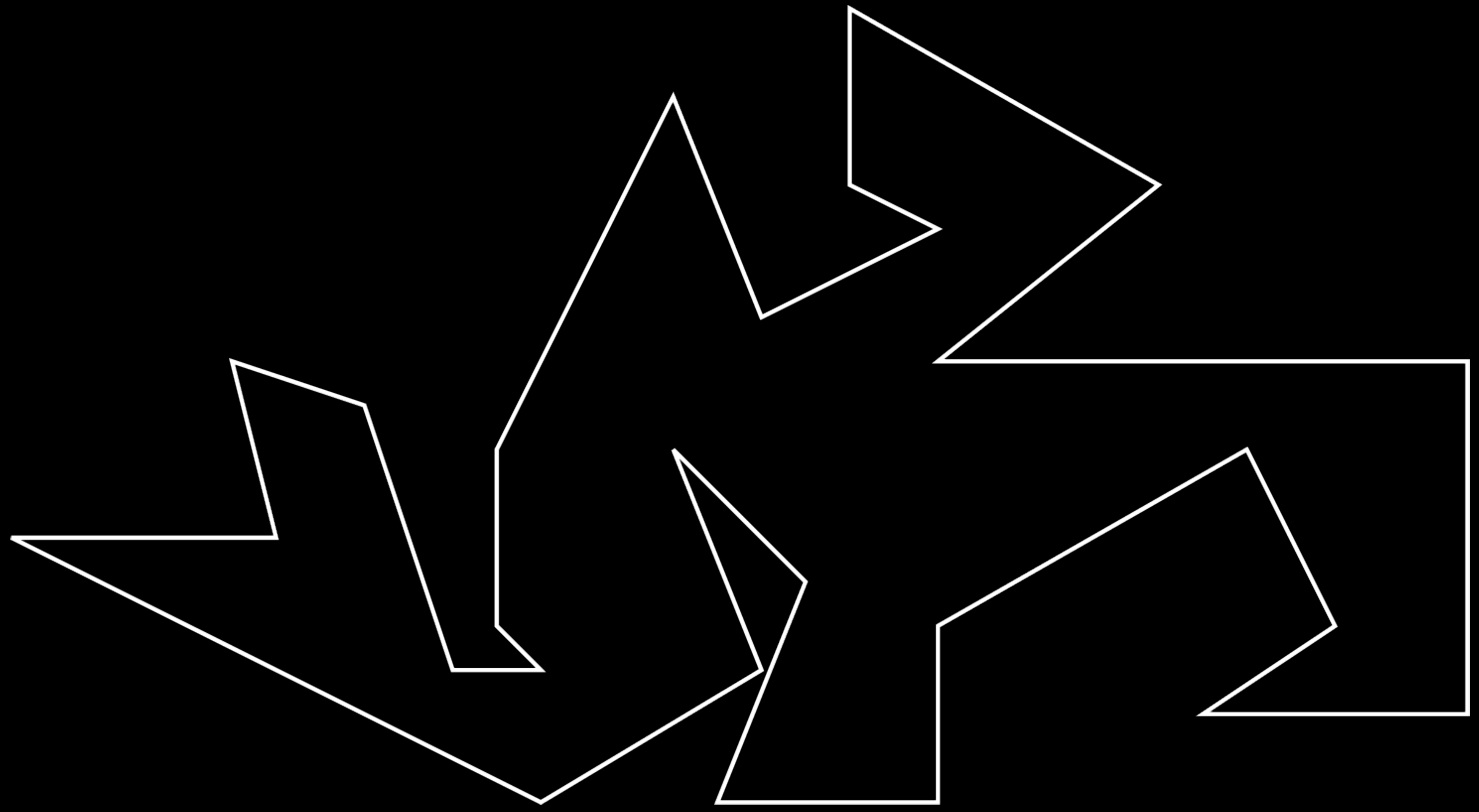
- Simple polygon with *n* vertices: $\lfloor \frac{n}{3} \rfloor$ are sometimes necessary and always sufficient. [Chvátal ’75]

The Art Gallery Problem (AGP)

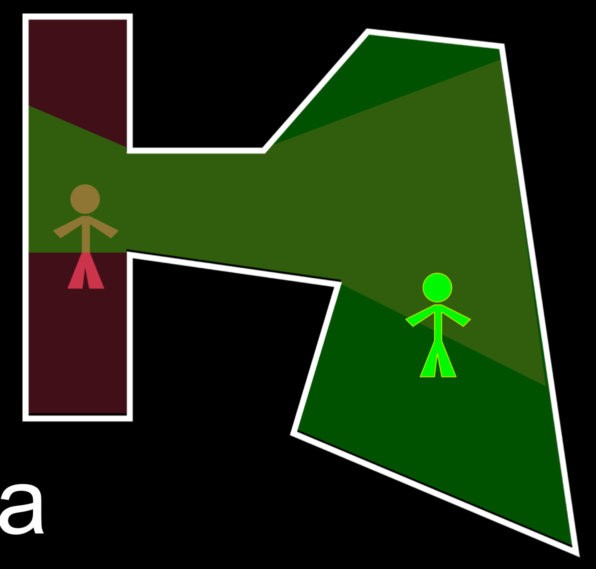


So-called “Art Gallery Theorems”: x guards are always sufficient and sometimes necessary to guard a polygon with n vertices (polygon from a specific class)

- Simple polygon with *n* vertices: $\lfloor \frac{n}{3} \rfloor$ are sometimes necessary and always sufficient. [Chvátal ’75]

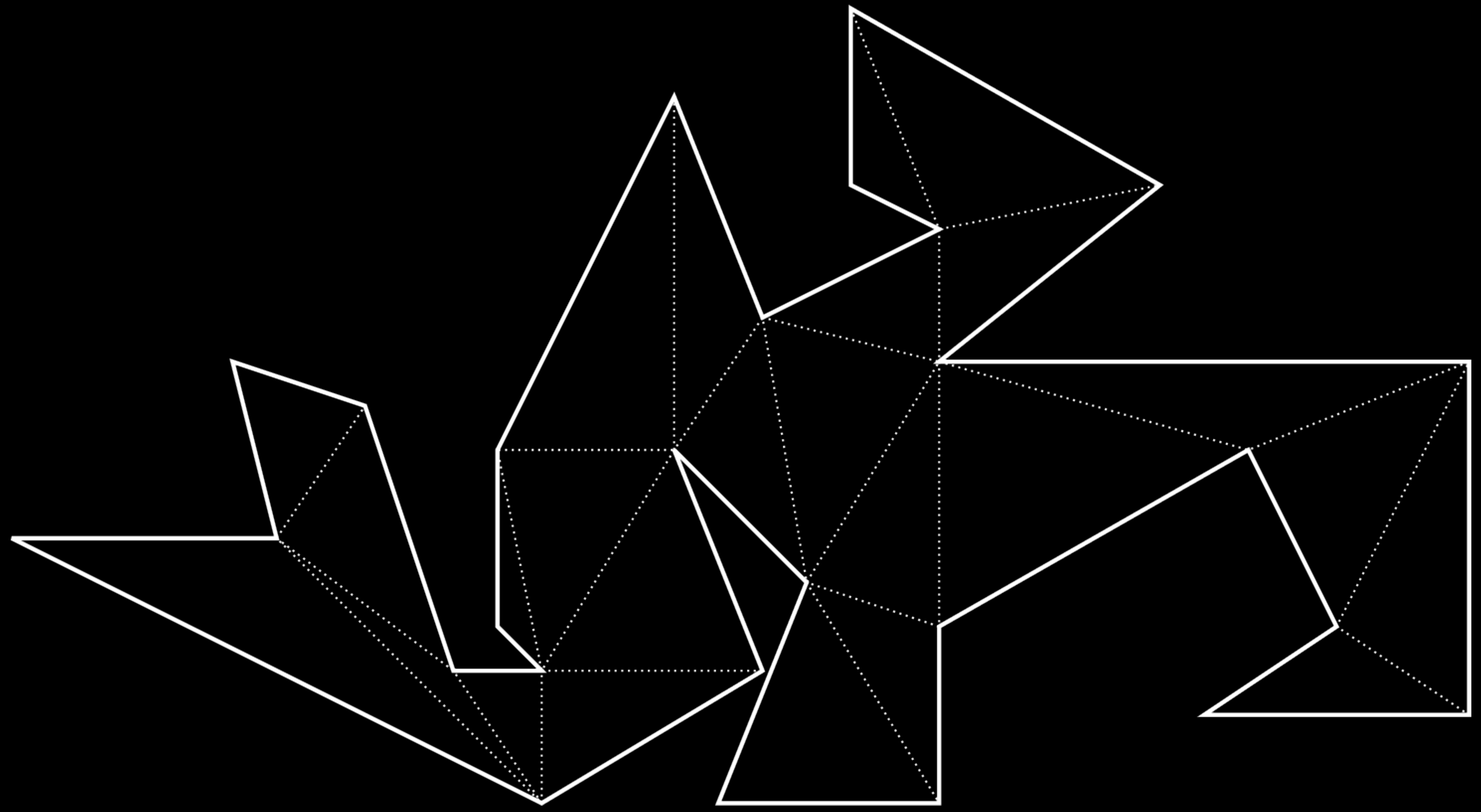


The Art Gallery Problem (AGP)

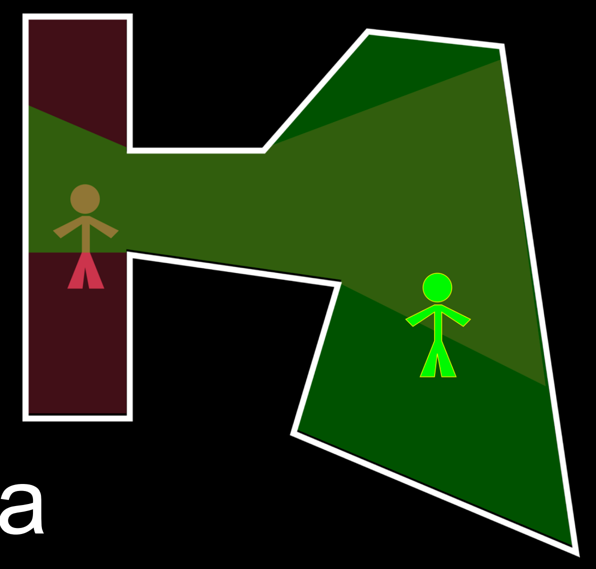


So-called “Art Gallery Theorems”: x guards are always sufficient and sometimes necessary to guard a polygon with n vertices (polygon from a specific class)

- Simple polygon with *n* vertices: $\lfloor \frac{n}{3} \rfloor$ are sometimes necessary and always sufficient. [Chvátal ’75]

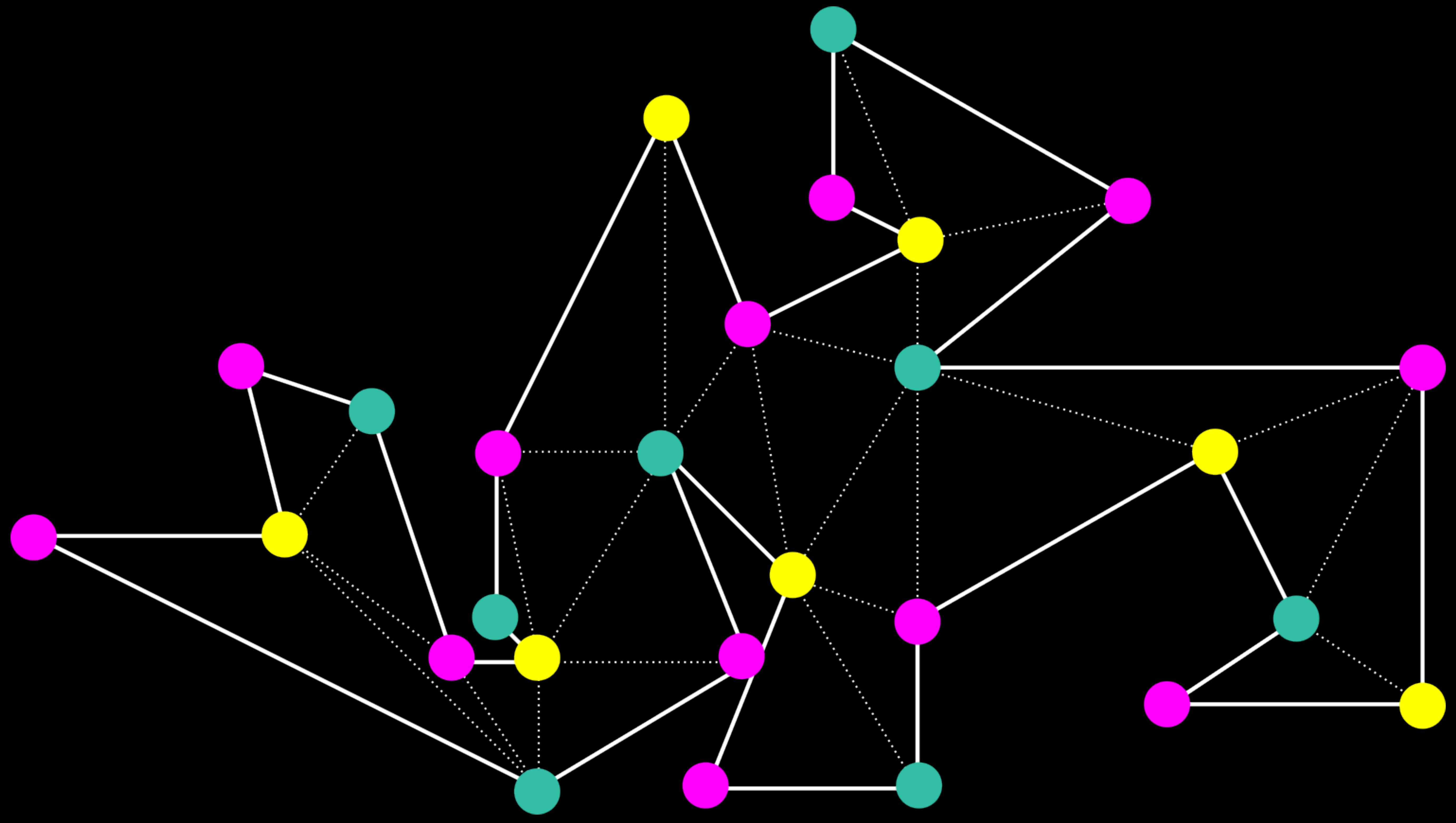


The Art Gallery Problem (AGP)

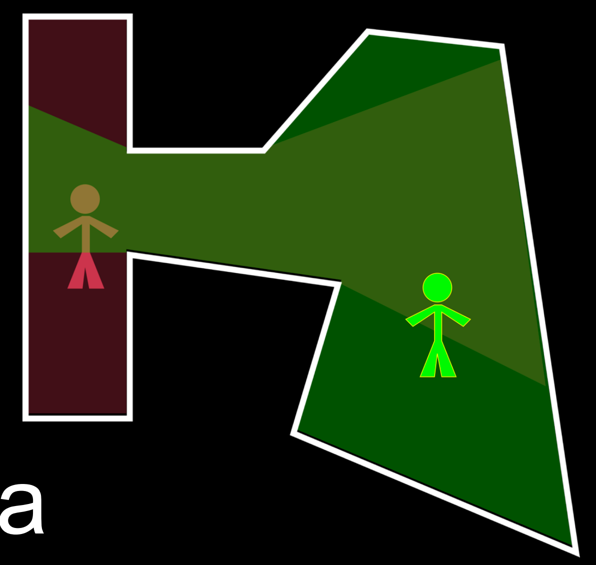


So-called “Art Gallery Theorems”: x guards are always sufficient and sometimes necessary to guard a polygon with n vertices (polygon from a specific class)

- Simple polygon with *n* vertices: $\lfloor \frac{n}{3} \rfloor$ are sometimes necessary and always sufficient. [Chvátal ’75]

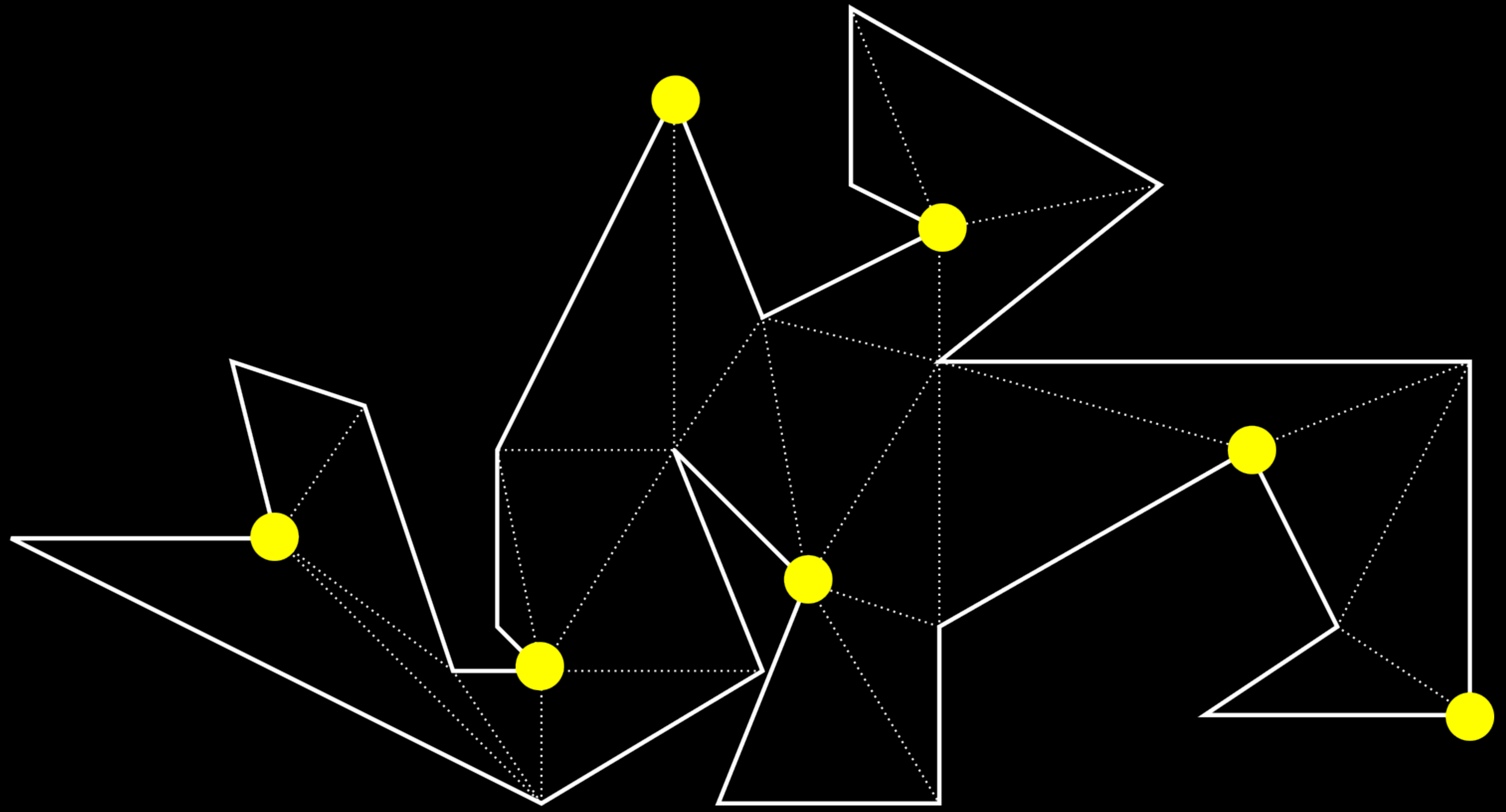


The Art Gallery Problem (AGP)

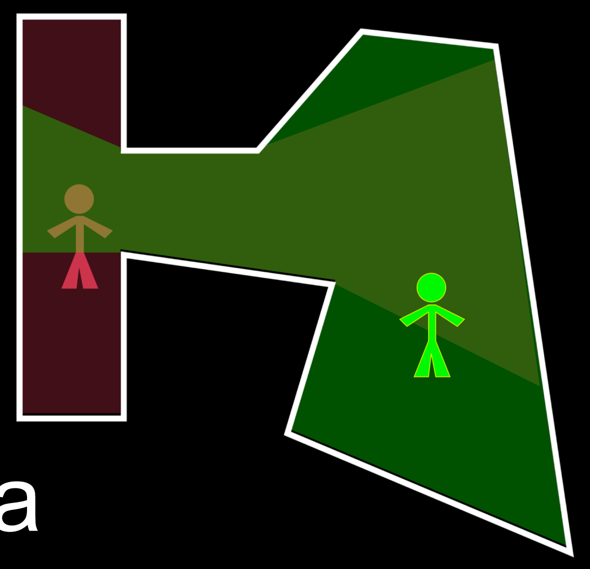


So-called "Art Gallery Theorems": x guards are always sufficient and sometimes necessary to guard a polygon with n vertices (polygon from a specific class)

- Simple polygon with n vertices: $\lfloor \frac{n}{3} \rfloor$ are sometimes necessary and always sufficient. [Chvátal '75]



The Art Gallery Problem (AGP)



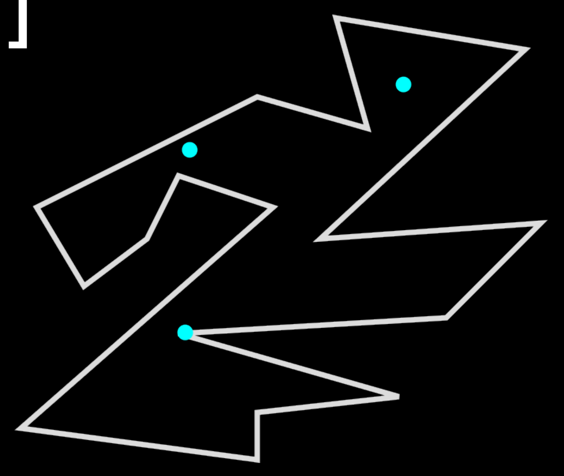
So-called “Art Gallery Theorems”: x guards are always sufficient and sometimes necessary to guard a polygon with n vertices (polygon from a specific class)

- Simple polygon with *n* vertices: $\lfloor \frac{n}{3} \rfloor$ are sometimes necessary and always sufficient. [Chvátal ’75]

Computational Complexity

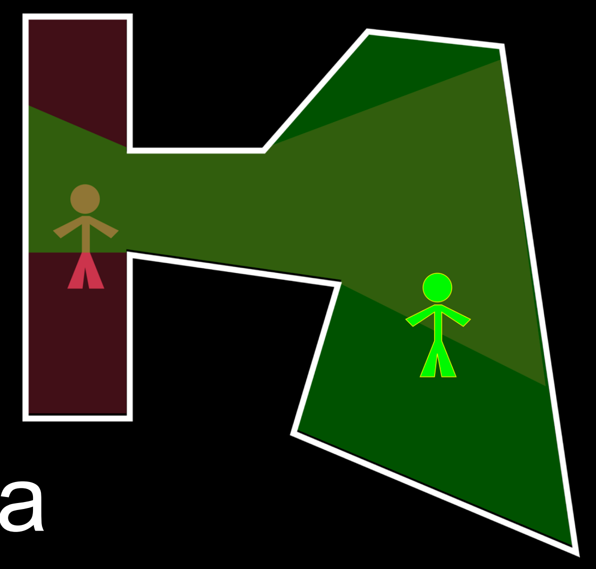
- The AGP is NP-hard for point guards with holes [O’Rourke & Supowit 1983] , vertex guards without holes [Lee & Lin 1986], point guards without holes [Aggarwal 1986]; point guards without holes is ∃R-hard [Abrahamsen et al. 2021]

point guards



vertex guards

The Art Gallery Problem (AGP)

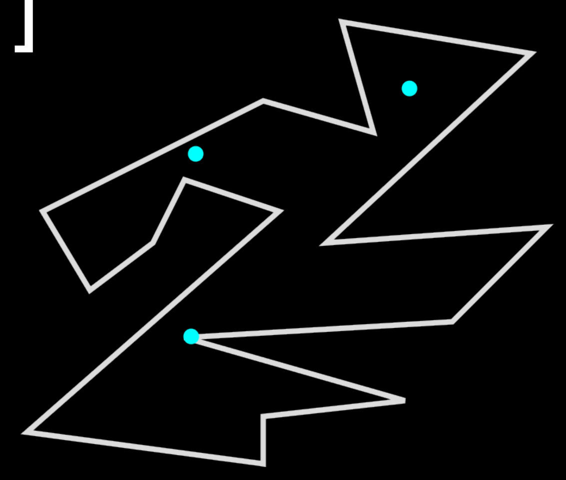
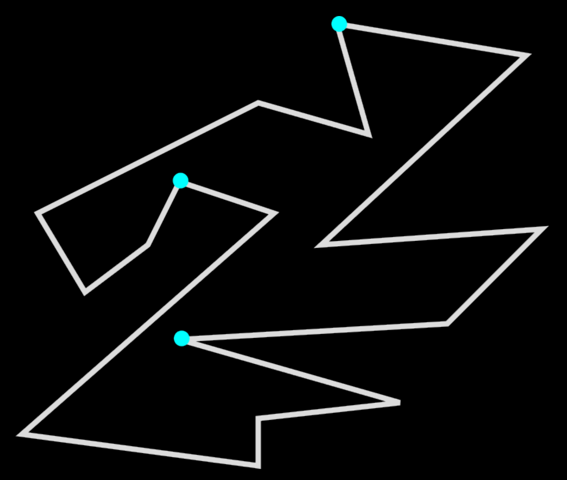


So-called "Art Gallery Theorems": x guards are always sufficient and sometimes necessary to guard a polygon with n vertices (polygon from a specific class)

- Simple polygon with n vertices: $\lfloor \frac{n}{3} \rfloor$ are sometimes necessary and always sufficient. [Chvátal '75]

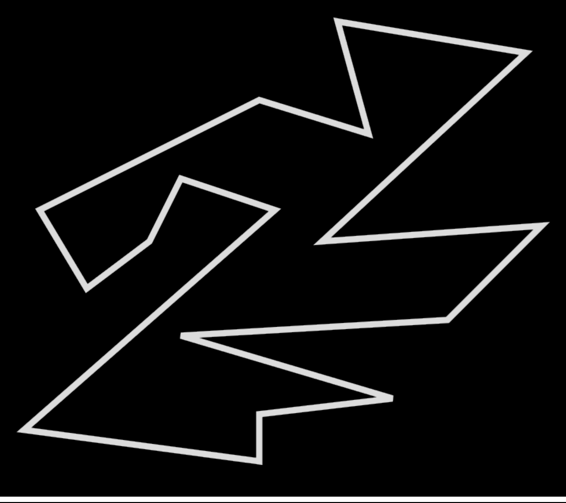
Computational Complexity

- The AGP is NP-hard for point guards with holes [O'Rourke & Supowit 1983] , vertex guards without holes [Lee & Lin 1986], point guards without holes [Aggarwal 1986]; point guards without holes is ∃R-hard [Abrahamsen et al. 2021]

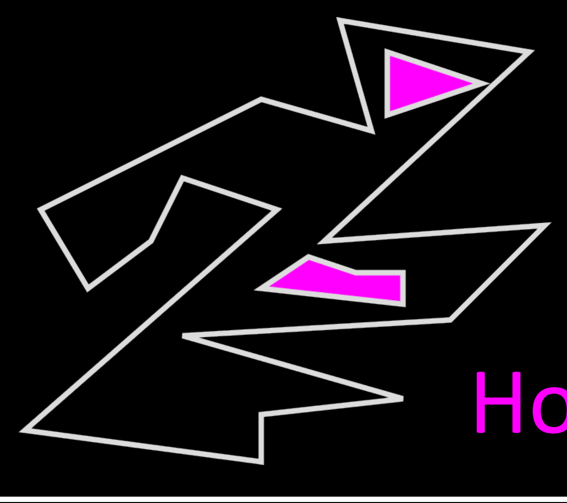
point guards   vertex guards

Simple polygon:

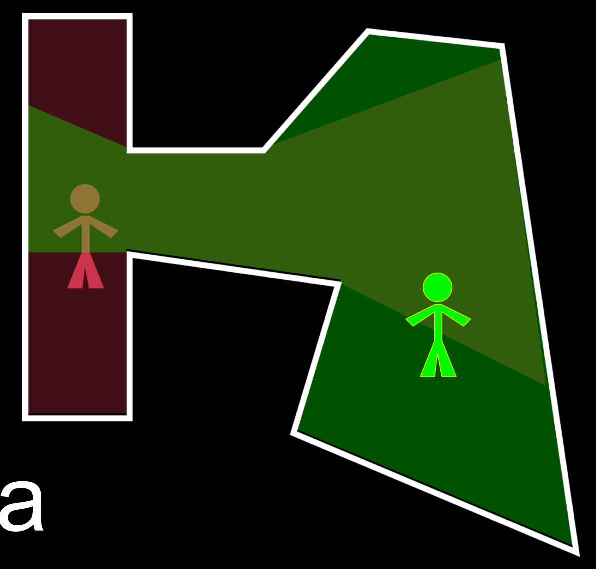
- Does not intersect itself
- No holes



⟺



The Art Gallery Problem (AGP)



So-called “Art Gallery Theorems”: x guards are always sufficient and sometimes necessary to guard a polygon with n vertices (polygon from a specific class)

- Simple polygon with *n* vertices: $\lfloor \frac{n}{3} \rfloor$ are sometimes necessary and always sufficient. [Chvátal ’75]

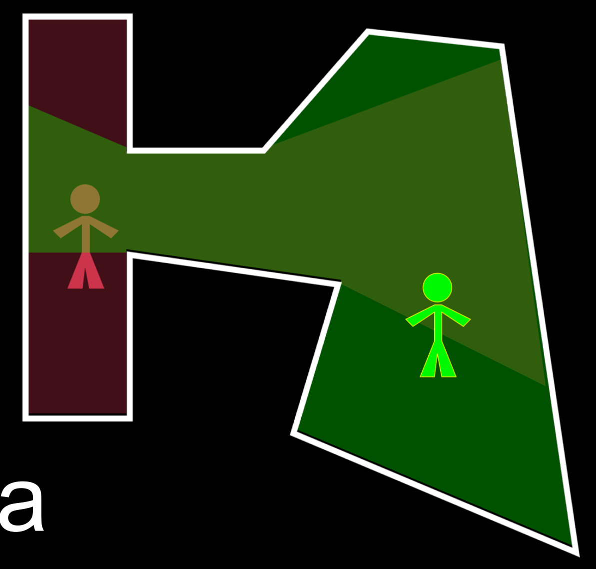
Computational Complexity

- The AGP is NP-hard for point guards with holes [O’Rourke & Supowit 1983] , vertex guards without holes [Lee & Lin 1986], point guards without holes [Aggarwal 1986]; point guards without holes is ∃R-hard [Abrahamsen et al. 2021]

Algorithms

- Depending on complexity: approximation algorithms, efficient algorithms for optimal solutions for many instances, heuristics; polytime algorithms

The Art Gallery Problem (AGP)



So-called “Art Gallery Theorems”: x guards are always sufficient and sometimes necessary to guard a polygon with n vertices (polygon from a specific class)

- Simple polygon with *n* vertices: $\lfloor \frac{n}{3} \rfloor$ are sometimes necessary and always sufficient. [Chvátal ’75]

Computational Complexity

- The AGP is NP-hard for point guards with holes [O’Rourke & Supowit 1983] , vertex guards without holes [Lee & Lin 1986], point guards without holes [Aggarwal 1986]; point guards without holes is ∃R-hard [Abrahamsen et al. 2021]

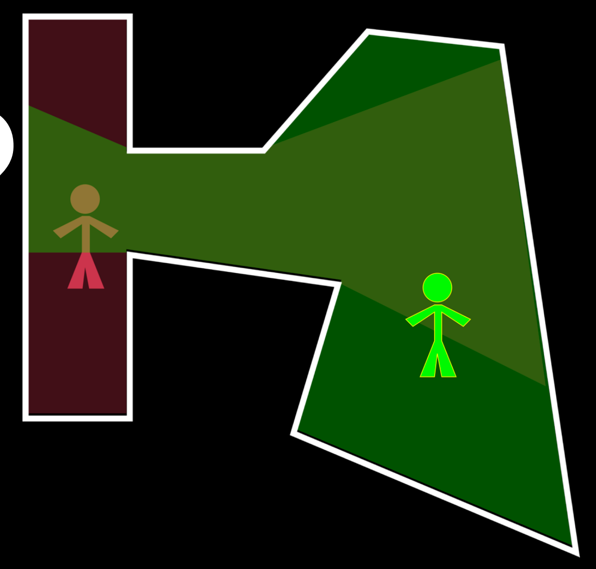
Algorithms

- Depending on complexity: approximation algorithms, efficient algorithms for optimal solutions for many instances, heuristics; polytime algorithms

Other structural results

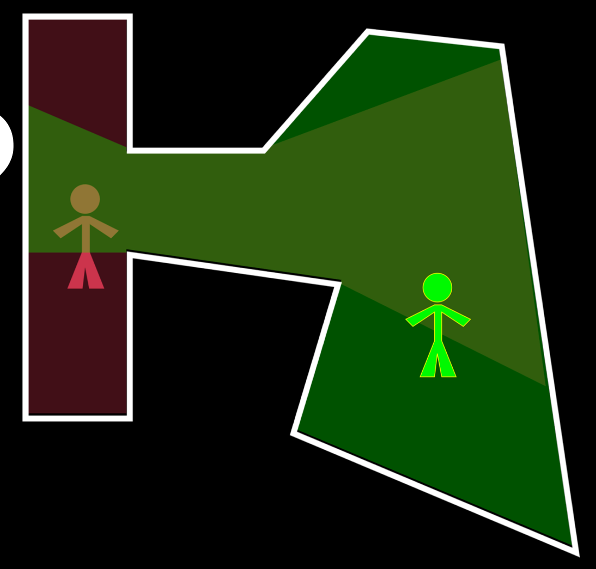
The Art Gallery Problem and Its Variants

The Art Gallery Problem (AGP)—and Its Variants?



We can alter:

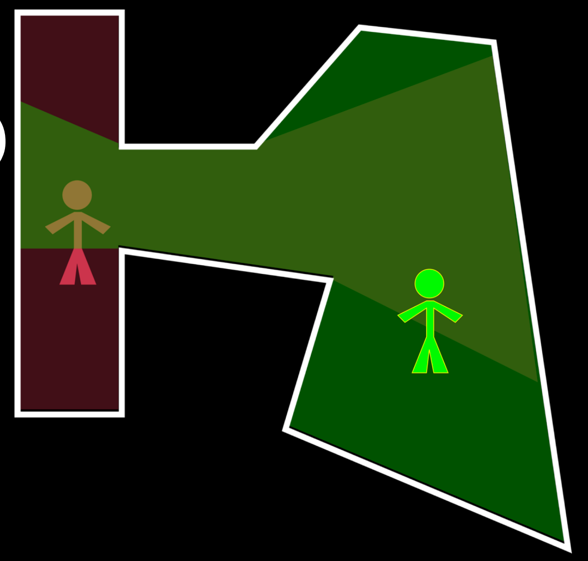
The Art Gallery Problem (AGP)—and Its Variants?



We can alter:

- Capabilities of the guards
- Environment to be guarded

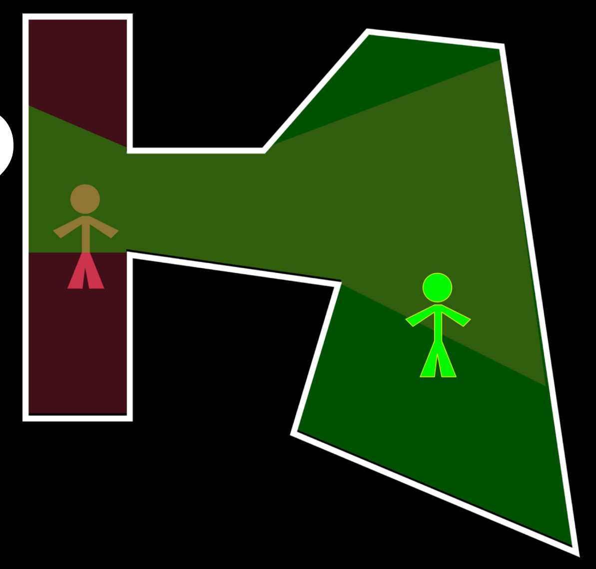
The Art Gallery Problem (AGP)—and Its Variants?



We can alter:

- Capabilities of the guards
- Environment to be guarded

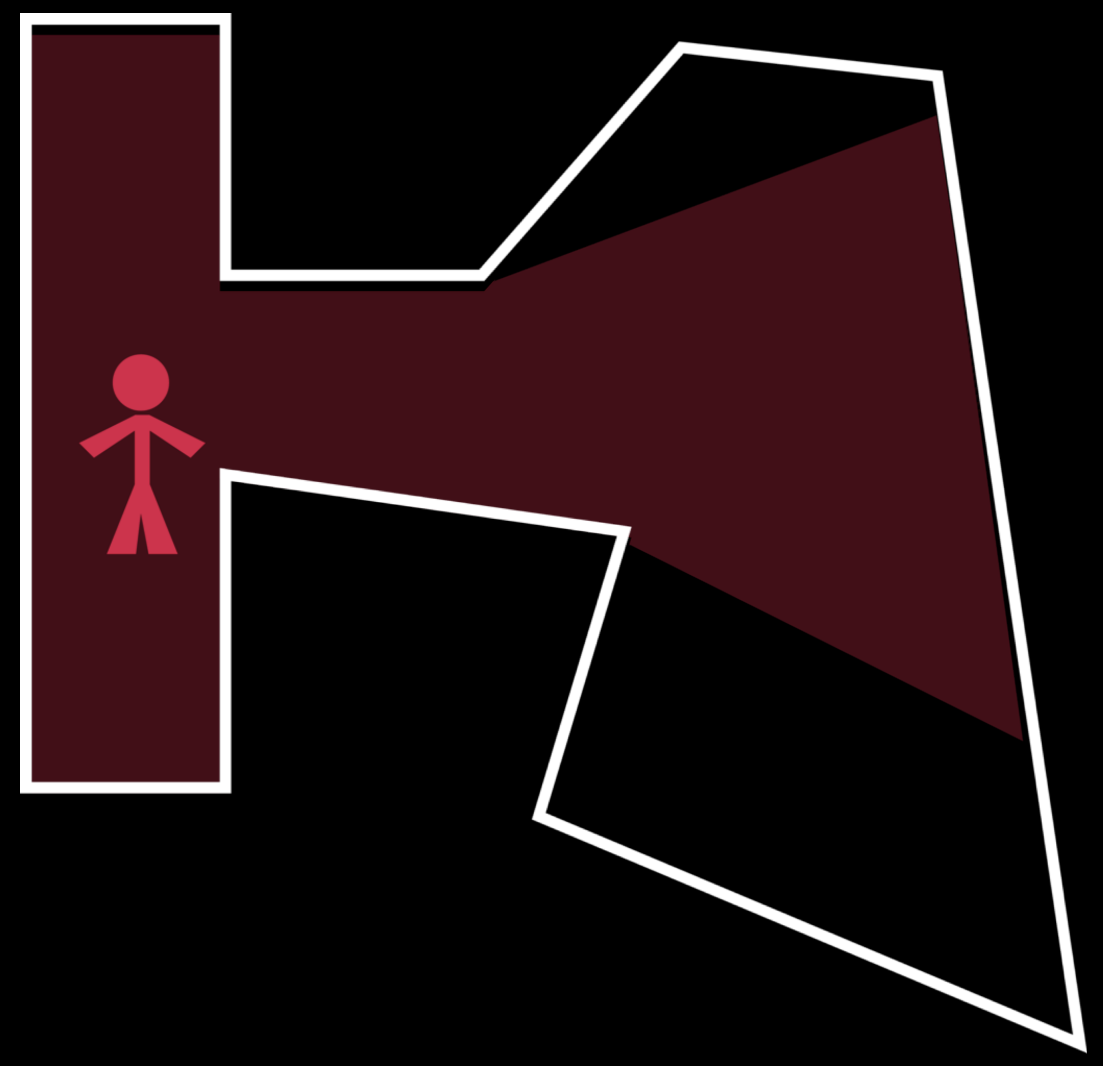
The Art Gallery Problem (AGP)—and Its Variants?



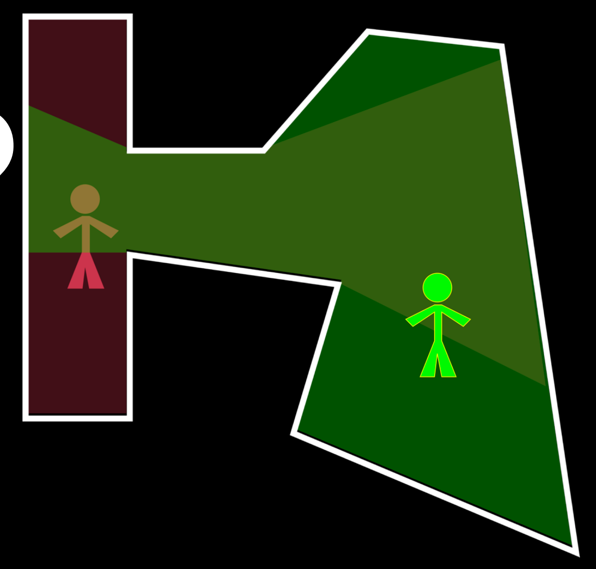
We can alter:

- Capabilities of the guards
- Environment to be guarded

k-transmitter:



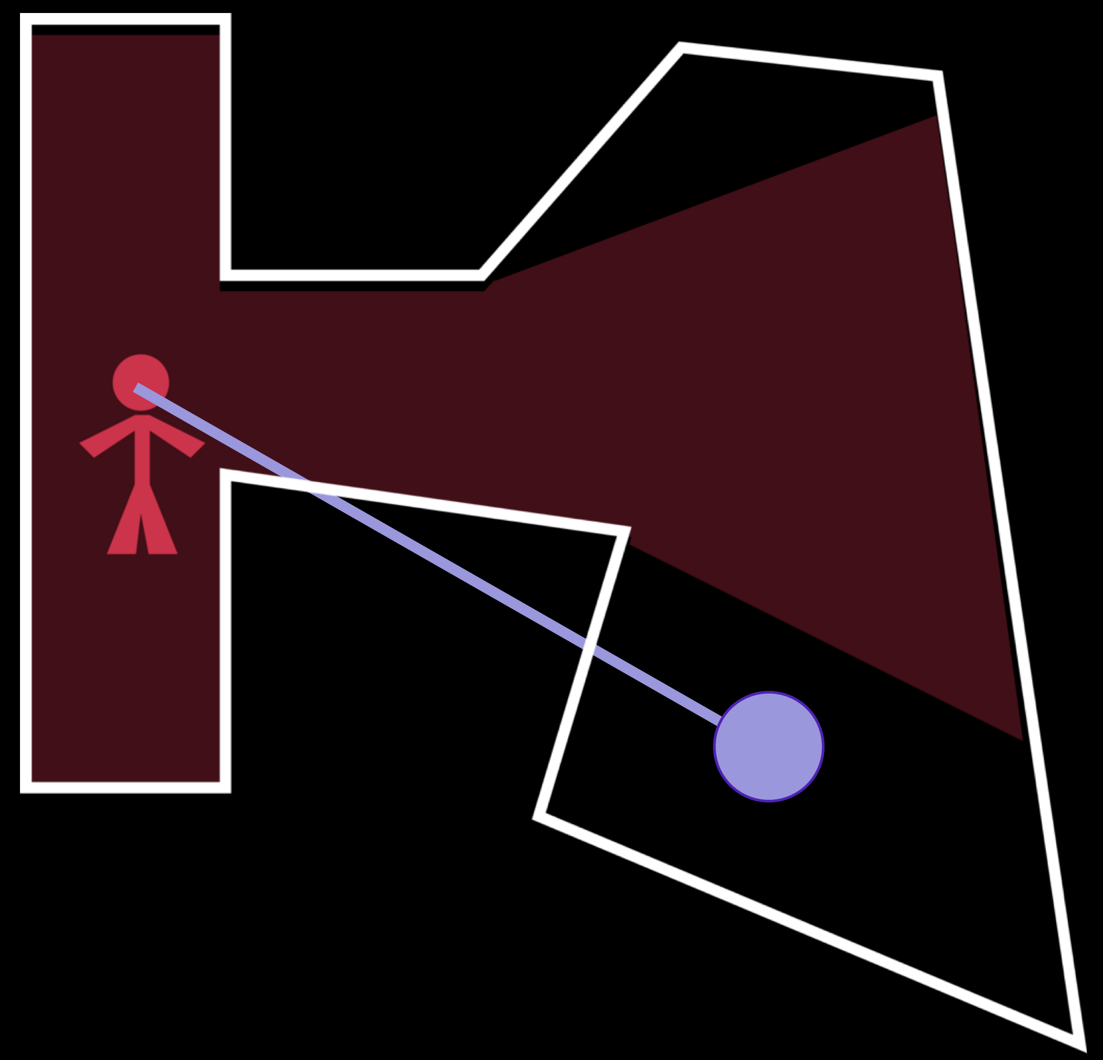
The Art Gallery Problem (AGP)—and Its Variants?



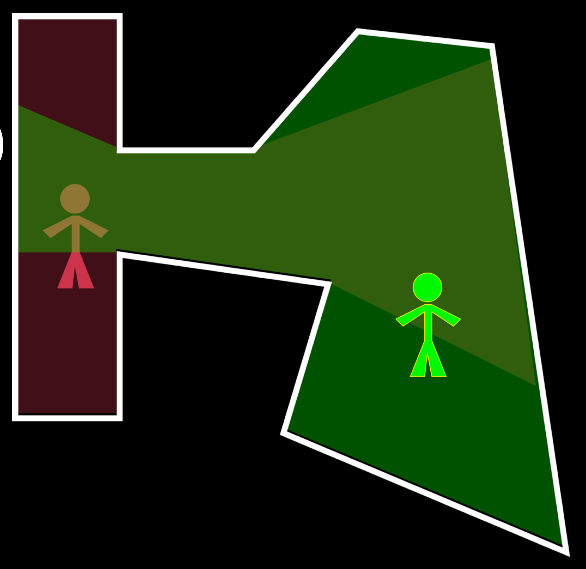
We can alter:

- Capabilities of the guards
- Environment to be guarded

k-transmitter:



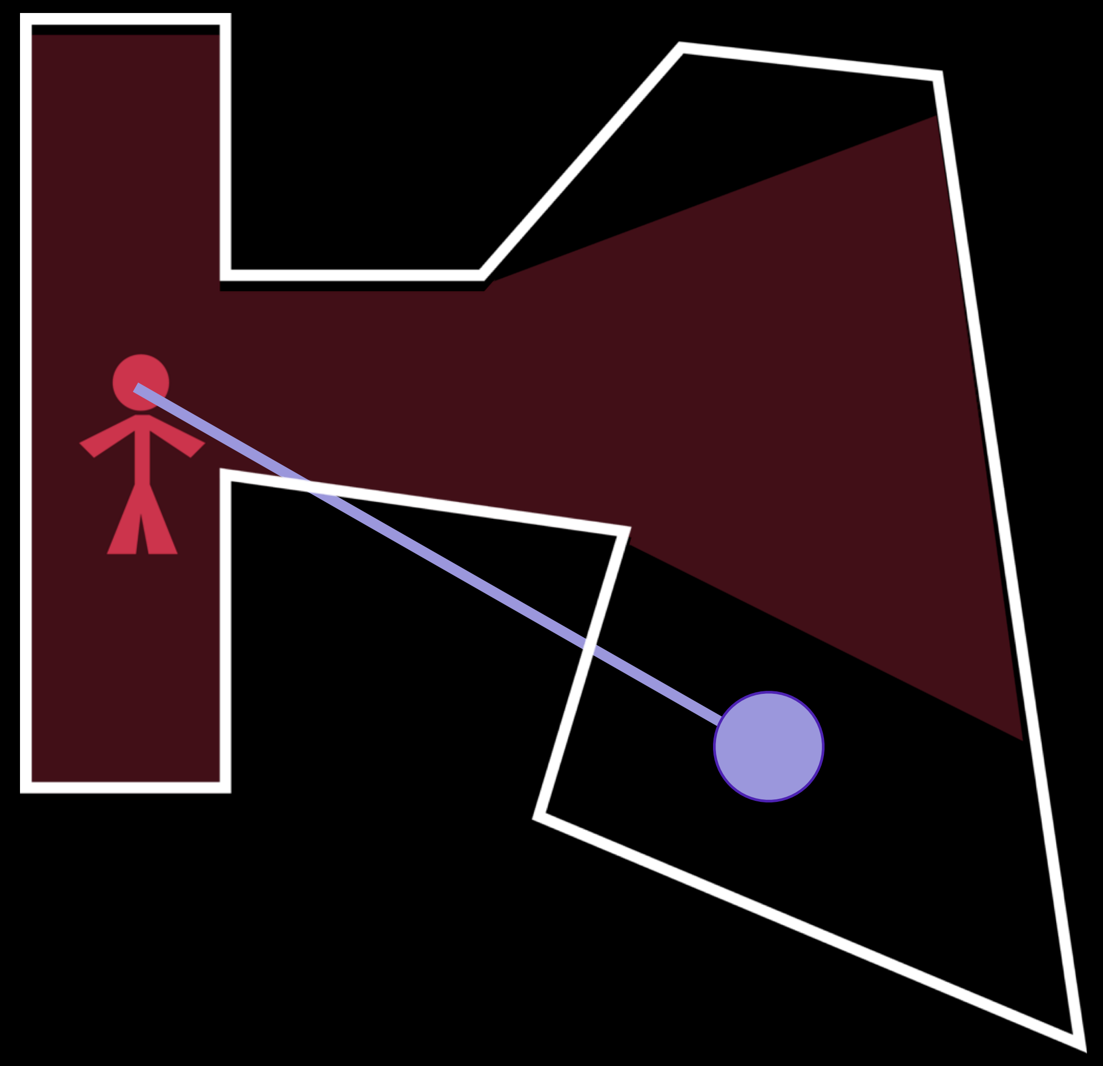
The Art Gallery Problem (AGP)—and Its Variants?



We can alter:

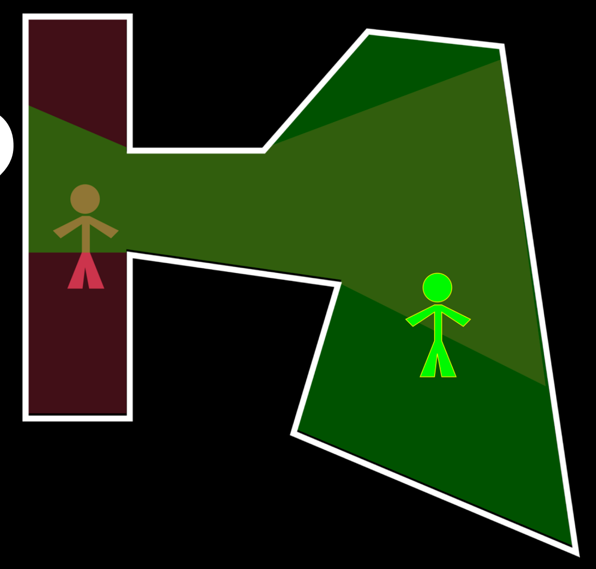
- Capabilities of the guards
- Environment to be guarded

k-transmitter:



Line crosses at most 2 walls
⇒visible from the 2-transmitter

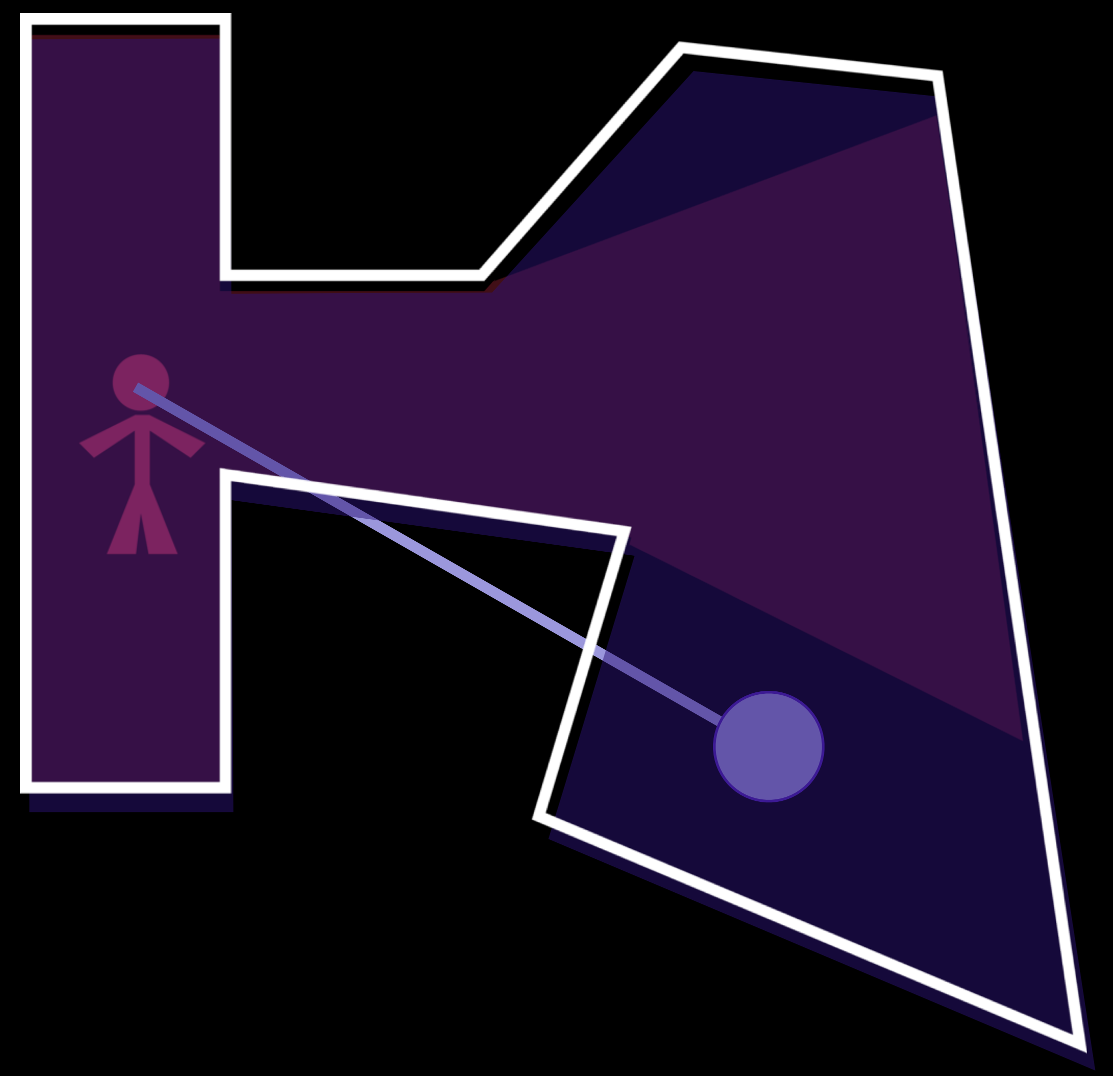
The Art Gallery Problem (AGP)—and Its Variants?



We can alter:

- Capabilities of the guards

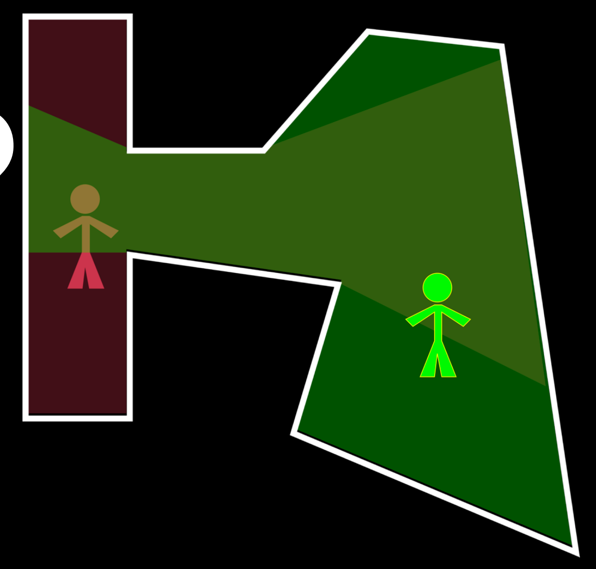
k-transmitter:



Line crosses at most 2 walls
⇒visible from the 2-transmitter

- Environment to be guarded

The Art Gallery Problem (AGP)—and Its Variants?

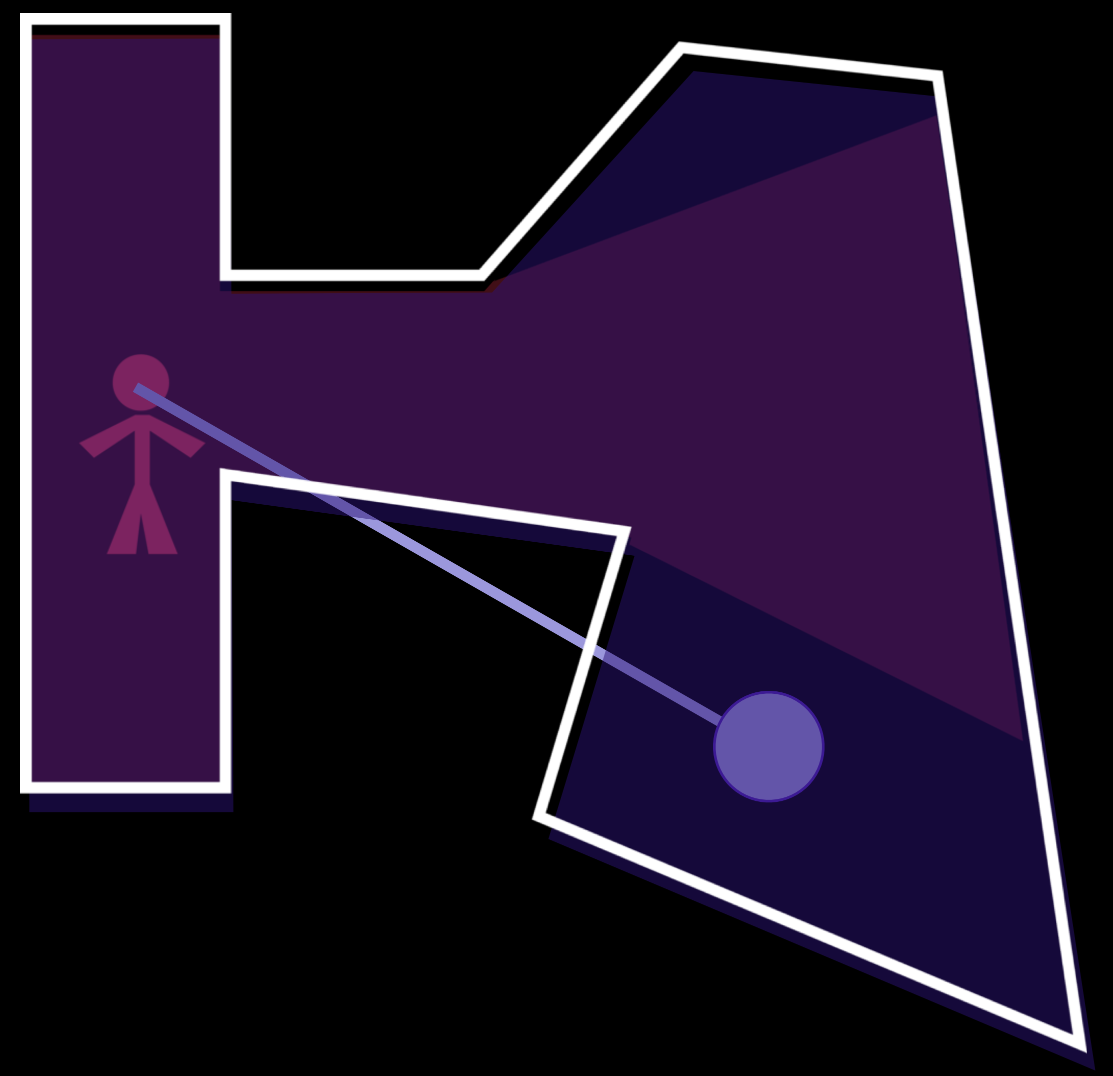


We can alter:

- Capabilities of the guards
- Environment to be guarded

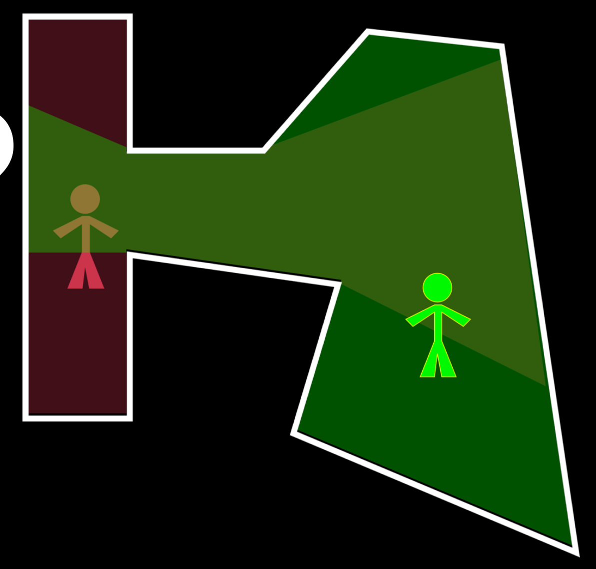
k-transmitter:

Fading:



Line crosses at most 2 walls
⇒visible from the 2-transmitter

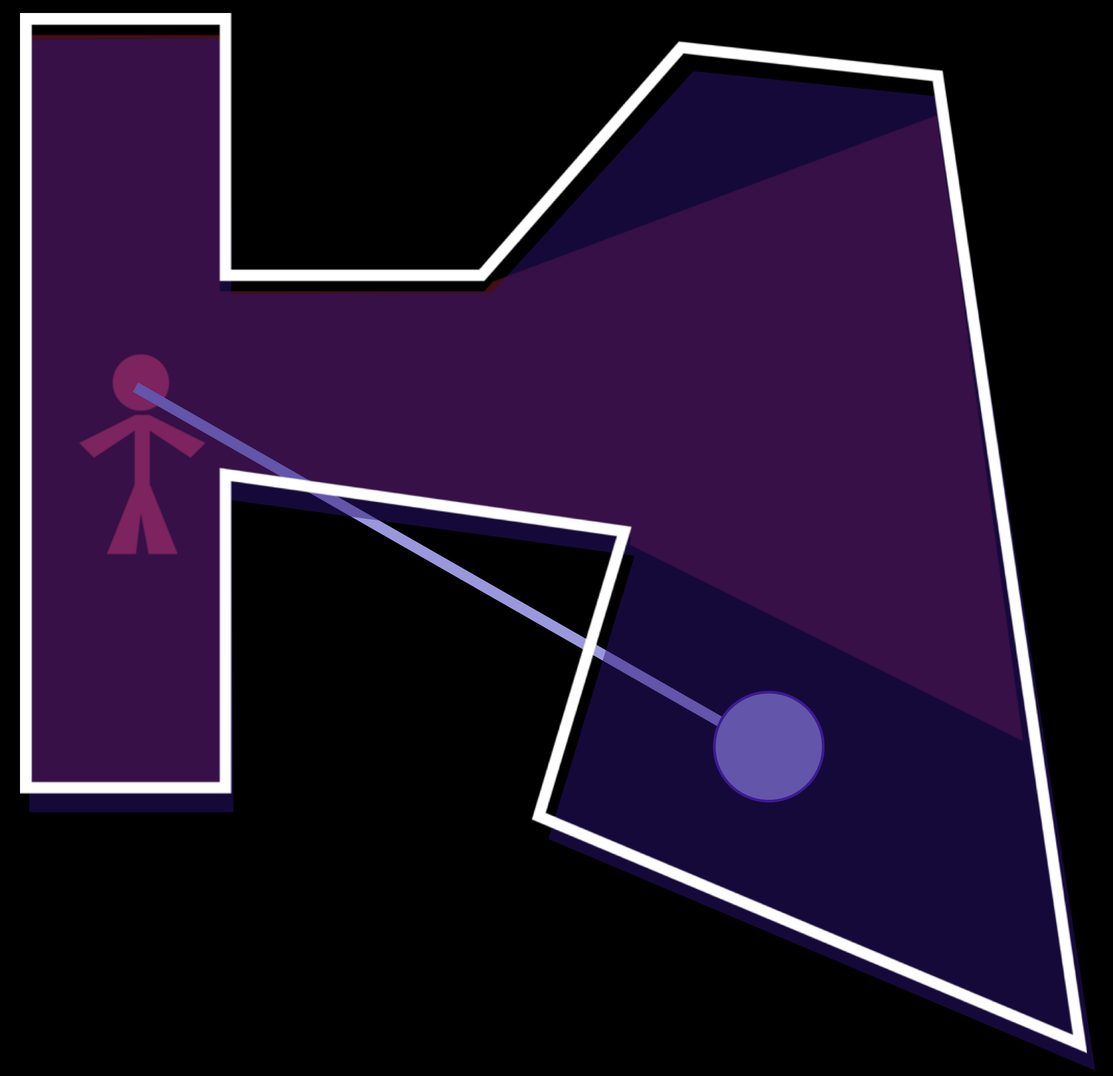
The Art Gallery Problem (AGP)—and Its Variants?



We can alter:

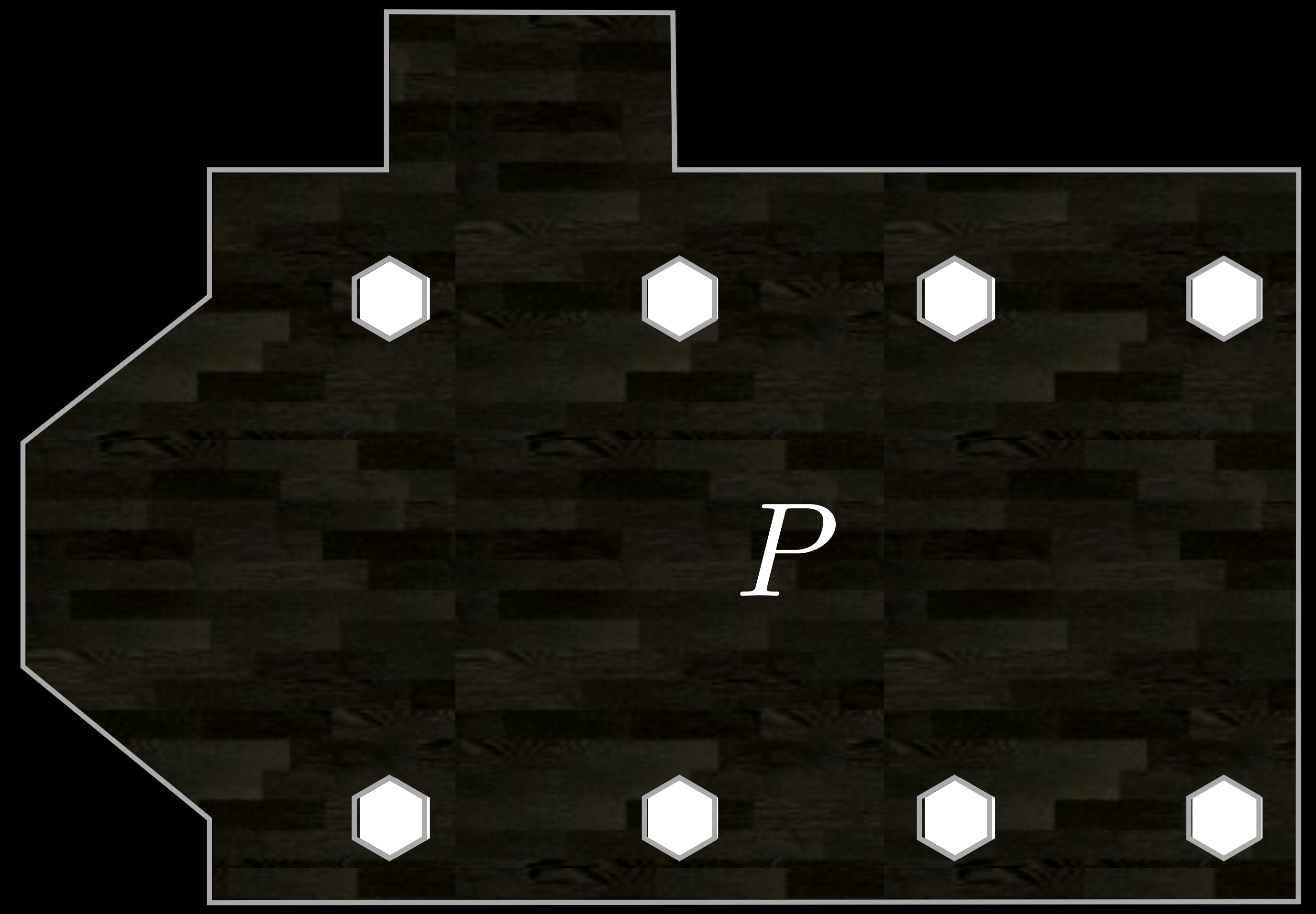
- Capabilities of the guards
- Environment to be guarded

k-transmitter:

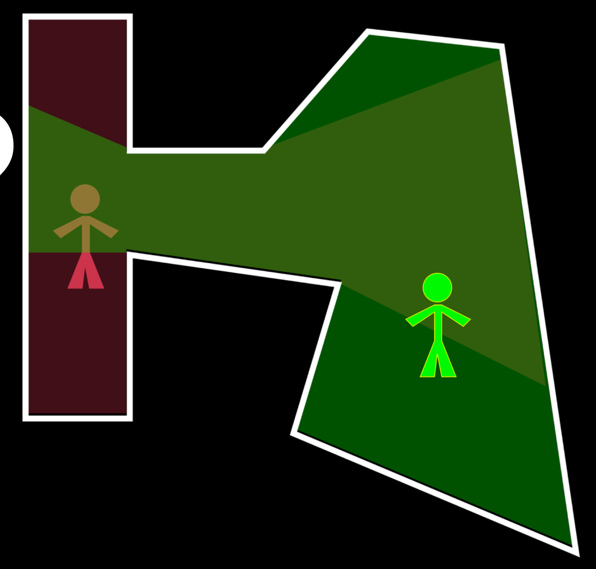


Line crosses at most 2 walls
⇒visible from the 2-transmitter

Fading:



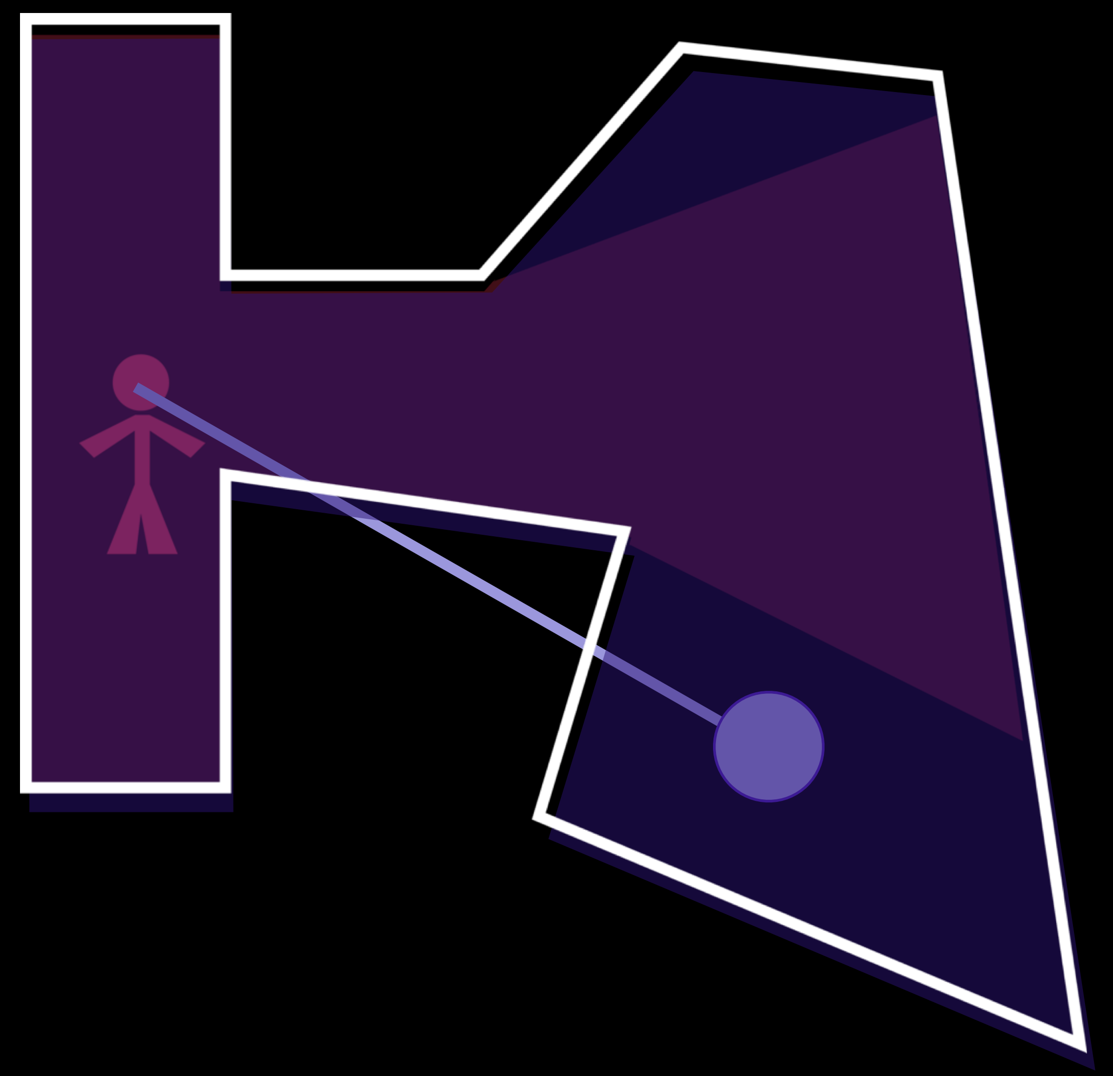
The Art Gallery Problem (AGP)—and Its Variants?



We can alter:

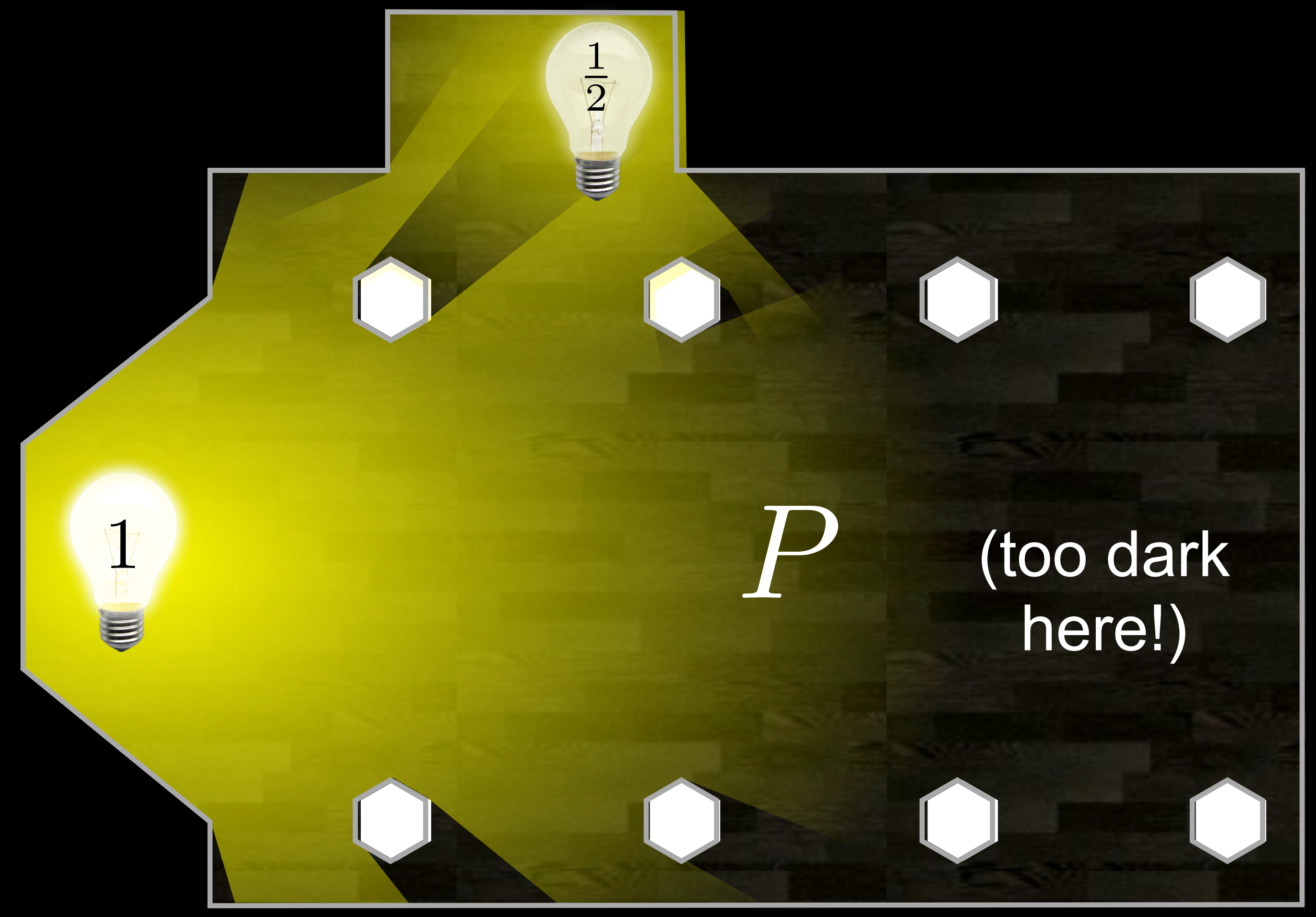
- Capabilities of the guards
- Environment to be guarded

k-transmitter:



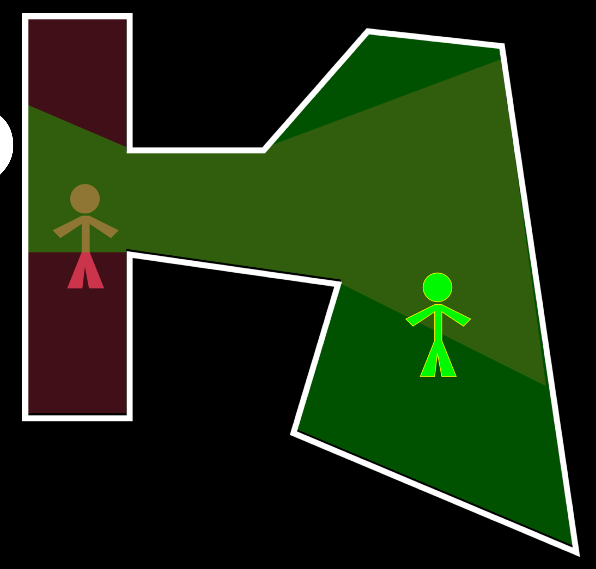
Line crosses at most 2 walls
⇒visible from the 2-transmitter

Fading:



Place lights,
assign energy (= brightness).

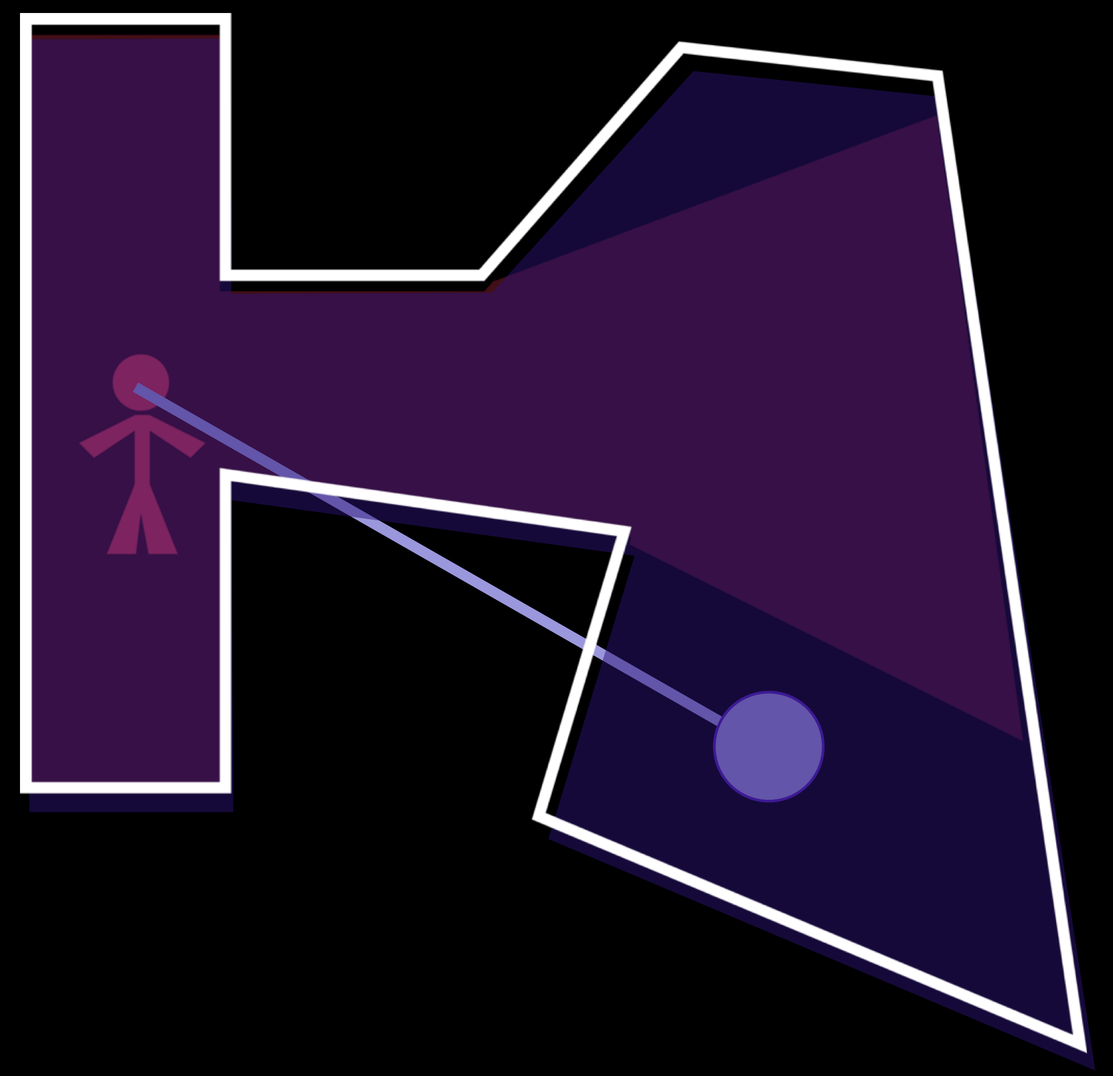
The Art Gallery Problem (AGP)—and Its Variants?



We can alter:

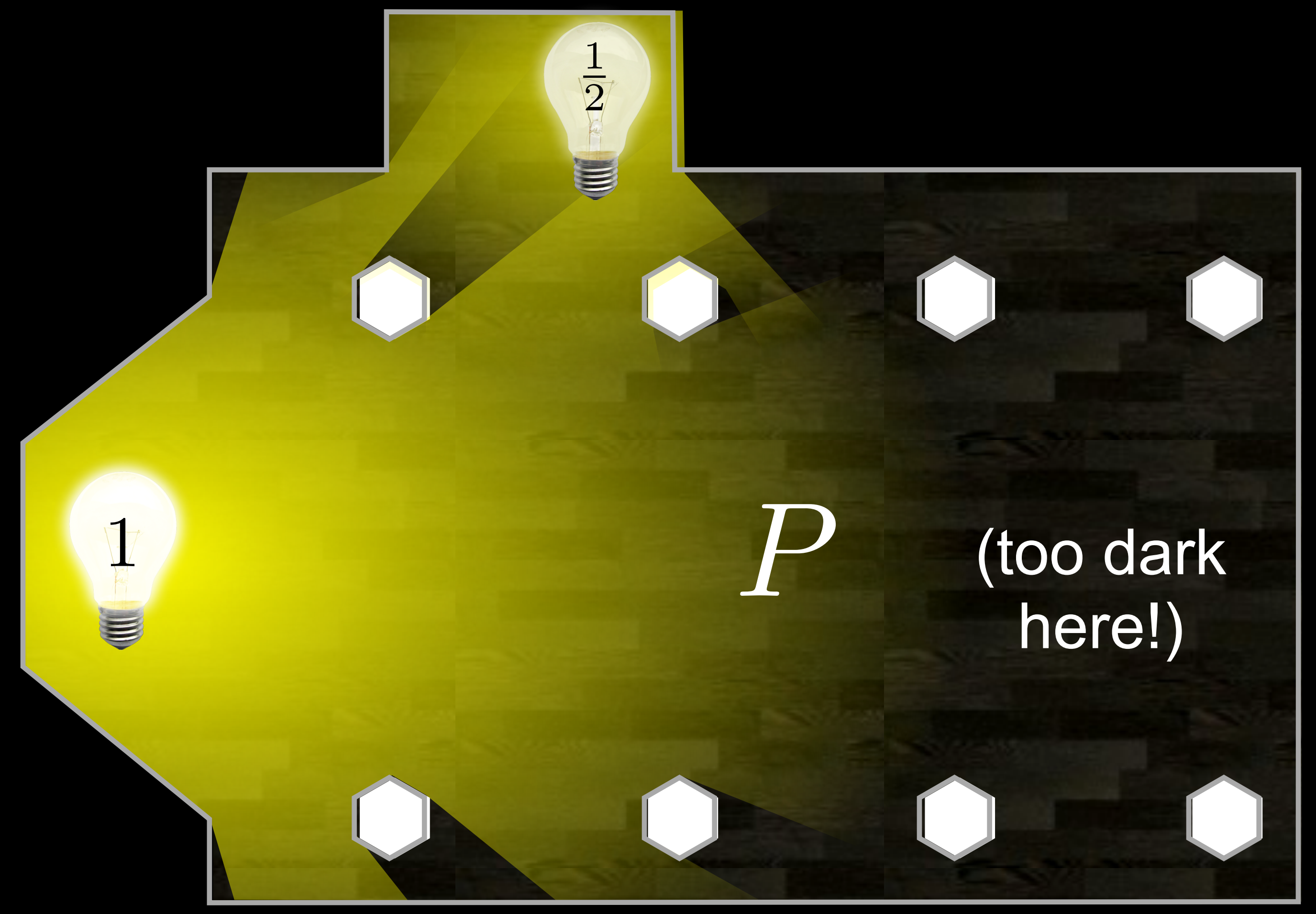
- Capabilities of the guards
- Environment to be guarded

k-transmitter:



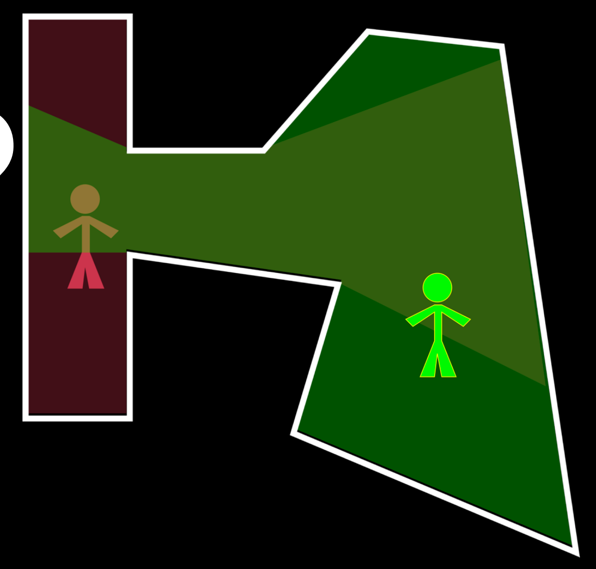
Line crosses at most 2 walls
⇒visible from the 2-transmitter

Fading:



Place lights,
assign energy (= brightness).
"Sufficiently" (normalize to 1) light everything — with fading!

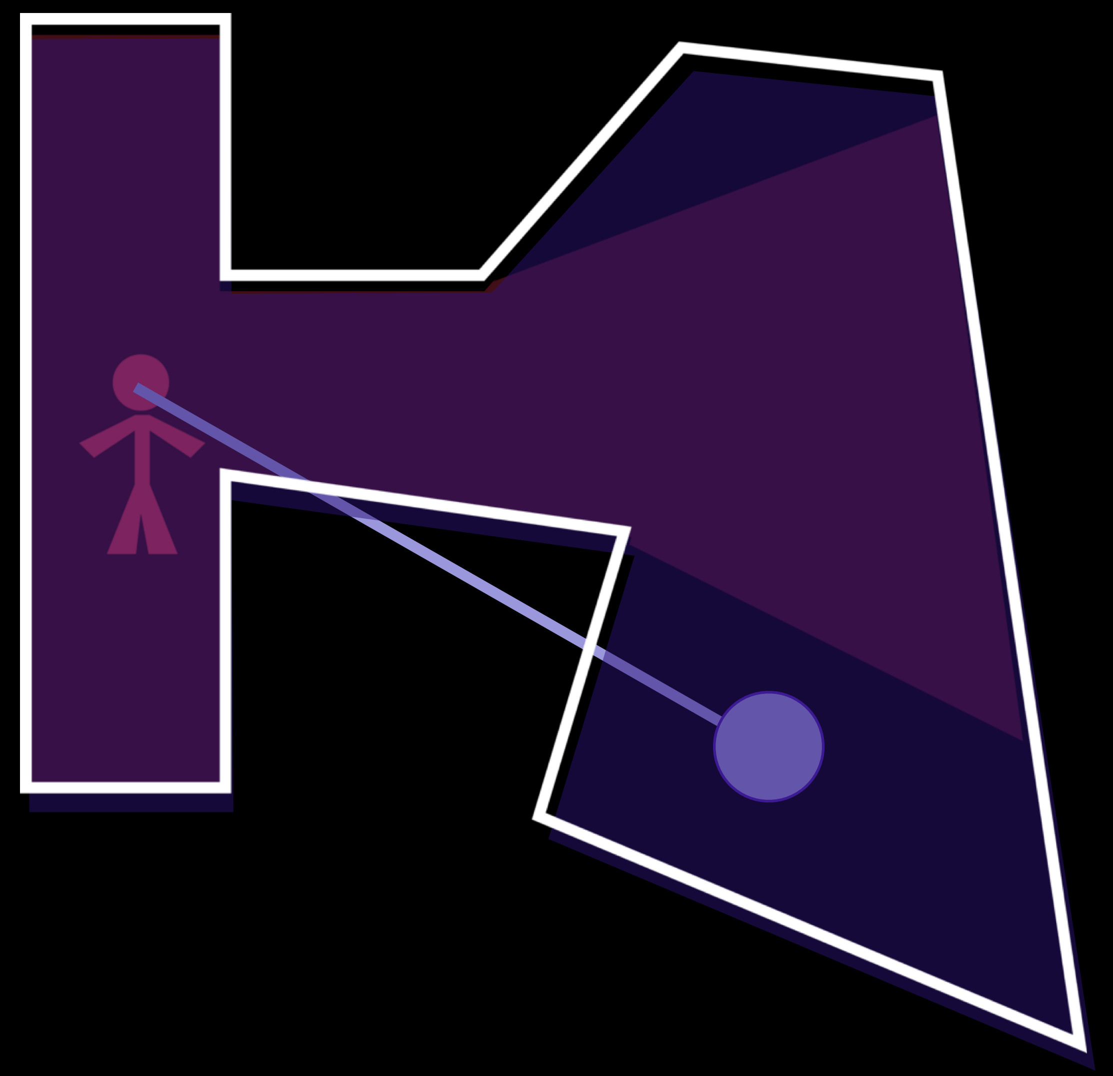
The Art Gallery Problem (AGP)—and Its Variants?



We can alter:

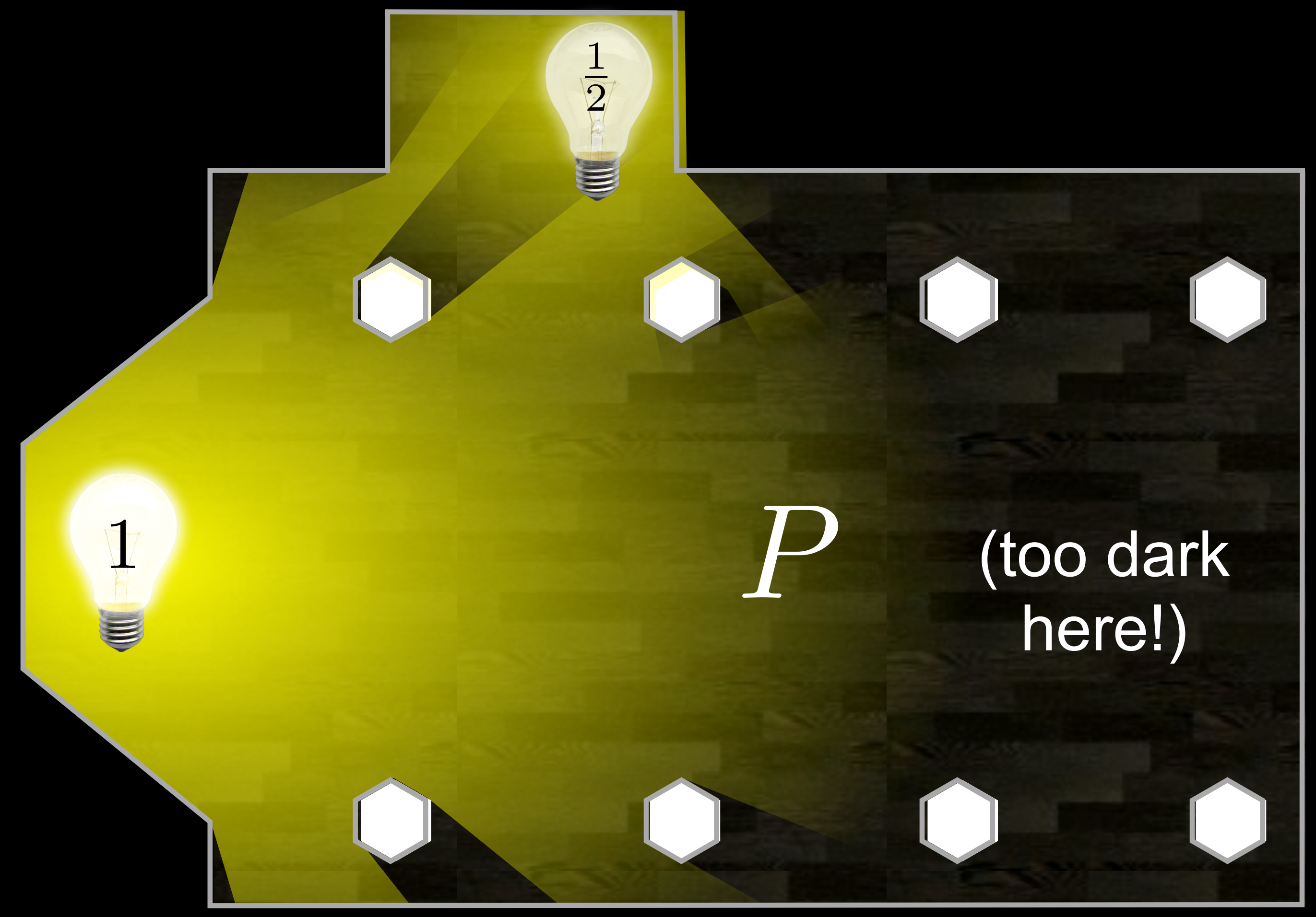
- Capabilities of the guards
- Environment to be guarded

k-transmitter:



Line crosses at most 2 walls
⇒visible from the 2-transmitter

Fading:

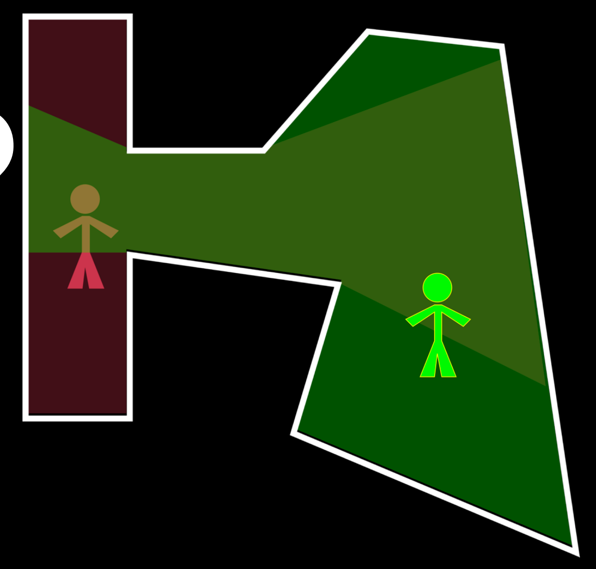


Place lights,
assign energy (= brightness).

“Sufficiently” (normalize to 1)
light everything — with fading!

Minimize total energy.

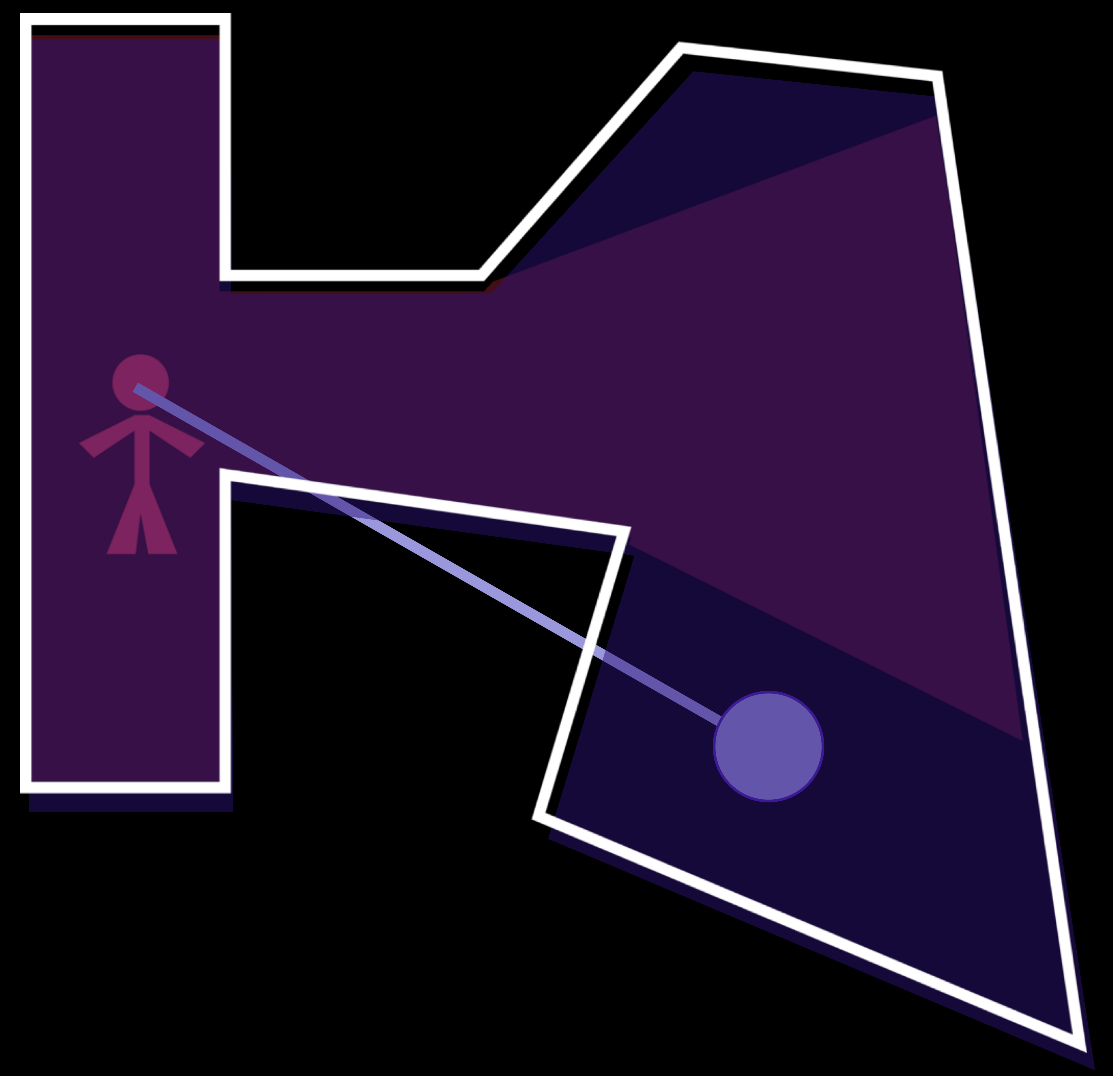
The Art Gallery Problem (AGP)—and Its Variants?



We can alter:

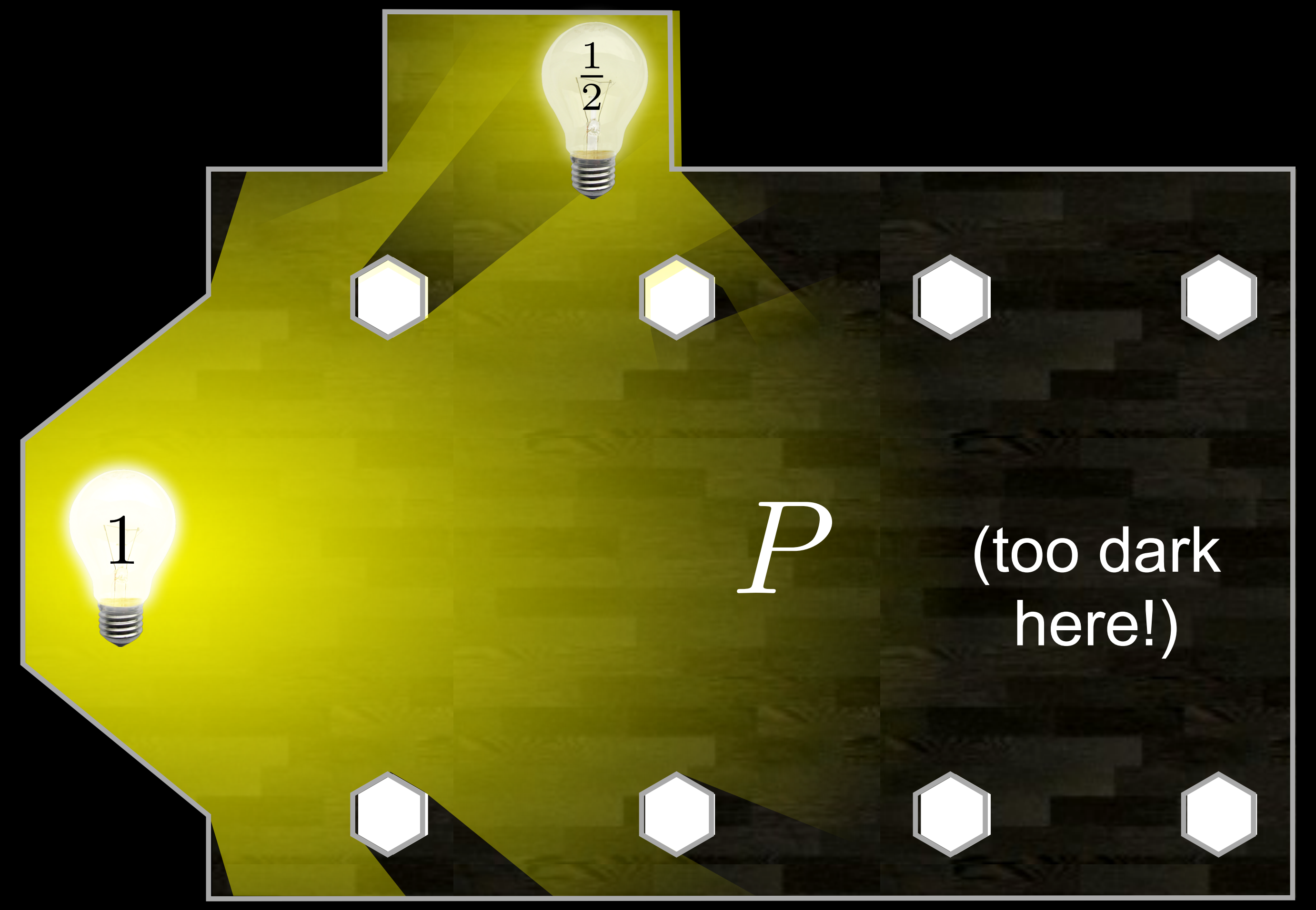
- Capabilities of the guards
- Environment to be guarded

k-transmitter:



Line crosses at most 2 walls
⇒visible from the 2-transmitter

Fading:

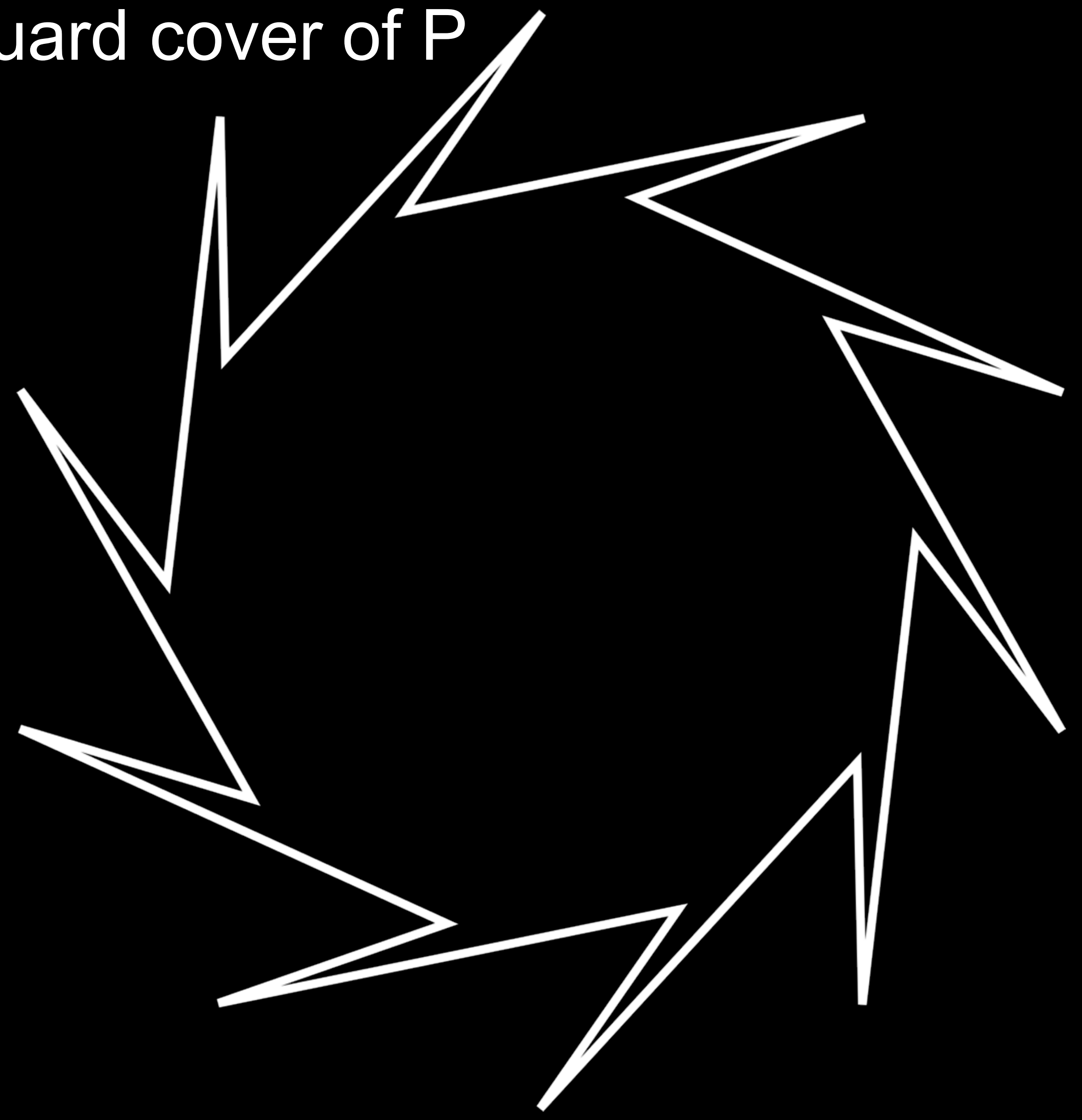


Place lights,
assign energy (= brightness).
“Sufficiently” (normalize to 1) light everything — with fading!
Minimize total energy.

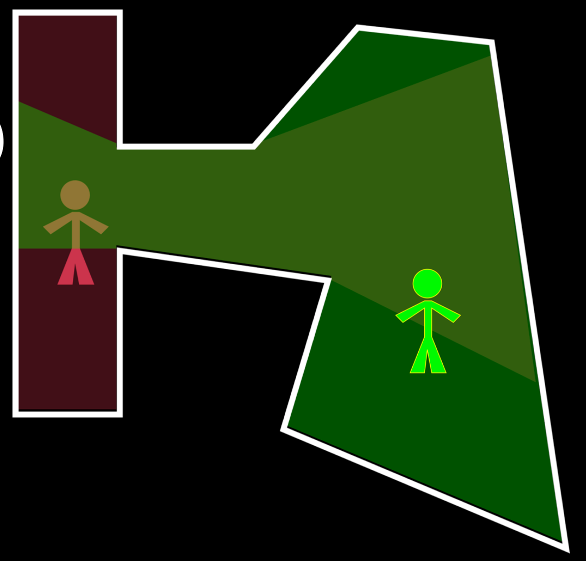
Chromatic AGP:

Given: a polygon P

Task: find a min guard cover of P



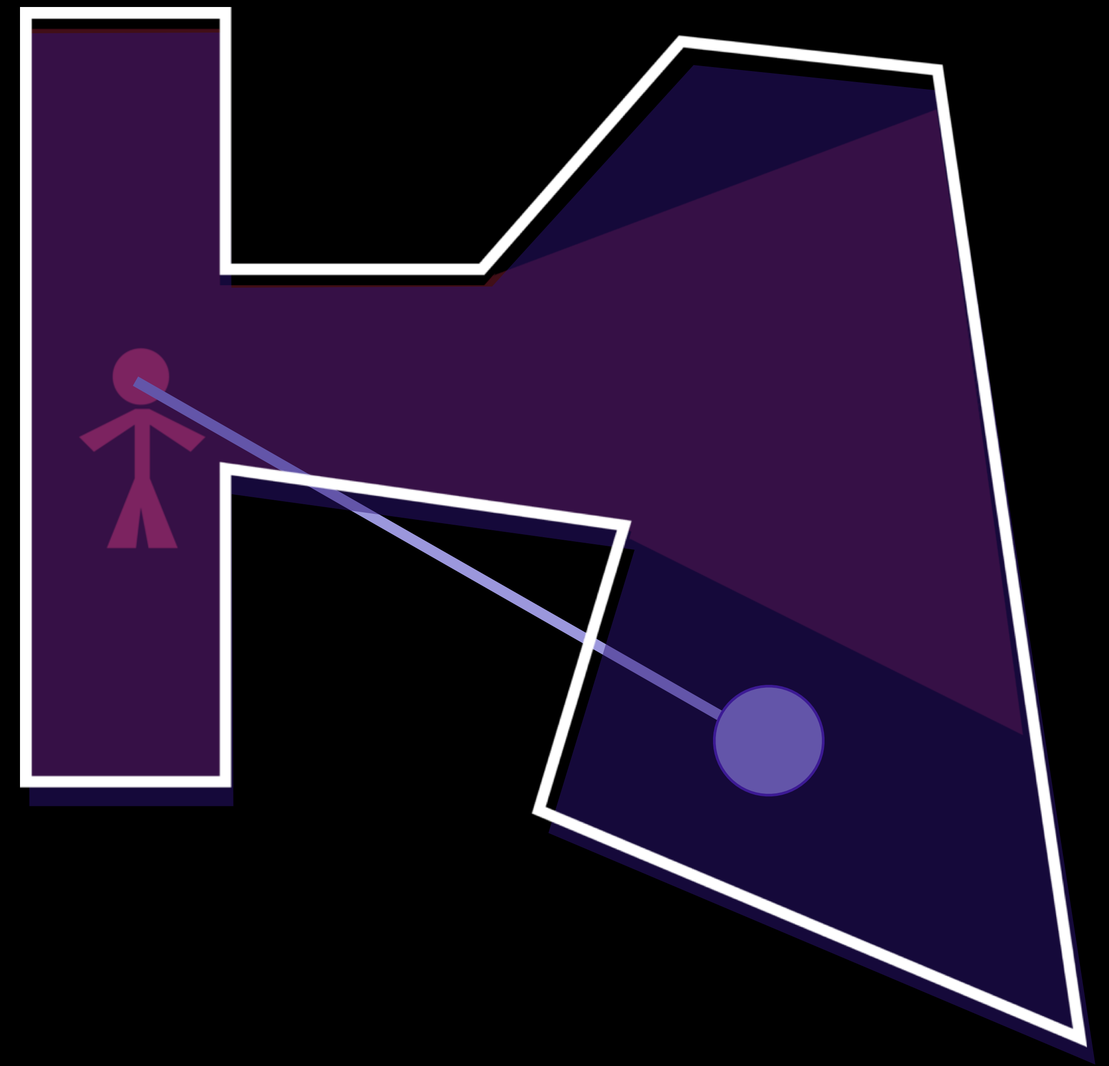
The Art Gallery Problem (AGP)—and Its Variants?



We can alter:

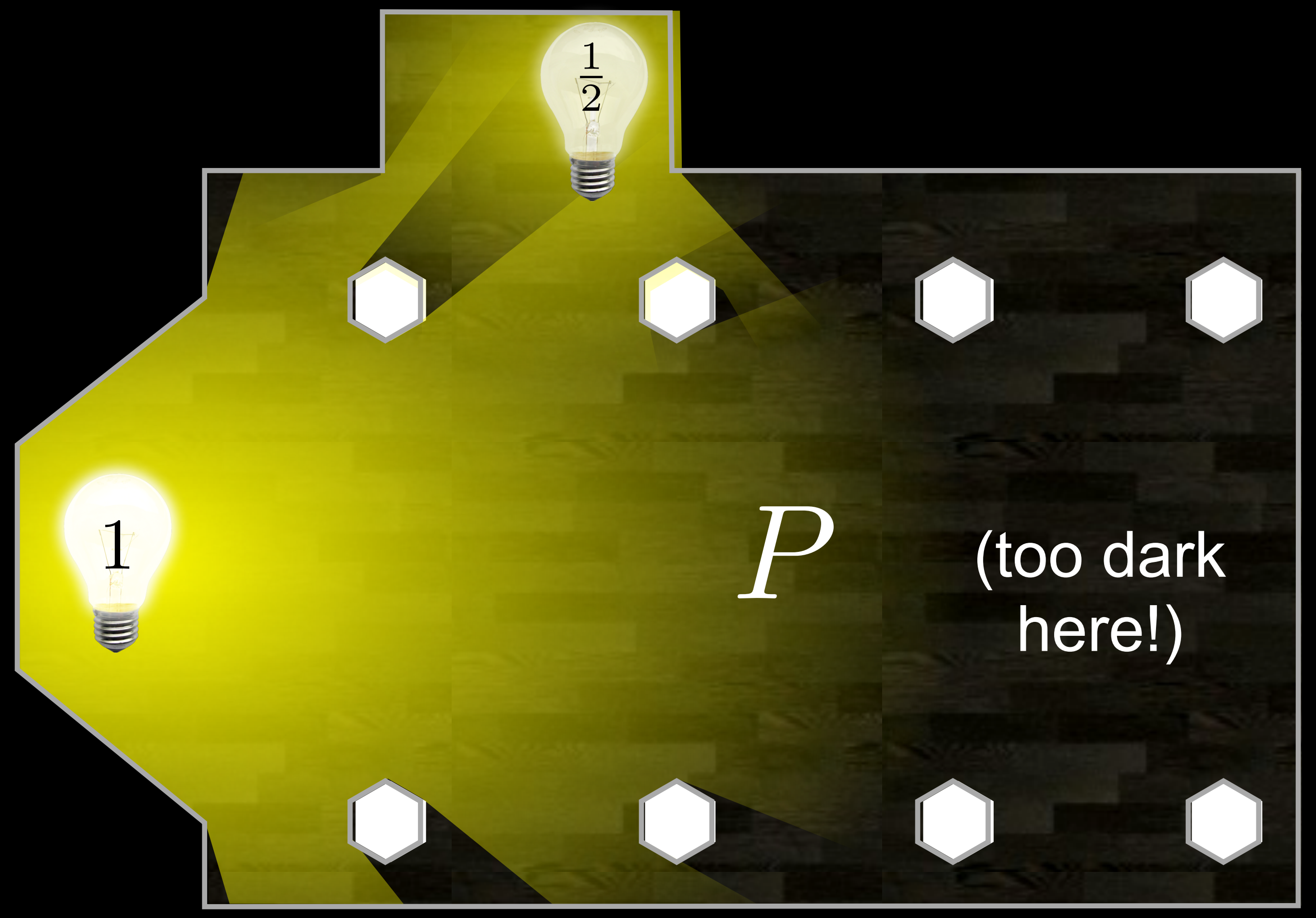
- Capabilities of the guards
- Environment to be guarded

k-transmitter:



Line crosses at most 2 walls
⇒visible from the 2-transmitter

Fading:

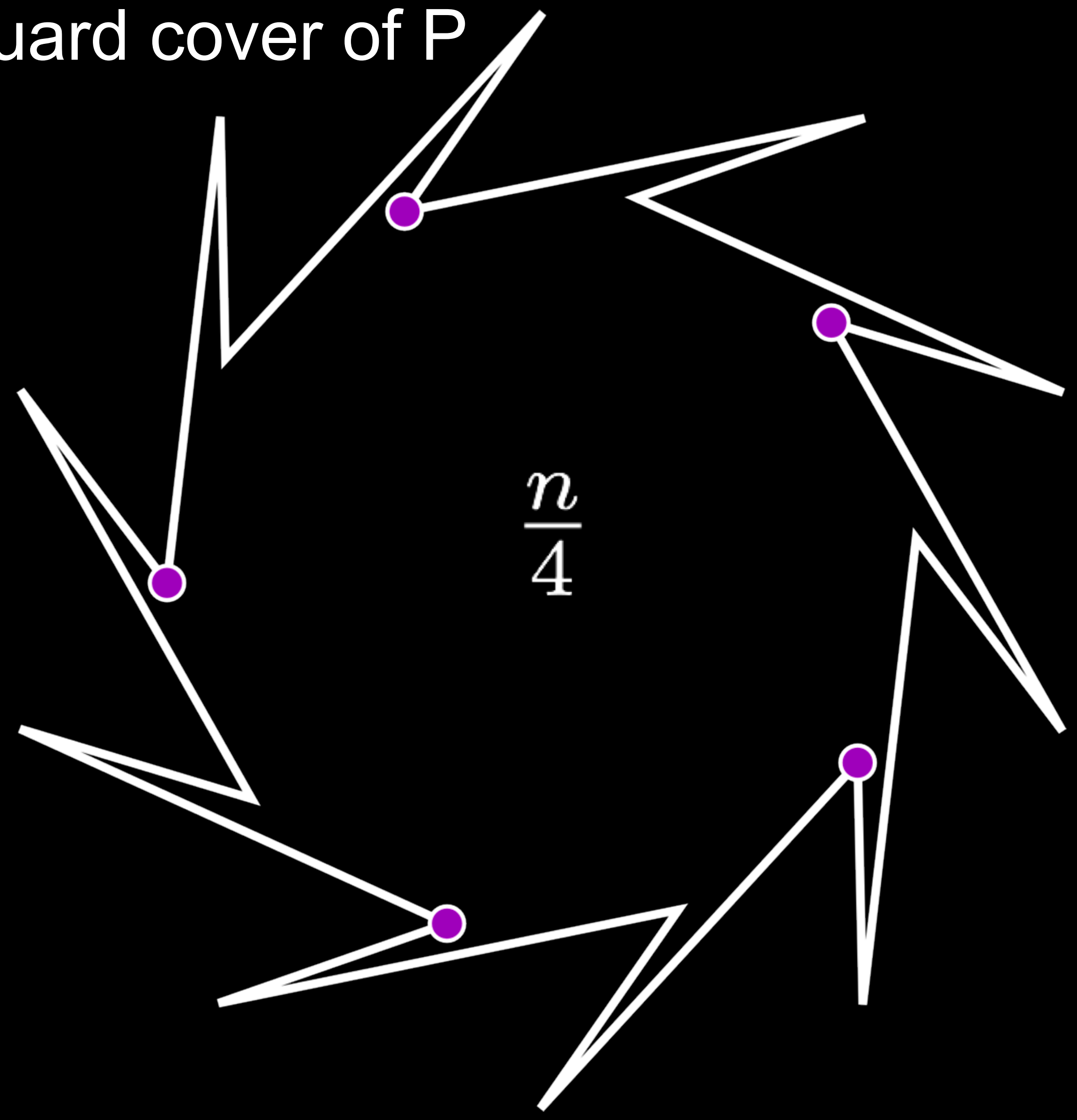


Place lights,
assign energy (= brightness).
"Sufficiently" (normalize to 1)
light everything — with fading!
Minimize total energy.

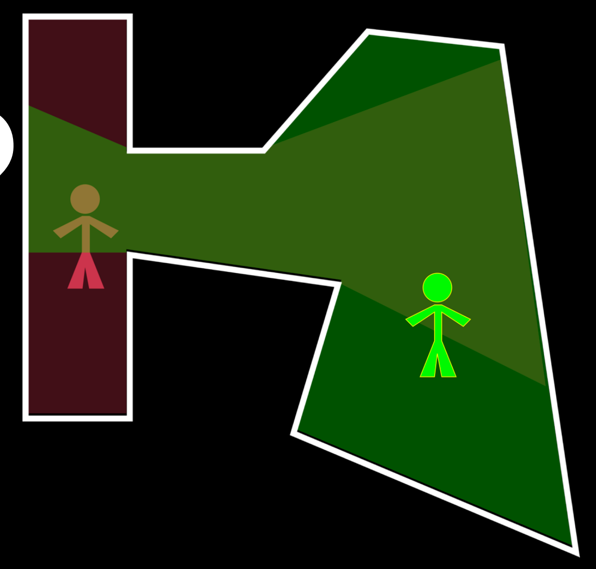
Chromatic AGP:

Given: a polygon P

Task: find a min guard cover of P



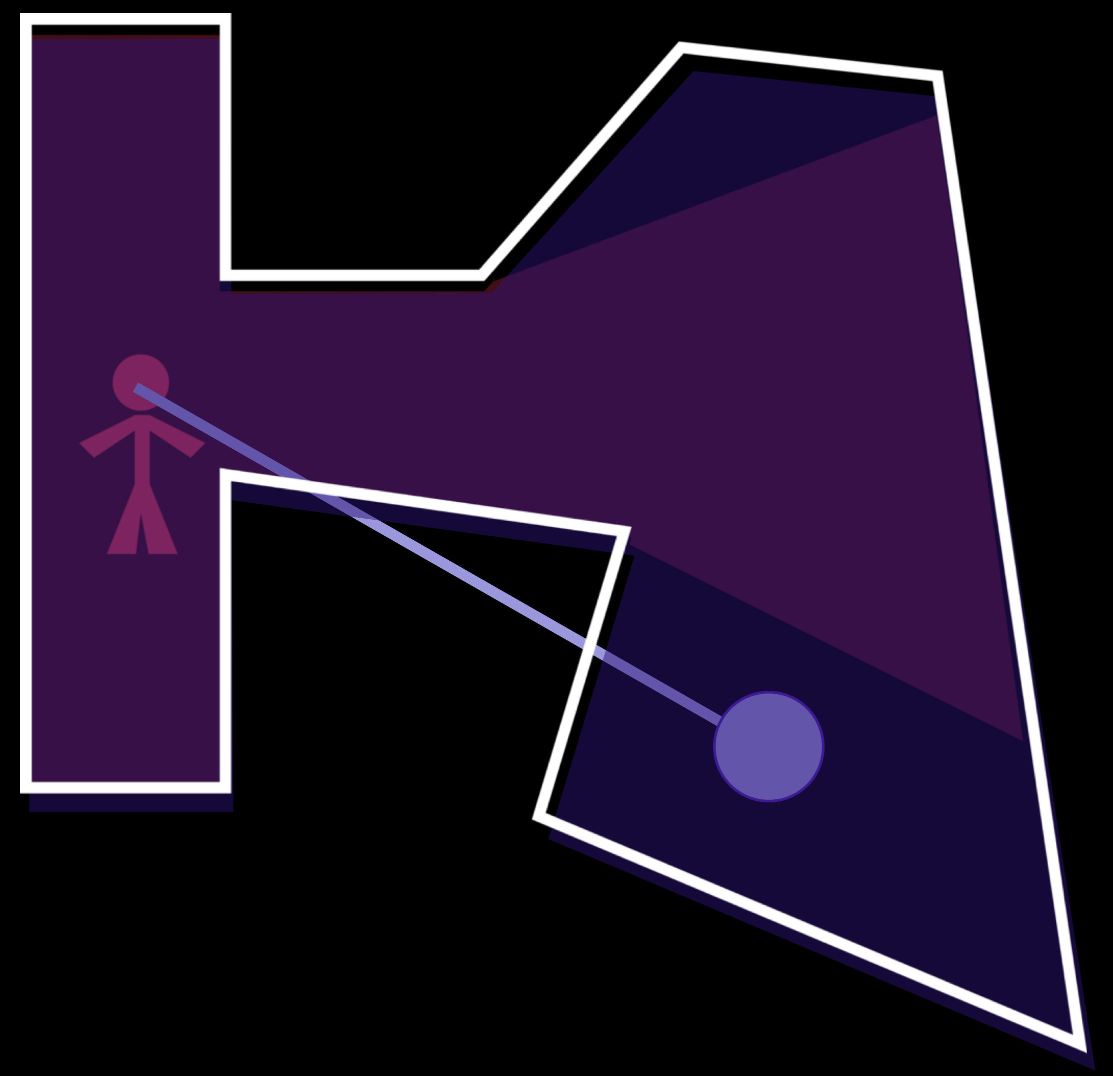
The Art Gallery Problem (AGP)—and Its Variants?



We can alter:

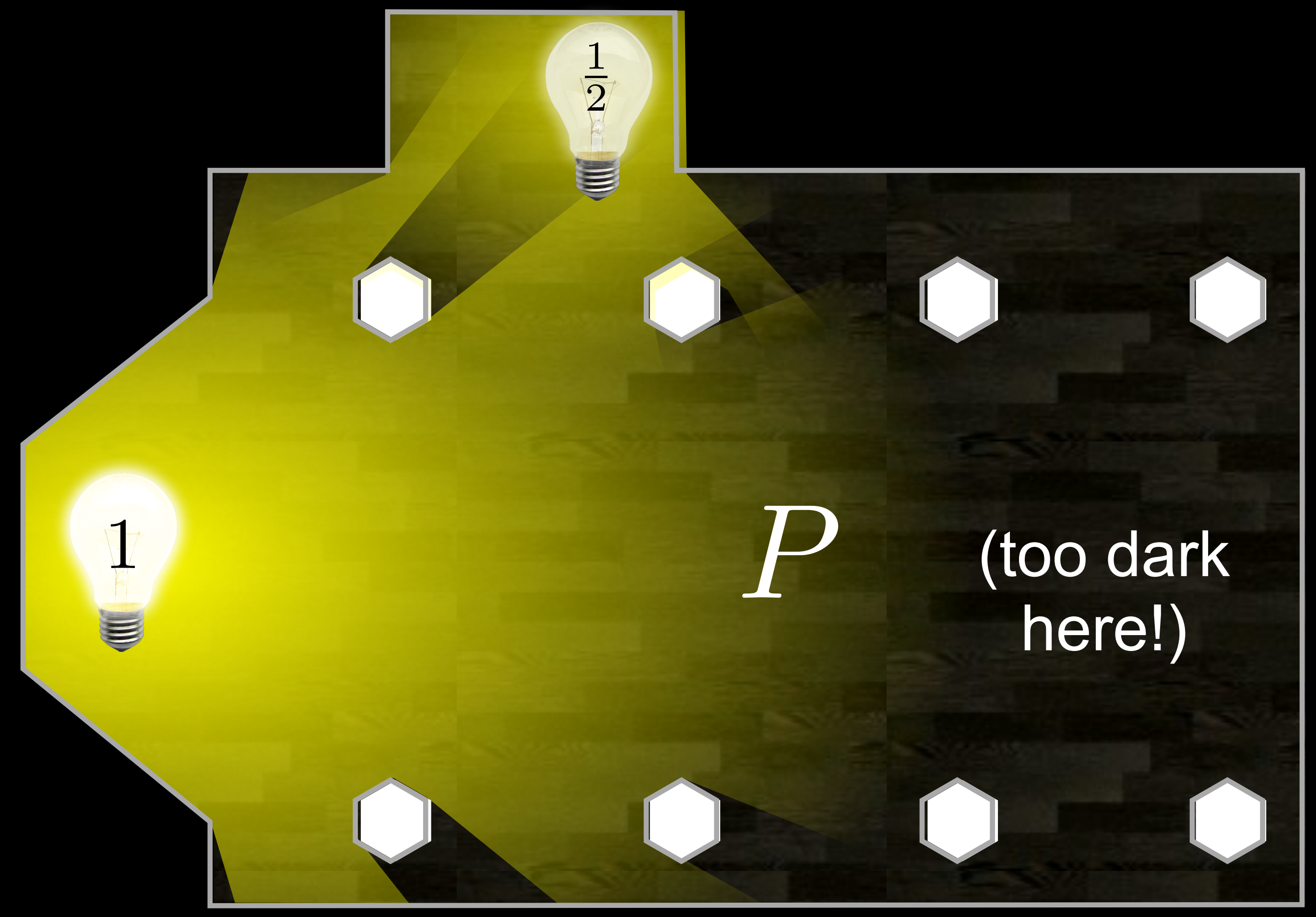
- Capabilities of the guards
- Environment to be guarded

k-transmitter:



Line crosses at most 2 walls
⇒visible from the 2-transmitter

Fading:

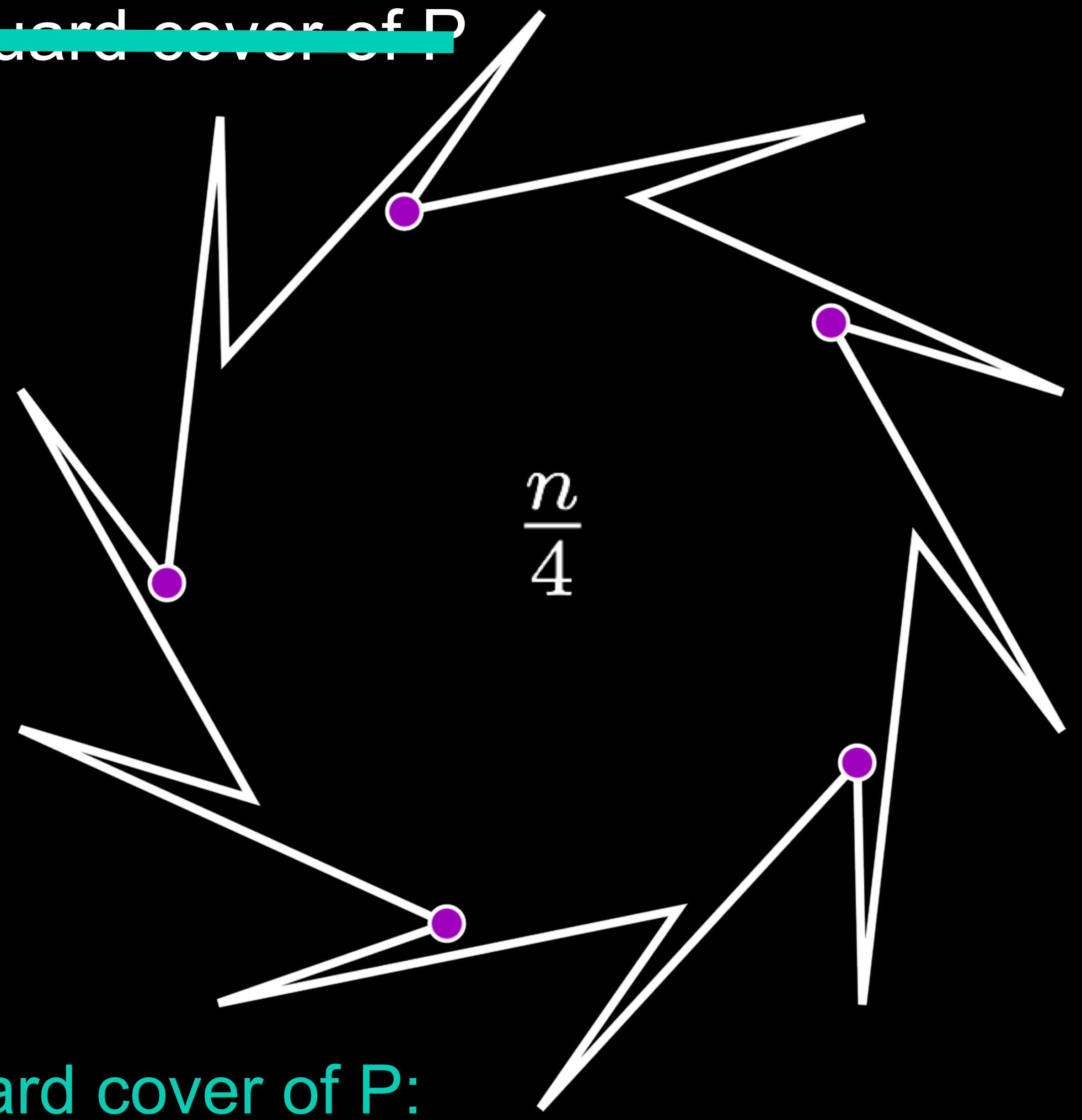


Place lights,
assign energy (= brightness).
"Sufficiently" (normalize to 1) light everything — with fading!
Minimize total energy.

Chromatic AGP:

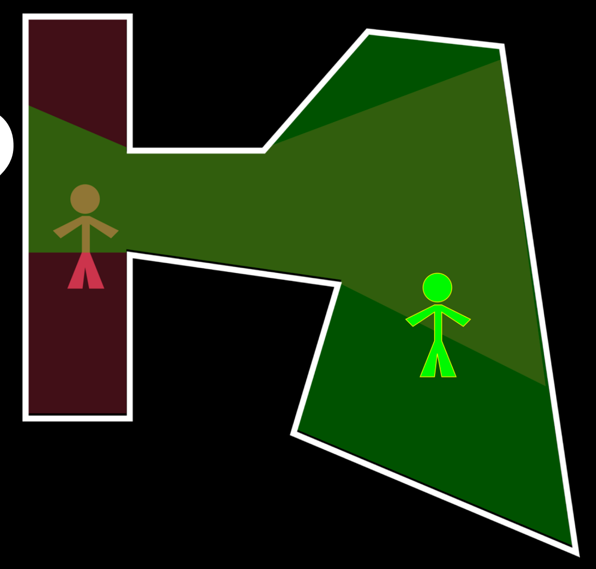
Given: a polygon P

Task: ~~find a min guard cover of P~~



Find a **colored** guard cover of P:
No point in P is seen by two guards of the same color.

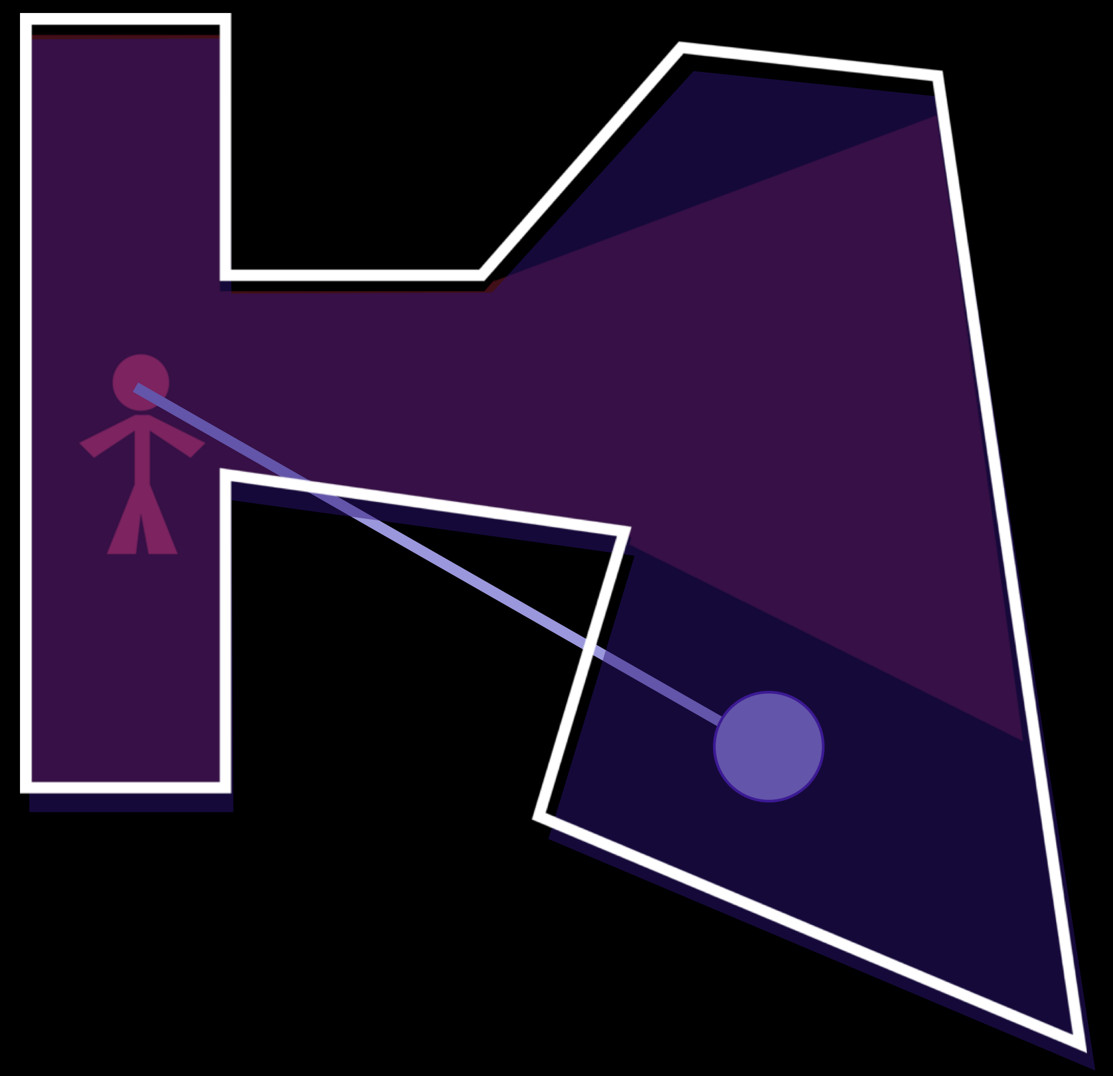
The Art Gallery Problem (AGP)—and Its Variants?



We can alter:

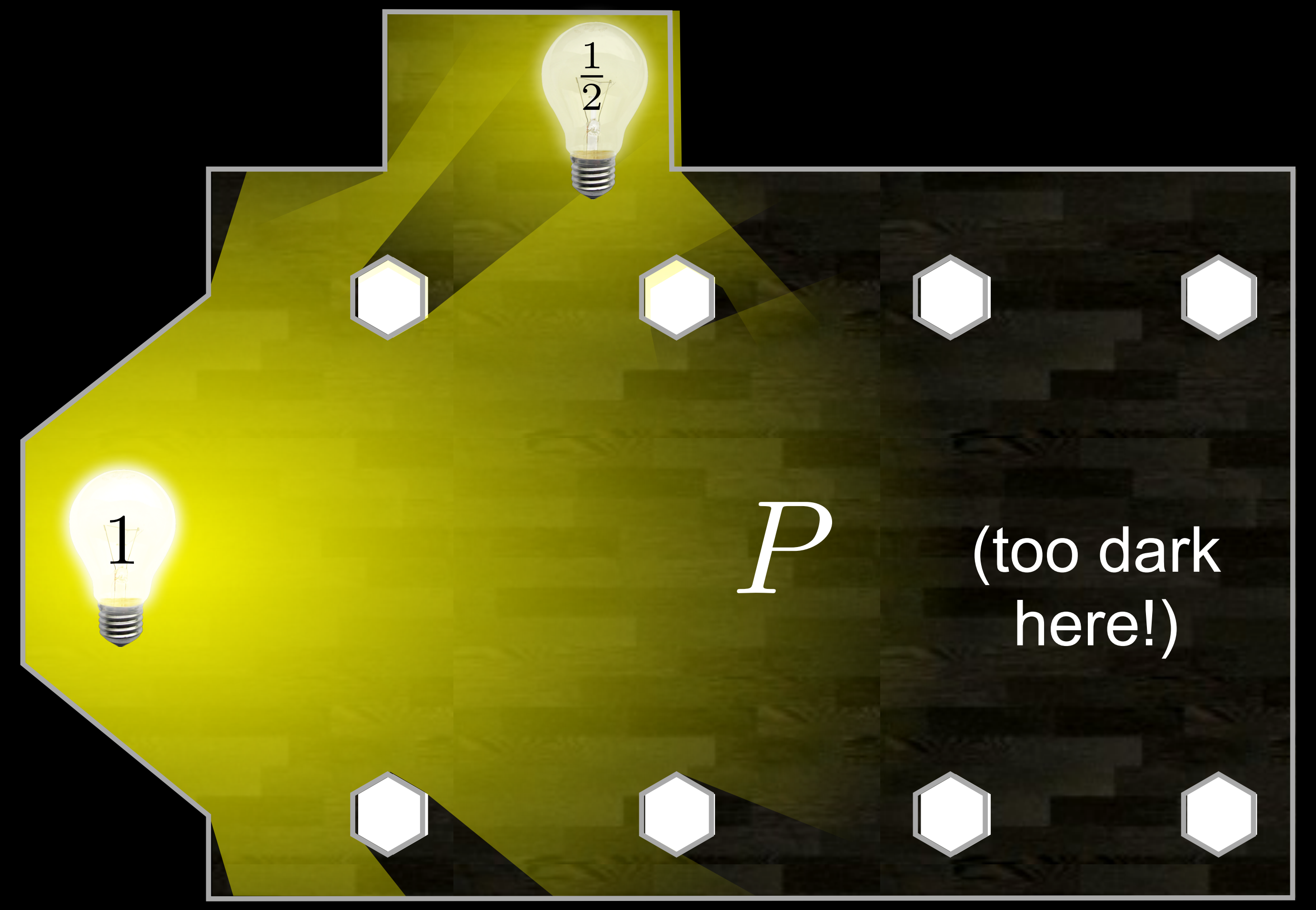
- Capabilities of the guards
- Environment to be guarded

k-transmitter:



Line crosses at most 2 walls
⇒visible from the 2-transmitter

Fading:

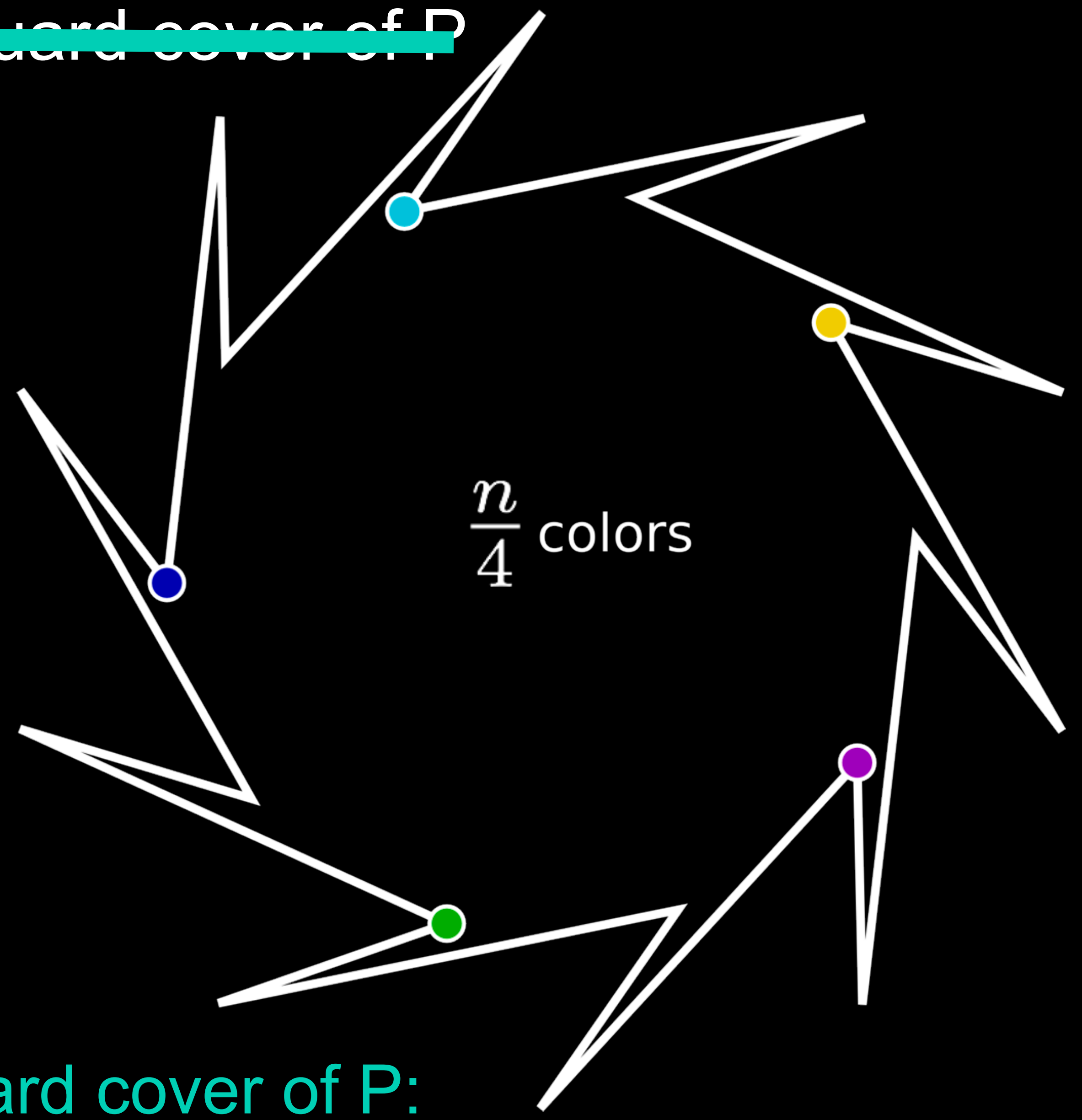


Place lights,
assign energy (= brightness).
"Sufficiently" (normalize to 1) light everything — with fading!
Minimize total energy.

Chromatic AGP:

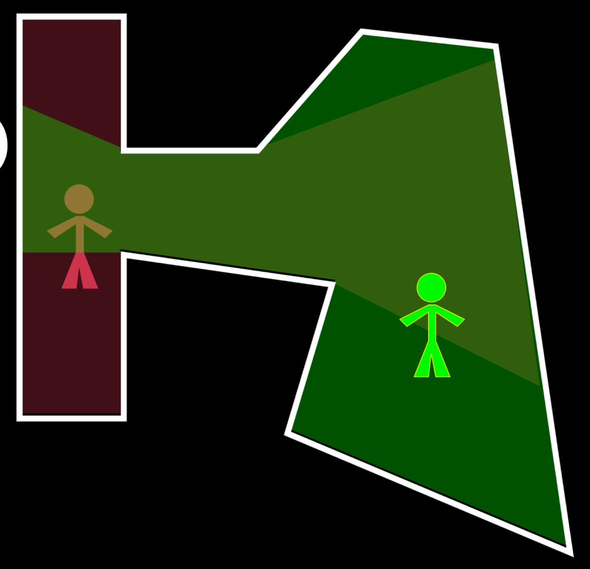
Given: a polygon P

Task: ~~find a min guard cover of P~~



Find a **colored** guard cover of P:
No point in P is seen by two guards of the same color.

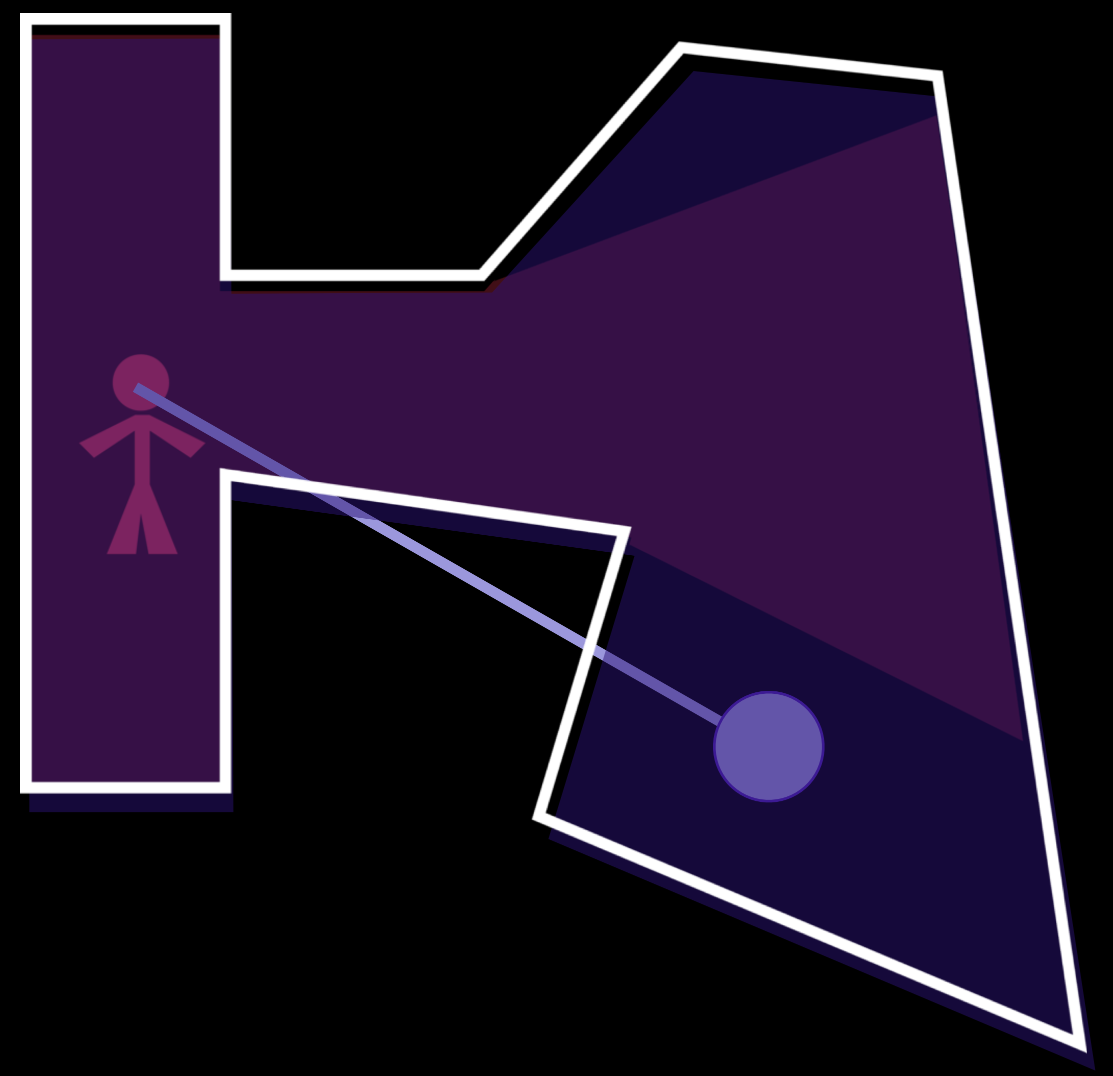
The Art Gallery Problem (AGP)—and Its Variants?



We can alter:

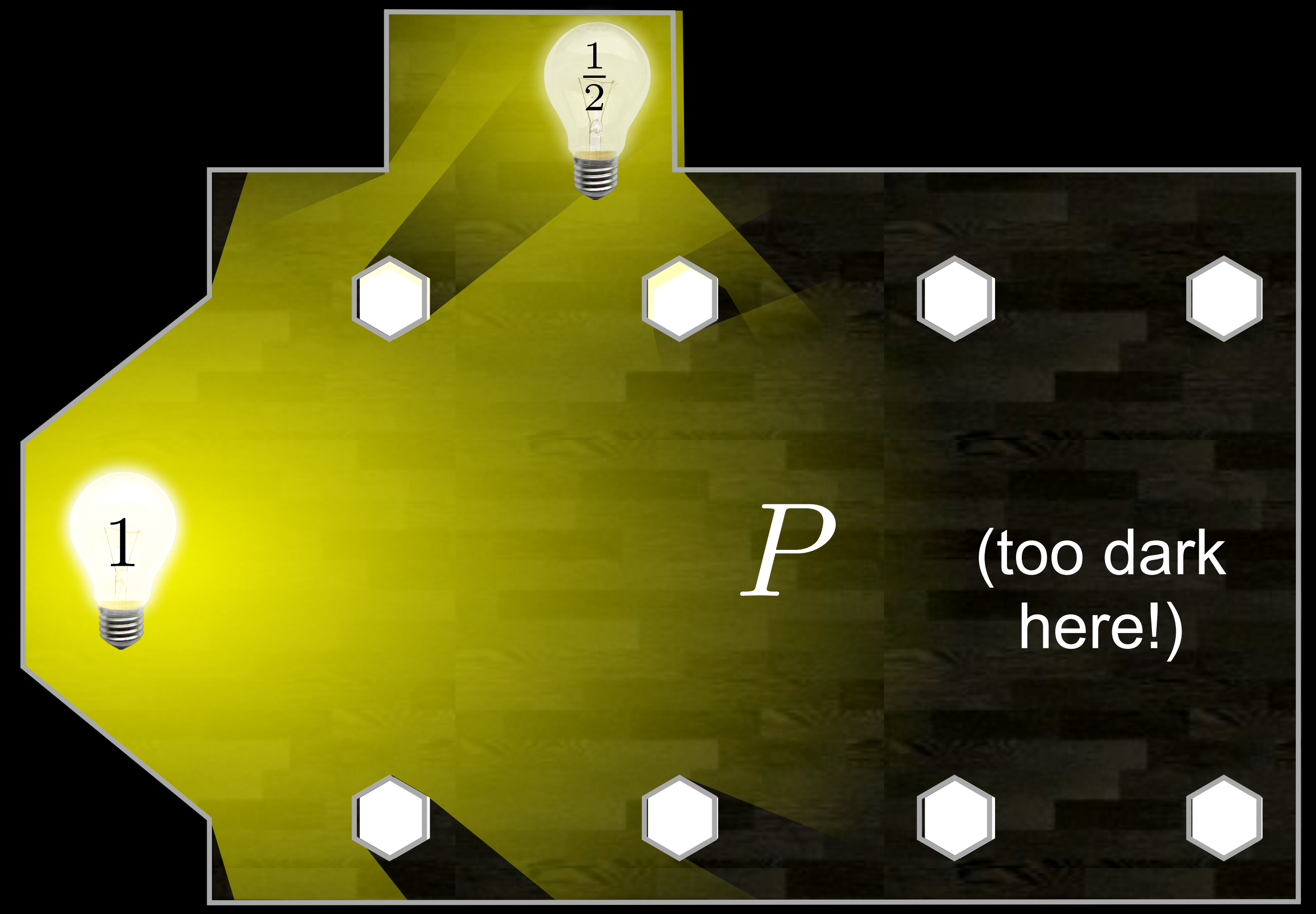
- Capabilities of the guards
- Environment to be guarded

k-transmitter:



Line crosses at most 2 walls
⇒visible from the 2-transmitter

Fading:

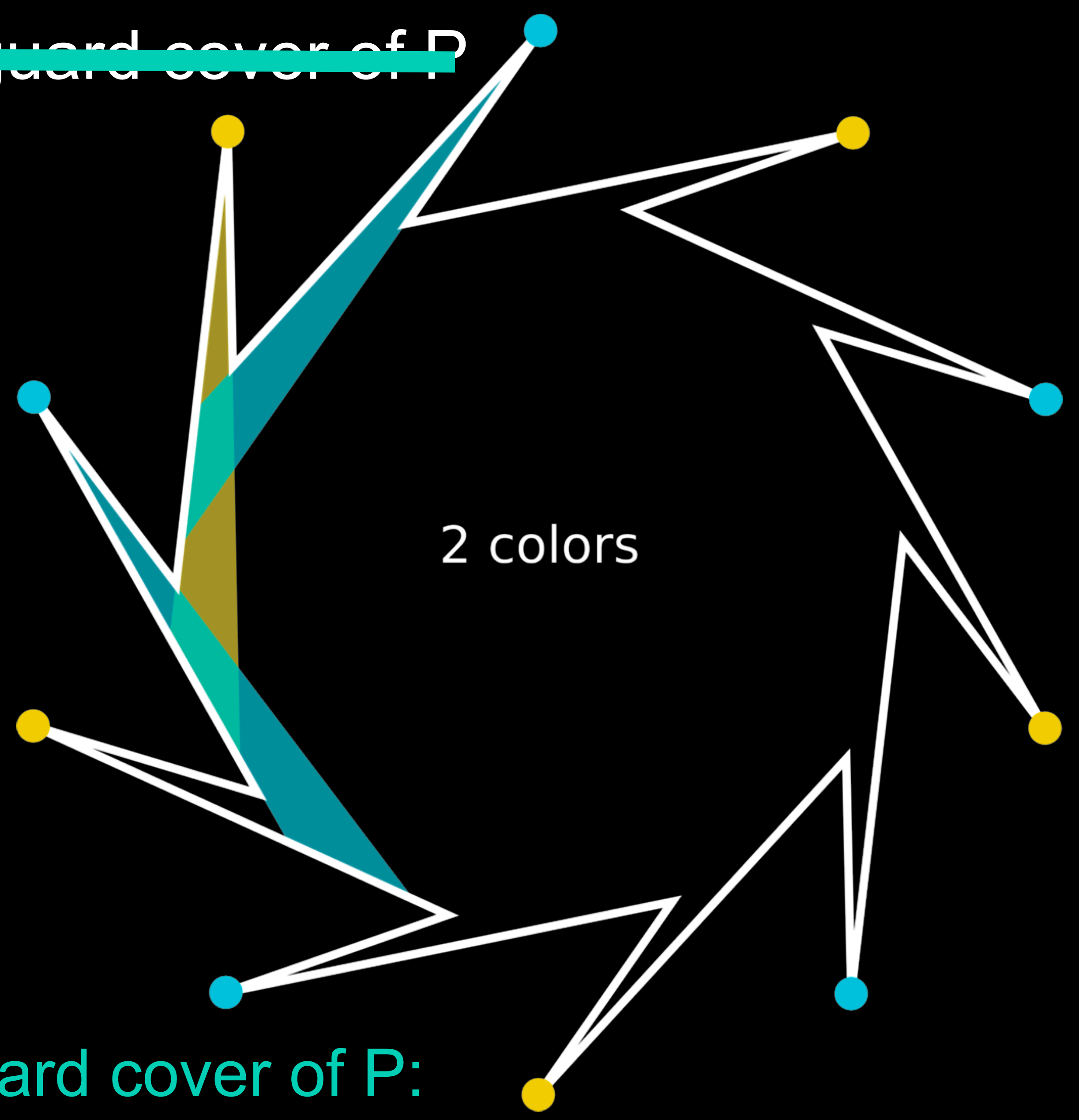


Place lights,
assign energy (= brightness).
“Sufficiently” (normalize to 1)
light everything — with fading!
Minimize total energy.

Chromatic AGP:

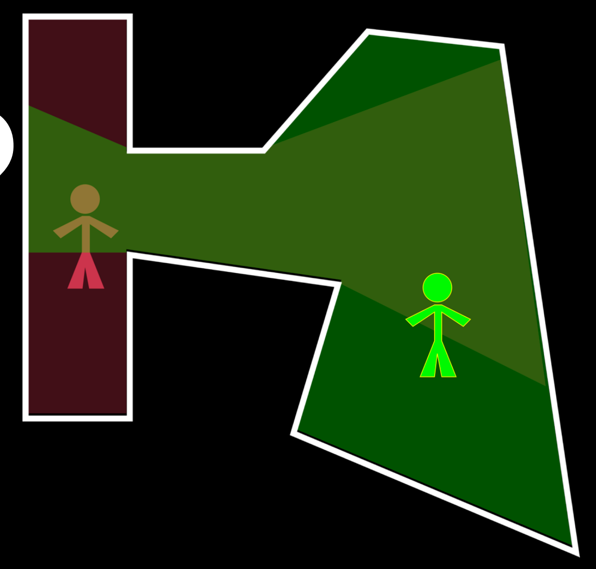
Given: a polygon P

Task: ~~find a min guard cover of P~~



Find a **colored** guard cover of P:
No point in P is seen by two guards
of the same color.

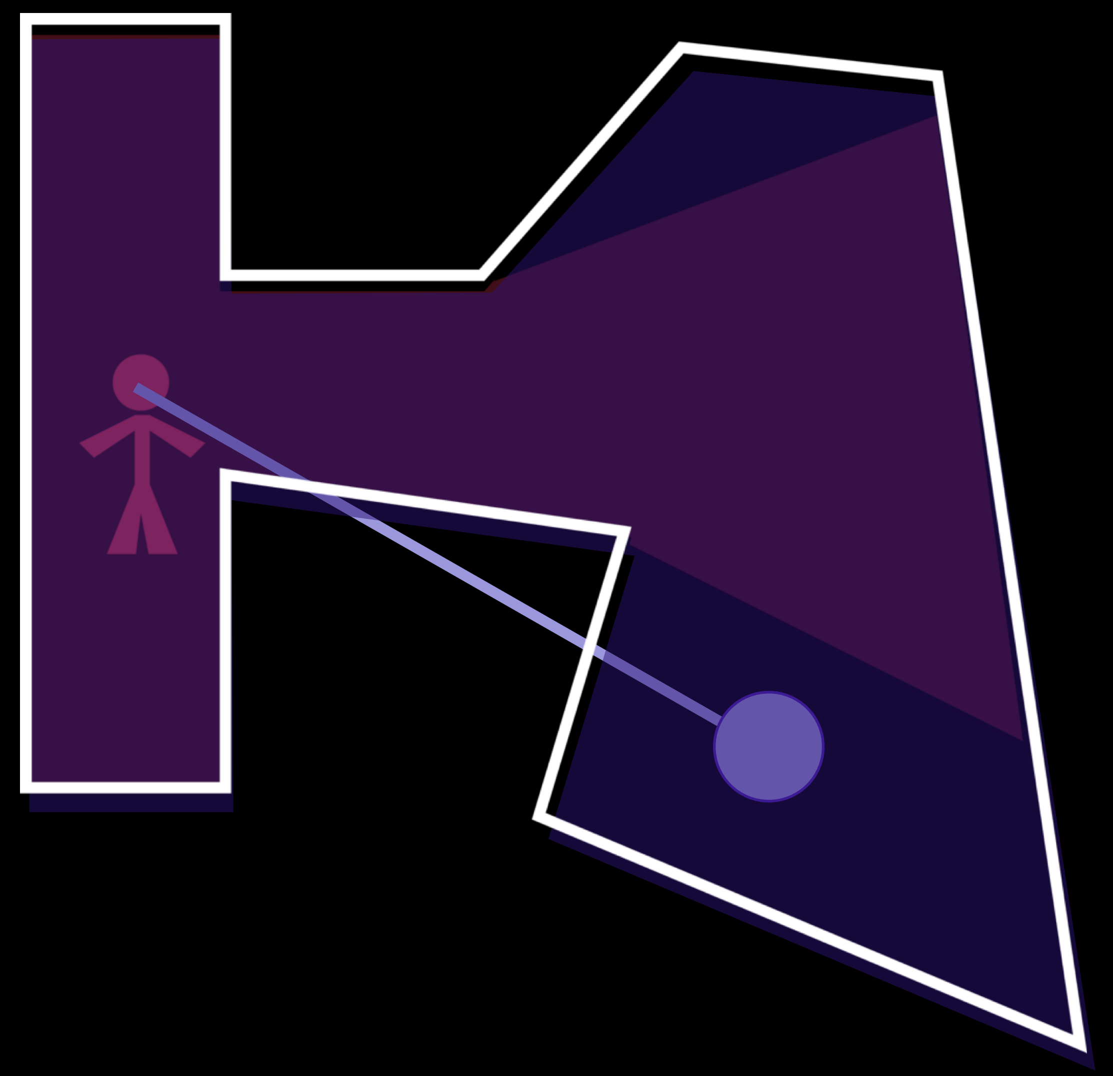
The Art Gallery Problem (AGP)—and Its Variants?



We can alter:

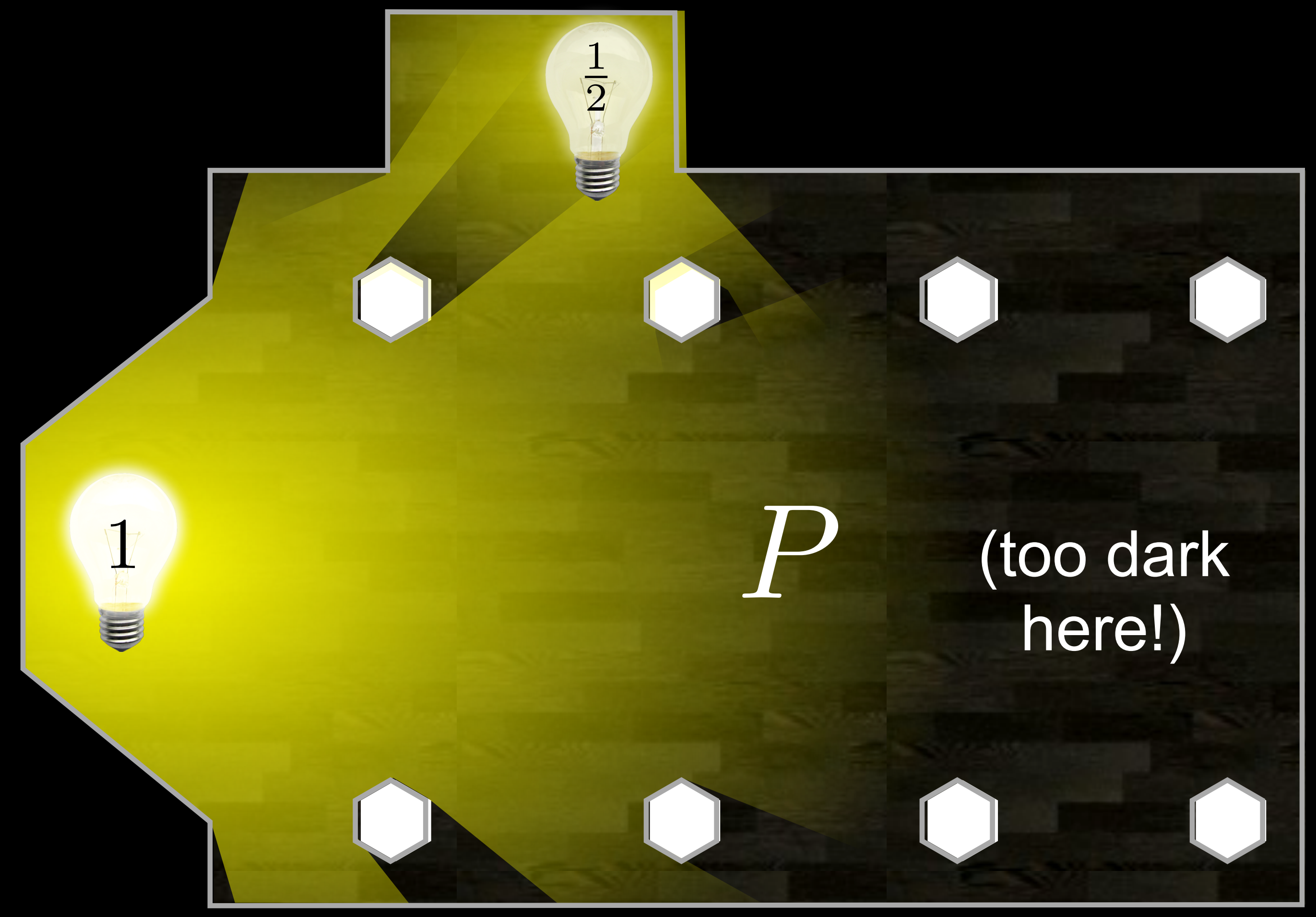
- Capabilities of the guards
- Environment to be guarded

k-transmitter:



Line crosses at most 2 walls
⇒visible from the 2-transmitter

Fading:

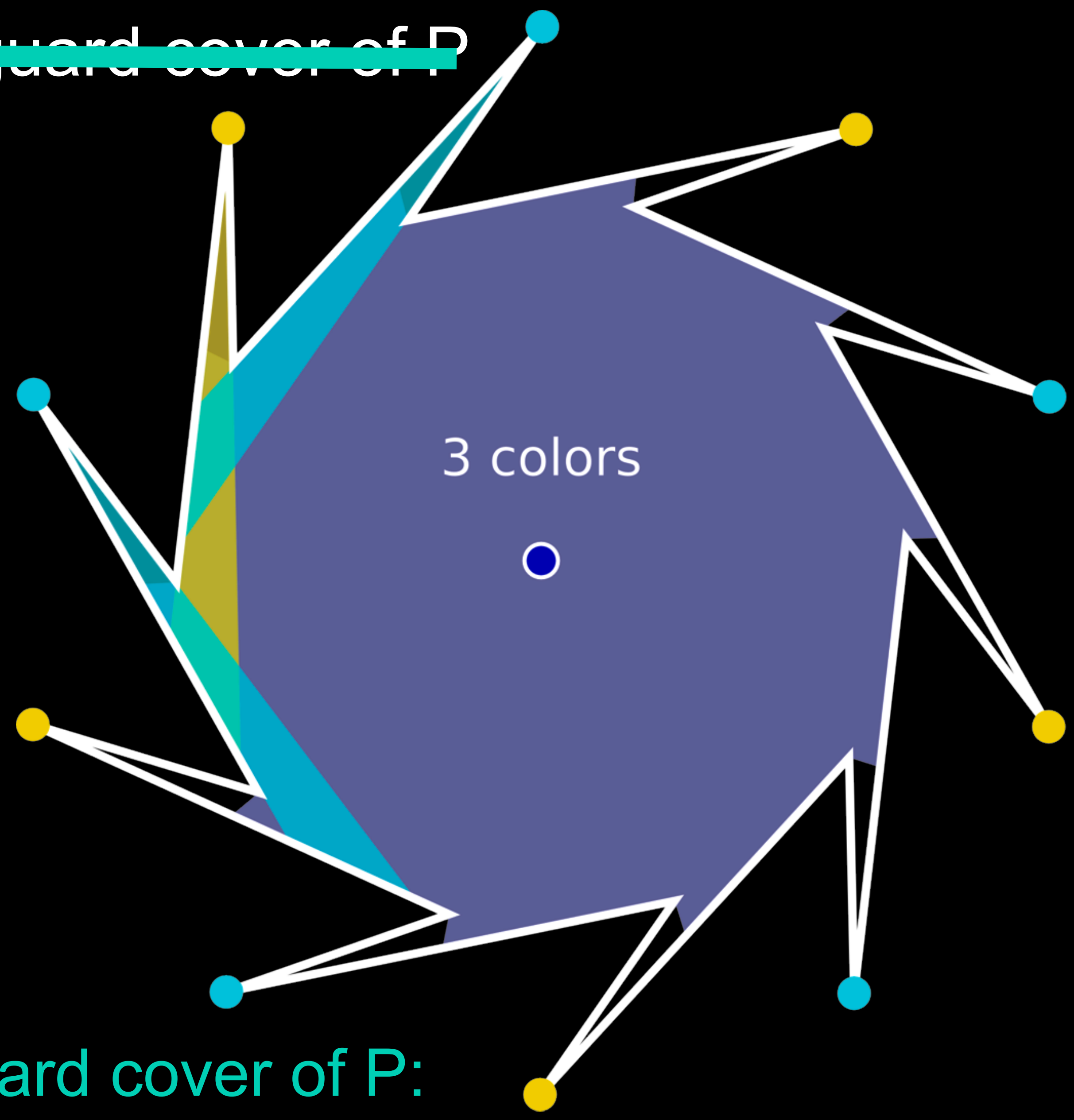


Place lights,
assign energy (= brightness).
"Sufficiently" (normalize to 1) light everything — with fading!
Minimize total energy.

Chromatic AGP:

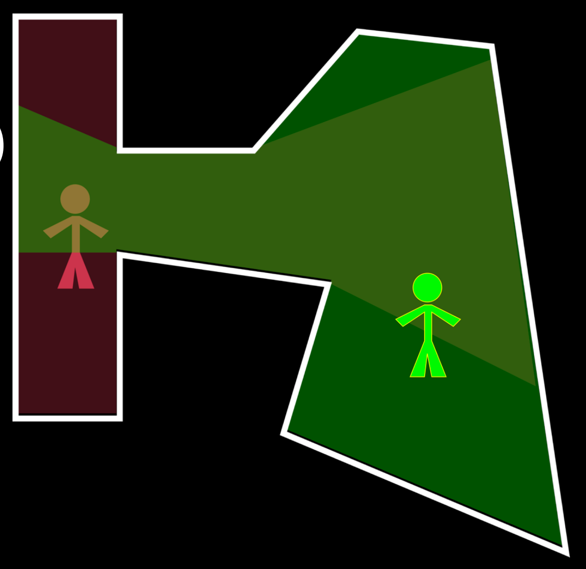
Given: a polygon P

Task: ~~find a min guard cover of P~~



Find a **colored** guard cover of P:
No point in P is seen by two guards of the same color.

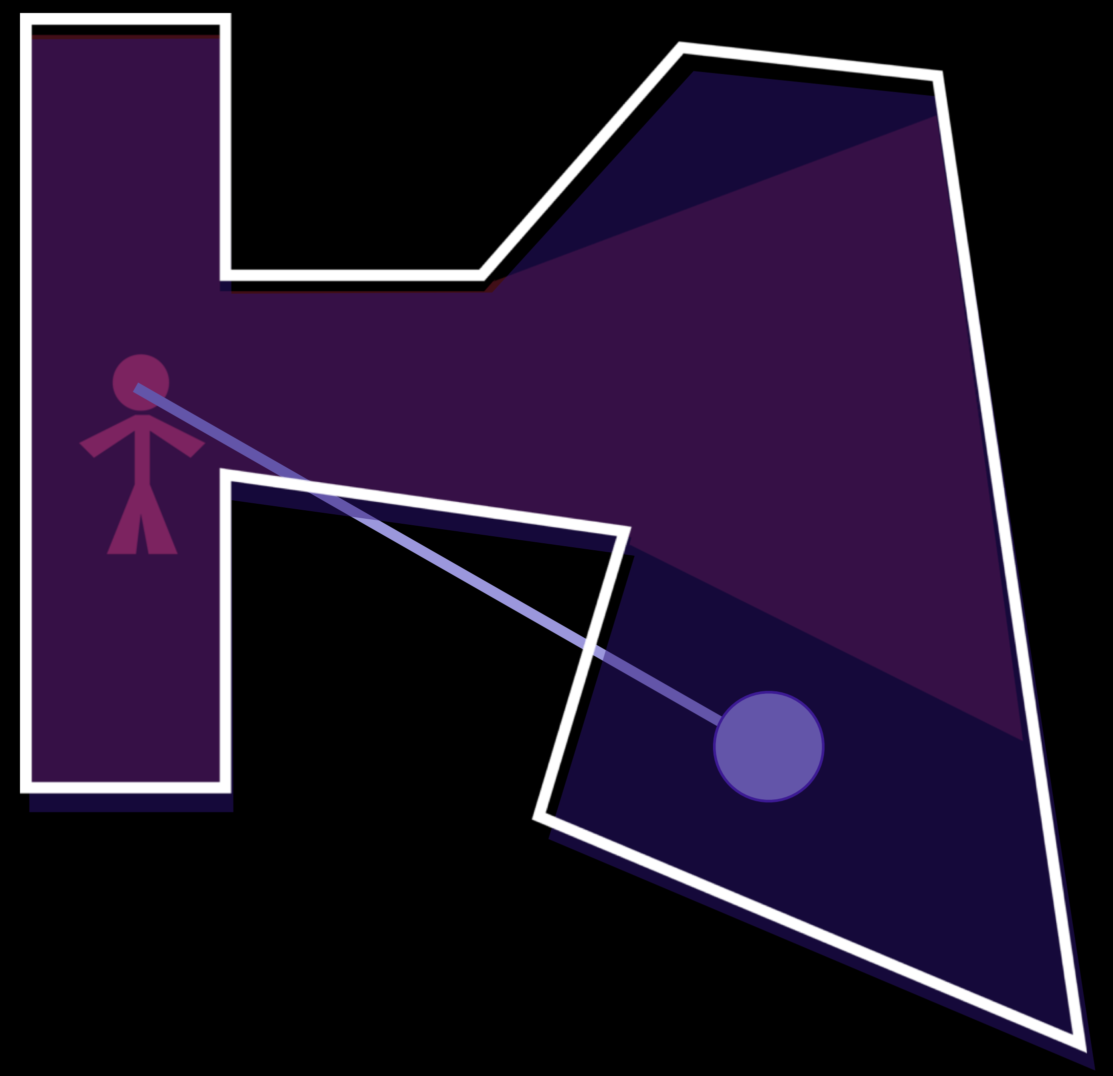
The Art Gallery Problem (AGP)—and Its Variants?



We can alter:

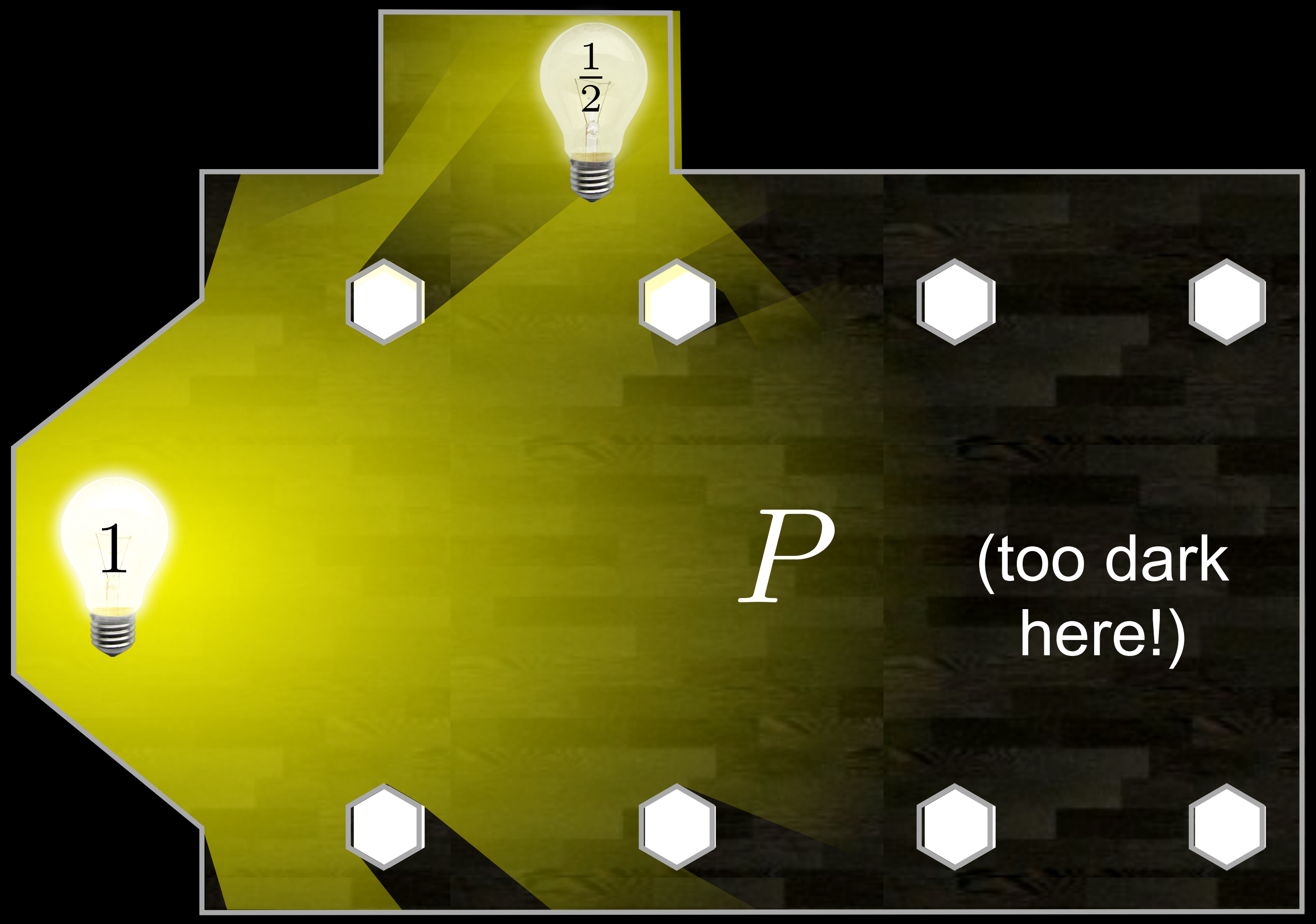
- Capabilities of the guards
- Environment to be guarded

k-transmitter:



Line crosses at most 2 walls
⇒visible from the 2-transmitter

Fading:



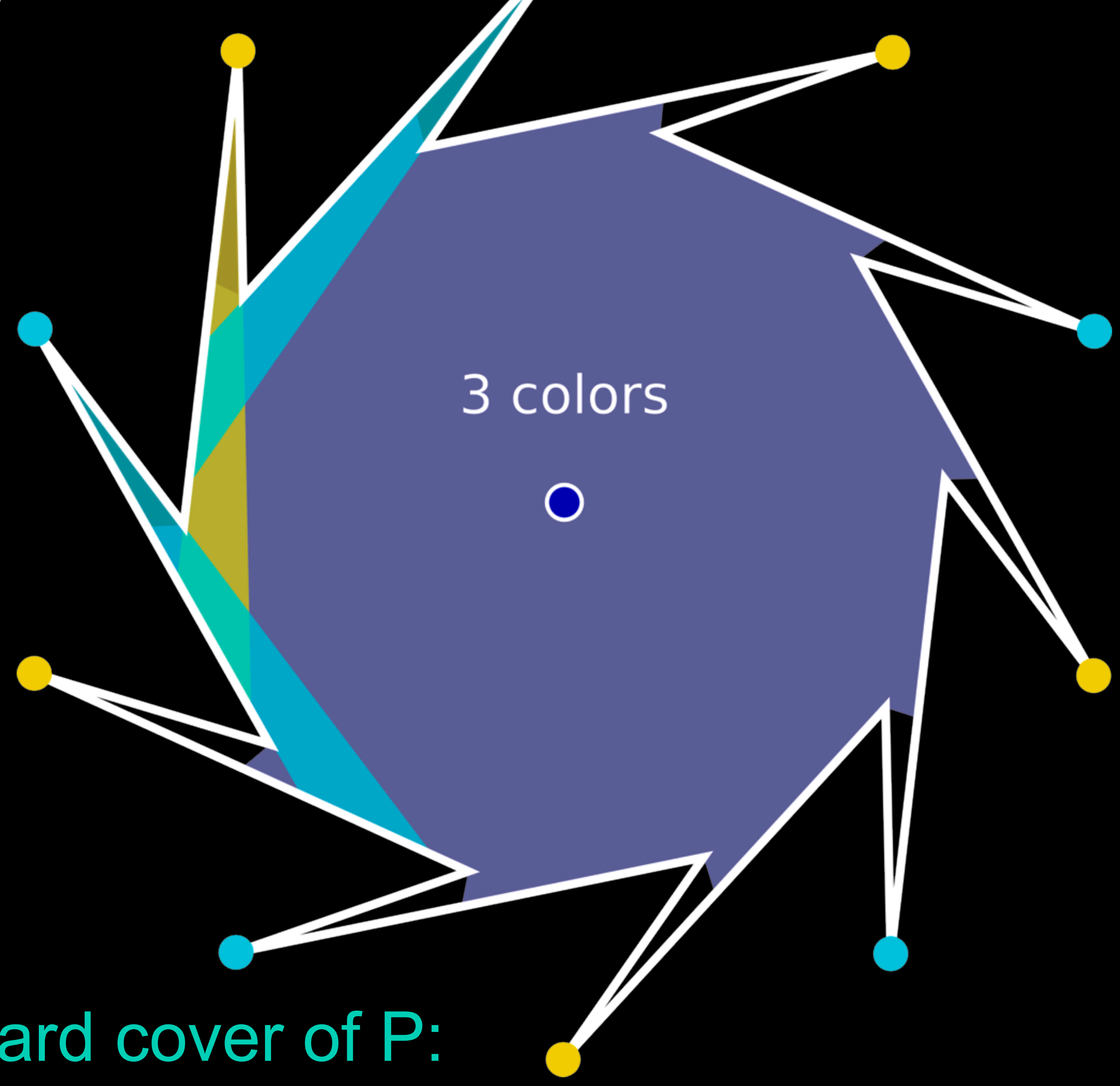
Place lights,
assign energy (= brightness).
“Sufficiently” (normalize to 1) light everything — with fading!
Minimize total energy.

Chromatic AGP:

We do not care about the number of guards, but about the number of colors!

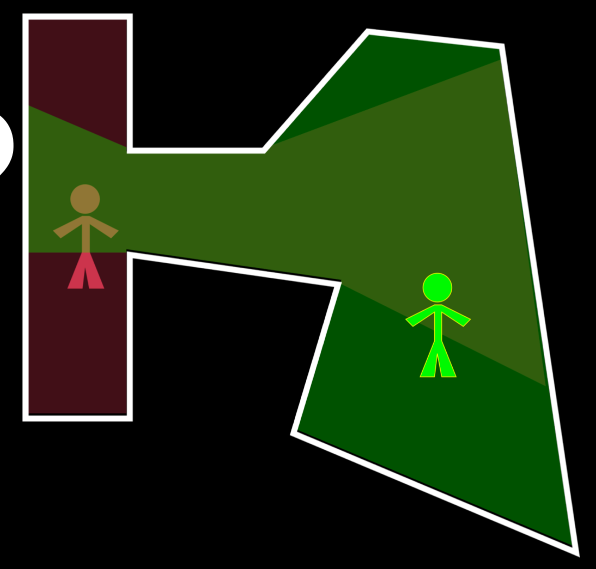
Given: a polygon P

Task: ~~find a min guard cover of P~~



Find a **colored** guard cover of P:
No point in P is seen by two guards of the same color.

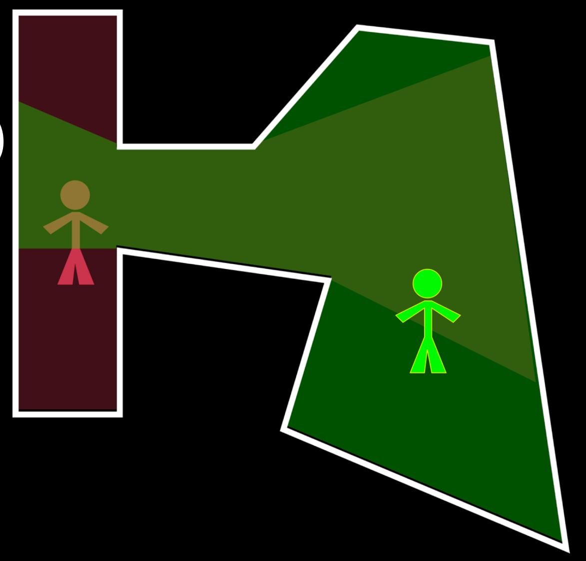
The Art Gallery Problem (AGP)—and Its Variants?



We can alter:

- Capabilities of the guards
- Environment to be guarded

The Art Gallery Problem (AGP)—and Its Variants?

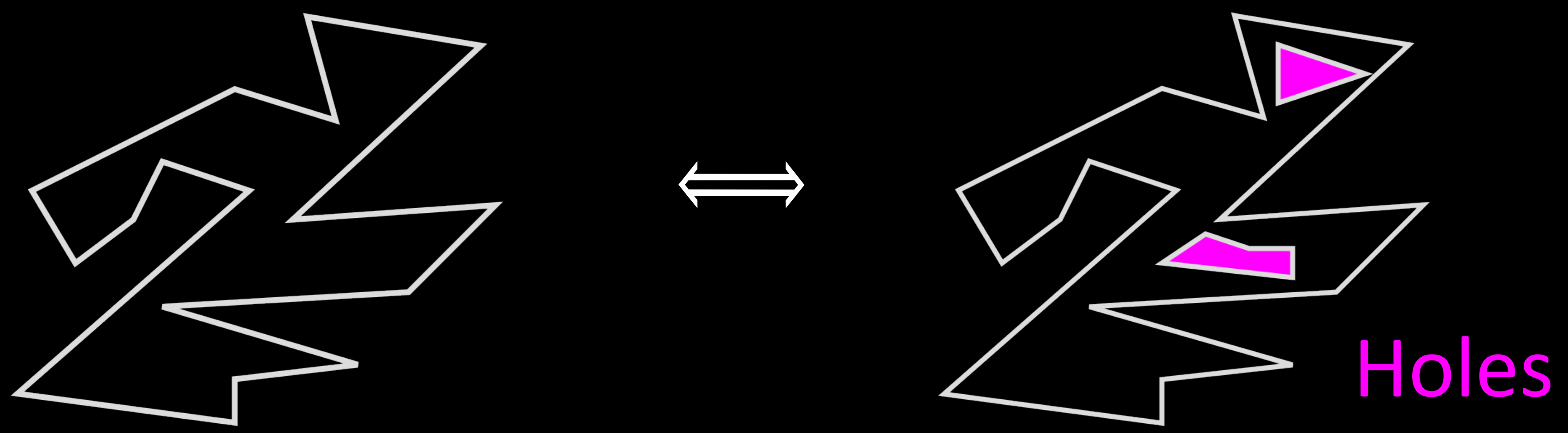


We can alter:

- Capabilities of the guards
- Environment to be guarded

Alter the polygon class:

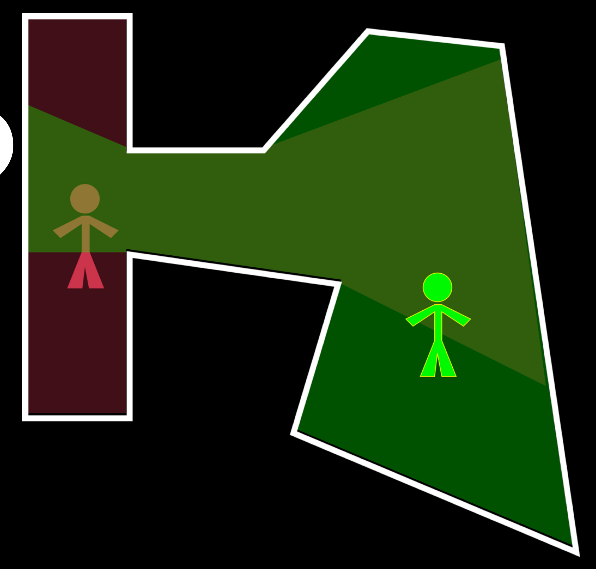
Traditionally:
Simple polygons or polygons with holes



Simple polygon:

- Does not intersect itself
- No holes

The Art Gallery Problem (AGP)—and Its Variants?

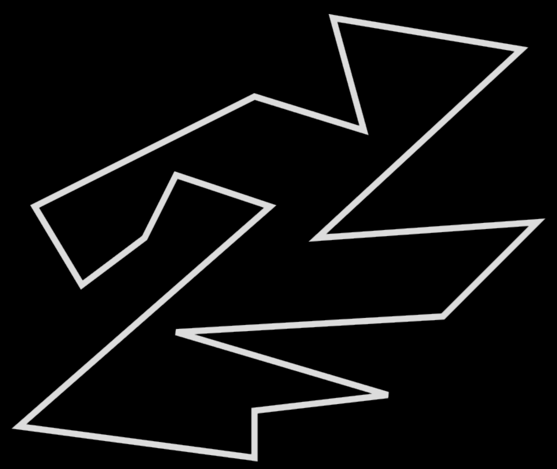


We can alter:

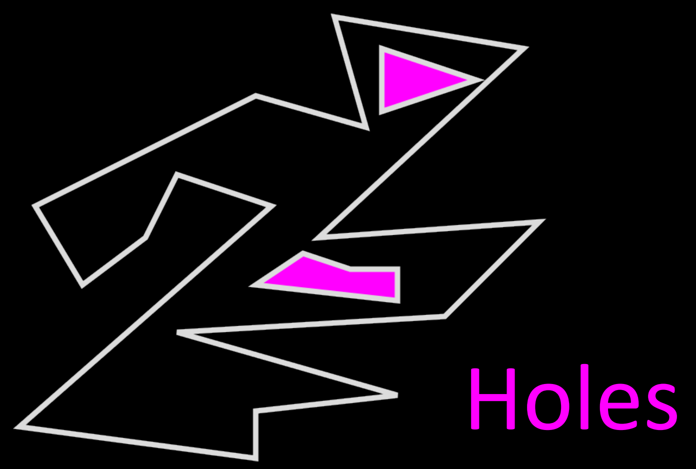
- Capabilities of the guards
- Environment to be guarded

Alter the polygon class:

Traditionally:
Simple polygons or polygons with holes



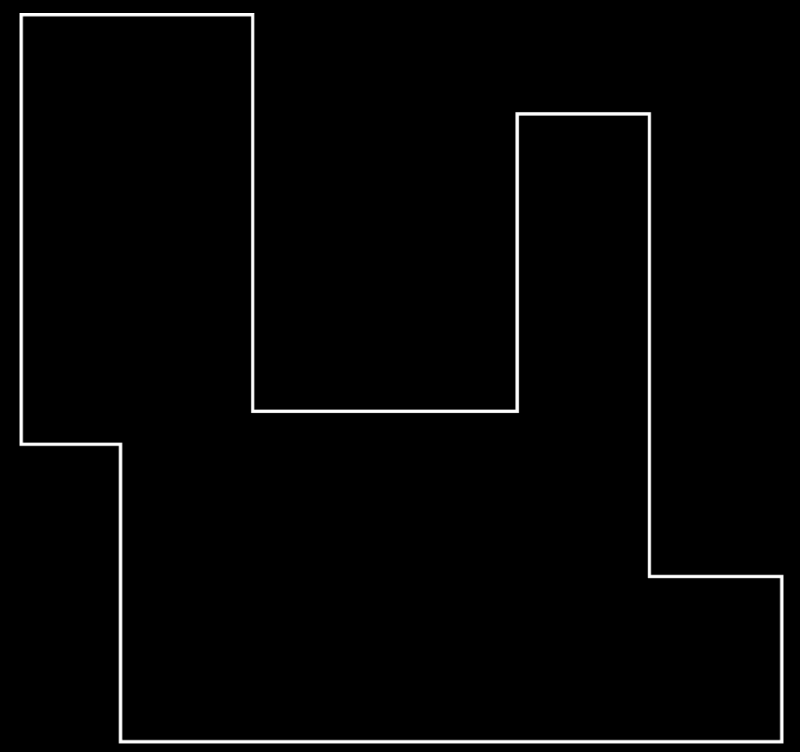
⟺



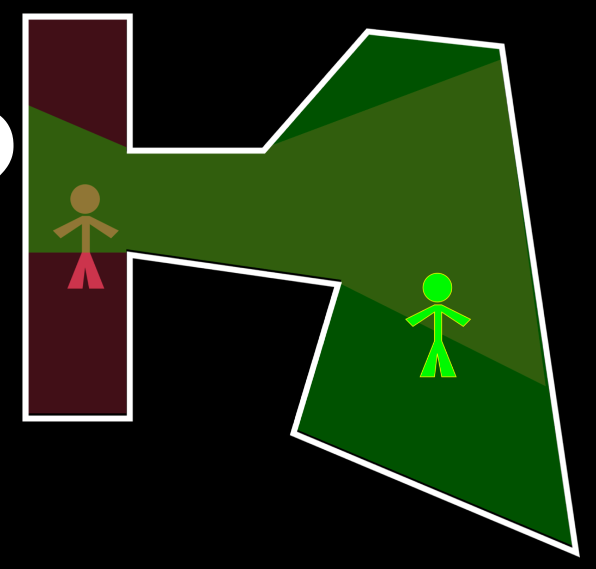
Simple polygon:

- Does not intersect itself
- No holes

Rectilinear polygons



The Art Gallery Problem (AGP)—and Its Variants?

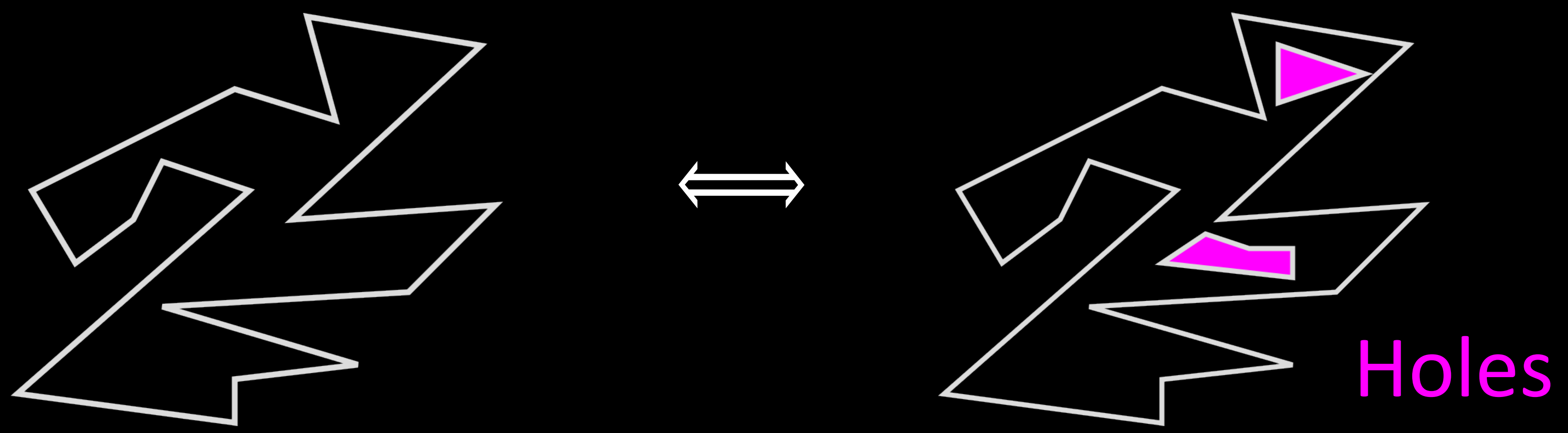


We can alter:

- Capabilities of the guards
- Environment to be guarded

Alter the polygon class:

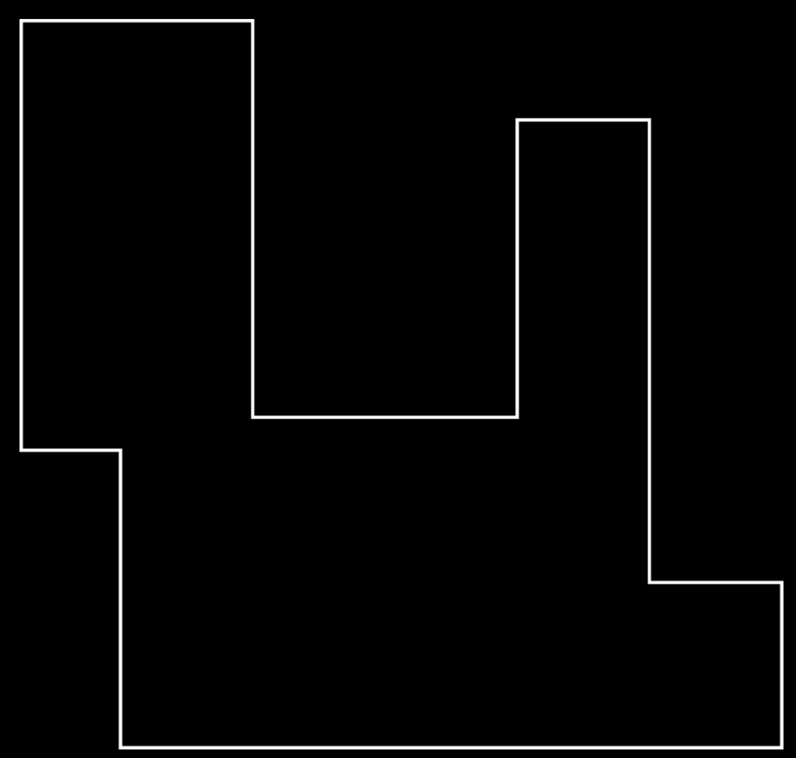
Traditionally:
Simple polygons or polygons with holes



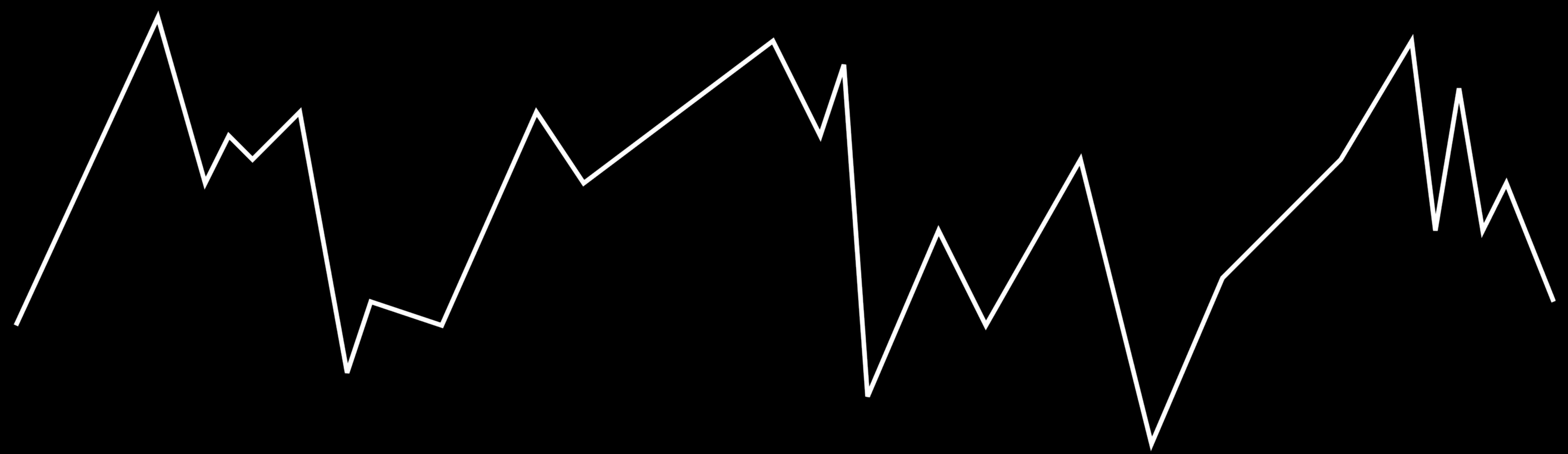
Simple polygon:

- Does not intersect itself
- No holes

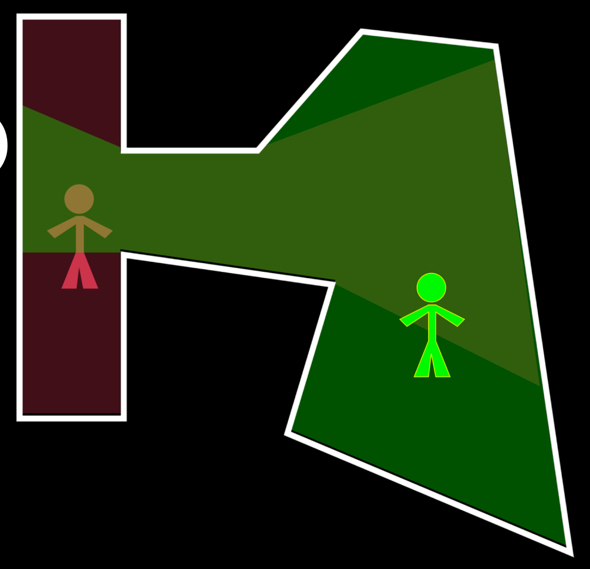
Rectilinear polygons



Guard a 1.5D-Terrain



The Art Gallery Problem (AGP)—and Its Variants?

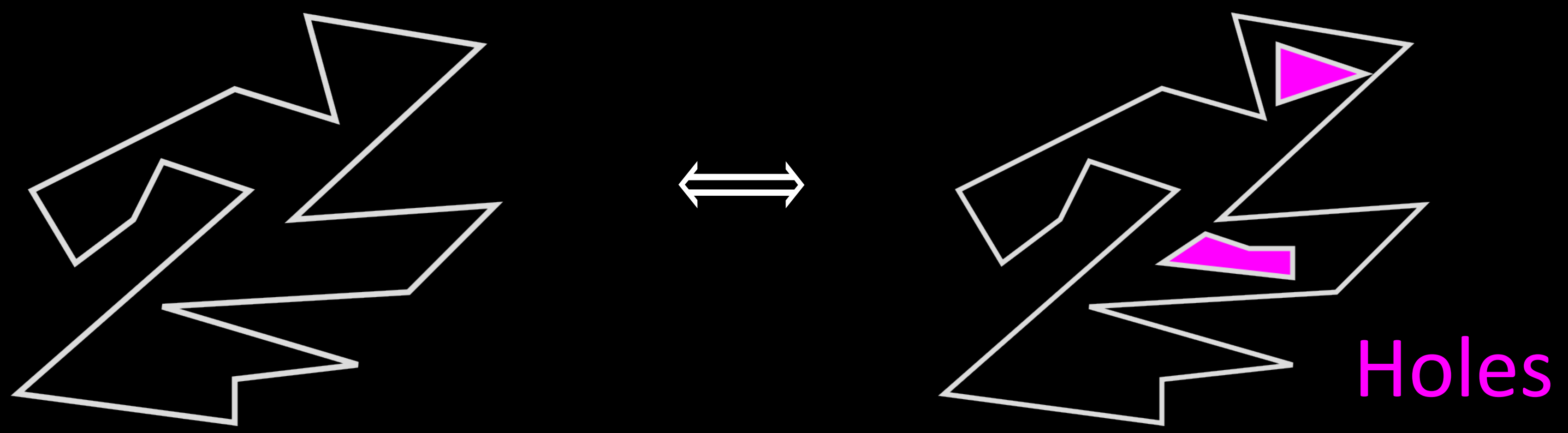


We can alter:

- Capabilities of the guards
- Environment to be guarded

Alter the polygon class:

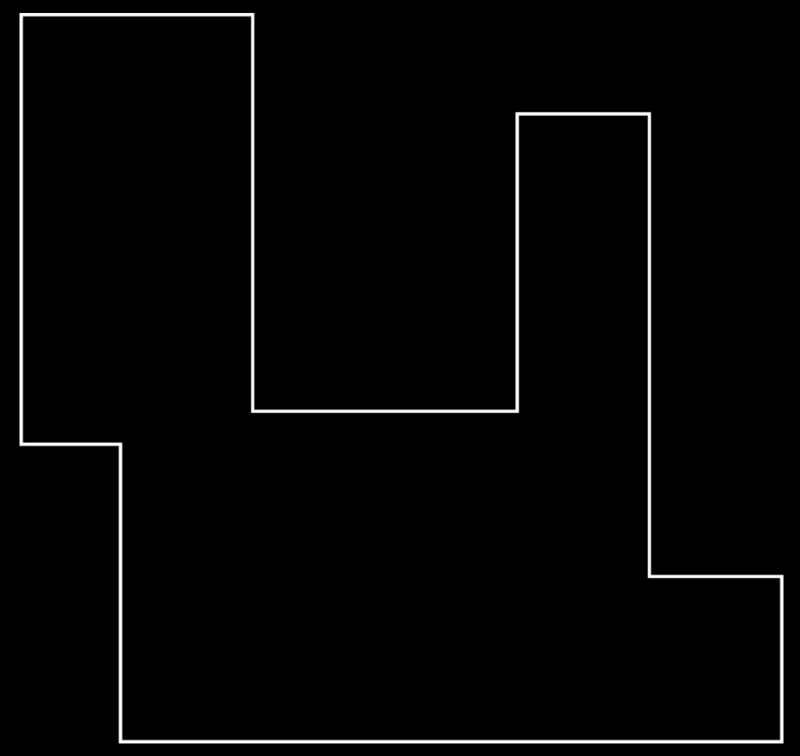
Traditionally:
Simple polygons or polygons with holes



Simple polygon:

- Does not intersect itself
- No holes

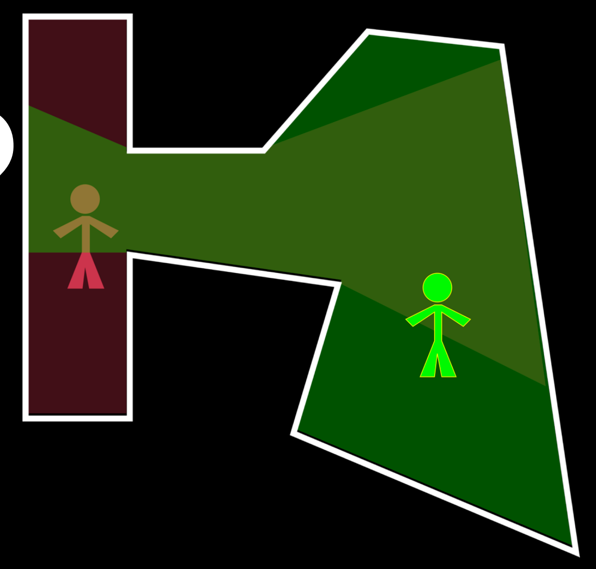
Rectilinear polygons



Guard a 1.5D-Terrain

- With guards on the terrain

The Art Gallery Problem (AGP)—and Its Variants?

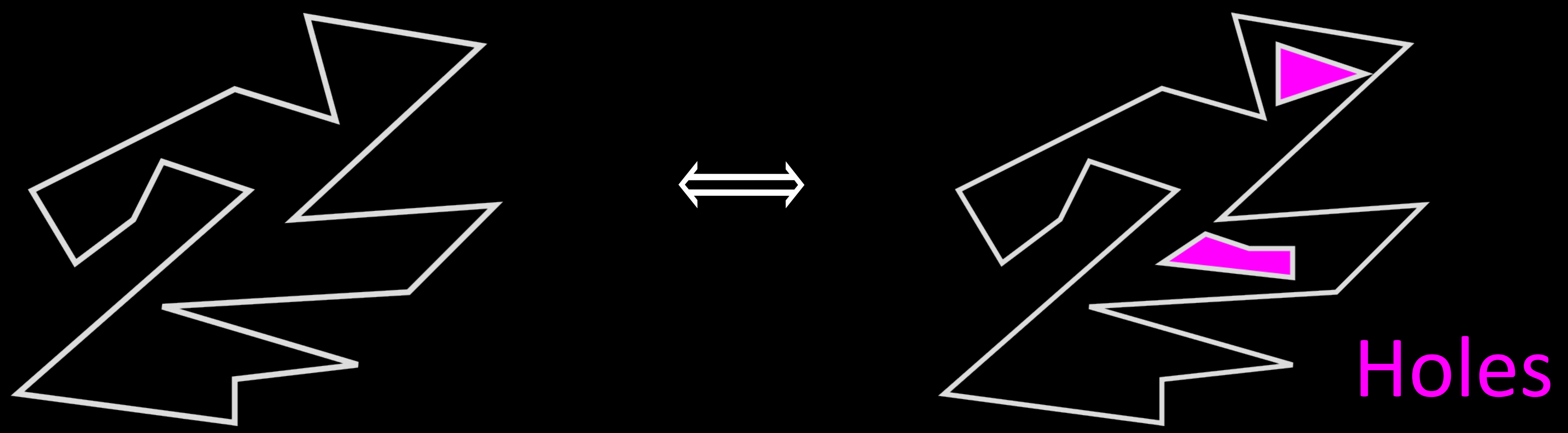


We can alter:

- Capabilities of the guards
- Environment to be guarded

Alter the polygon class:

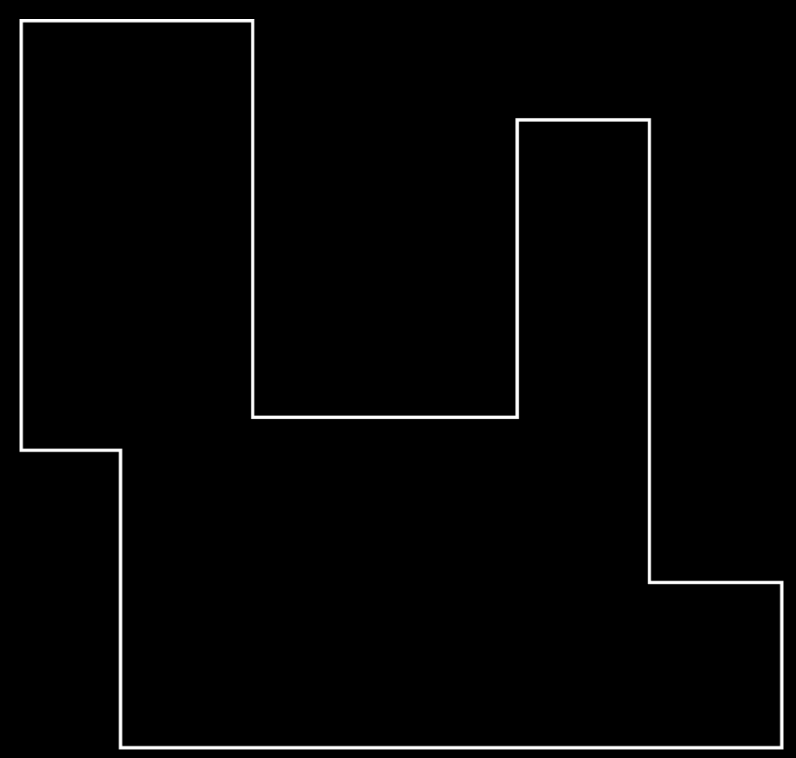
Traditionally:
Simple polygons or polygons with holes



Simple polygon:

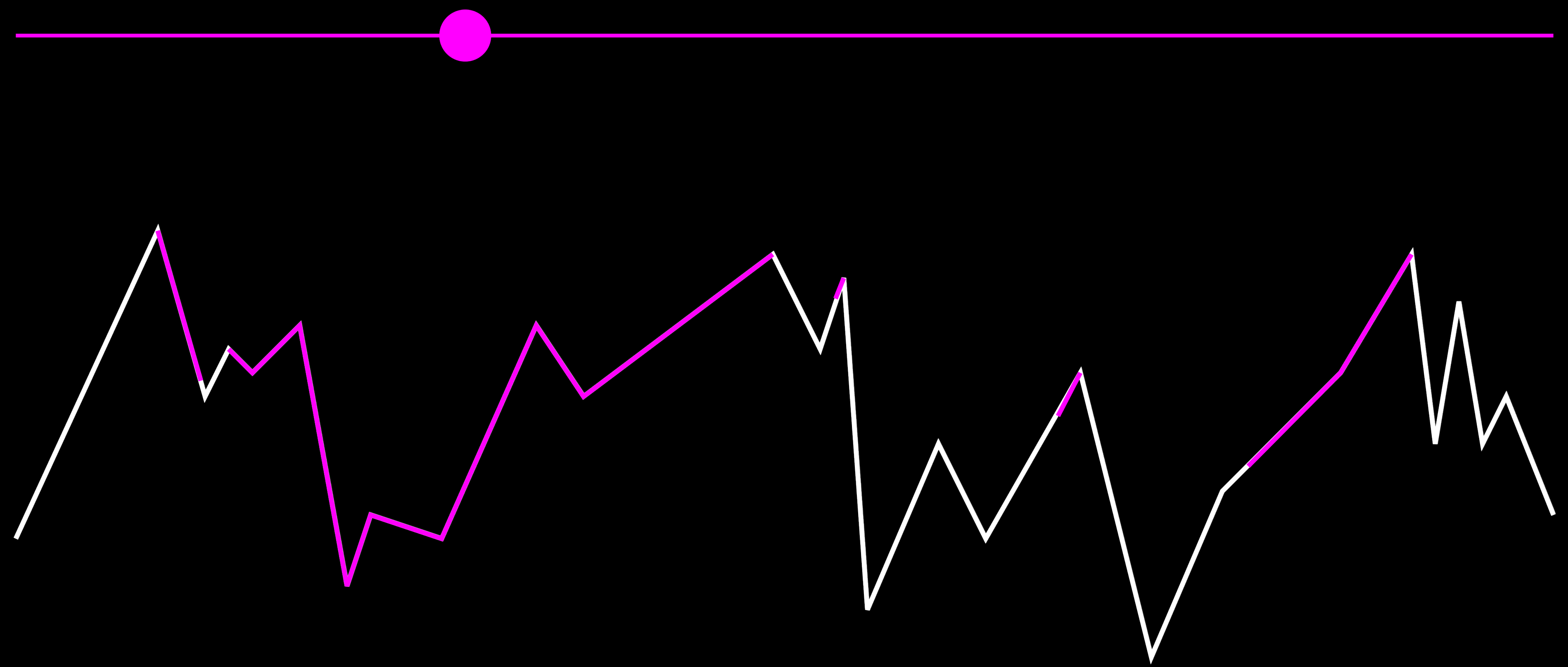
- Does not intersect itself
- No holes

Rectilinear polygons



Guard a 1.5D-Terrain

- With guards on the terrain
- With guards on an altitude line above the terrain



k-Hop Visibility

Motivation

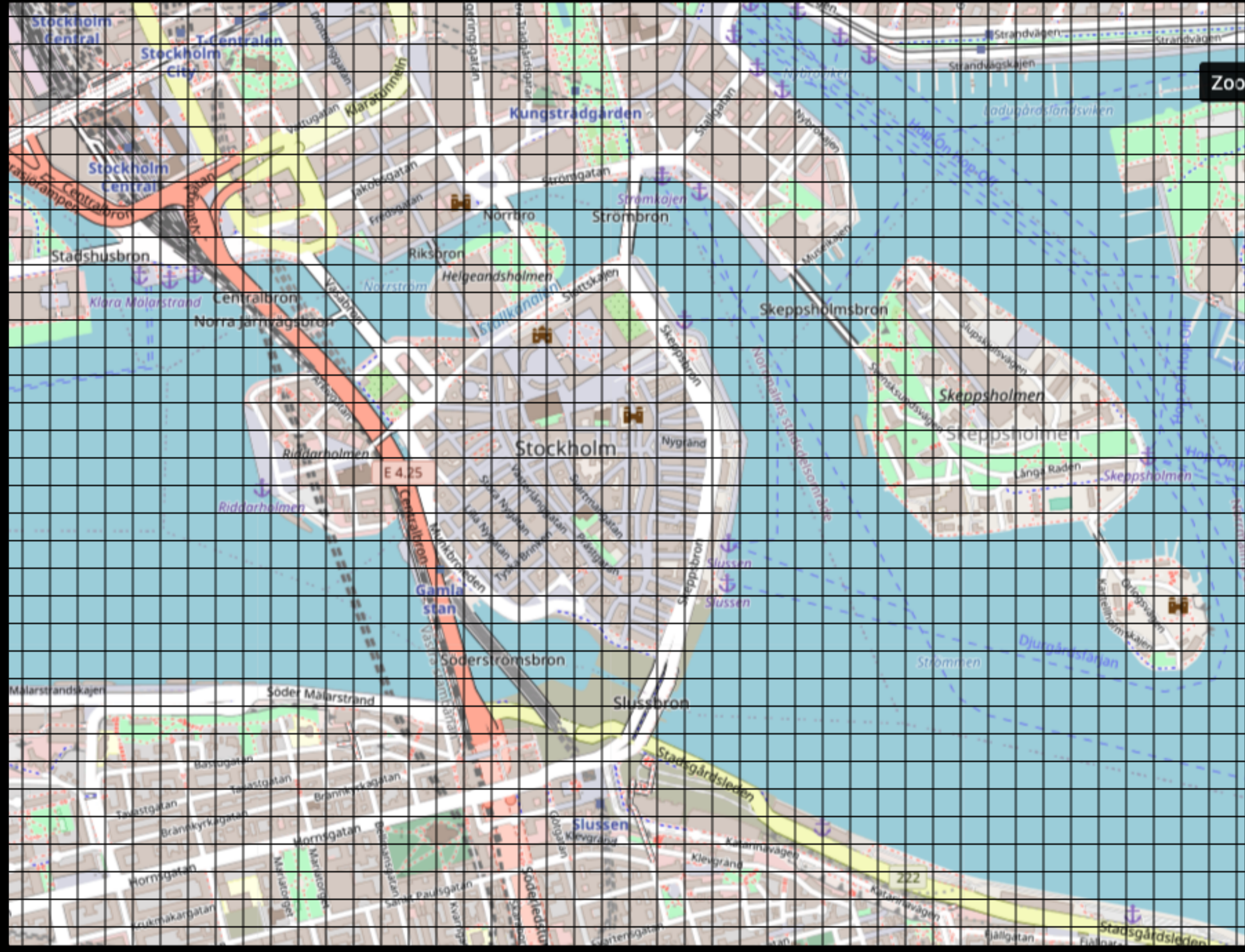
- Serve a city with carsharing (CS) stations:

Motivation

- Serve a city with carsharing (CS) stations:
  - Demand in granularity of square cells

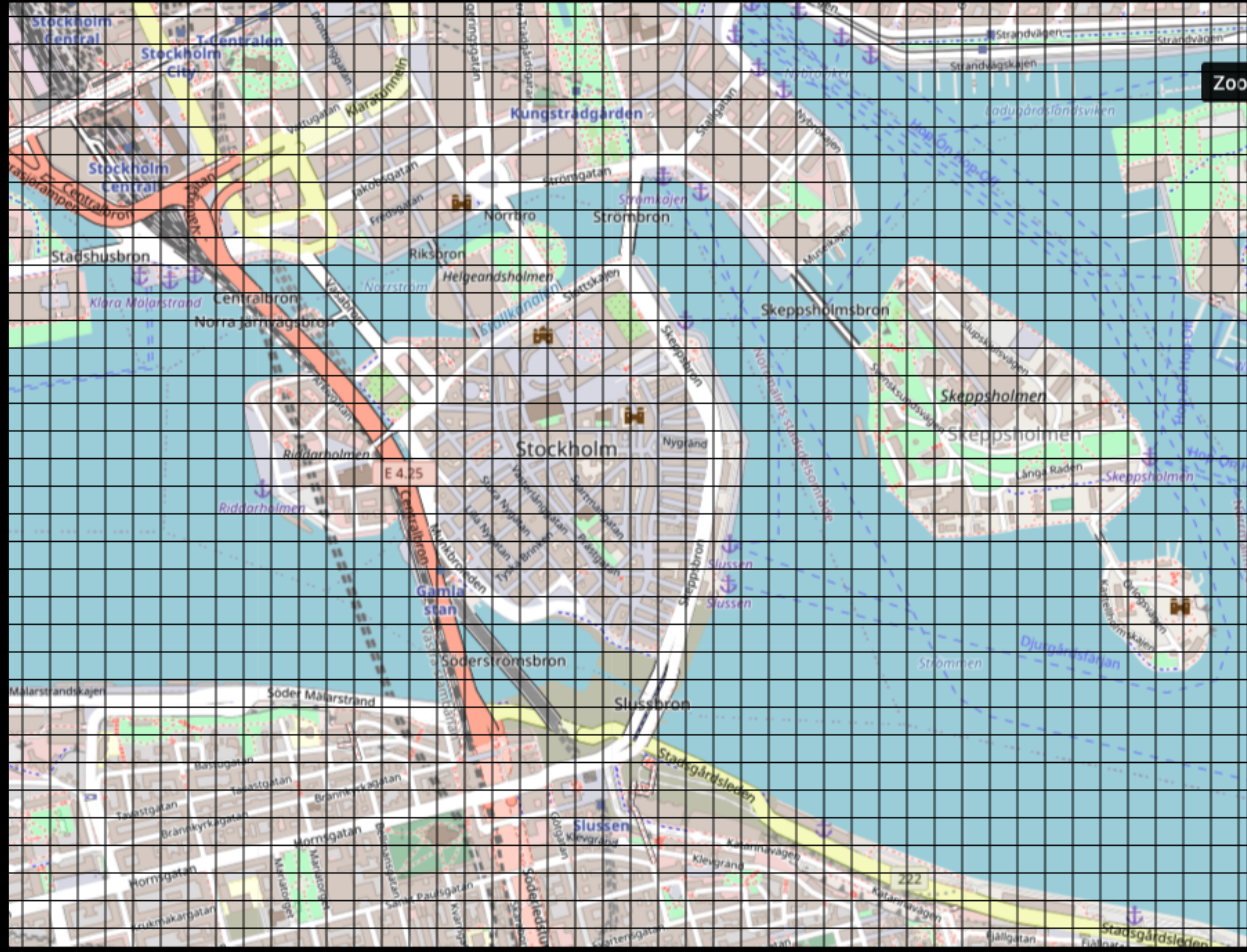
Motivation

- Serve a city with carsharing (CS) stations:
  - Demand in granularity of square cells

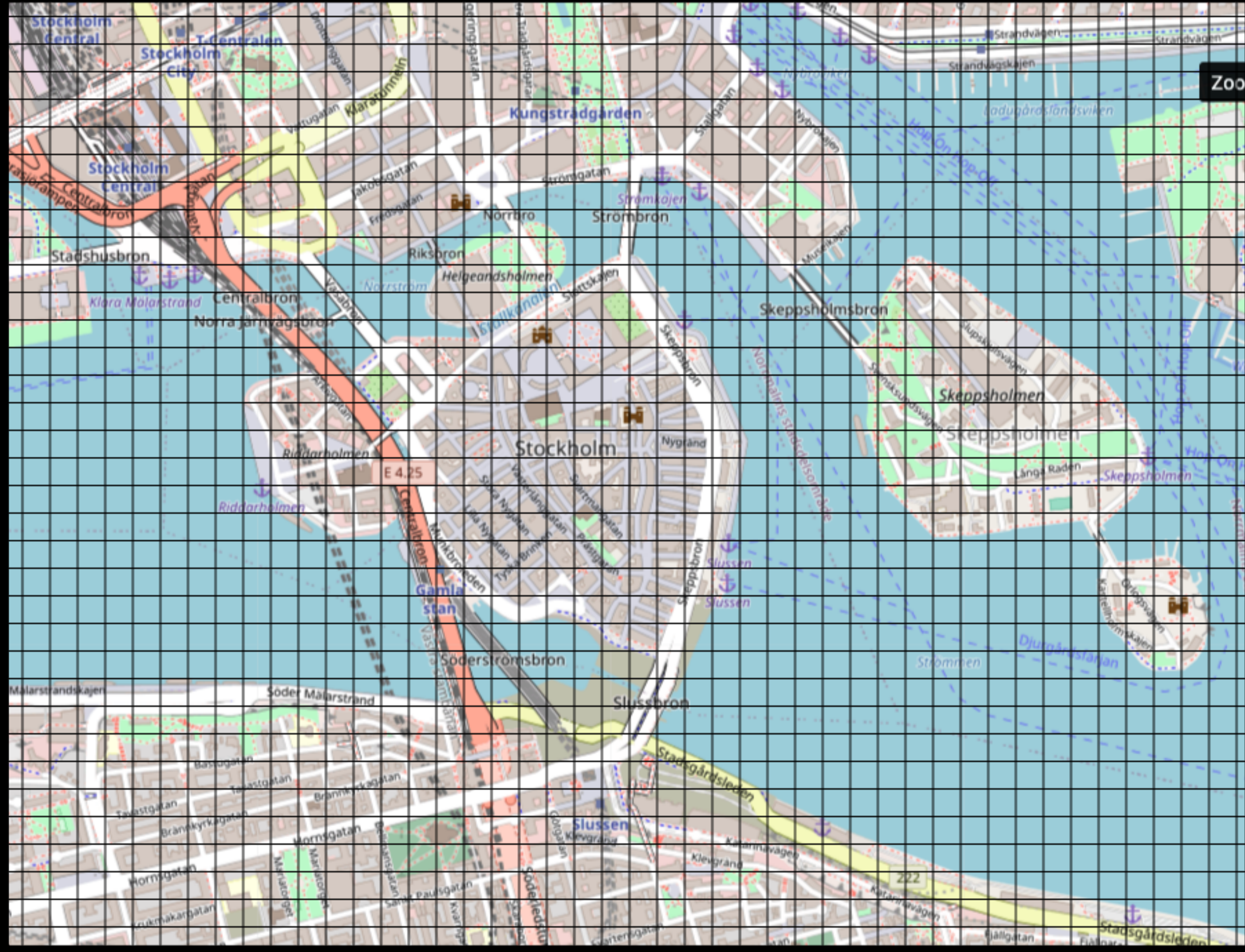


Motivation

- Serve a city with carsharing (CS) stations:
  - Demand in granularity of square cells
  - Customers willing to walk a certain distance



Motivation



- Serve a city with carsharing (CS) stations:
  - Demand in granularity of square cells
  - Customers willing to walk a certain distance
  - Same distance bound for the complete city

Motivation

- Serve a city with carsharing (CS) stations:
  - Demand in granularity of square cells
  - Customers willing to walk a certain distance
  - Same distance bound for the complete city

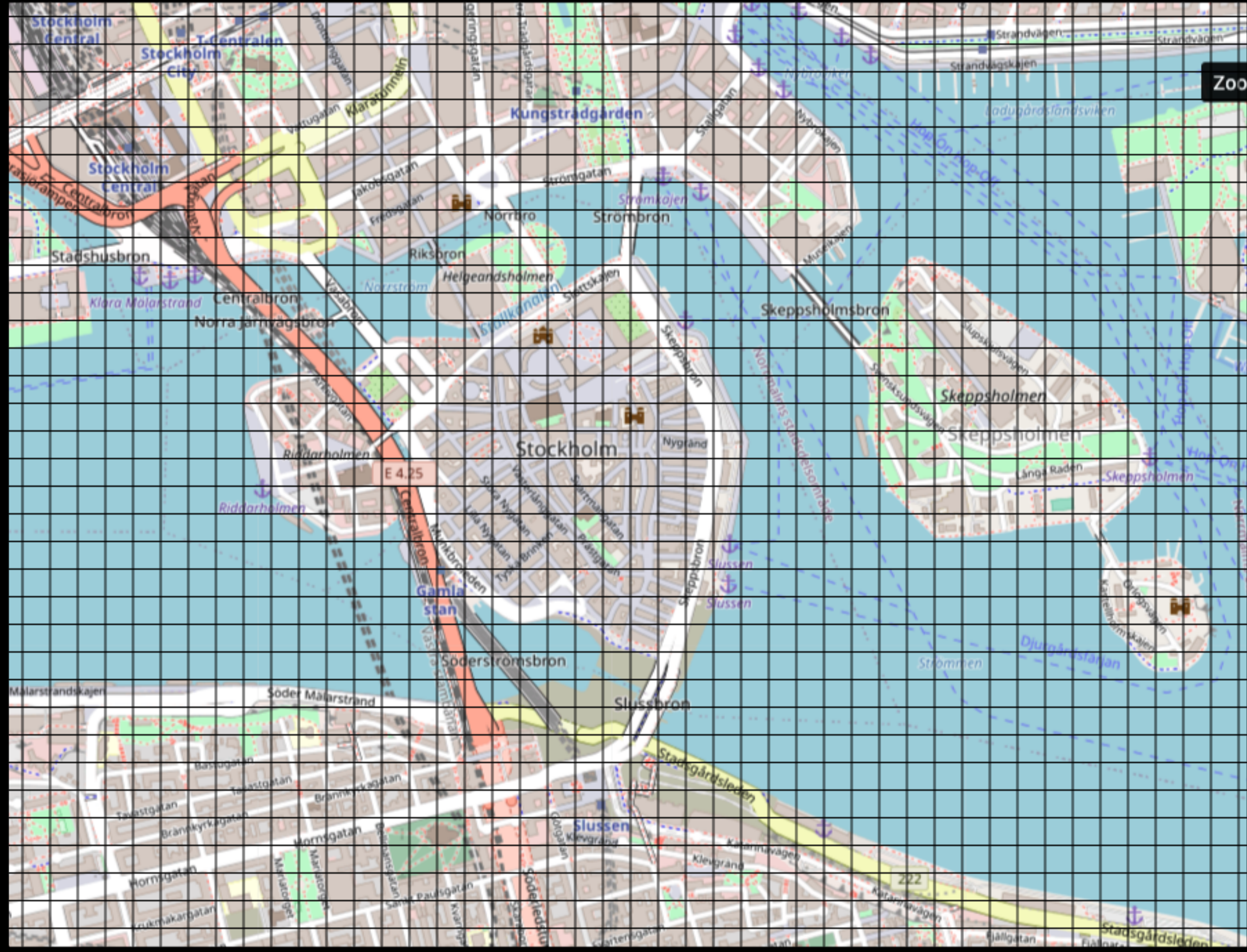
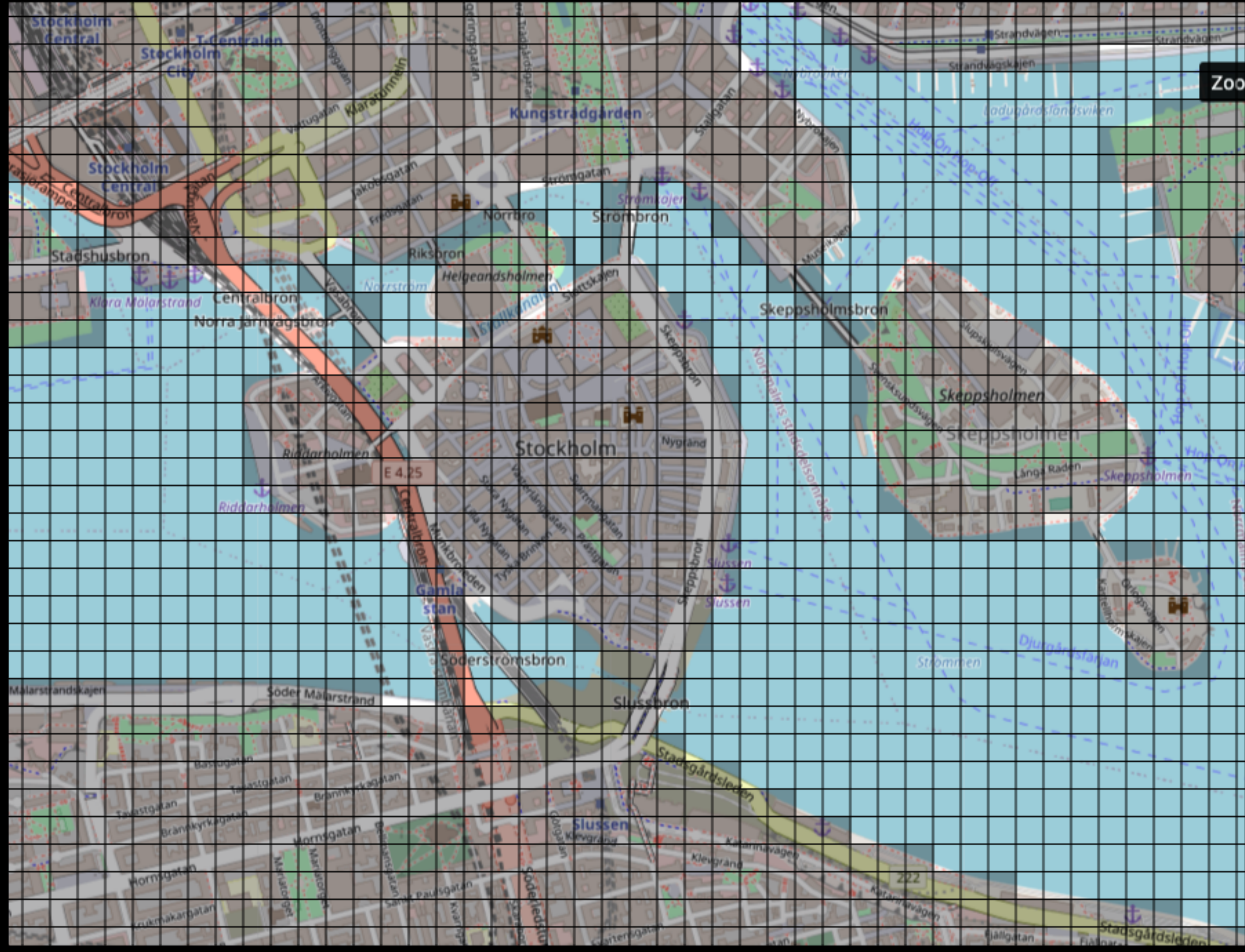


Image source: Gunnar Flötteröd, Waterborne Urban Mobility, Final Project Report

LINKÖPING UNIVERSITY

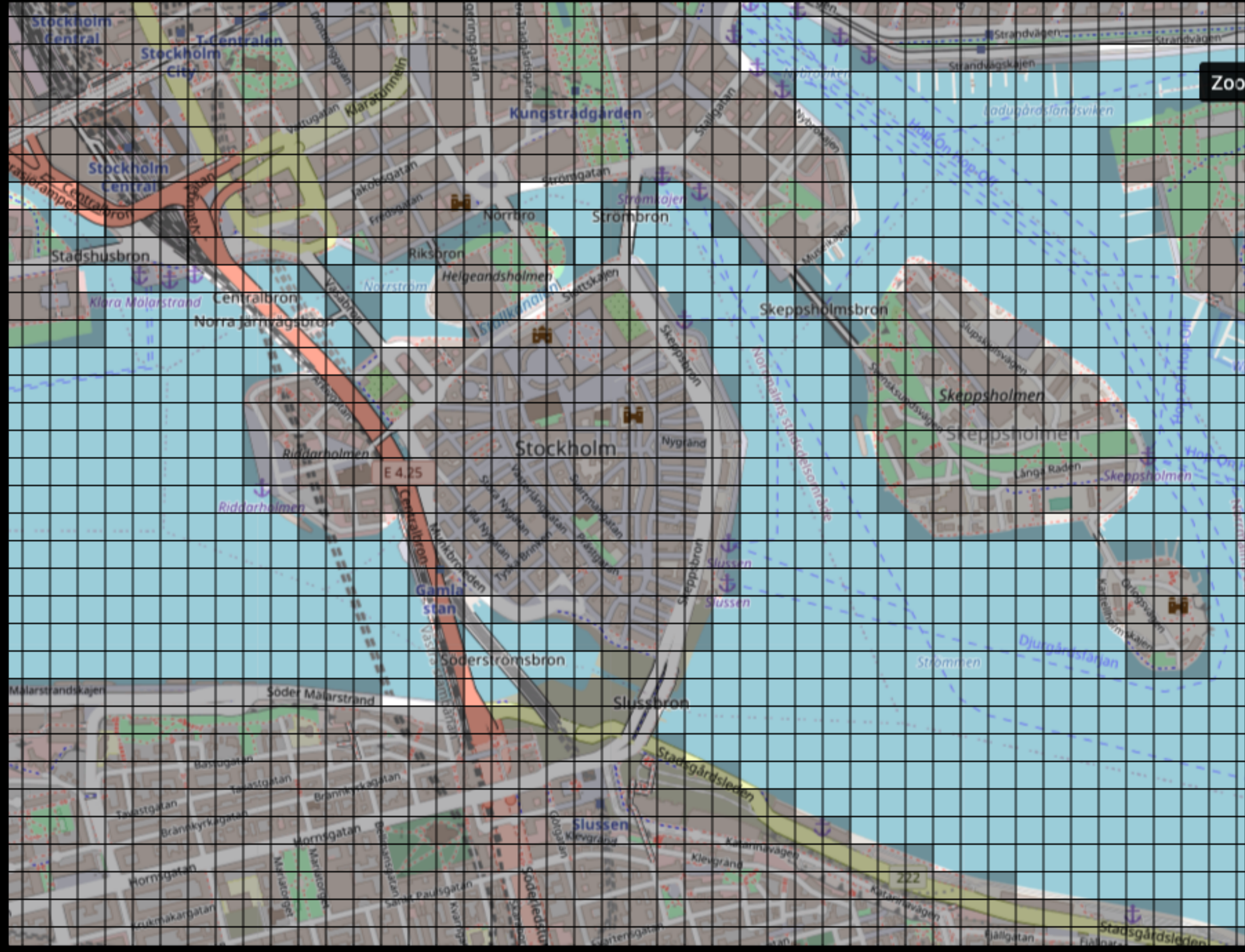
Motivation

- Serve a city with carsharing (CS) stations:
  - Demand in granularity of square cells
  - Customers willing to walk a certain distance
  - Same distance bound for the complete city
  - City ➜ Polyomino (➜ alter the environment)

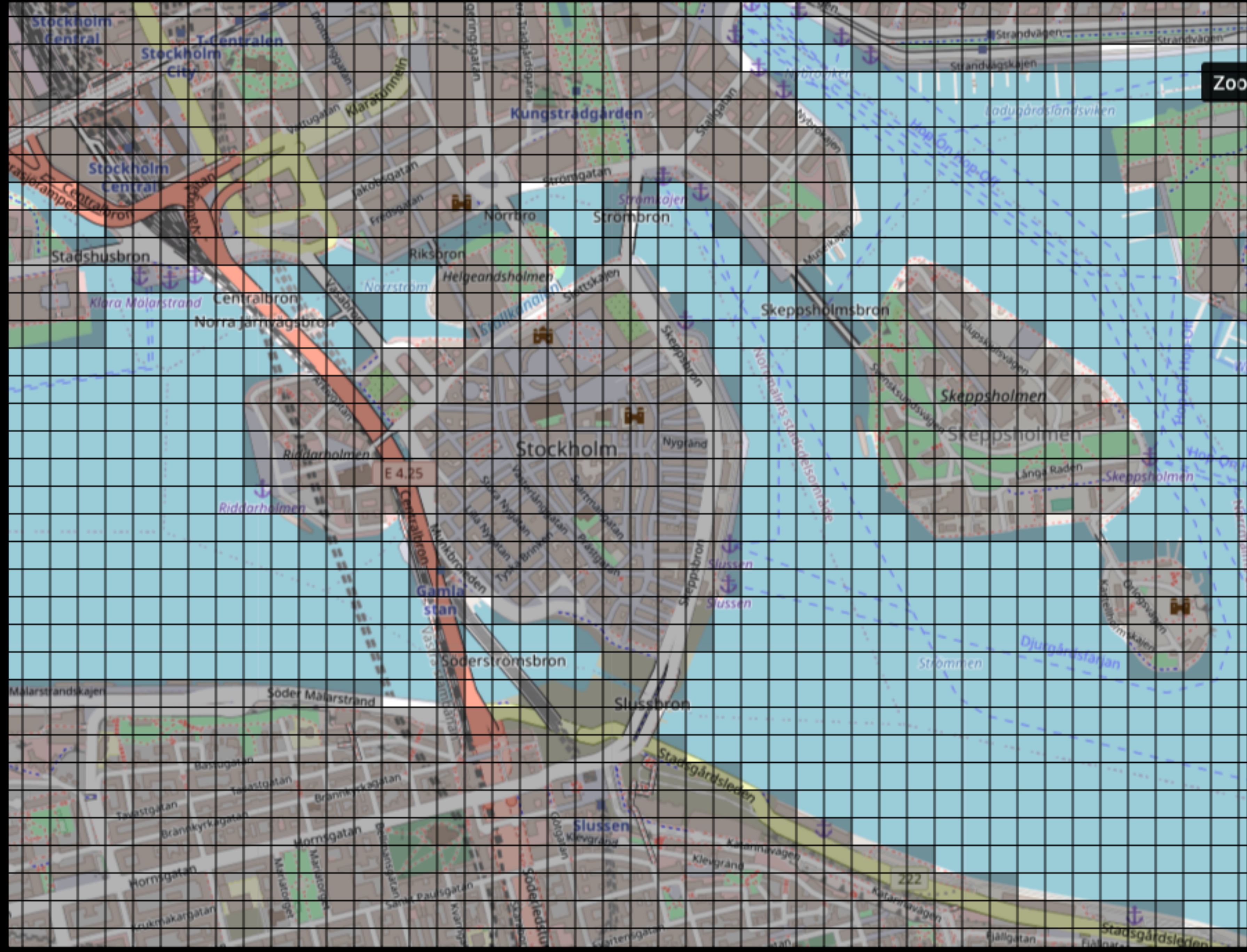


Motivation

- Serve a city with carsharing (CS) stations:
  - Demand in granularity of square cells
  - Customers willing to walk a certain distance
  - Same distance bound for the complete city
  - City ➜ Polyomino (➜ alter the environment)
  - Walking only within the polyomino

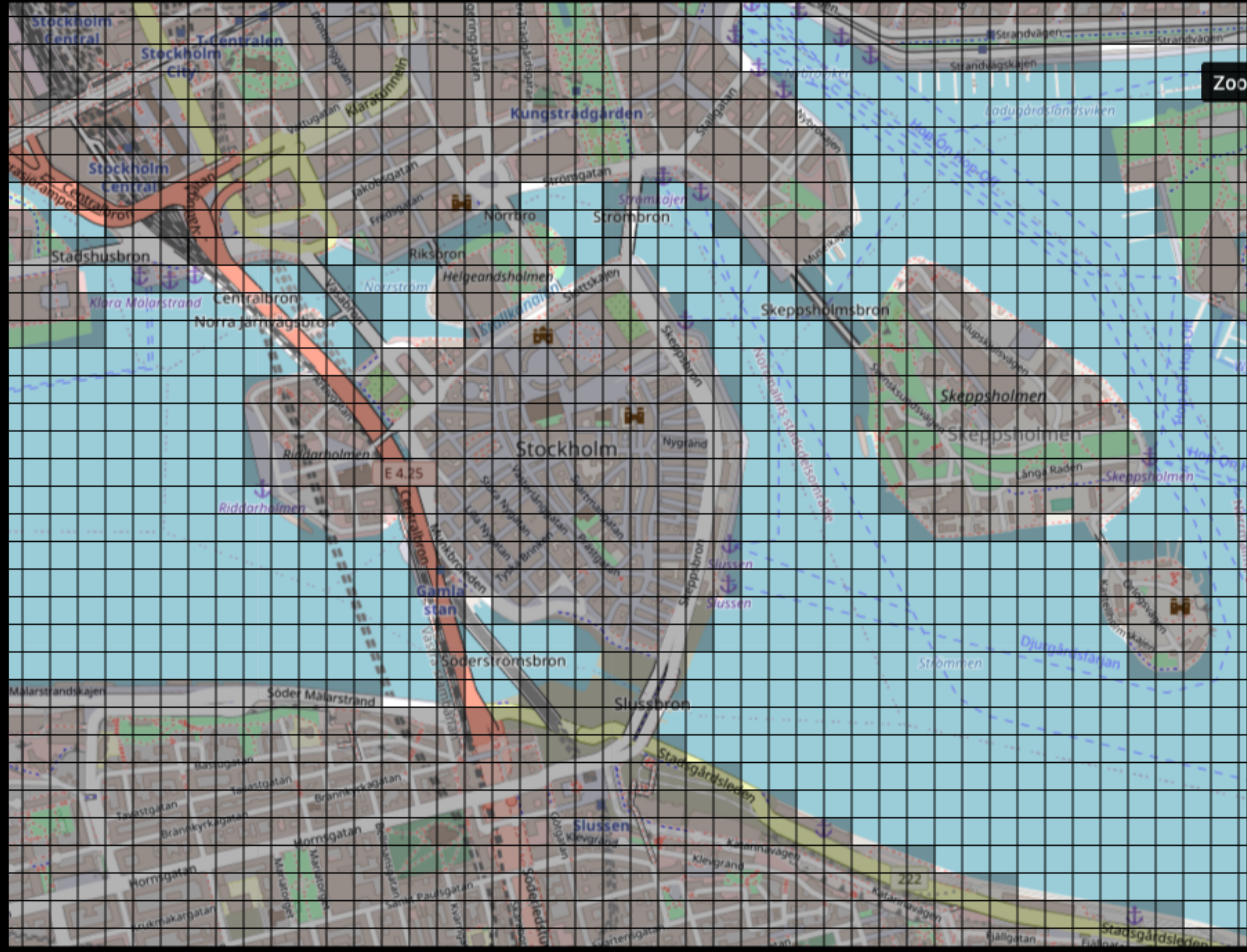


Motivation



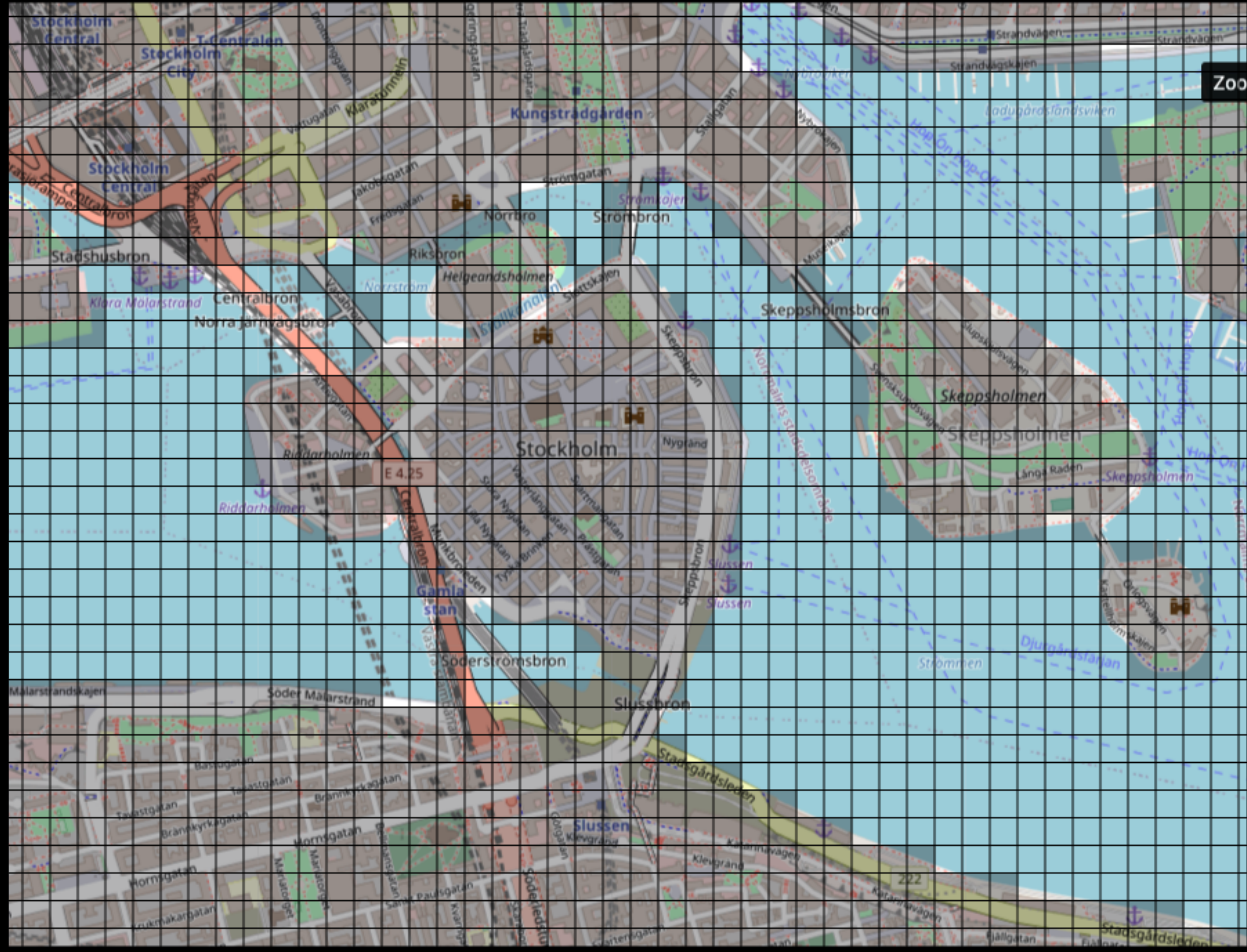
- Serve a city with carsharing (CS) stations:
  - Demand in granularity of square cells
  - Customers willing to walk a certain distance
  - Same distance bound for the complete city
  - City ➜ Polyomino (➜ alter the environment)
  - Walking only within the polyomino
- Goal: Place as few CS stations as possible to serve the complete city

Motivation

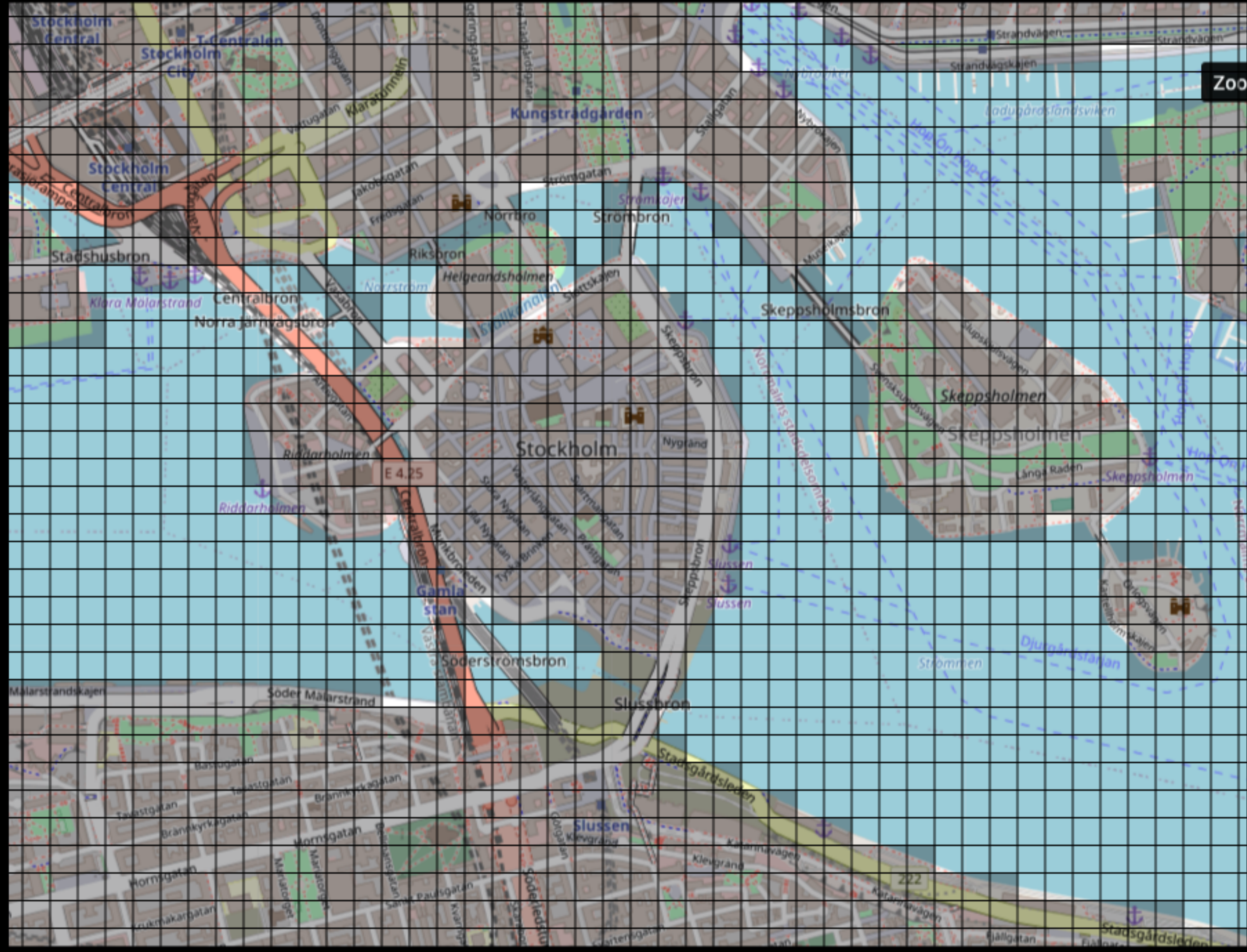

Kungsträdgården
Stadshusbron
Centralbron
Riksbron
Norrbro
Strömbron
Helgeandsholmen
Skeppsholmsbron
Stockholm
Skeppsholmen
Riddarholmen
Söderströmsbron
Slussbron
Slussen

Motivation

- So, what can a station serve?



Motivation



- So, what can a station serve?
  - All unit squares of the polyomino reachable when walking inside the polyomino for at most the given walking range

Motivation

- So, what can a station serve?
  - All unit squares of the polyomino reachable when walking inside the polyomino for at most the given walking range
  - Walking range k

k=5

Motivation

- So, what can a station serve?
  - All unit squares of the polyomino reachable when walking inside the polyomino for at most the given walking range
  - Walking range *k*
  - “Visibility”: We can look around corners for $k \geq 2$

k=5

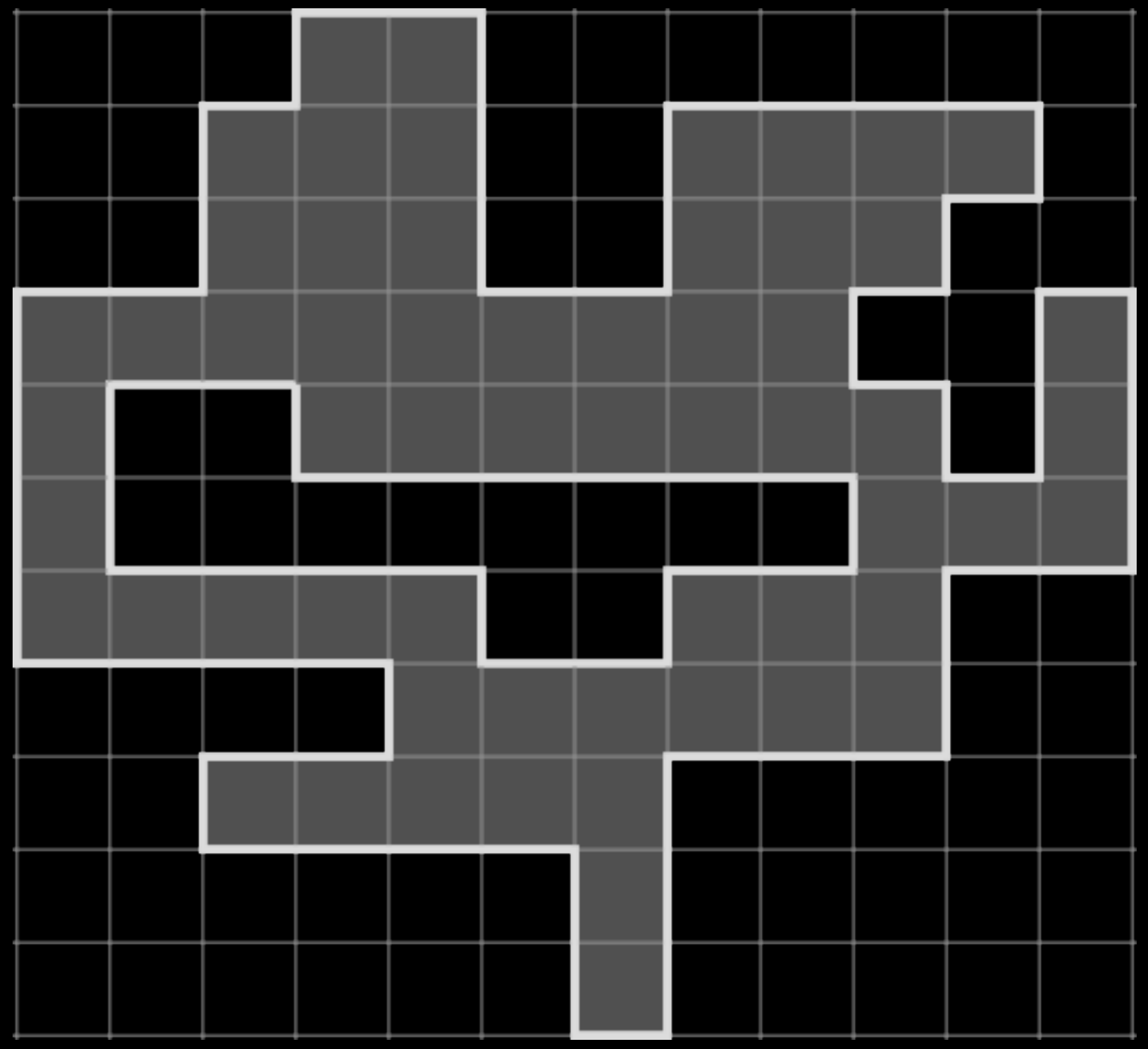
Motivation

k=5

- So, what can a station serve?
  - All unit squares of the polyomino reachable when walking inside the polyomino for at most the given walking range
  - Walking range k
  - "Visibility": We can look around corners for $k \geq 2$
  - ➡ Alter the capabilities of the guard

k-Hop Visibility: Formal Definition

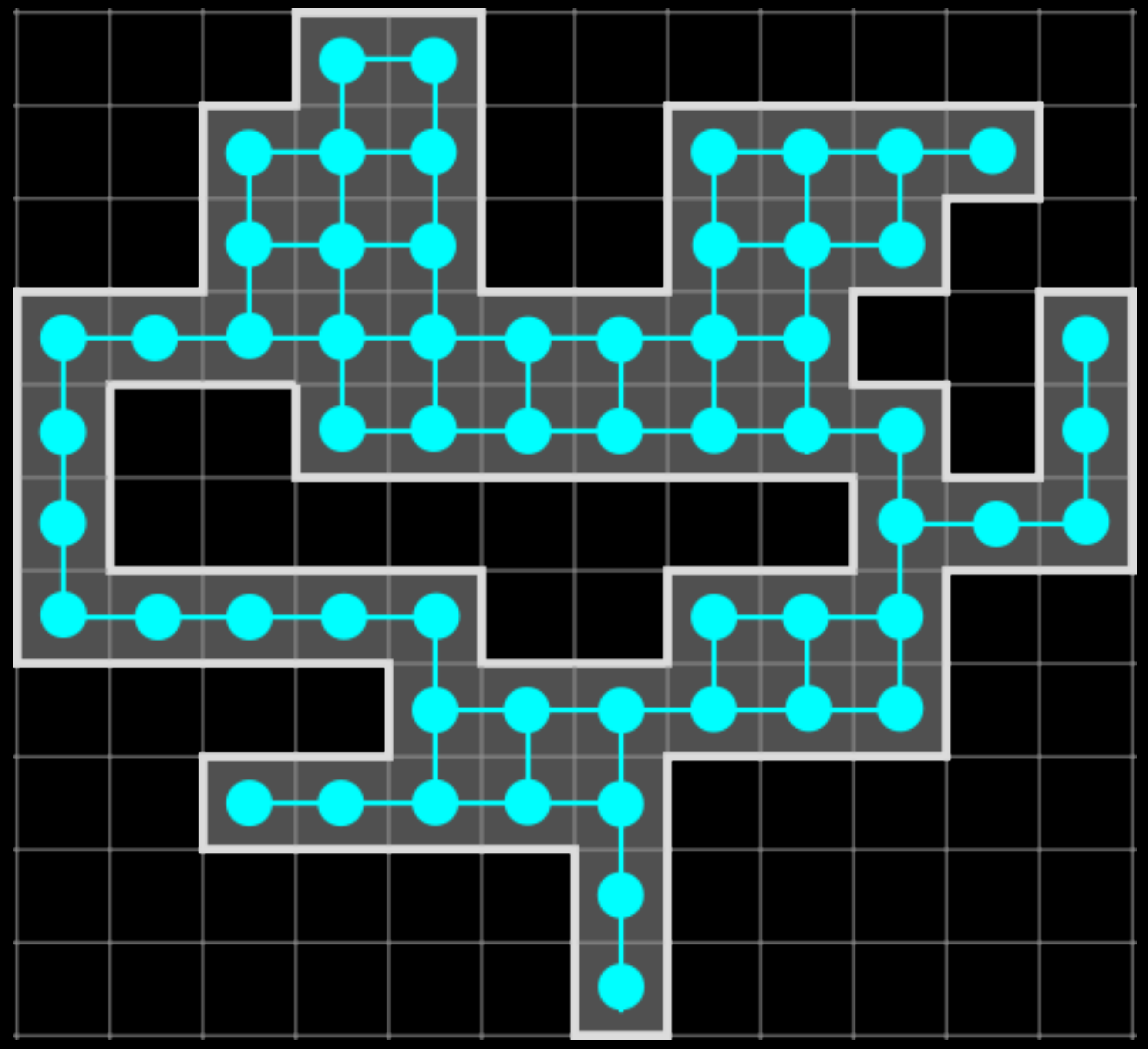
k-Hop Visibility: Formal Definition



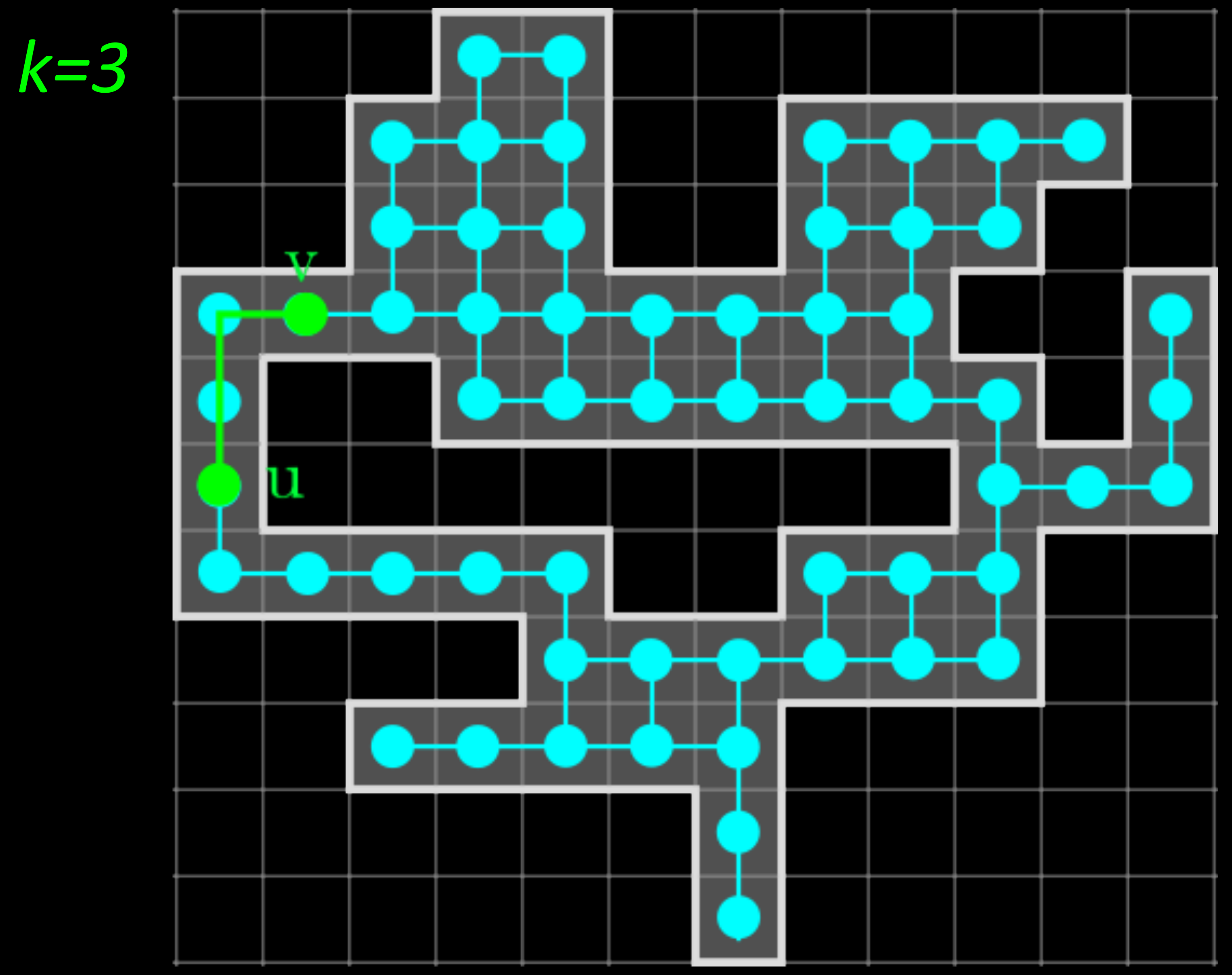
- Polyomino: connected polygon P in plane, formed by joining m unit squares on the square lattice

LINKÖPING UNIVERSITY

k-Hop Visibility: Formal Definition



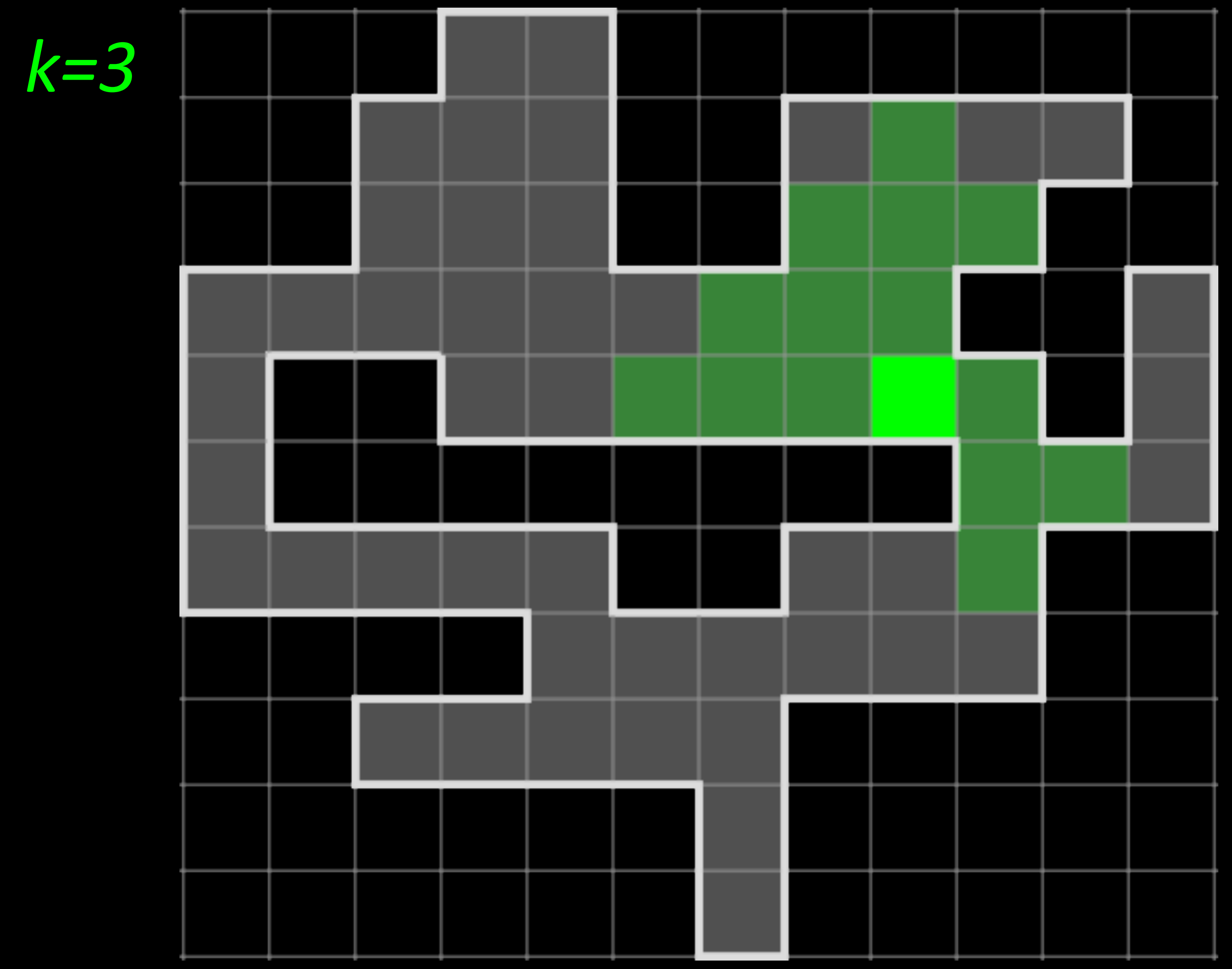
- Polyomino: connected polygon P in plane, formed by joining m unit squares on the square lattice
- Dual graph G_P is a grid graph



k-Hop Visibility: Formal Definition

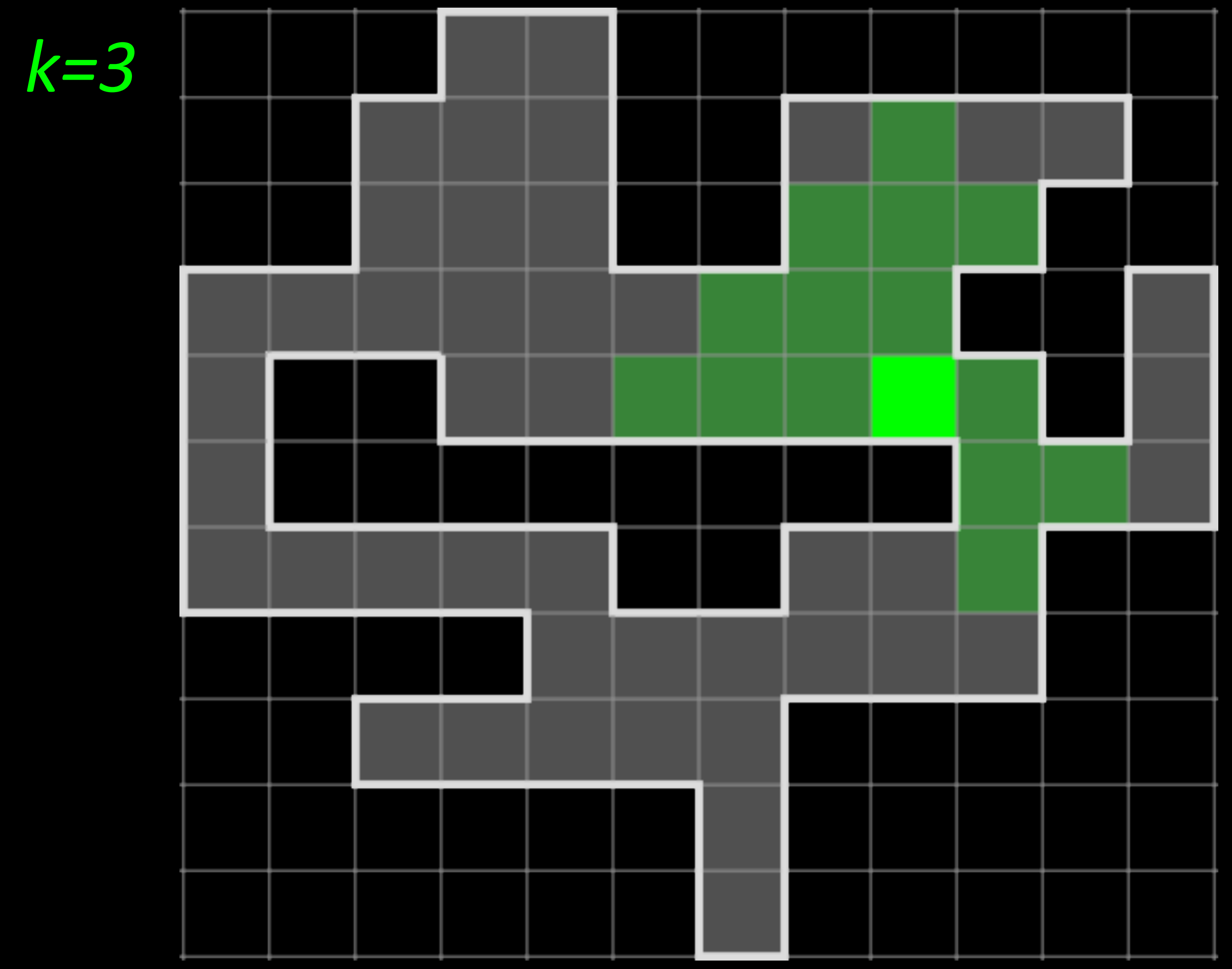
- Polyomino: connected polygon P in plane, formed by joining m unit squares on the square lattice
- Dual graph G_P is a grid graph
- Unit square $v \in P$ *k-hop visible* to unit square $u \in P$, if shortest path from u to v in G_P has length at most k.

k-Hop Visibility: Formal Definition



- Polyomino: connected polygon P in plane, formed by joining m unit squares on the square lattice
- Dual graph G_P is a grid graph
- Unit square $v \in P$ *k-hop visible* to unit square $u \in P$, if shortest path from u to v in G_P has length at most k.

k-Hop Visibility: Formal Definition



- Polyomino: connected polygon *P* in plane, formed by joining *m* unit squares on the square lattice
- Dual graph G_P is a grid graph
- Unit square $v \in P$ *k-hop visible* to unit square $u \in P$, if shortest path from *u* to *v* in G_P has length at most *k*.

Minimum *k*-hop Guarding Problem (M*k*GP)

Given: Polyomino *P*, range *k*

Find: Minimum cardinality unit-square guard cover in *P* under *k*-hop visibility.

Alternative Formulation

Alternative Formulation

Minimum *k*-Hop Dominating Set Problem (M*k*DSP)

Given: Graph G

Alternative Formulation

Minimum *k*-Hop Dominating Set Problem (M*k*DSP)

Given: Graph G

Find: Minimum cardinality $D_k \subseteq V(G)$, each graph vertex connected to vertex in D_k with a path of length at most k.

Alternative Formulation

Minimum *k*-Hop Dominating Set Problem (M*k*DSP)

Given: Graph G

Find: Minimum cardinality $D_k \subseteq V(G)$, each graph vertex connected to vertex in D_k with a path of length at most k.

M*k*DSP is NP-complete in general graphs.

Alternative Formulation

Minimum *k*-Hop Dominating Set Problem (M*k*DSP)

Given: Graph G

Find: Minimum cardinality $D_k \subseteq V(G)$, each graph vertex connected to vertex in D_k with a path of length at most k.

M*k*DSP is NP-complete in general graphs.

➡ We want to solve M*k*DSP in grid graphs

What Do We Know About Guarding Polyominoes + Thin Polygons and About the Minimum k-Hop Dominating Set Problem?

If the dual graph of the partition obtained by extending all polygon edges through incident reflex vertices is a tree

Guarding Polyominoes + Thin Orthogonal Polygons

LINKÖPING UNIVERSITY

[*] Ana Paula Tomás. Guarding thin orthogonal polygons is hard. In Fundamentals of Computation Theory (FCT), pages 305–316, 2013.
[#] Therese C. Biedl and Saeed Mehrabi. On r-guarding thin orthogonal polygons. In International Symposium on Algorithms and Computation (ISAAC), pages 17:1–17:13, 2016.

If the dual graph of the partition obtained by extending all polygon edges through incident reflex vertices is a tree

Guarding Polyominoes + Thin Orthogonal Polygons

- Thin orthogonal polygons, under original definition of visibility, computing minimum guard set [*]:

[*] Ana Paula Tomás. Guarding thin orthogonal polygons is hard. In Fundamentals of Computation Theory (FCT), pages 305–316, 2013.
[#] Therese C. Biedl and Saeed Mehrabi. On r-guarding thin orthogonal polygons. In International Symposium on Algorithms and Computation (ISAAC), pages 17:1–17:13, 2016.

If the dual graph of the partition obtained by extending all polygon edges through incident reflex vertices is a tree

Guarding Polyominoes + Thin Orthogonal Polygons

- Thin orthogonal polygons, under original definition of visibility, computing minimum guard set [*]:
  - NP-hard for point guards

[*] Ana Paula Tomás. Guarding thin orthogonal polygons is hard. In Fundamentals of Computation Theory (FCT), pages 305–316, 2013.
[#] Therese C. Biedl and Saeed Mehrabi. On r-guarding thin orthogonal polygons. In International Symposium on Algorithms and Computation (ISAAC), pages 17:1–17:13, 2016.

If the dual graph of the partition obtained by extending all polygon edges through incident reflex vertices is a tree

Guarding Polyominoes + Thin Orthogonal Polygons

- Thin orthogonal polygons, under original definition of visibility, computing minimum guard set [*]:
  - NP-hard for point guards
  - APX-hard for vertex or boundary guards

LINKÖPING UNIVERSITY

[*] Ana Paula Tomás. Guarding thin orthogonal polygons is hard. In Fundamentals of Computation Theory (FCT), pages 305–316, 2013.
[#] Therese C. Biedl and Saeed Mehrabi. On r-guarding thin orthogonal polygons. In International Symposium on Algorithms and Computation (ISAAC), pages 17:1–17:13, 2016.

If the dual graph of the partition obtained by extending all polygon edges through incident reflex vertices is a tree

Guarding Polyominoes + Thin Orthogonal Polygons

- Thin orthogonal polygons, under original definition of visibility, computing minimum guard set [*]:
  - NP-hard for point guards
  - APX-hard for vertex or boundary guards
- Thin orthogonal polygons, rectilinear visibility, computing minimum guard set [#]:

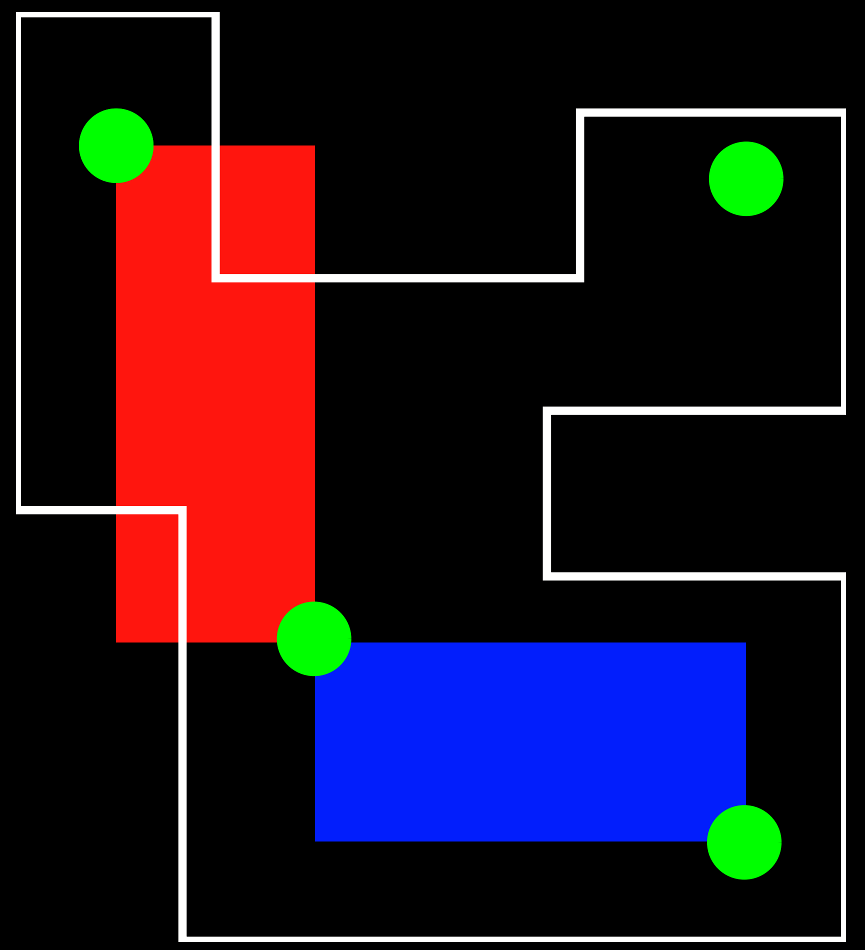
[*] Ana Paula Tomás. Guarding thin orthogonal polygons is hard. In Fundamentals of Computation Theory (FCT), pages 305–316, 2013.
[#] Therese C. Biedl and Saeed Mehrabi. On r-guarding thin orthogonal polygons. In International Symposium on Algorithms and Computation (ISAAC), pages 17:1–17:13, 2016.

If the dual graph of the partition obtained by extending all polygon edges through incident reflex vertices is a tree

Guarding Polyominoes + Thin Orthogonal Polygons

- Thin orthogonal polygons, under original definition of visibility, computing minimum guard set [*]:
  - NP-hard for point guards
  - APX-hard for vertex or boundary guards
- Thin orthogonal polygons, rectilinear visibility, computing minimum guard set [#]:

Rectilinear visibility/ r-visibility:



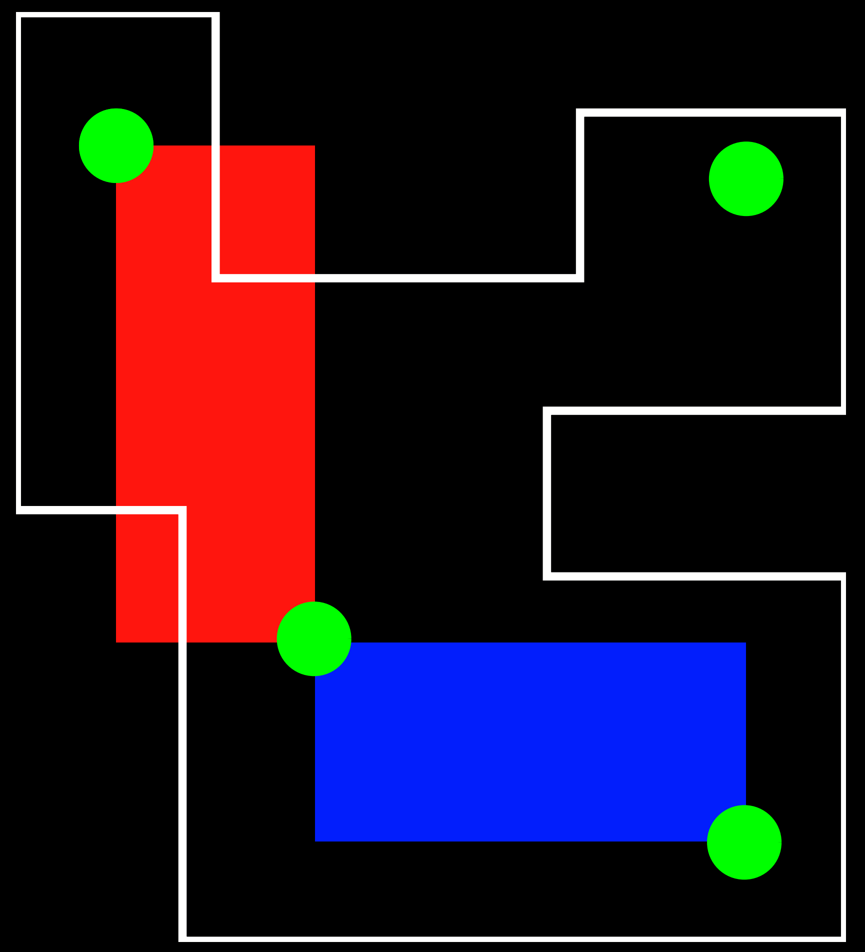
Two points are r-visible to each other if there exists a rectangle in P that contains both points.

[*] Ana Paula Tomás. Guarding thin orthogonal polygons is hard. In Fundamentals of Computation Theory (FCT), pages 305–316, 2013.
[#] Therese C. Biedl and Saeed Mehrabi. On r-guarding thin orthogonal polygons. In International Symposium on Algorithms and Computation (ISAAC), pages 17:1–17:13, 2016.

If the dual graph of the partition obtained by extending all polygon edges through incident reflex vertices is a tree

Guarding Polyominoes + Thin Orthogonal Polygons

- Thin orthogonal polygons, under original definition of visibility, computing minimum guard set [*]:
  - NP-hard for point guards
  - APX-hard for vertex or boundary guards
- Thin orthogonal polygons, rectilinear visibility, computing minimum guard set [#]:
  - NP-hard in polygons with holes

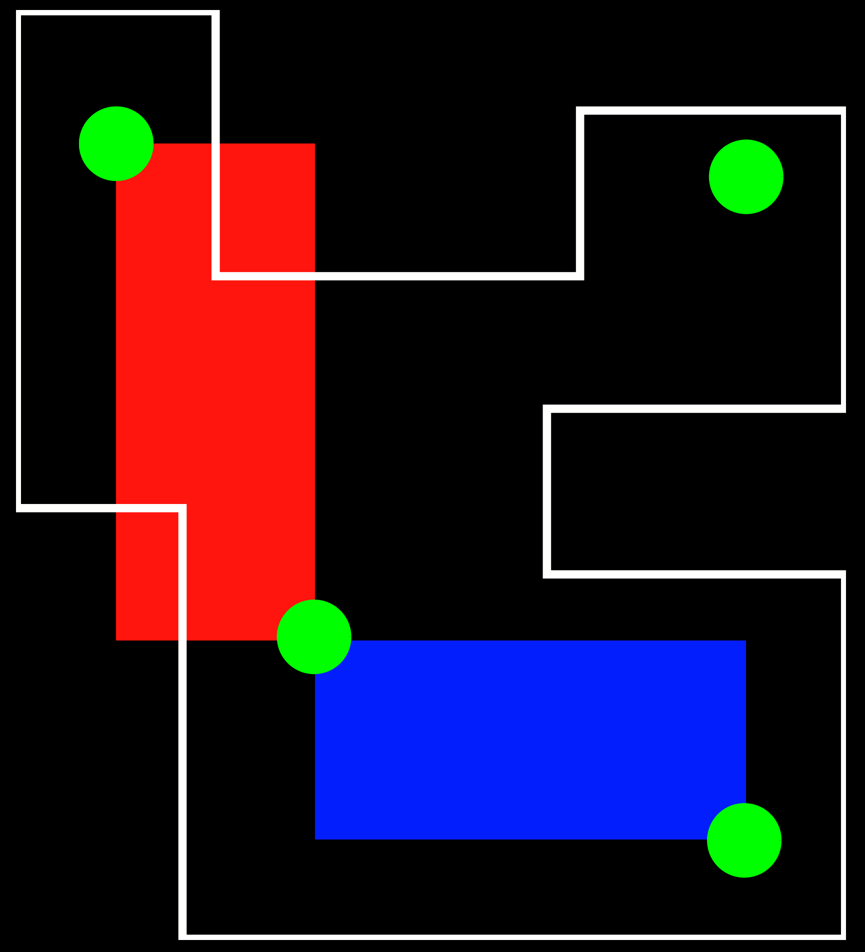


[*] Ana Paula Tomás. Guarding thin orthogonal polygons is hard. In Fundamentals of Computation Theory (FCT), pages 305–316, 2013.
[#] Therese C. Biedl and Saeed Mehrabi. On r-guarding thin orthogonal polygons. In International Symposium on Algorithms and Computation (ISAAC), pages 17:1–17:13, 2016.

If the dual graph of the partition obtained by extending all polygon edges through incident reflex vertices is a tree

Guarding Polyominoes + Thin Orthogonal Polygons

- Thin orthogonal polygons, under original definition of visibility, computing minimum guard set [*]:
  - NP-hard for point guards
  - APX-hard for vertex or boundary guards
- Thin orthogonal polygons, rectilinear visibility, computing minimum guard set [#]:
  - NP-hard in polygons with holes
  - Linear-time algorithm for tree polygons

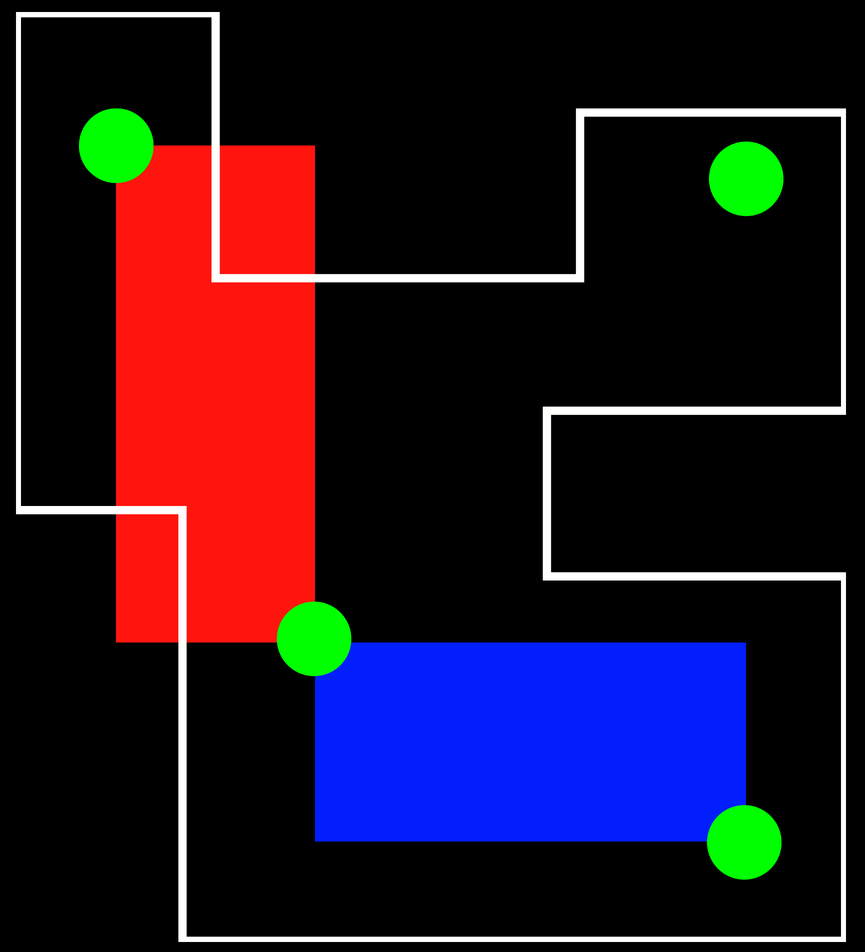


[*] Ana Paula Tomás. Guarding thin orthogonal polygons is hard. In Fundamentals of Computation Theory (FCT), pages 305–316, 2013.
[#] Therese C. Biedl and Saeed Mehrabi. On r-guarding thin orthogonal polygons. In International Symposium on Algorithms and Computation (ISAAC), pages 17:1–17:13, 2016.

If the dual graph of the partition obtained by extending all polygon edges through incident reflex vertices is a tree

Guarding Polyominoes + Thin Orthogonal Polygons

- Thin orthogonal polygons, under original definition of visibility, computing minimum guard set [*]:
  - NP-hard for point guards
  - APX-hard for vertex or boundary guards
- Thin orthogonal polygons, rectilinear visibility, computing minimum guard set [#]:
  - NP-hard in polygons with holes
  - Linear-time algorithm for tree polygons
  - Generalizes to polygons with h holes or thickness t (dual graph does not contain an induced $(t+1)$x$(t+1)$ grid)➜ fixed-parameter tractable in $t+h$



[*] Ana Paula Tomás. Guarding thin orthogonal polygons is hard. In Fundamentals of Computation Theory (FCT), pages 305–316, 2013.
[#] Therese C. Biedl and Saeed Mehrabi. On r-guarding thin orthogonal polygons. In International Symposium on Algorithms and Computation (ISAAC), pages 17:1–17:13, 2016.

Guarding Polyominoes + Thin Orthogonal Polygons

[*] Therese C. Biedl and Saeed Mehrabi. On orthogonally guarding orthogonal polygons with bounded treewidth. Algorithmica, 83(2):641–666, 2021.
[#] Chris Worman and J. Mark Keil. Polygon decomposition and the orthogonal art gallery problem. International Journal of Computational Geometry & Applications, 17(2):105–138, 2007.

Guarding Polyominoes + Thin Orthogonal Polygons

- Orthogonal polygons with bounded treewidth [*]:

[*] Therese C. Biedl and Saeed Mehrabi. On orthogonally guarding orthogonal polygons with bounded treewidth. Algorithmica, 83(2):641–666, 2021.
[#] Chris Worman and J. Mark Keil. Polygon decomposition and the orthogonal art gallery problem. International Journal of Computational Geometry & Applications, 17(2):105–138, 2007.

Guarding Polyominoes + Thin Orthogonal Polygons

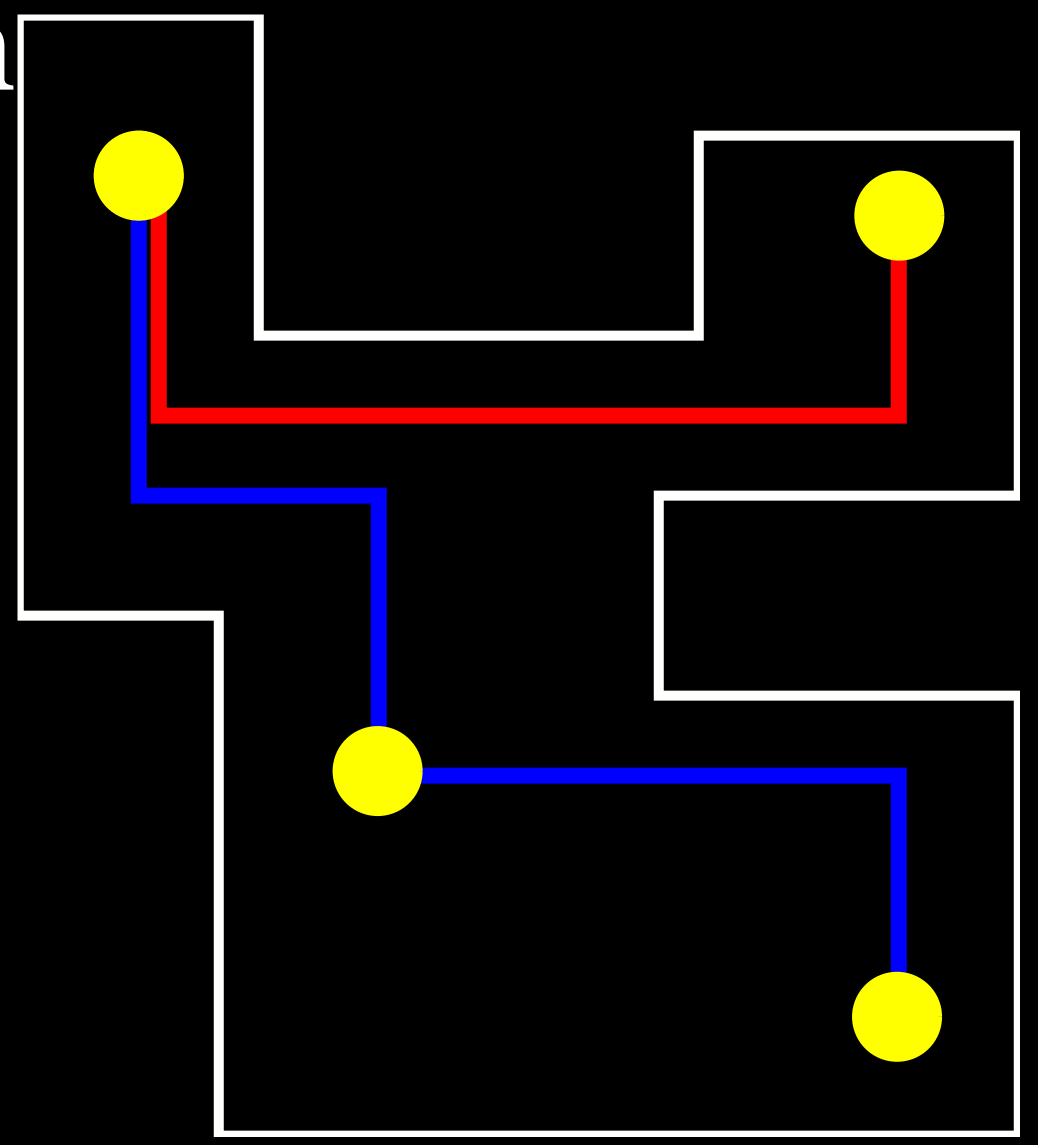
- Orthogonal polygons with bounded treewidth [*]:
  - For rectilinear visibility, staircase visibility, limited-turn path visibility

[*] Therese C. Biedl and Saeed Mehrabi. On orthogonally guarding orthogonal polygons with bounded treewidth. Algorithmica, 83(2):641–666, 2021.
[#] Chris Worman and J. Mark Keil. Polygon decomposition and the orthogonal art gallery problem. International Journal of Computational Geometry & Applications, 17(2):105–138, 2007.

Guarding Polyominoes + Thin Orthogonal Polygons

- Orthogonal polygons with bounded treewidth [*]:
  - For rectilinear visibility, staircase visibility, limited-turn path visibility

Staircase visibility/ s-visiblity:

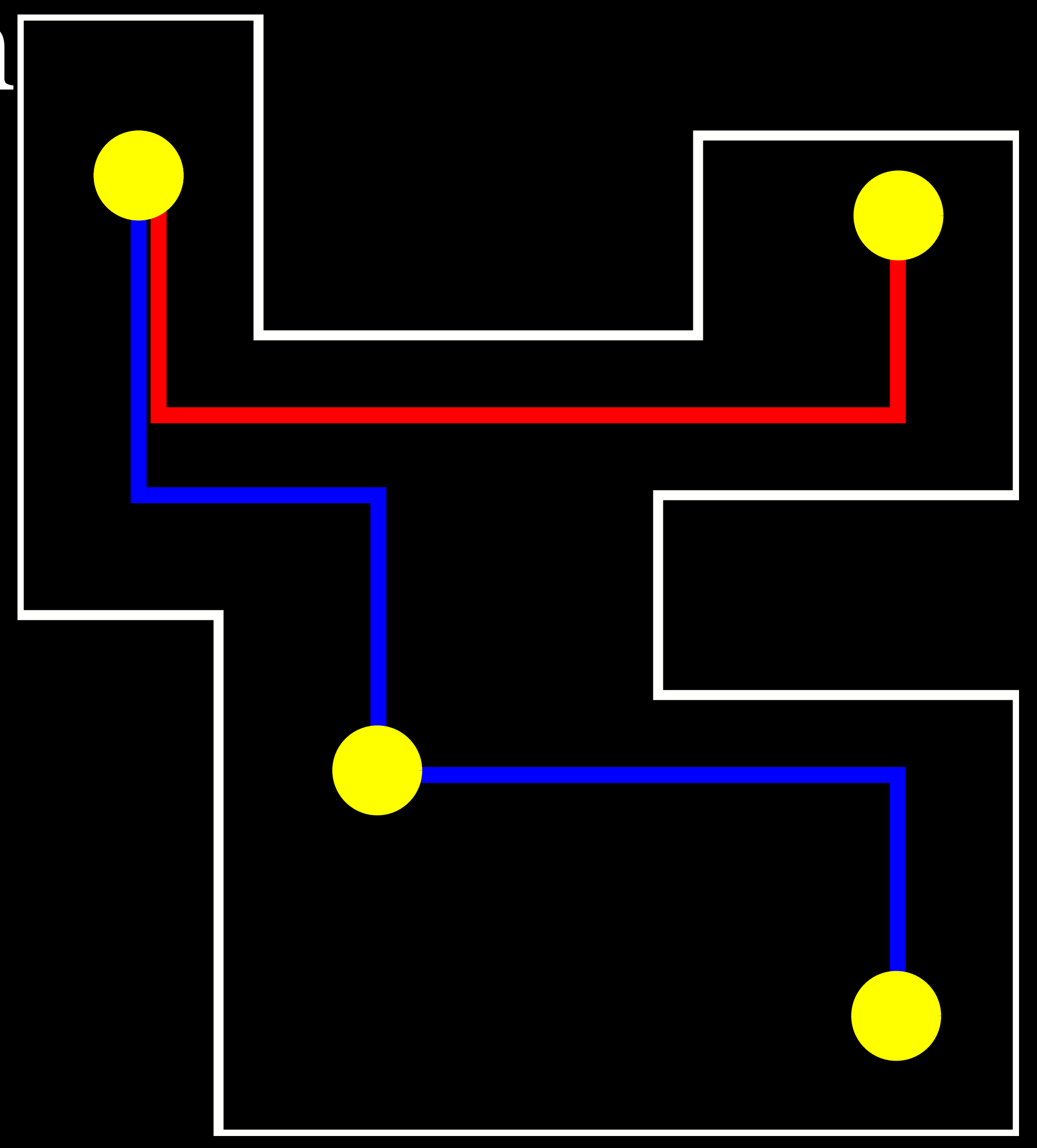


Two points are s-visible to each other if there exists a staircase path in P that connects them.

[*] Therese C. Biedl and Saeed Mehrabi. On orthogonally guarding orthogonal polygons with bounded treewidth. Algorithmica, 83(2):641–666, 2021.
[#] Chris Worman and J. Mark Keil. Polygon decomposition and the orthogonal art gallery problem. International Journal of Computational Geometry & Applications, 17(2):105–138, 2007.

Guarding Polyominoes + Thin Orthogonal Polygons

- Orthogonal polygons with bounded treewidth [*]:
  - For rectilinear visibility, staircase visibility, limited-turn path visibility
  - Linear-time solvable

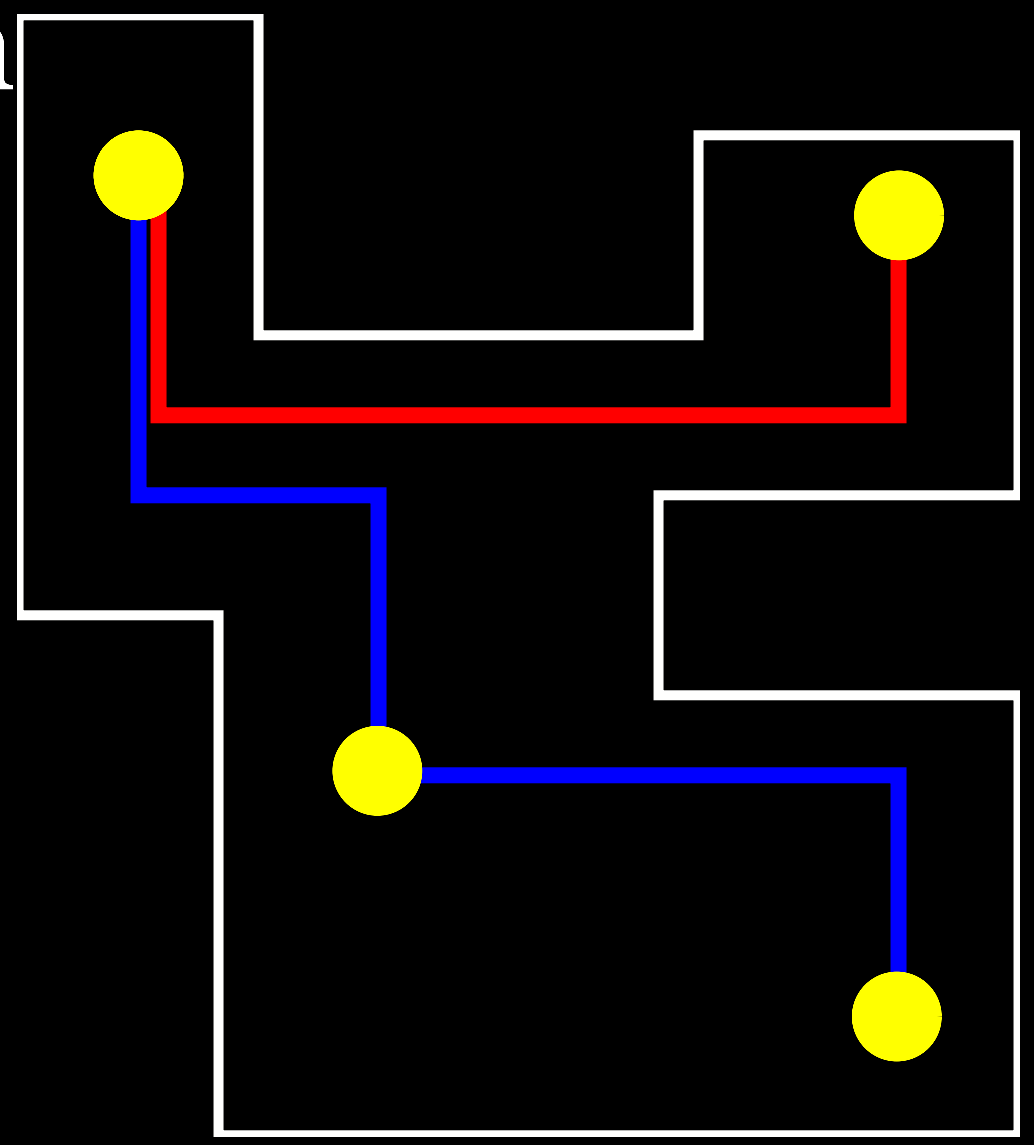


[*] Therese C. Biedl and Saeed Mehrabi. On orthogonally guarding orthogonal polygons with bounded treewidth. Algorithmica, 83(2):641–666, 2021.

[#] Chris Worman and J. Mark Keil. Polygon decomposition and the orthogonal art gallery problem. International Journal of Computational Geometry & Applications, 17(2):105–138, 2007.

Guarding Polyominoes + Thin Orthogonal Polygons

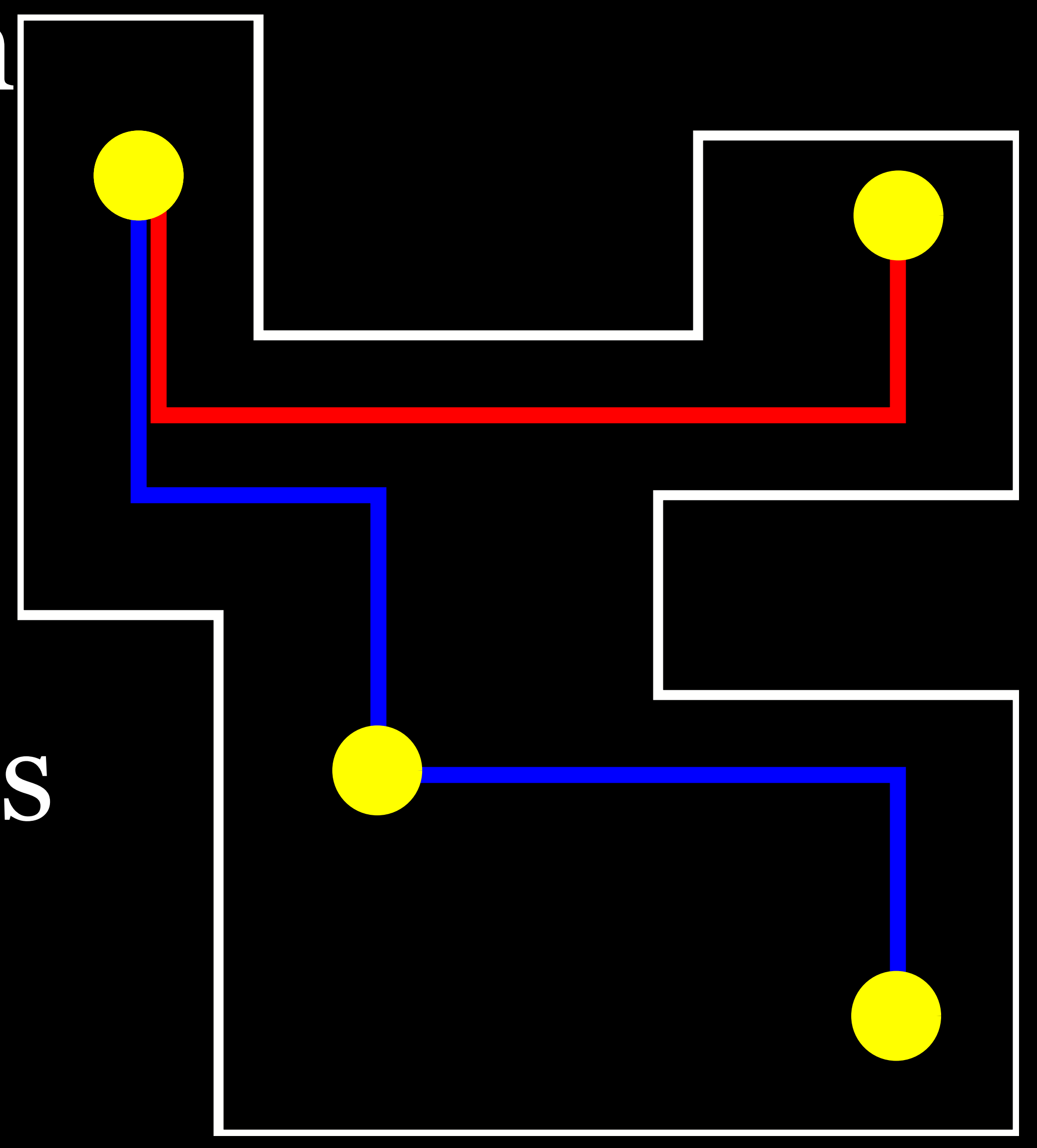
- Orthogonal polygons with bounded treewidth [*]:
  - For rectilinear visibility, staircase visibility, limited-turn path visibility
  - Linear-time solvable
- Rectilinear visibility [#]:



[*] Therese C. Biedl and Saeed Mehrabi. On orthogonally guarding orthogonal polygons with bounded treewidth. Algorithmica, 83(2):641–666, 2021.
[#] Chris Worman and J. Mark Keil. Polygon decomposition and the orthogonal art gallery problem. International Journal of Computational Geometry & Applications, 17(2):105–138, 2007.

Guarding Polyominoes + Thin Orthogonal Polygons

- Orthogonal polygons with bounded treewidth [*]:
  - For rectilinear visibility, staircase visibility, limited-turn path visibility
  - Linear-time solvable
- Rectilinear visibility [#]:
  - Used that underlying graph is perfect + this problem variant is equivalent to clique cover

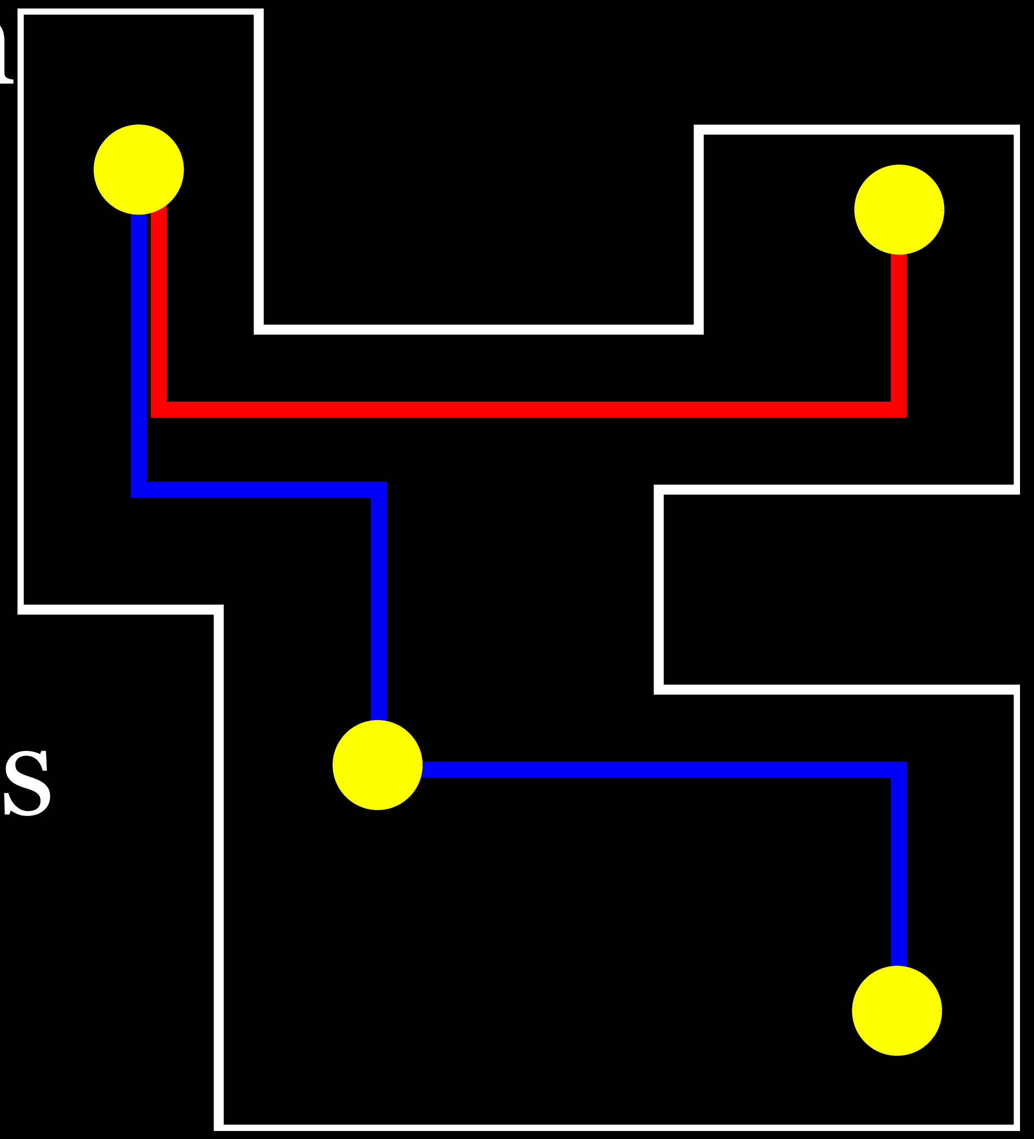


[*] Therese C. Biedl and Saeed Mehrabi. On orthogonally guarding orthogonal polygons with bounded treewidth. Algorithmica, 83(2):641–666, 2021.
[#] Chris Worman and J. Mark Keil. Polygon decomposition and the orthogonal art gallery problem. International Journal of Computational Geometry & Applications, 17(2):105–138, 2007.

Guarding Polyominoes + Thin Orthogonal Polygons

- Orthogonal polygons with bounded treewidth [*]:
  - For rectilinear visibility, staircase visibility, limited-turn path visibility
  - Linear-time solvable
- Rectilinear visibility [#]:
  - Used that underlying graph is perfect + this problem variant is equivalent to clique cover

  ➡ Linear-time algorithm



[*] Therese C. Biedl and Saeed Mehrabi. On orthogonally guarding orthogonal polygons with bounded treewidth. Algorithmica, 83(2):641–666, 2021.
[#] Chris Worman and J. Mark Keil. Polygon decomposition and the orthogonal art gallery problem. International Journal of Computational Geometry & Applications, 17(2):105–138, 2007.

Guarding Polyominoes + Thin Orthogonal Polygons

[*] Therese C. Biedl, Mohammad Tanvir Irfan, Justin Iwerks, Joondong Kim, and Joseph S. B. Mitchell. Guarding polyominoes. In Symp.Computational Geom., p. 387–396, 2011
[#] Chuzo Iwamoto and Toshihiko Kume. Computational complexity of the r-visibility guard set problem for polyominoes. In Japanese Conference on Discrete and Computational Geometry and Graphs, pages 87–95, 2013
[+] Val Pinciu. Guarding polyominoes, polycubes and polyhypercubes. Electronic Notes in Discrete Mathematics, 49:159–166, 2015

Unit square visible from a guard if and only if the guard sees all points of the unit square under ordinary visibility

Guarding Polyominoes + Thin Orthogonal Polygons

- Simple polyominoes, all-or-nothing visibility and ordinary visibility[*]: NP-hard

[*] Therese C. Biedl, Mohammad Tanvir Irfan, Justin Iwerks, Joondong Kim, and Joseph S. B. Mitchell. Guarding polyominoes. In Symp.Computational Geom., p. 387–396, 2011
[#] Chuzo Iwamoto and Toshihiko Kume. Computational complexity of the r-visibility guard set problem for polyominoes. In Japanese Conference on Discrete and Computational Geometry and Graphs, pages 87–95, 2013
[+] Val Pinciu. Guarding polyominoes, polycubes and polyhypercubes. Electronic Notes in Discrete Mathematics, 49:159–166, 2015

Unit square visible from a guard if and only if the guard sees all points of the unit square under ordinary visibility

Guarding Polyominoes + Thin Orthogonal Polygons

- Simple polyominoes, all-or-nothing visibility and ordinary visibility[*]: NP-hard
- Thin polyominoes (dual is tree), all-or-nothing visibility with limited visibility range [*]: polytime algorithm

[*] Therese C. Biedl, Mohammad Tanvir Irfan, Justin Iwerks, Joondong Kim, and Joseph S. B. Mitchell. Guarding polyominoes. In Symp.Computational Geom., p. 387–396, 2011

[#] Chuzo Iwamoto and Toshihiko Kume. Computational complexity of the r-visibility guard set problem for polyominoes. In Japanese Conference on Discrete and Computational Geometry and Graphs, pages 87–95, 2013

[+] Val Pinciu. Guarding polyominoes, polycubes and polyhypercubes. Electronic Notes in Discrete Mathematics, 49:159–166, 2015

Unit square visible from a guard if and only if the guard sees all points of the unit square under ordinary visibility

Guarding Polyominoes + Thin Orthogonal Polygons

- Simple polyominoes, all-or-nothing visibility and ordinary visibility[*]: NP-hard
- Thin polyominoes (dual is tree), all-or-nothing visibility with limited visibility range [*]: polytime algorithm
- Polyominoes with holes, rectilinear visibility [#]: NP-hard

[*] Therese C. Biedl, Mohammad Tanvir Irfan, Justin Iwerks, Joondong Kim, and Joseph S. B. Mitchell. Guarding polyominoes. In Symp.Computational Geom., p. 387–396, 2011
[#] Chuzo Iwamoto and Toshihiko Kume. Computational complexity of the r-visibility guard set problem for polyominoes. In Japanese Conference on Discrete and Computational Geometry and Graphs, pages 87–95, 2013
[+] Val Pinciu. Guarding polyominoes, polycubes and polyhypercubes. Electronic Notes in Discrete Mathematics, 49:159–166, 2015

Unit square visible from a guard if and only if the guard sees all points of the unit square under ordinary visibility

Guarding Polyominoes + Thin Orthogonal Polygons

- Simple polyominoes, all-or-nothing visibility and ordinary visibility[*]: NP-hard
- Thin polyominoes (dual is tree), all-or-nothing visibility with limited visibility range [*]: polytime algorithm
- Polyominoes with holes, rectilinear visibility [#]: NP-hard
- Generalized to polycubes [+]

[*] Therese C. Biedl, Mohammad Tanvir Irfan, Justin Iwerks, Joondong Kim, and Joseph S. B. Mitchell. Guarding polyominoes. In Symp.Computational Geom., p. 387–396, 2011
[#] Chuzo Iwamoto and Toshihiko Kume. Computational complexity of the r-visibility guard set problem for polyominoes. In Japanese Conference on Discrete and Computational Geometry and Graphs, pages 87–95, 2013
[+] Val Pinciu. Guarding polyominoes, polycubes and polyhypercubes. Electronic Notes in Discrete Mathematics, 49:159–166, 2015

Minimum k-Hop Dominating Set Problem

[*] Alan D. Amis, Ravi Prakash, Dung T. Huynh, and Thai H.P. Vuong. Max-min d-cluster formation in wireless ad hoc networks. In Conference on Computer Communications, pages 32–41, 2000
[#] Partha Basuchowdhuri and Subhashis Majumder. Finding influential nodes in social networks using minimum k-hop dominating set. In International Conference on Applied Algorithms (ICAA), pages 137–151, 2014
[+] Sukhamay Kundu and Subhashis Majumder. A linear time algorithm for optimal k-hop dominating set of a tree. Information Processing Letters, 116(2):197–202, 2016
[**] A. Karim Abu-Affash, Paz Carmi, and Adi Krasin. A linear-time algorithm for minimum k-hop dominating set of a cactus graph. Discrete Applied Mathematics, 320:488–499, 2022
[##] Glencora Borradaile and Hung Le. Optimal dynamic program for r-domination problems over tree decompositions. In International Symposium on Parameterized and Exact Computation (IPEC), pages 8:1–8:23, 2017.
[++] Erik D. Demaine, Fedor V. Fomin, Mohammad Taghi Hajiaghayi, and Dimitrios M. Thilikos. Fixed-parameter algorithms for (k, r)-center in planar graphs and map graphs. ACM Transactions on Algorithms, 1(1):33–47, 2005

Minimum k-Hop Dominating Set Problem

- NP-complete in general graphs [*,#]

[*] Alan D. Amis, Ravi Prakash, Dung T. Huynh, and Thai H.P. Vuong. Max-min d-cluster formation in wireless ad hoc networks. In Conference on Computer Communications, pages 32–41, 2000
[#] Partha Basuchowdhuri and Subhashis Majumder. Finding influential nodes in social networks using minimum k-hop dominating set. In International Conference on Applied Algorithms (ICAA), pages 137–151, 2014
[+] Sukhamay Kundu and Subhashis Majumder. A linear time algorithm for optimal k-hop dominating set of a tree. Information Processing Letters, 116(2):197–202, 2016
[**] A. Karim Abu-Affash, Paz Carmi, and Adi Krasin. A linear-time algorithm for minimum k-hop dominating set of a cactus graph. Discrete Applied Mathematics, 320:488–499, 2022
[##] Glencora Borradaile and Hung Le. Optimal dynamic program for r-domination problems over tree decompositions. In International Symposium on Parameterized and Exact Computation (IPEC), pages 8:1–8:23, 2017.
[++] Erik D. Demaine, Fedor V. Fomin, Mohammad Taghi Hajiaghayi, and Dimitrios M. Thilikos. Fixed-parameter algorithms for (k, r)-center in planar graphs and map graphs. ACM Transactions on Algorithms, 1(1):33–47, 2005

Minimum k-Hop Dominating Set Problem

- NP-complete in general graphs [*,#]
- Trees [+]: can be solved in linear time

[*] Alan D. Amis, Ravi Prakash, Dung T. Huynh, and Thai H.P. Vuong. Max-min d-cluster formation in wireless ad hoc networks. In Conference on Computer Communications, pages 32–41, 2000
[#] Partha Basuchowdhuri and Subhashis Majumder. Finding influential nodes in social networks using minimum k-hop dominating set. In International Conference on Applied Algorithms (ICAA), pages 137–151, 2014
[+] Sukhamay Kundu and Subhashis Majumder. A linear time algorithm for optimal k-hop dominating set of a tree. Information Processing Letters, 116(2):197–202, 2016
[**] A. Karim Abu-Affash, Paz Carmi, and Adi Krasin. A linear-time algorithm for minimum k-hop dominating set of a cactus graph. Discrete Applied Mathematics, 320:488–499, 2022
[##] Glencora Borradaile and Hung Le. Optimal dynamic program for r-domination problems over tree decompositions. In International Symposium on Parameterized and Exact Computation (IPEC), pages 8:1–8:23, 2017.
[++] Erik D. Demaine, Fedor V. Fomin, Mohammad Taghi Hajiaghayi, and Dimitrios M. Thilikos. Fixed-parameter algorithms for (k, r)-center in planar graphs and map graphs. ACM Transactions on Algorithms, 1(1):33–47, 2005

Minimum k-Hop Dominating Set Problem

- NP-complete in general graphs [*,#]
- Trees [+]: can be solved in linear time
- Recently [**]: simplified algorithm for trees + linear-time algorithm for cactus graphs

[*] Alan D. Amis, Ravi Prakash, Dung T. Huynh, and Thai H.P. Vuong. Max-min d-cluster formation in wireless ad hoc networks. In Conference on Computer Communications, pages 32–41, 2000
[#] Partha Basuchowdhuri and Subhashis Majumder. Finding influential nodes in social networks using minimum k-hop dominating set. In International Conference on Applied Algorithms (ICAA), pages 137–151, 2014
[+] Sukhamay Kundu and Subhashis Majumder. A linear time algorithm for optimal k-hop dominating set of a tree. Information Processing Letters, 116(2):197–202, 2016
[**] A. Karim Abu-Affash, Paz Carmi, and Adi Krasin. A linear-time algorithm for minimum k-hop dominating set of a cactus graph. Discrete Applied Mathematics, 320:488–499, 2022
[##] Glencora Borradaile and Hung Le. Optimal dynamic program for r-domination problems over tree decompositions. In International Symposium on Parameterized and Exact Computation (IPEC), pages 8:1–8:23, 2017.
[++] Erik D. Demaine, Fedor V. Fomin, Mohammad Taghi Hajiaghayi, and Dimitrios M. Thilikos. Fixed-parameter algorithms for (k, r)-center in planar graphs and map graphs. ACM Transactions on Algorithms, 1(1):33–47, 2005

Minimum k-Hop Dominating Set Problem

- NP-complete in general graphs [*,#]
- Trees [+]: can be solved in linear time
- Recently [**]: simplified algorithm for trees + linear-time algorithm for cactus graphs
- Graphs with treewidth *tw* [##]: dynamic program runs in $O((2k+1)^{tw}\, n)$

[*] Alan D. Amis, Ravi Prakash, Dung T. Huynh, and Thai H.P. Vuong. Max-min d-cluster formation in wireless ad hoc networks. In Conference on Computer Communications, pages 32–41, 2000
[#] Partha Basuchowdhuri and Subhashis Majumder. Finding influential nodes in social networks using minimum k-hop dominating set. In International Conference on Applied Algorithms (ICAA), pages 137–151, 2014
[+] Sukhamay Kundu and Subhashis Majumder. A linear time algorithm for optimal k-hop dominating set of a tree. Information Processing Letters, 116(2):197–202, 2016
[**] A. Karim Abu-Affash, Paz Carmi, and Adi Krasin. A linear-time algorithm for minimum k-hop dominating set of a cactus graph. Discrete Applied Mathematics, 320:488–499, 2022
[##] Glencora Borradaile and Hung Le. Optimal dynamic program for r-domination problems over tree decompositions. In International Symposium on Parameterized and Exact Computation (IPEC), pages 8:1–8:23, 2017.
[++] Erik D. Demaine, Fedor V. Fomin, Mohammad Taghi Hajiaghayi, and Dimitrios M. Thilikos. Fixed-parameter algorithms for (k, r)-center in planar graphs and map graphs. ACM Transactions on Algorithms, 1(1):33–47, 2005

Minimum k-Hop Dominating Set Problem

- NP-complete in general graphs [*,#]
- Trees [+]: can be solved in linear time
- Recently [**]: simplified algorithm for trees + linear-time algorithm for cactus graphs
- Graphs with treewidth tw [##]: dynamic program runs in $O((2k+1)^{tw}\, n)$
- Decision version (dominating set of cardinality ℓ?) [++]: fixed-parameter tractable

[*] Alan D. Amis, Ravi Prakash, Dung T. Huynh, and Thai H.P. Vuong. Max-min d-cluster formation in wireless ad hoc networks. In Conference on Computer Communications, pages 32–41, 2000

[#] Partha Basuchowdhuri and Subhashis Majumder. Finding influential nodes in social networks using minimum k-hop dominating set. In International Conference on Applied Algorithms (ICAA), pages 137–151, 2014

[+] Sukhamay Kundu and Subhashis Majumder. A linear time algorithm for optimal k-hop dominating set of a tree. Information Processing Letters, 116(2):197–202, 2016

[**] A. Karim Abu-Affash, Paz Carmi, and Adi Krasin. A linear-time algorithm for minimum k-hop dominating set of a cactus graph. Discrete Applied Mathematics, 320:488–499, 2022

[##] Glencora Borradaile and Hung Le. Optimal dynamic program for r-domination problems over tree decompositions. In International Symposium on Parameterized and Exact Computation (IPEC), pages 8:1–8:23, 2017.

[++] Erik D. Demaine, Fedor V. Fomin, Mohammad Taghi Hajiaghayi, and Dimitrios M. Thilikos. Fixed-parameter algorithms for (k, r)-center in planar graphs and map graphs. ACM Transactions on Algorithms, 1(1):33–47, 2005

Minimum k-Hop Dominating Set Problem

- NP-complete in general graphs [*,#]
- Trees [+]: can be solved in linear time
- Recently [**]: simplified algorithm for trees + linear-time algorithm for cactus graphs
- Graphs with treewidth tw [##]: dynamic program runs in $O((2k+1)^{tw}\, n)$
- Decision version (dominating set of cardinality ℓ?) [++]: fixed-parameter tractable
- Also: versions where the edges of the graph are weighted

[*] Alan D. Amis, Ravi Prakash, Dung T. Huynh, and Thai H.P. Vuong. Max-min d-cluster formation in wireless ad hoc networks. In Conference on Computer Communications, pages 32–41, 2000

[#] Partha Basuchowdhuri and Subhashis Majumder. Finding influential nodes in social networks using minimum k-hop dominating set. In International Conference on Applied Algorithms (ICAA), pages 137–151, 2014

[+] Sukhamay Kundu and Subhashis Majumder. A linear time algorithm for optimal k-hop dominating set of a tree. Information Processing Letters, 116(2):197–202, 2016

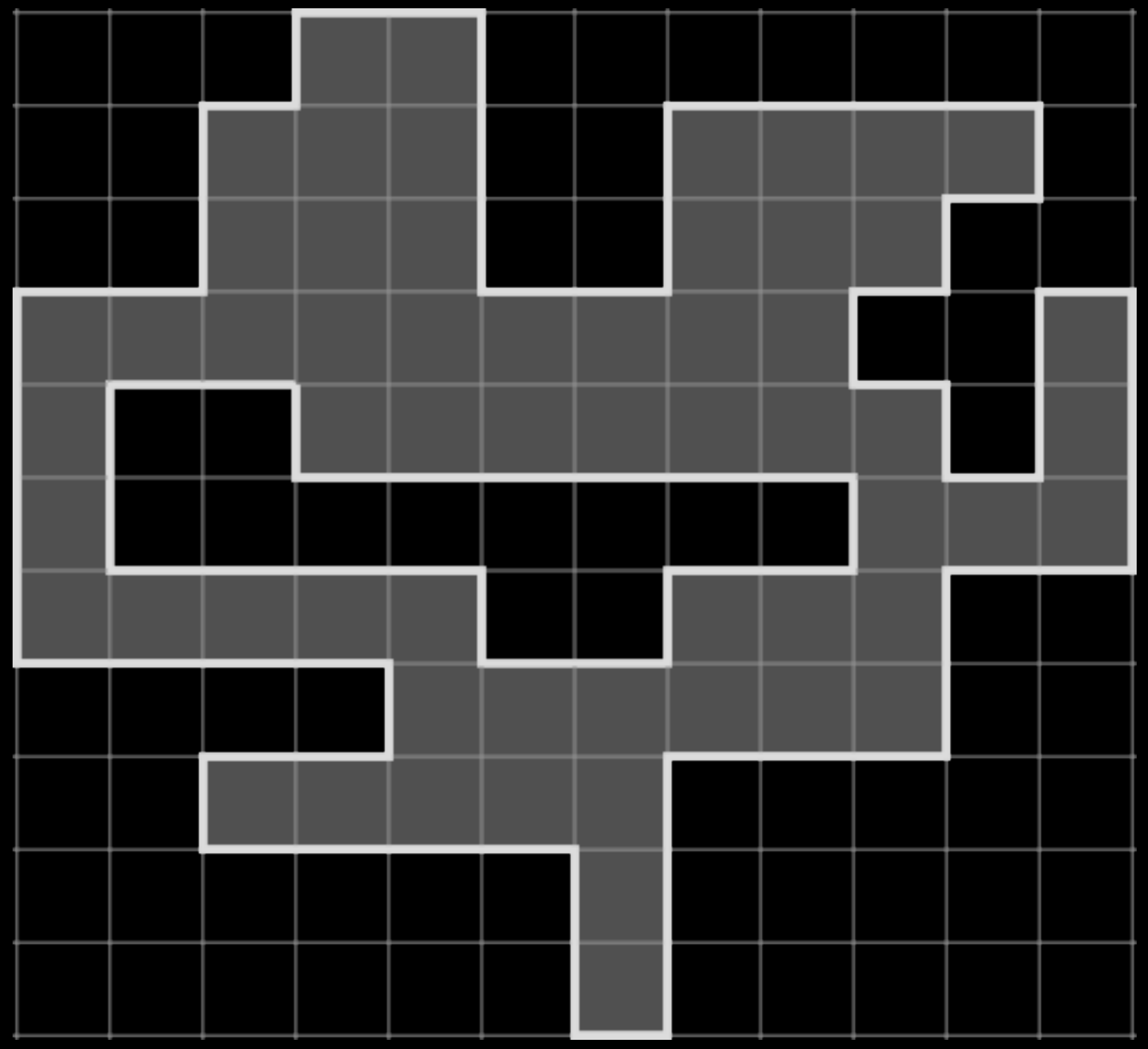
[**] A. Karim Abu-Affash, Paz Carmi, and Adi Krasin. A linear-time algorithm for minimum k-hop dominating set of a cactus graph. Discrete Applied Mathematics, 320:488–499, 2022

[##] Glencora Borradaile and Hung Le. Optimal dynamic program for r-domination problems over tree decompositions. In International Symposium on Parameterized and Exact Computation (IPEC), pages 8:1–8:23, 2017.

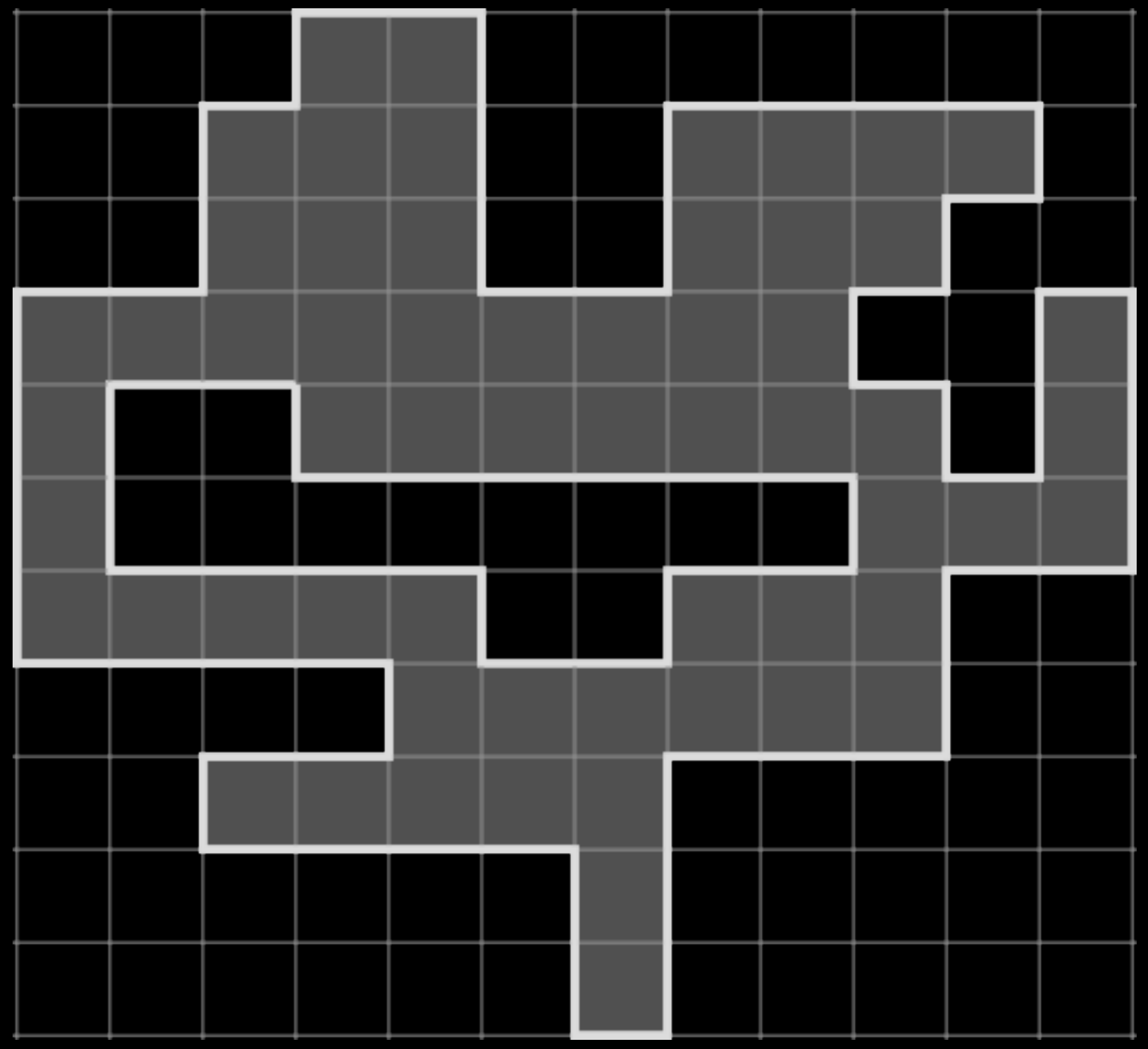
[++] Erik D. Demaine, Fedor V. Fomin, Mohammad Taghi Hajiaghayi, and Dimitrios M. Thilikos. Fixed-parameter algorithms for (k, r)-center in planar graphs and map graphs. ACM Transactions on Algorithms, 1(1):33–47, 2005

Some More Definitions

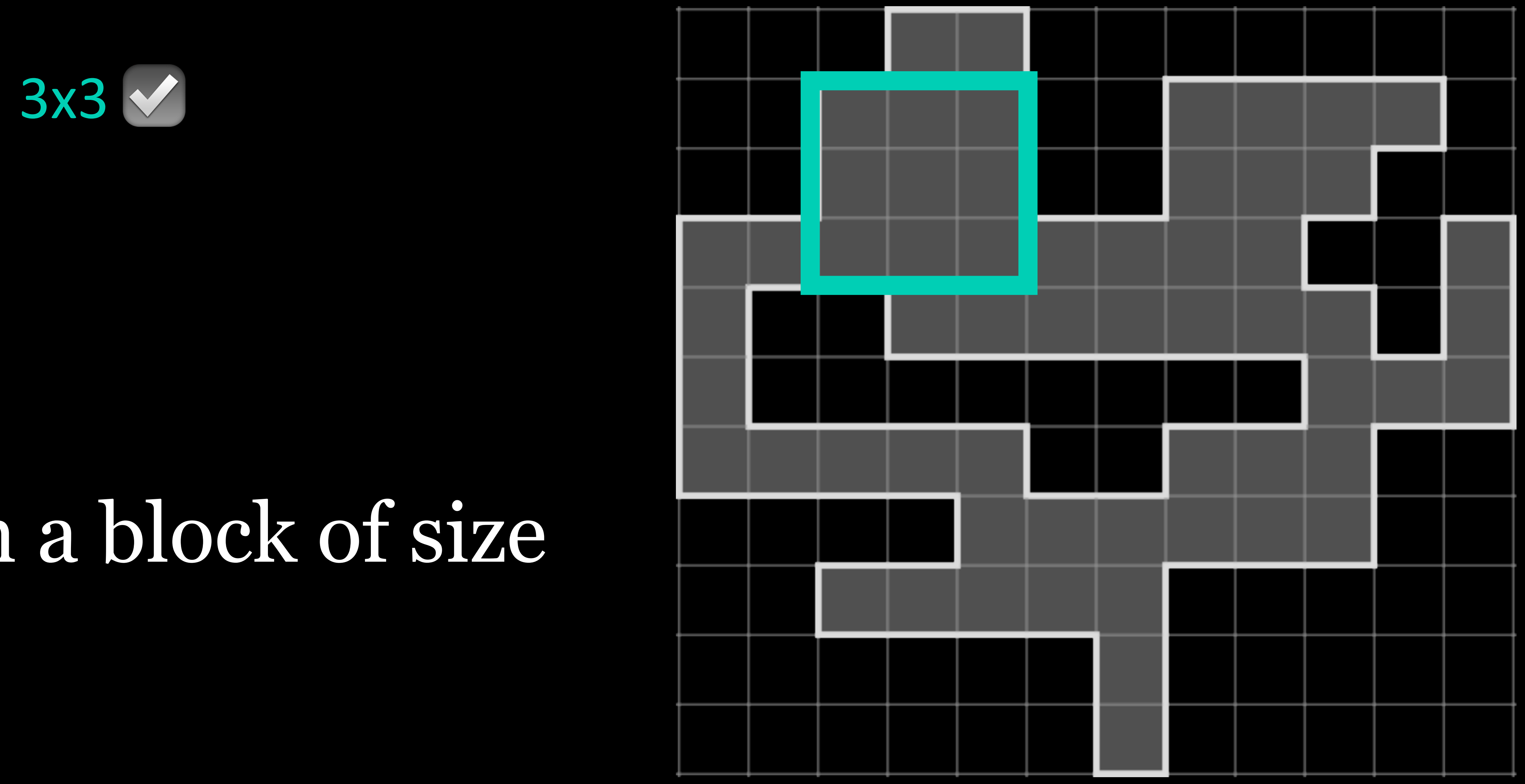
Definitions



Definitions



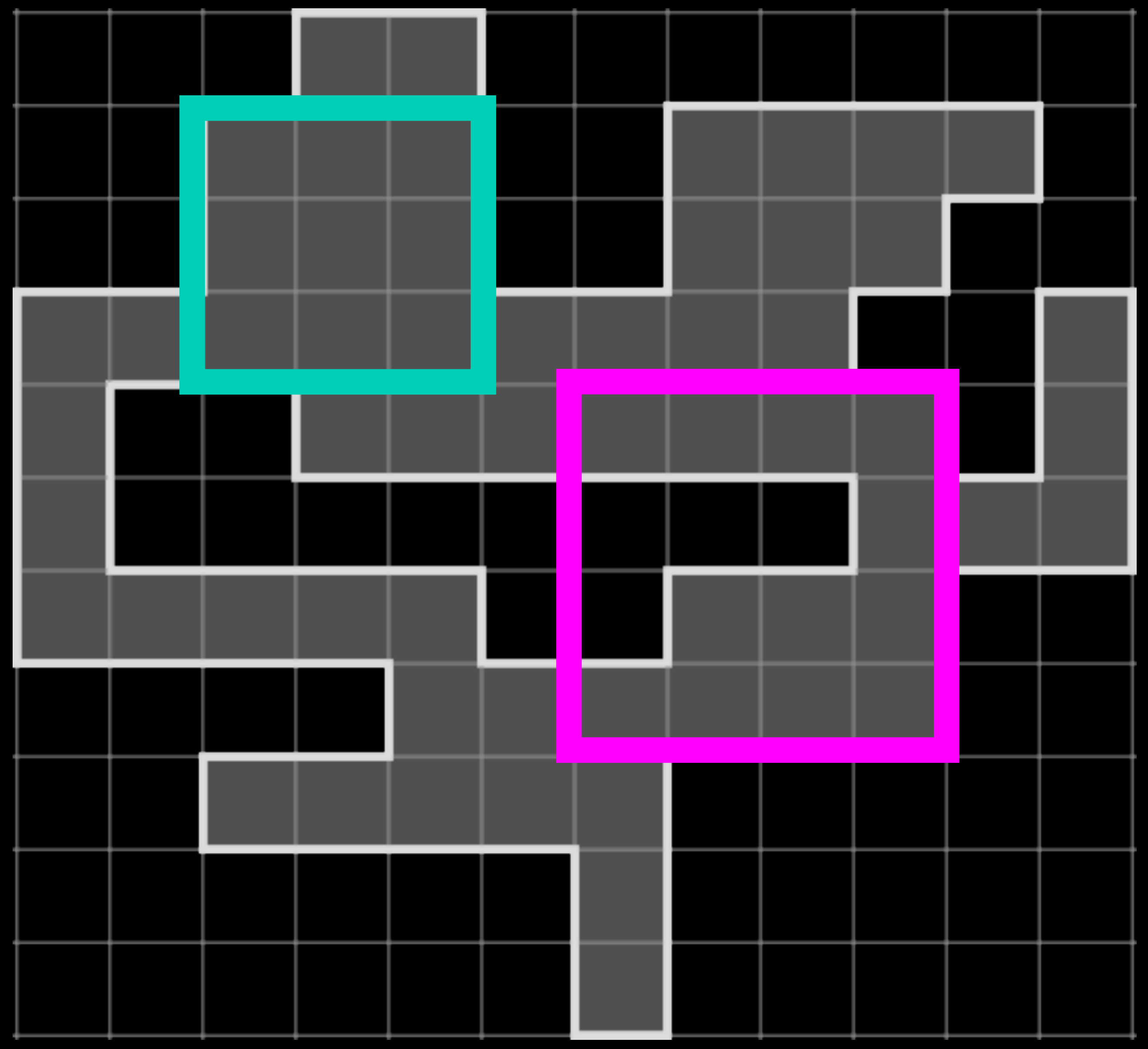
- Polyomino is *t-thin* if it does not contain a block of size (t+1)x(t+1)



Definitions

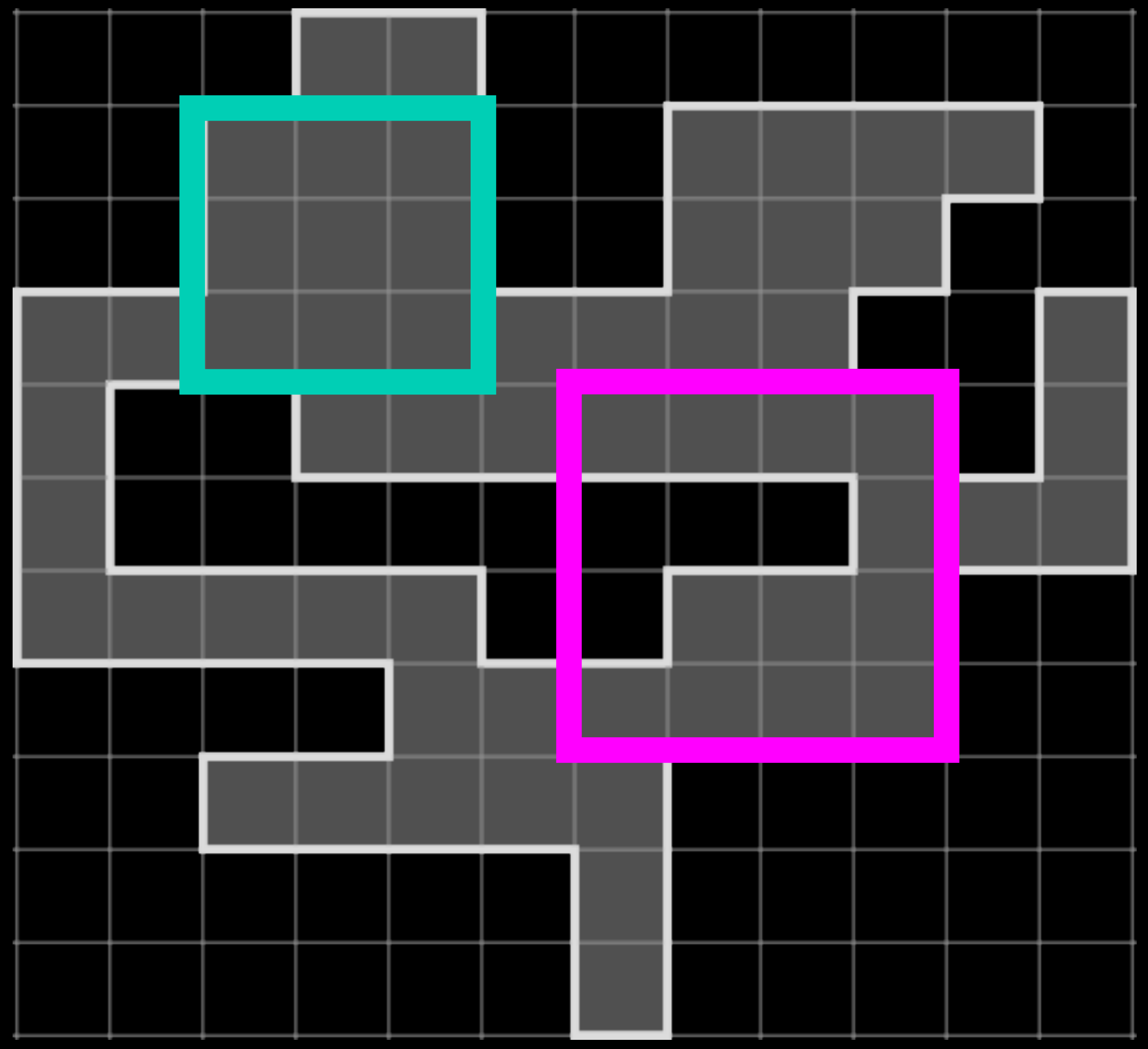
- Polyomino is *t-thin* if it does not contain a block of size (t+1)x(t+1)

Definitions



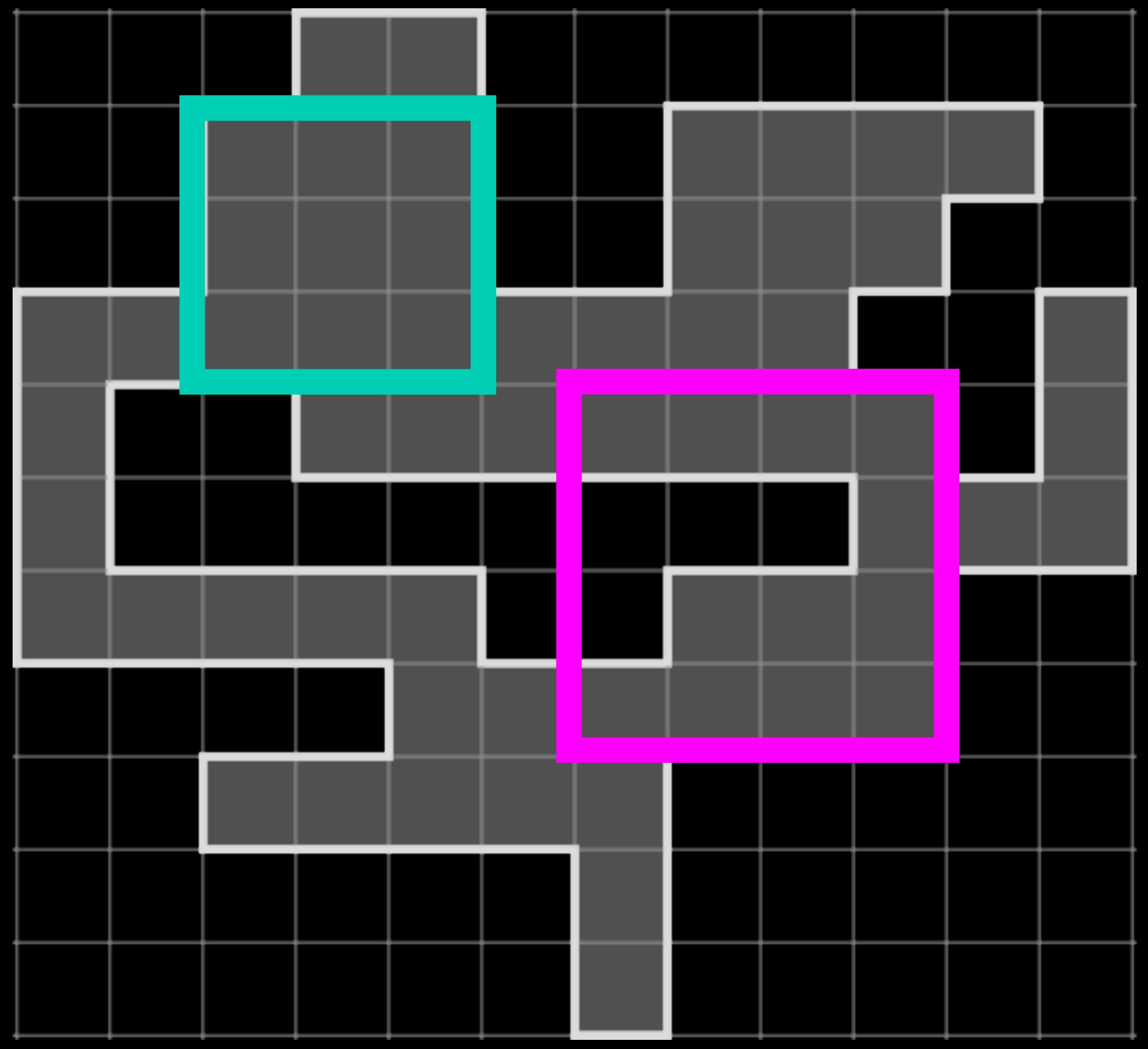
- Polyomino is *t-thin* if it does not contain a block of size (t+1)x(t+1)

Definitions



- Polyomino is *t-thin* if it does not contain a block of size (t+1)x(t+1)
- Analogously: grid graph G is *t-thin* if the polyomino for which G is the dual graph is t-thin

Definitions



- Polyomino is *t-thin* if it does not contain a block of size (t+1)x(t+1)
- Analogously: grid graph G is *t-thin* if the polyomino for which G is the dual graph is t-thin
- Polyomino is simple if it has no holes (every inner face in dual grid graph has unit area)

VC Dimension

What Is the VC Dimension?

What Is the VC Dimension?

- Defined by Vapnik and Chervonenkis

What Is the VC Dimension?

- Defined by Vapnik and Chervonenkis
- Measure of complexity of a set system

What Is the VC Dimension?

- Defined by Vapnik and Chervonenkis
- Measure of complexity of a set system
- For us:

What Is the VC Dimension?

- Defined by Vapnik and Chervonenkis
- Measure of complexity of a set system
- For us:

  Finite set of unit square (guards) D in a polyomino P is *shattered* if for any of the $2^{|D|}$ subsets $D_i \subseteq D$, there exists a unit square $u \in P$:

What Is the VC Dimension?

- Defined by Vapnik and Chervonenkis
- Measure of complexity of a set system
- For us:

  Finite set of unit square (guards) D in a polyomino P is *shattered* if for any of the $2^{|D|}$ subsets $D_i \subseteq D$, there exists a unit square $u \in P$:

  - Every unit square in D_i is k-hop visible from u (or: from every unit square in D_i unit square u is k-hop visible)

What Is the VC Dimension?

- Defined by Vapnik and Chervonenkis
- Measure of complexity of a set system
- For us:

  Finite set of unit square (guards) D in a polyomino P is *shattered* if for any of the $2^{|D|}$ subsets $D_i \subseteq D$, there exists a unit square $u \in P$:

  - Every unit square in D_i is k-hop visible from u (or: from every unit square in D_i unit square u is k-hop visible)
  - No unit square in $D \setminus D_i$ is k-hop visible from u (or: from no unit square in $D \setminus D_i$ unit square u is k-hop visible)

What Is the VC Dimension?

- Defined by Vapnik and Chervonenkis
- Measure of complexity of a set system
- For us:

  Finite set of unit square (guards) D in a polyomino P is *shattered* if for any of the $2^{|D|}$ subsets $D_i \subseteq D$, there exists a unit square $u \in P$:

  - Every unit square in D_i is k-hop visible from u (or: from every unit square in D_i unit square u is k-hop visible)
  - No unit square in $D \setminus D_i$ is k-hop visible from u (or: from no unit square in $D \setminus D_i$ unit square u is k-hop visible)

  Unit square u is a *viewpoint*.

What Is the VC Dimension?

- Defined by Vapnik and Chervonenkis
- Measure of complexity of a set system
- For us:

Finite set of unit square (guards) D in a polyomino P is *shattered* if for any of the $2^{|D|}$ subsets $D_i \subseteq D$, there exists a unit square $u \in P$:

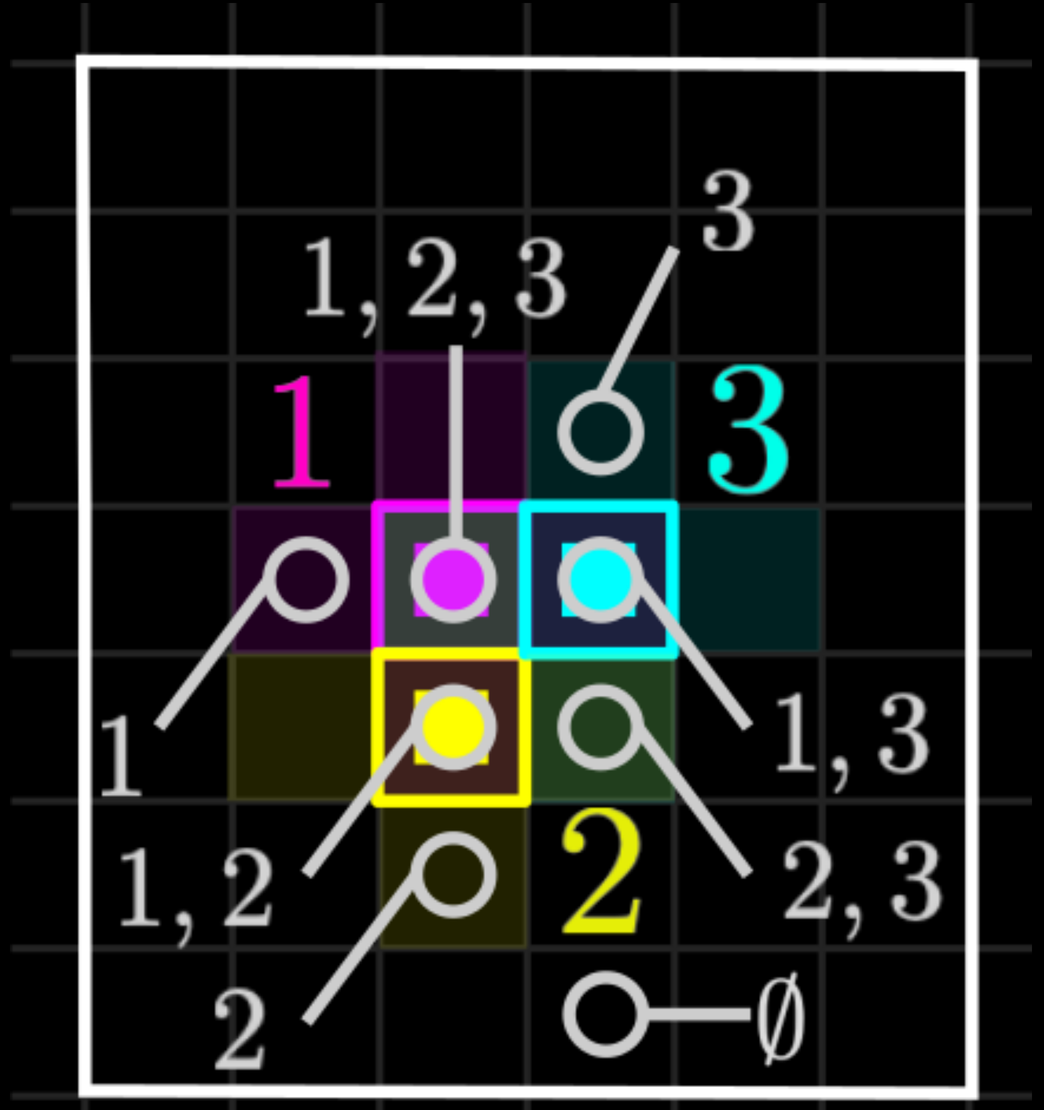
- Every unit square in D_i is k-hop visible from u (or: from every unit square in D_i unit square u is k-hop visible)
- No unit square in $D \setminus D_i$ is k-hop visible from u (or: from no unit square in $D \setminus D_i$ unit square u is k-hop visible)

Unit square u is a *viewpoint*.

VC dimension: largest d, such that there exists a polyomino P and a set of d unit-square guards D that can be shattered.

What Is the VC Dimension?

3 guards (*d*=3)
k=1
2^3=8 viewpoints



- Defined by Vapnik and Chervonenkis
- Measure of complexity of a set system
- For us:

  Finite set of unit square (guards) D in a polyomino P is *shattered* if for any of the $2^{|D|}$ subsets $D_i \subseteq D$, there exists a unit square $u \in P$:

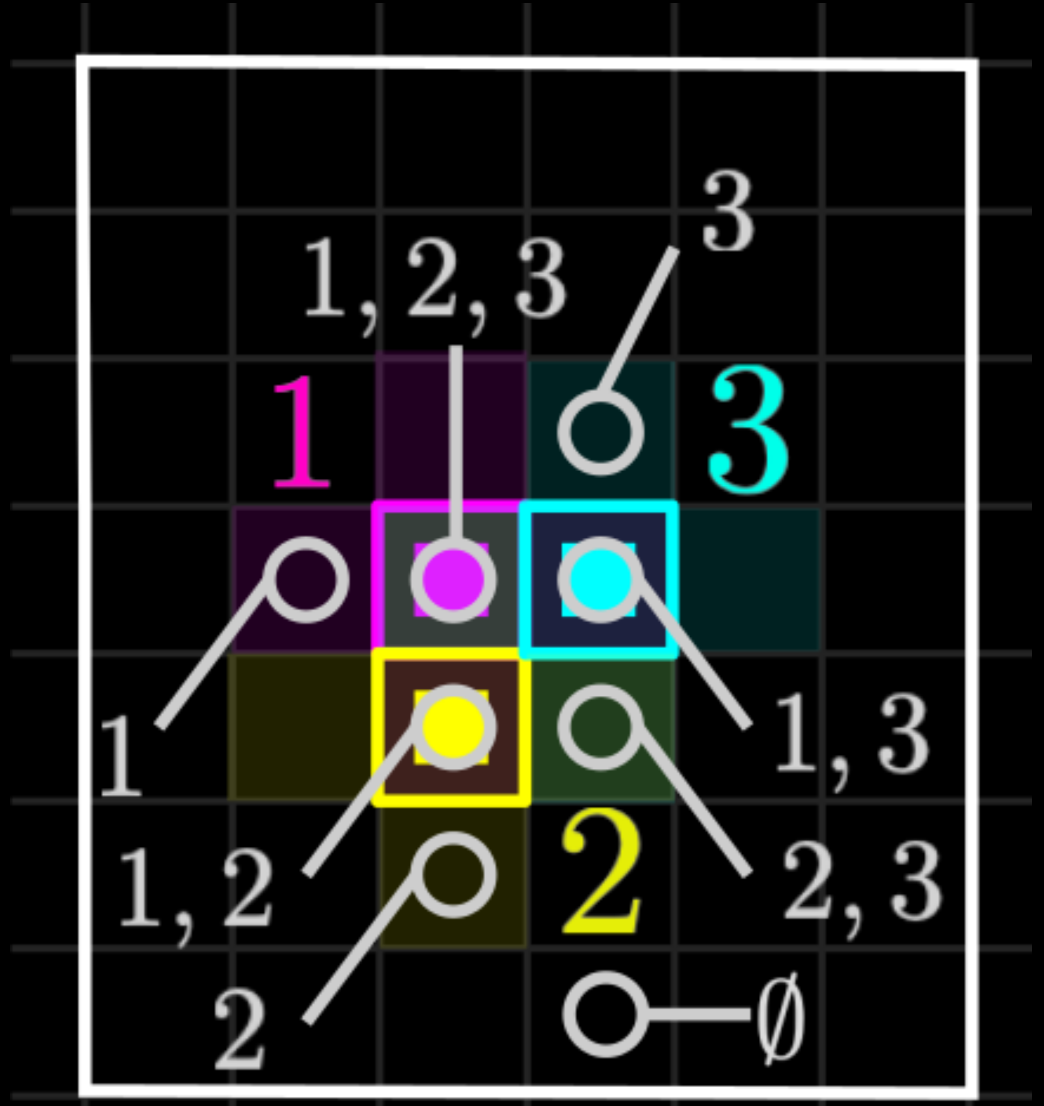
  - Every unit square in D_i is k-hop visible from u (or: from every unit square in D_i unit square u is k-hop visible)
  - No unit square in $D \setminus D_i$ is k-hop visible from u (or: from no unit square in $D \setminus D_i$ unit square u is k-hop visible)

  Unit square u is a *viewpoint*.

  VC dimension: largest d, such that there exists a polyomino P and a set of d unit-square guards D that can be shattered.

What Is the VC Dimension?

3 guards (d=3)
k=1
2^3=8 viewpoints



- Defined by Vapnik and Chervonenkis
- Measure of complexity of a set system
- For us:

  Finite set of unit square (guards) D in a polyomino P is *shattered* if for any of the $2^{|D|}$ subsets $D_i \subseteq D$, there exists a unit square $u \in P$:

  - Every unit square in D_i is k-hop visible from u (or: from every unit square in D_i unit square u is k-hop visible)
  - No unit square in $D \setminus D_i$ is k-hop visible from u (or: from no unit square in $D \setminus D_i$ unit square u is k-hop visible)

  Unit square u is a *viewpoint*.

  VC dimension: largest d, such that there exists a polyomino P and a set of d unit-square guards D that can be shattered.

  Has been studied for different guarding problems (e.g., for simple polygons, ordinary visibility VC dimension in [6,14])

VC Dimension

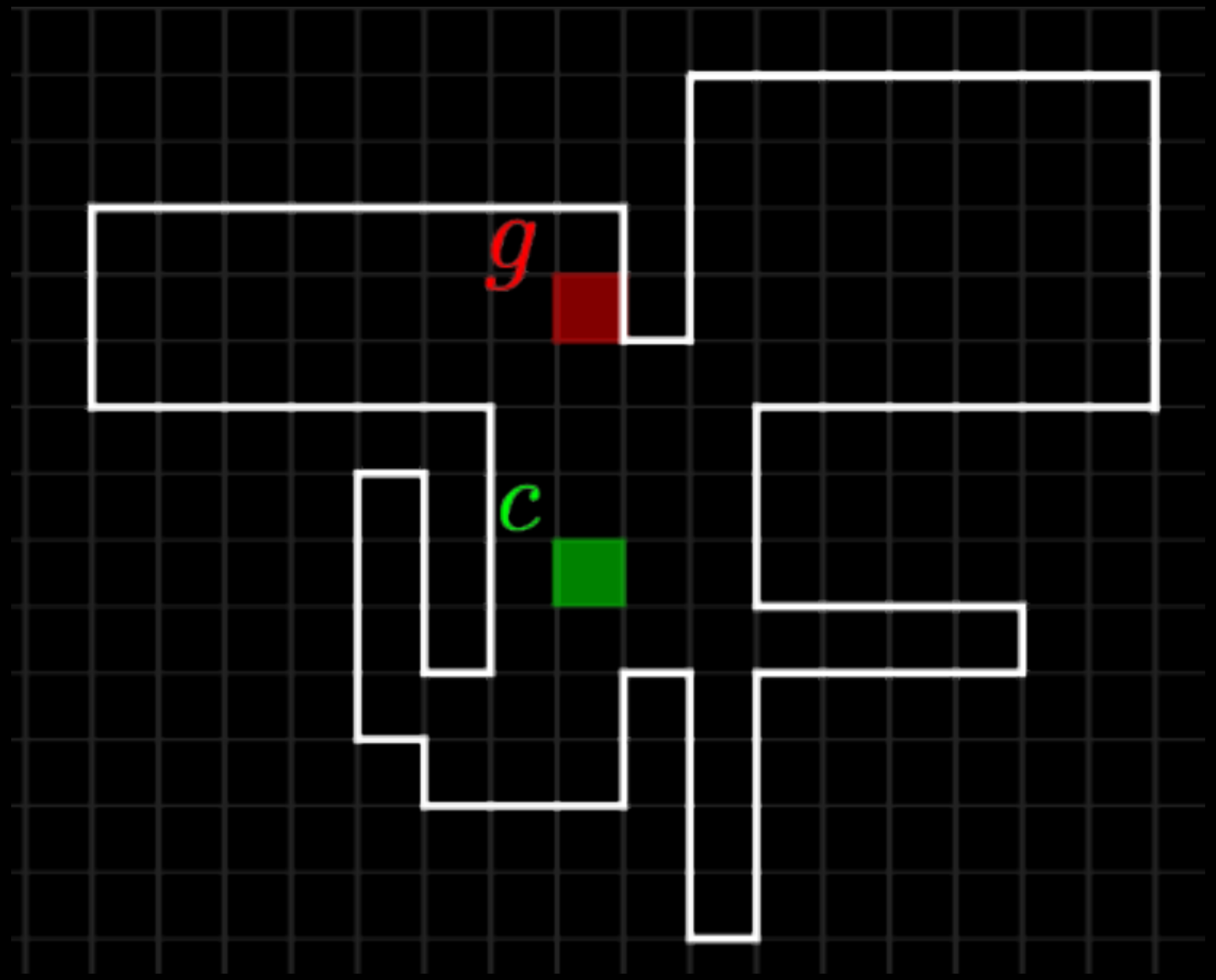
We show:

Theorem 1: For any $k \in \mathbb{N}$, the VC dimension of k-hop visibility in a simple polyomino is 3.

Theorem 2: For large enough $k \in \mathbb{N}$, the VC dimension of k-hop visibility in a polyomino with holes is 4.

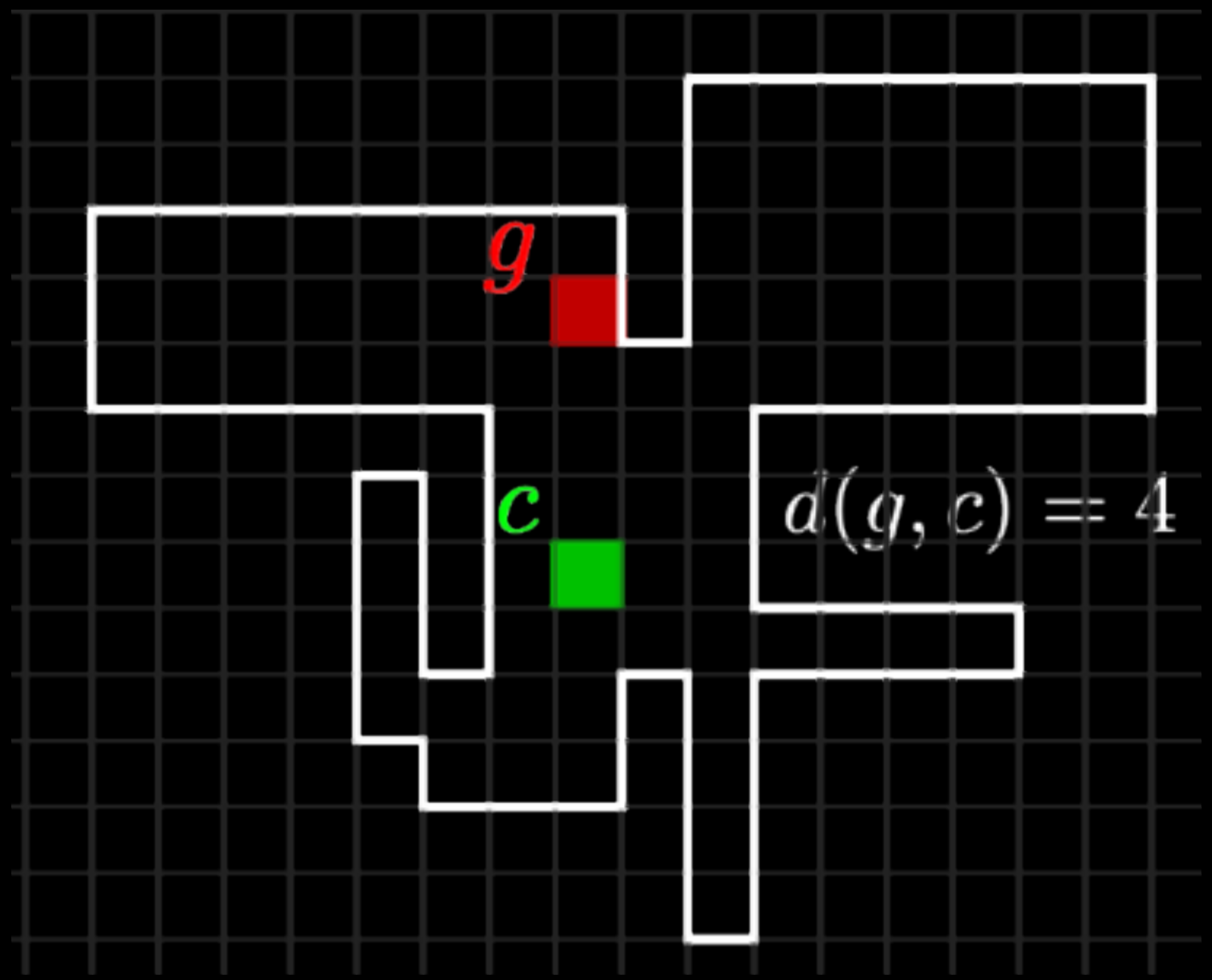
For Upper Bounds: Rest Budget

- *Rest budget* of unit square $g \in P$ at a unit square $c \in P$: $\mathrm{rb}(g,c)=\max\{k-d(g,c),0\}$



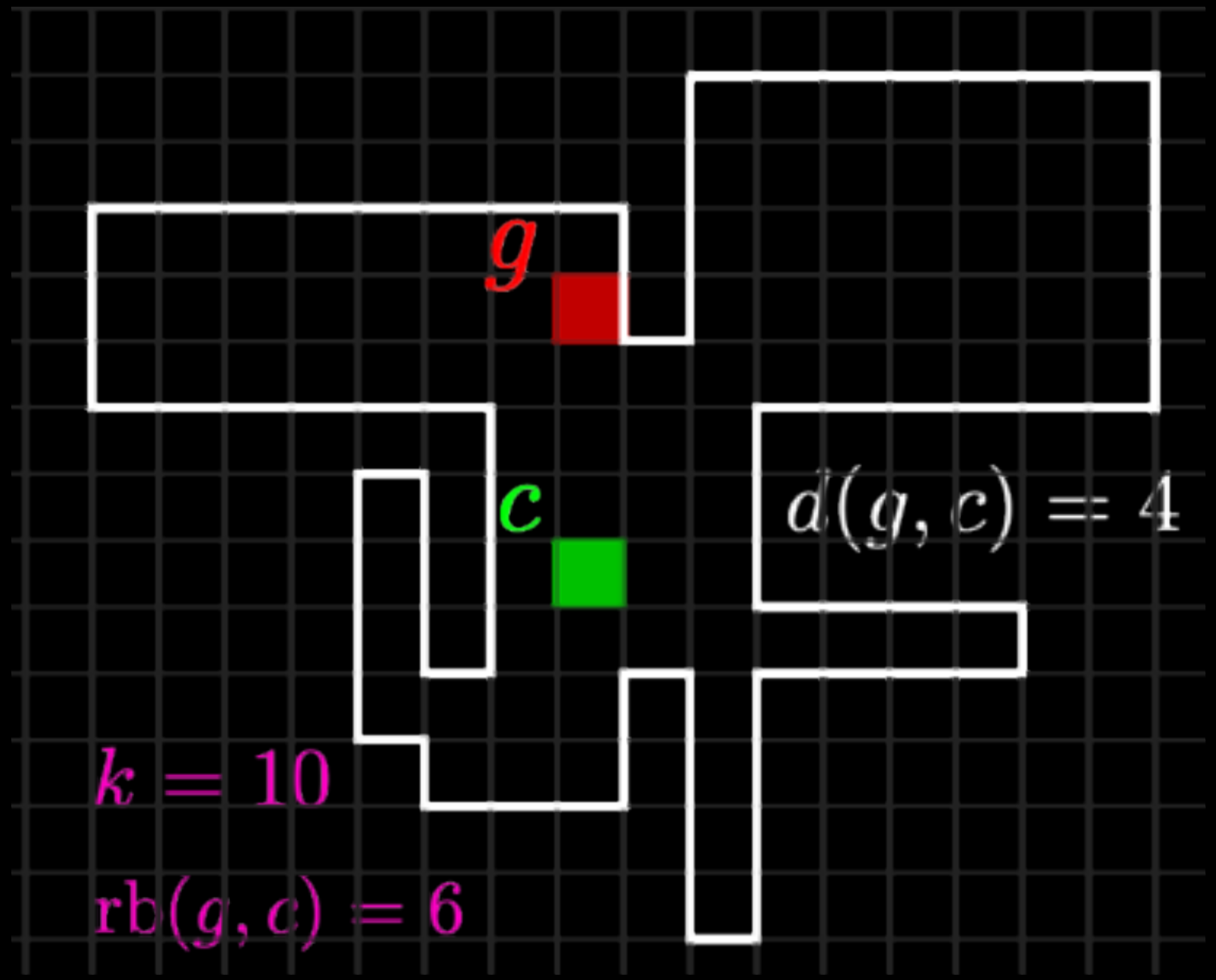
For Upper Bounds: Rest Budget

- *Rest budget* of unit square $g \in P$ at a unit square $c \in P$: $\mathrm{rb}(g,c)=\max\{k-d(g,c),0\}$



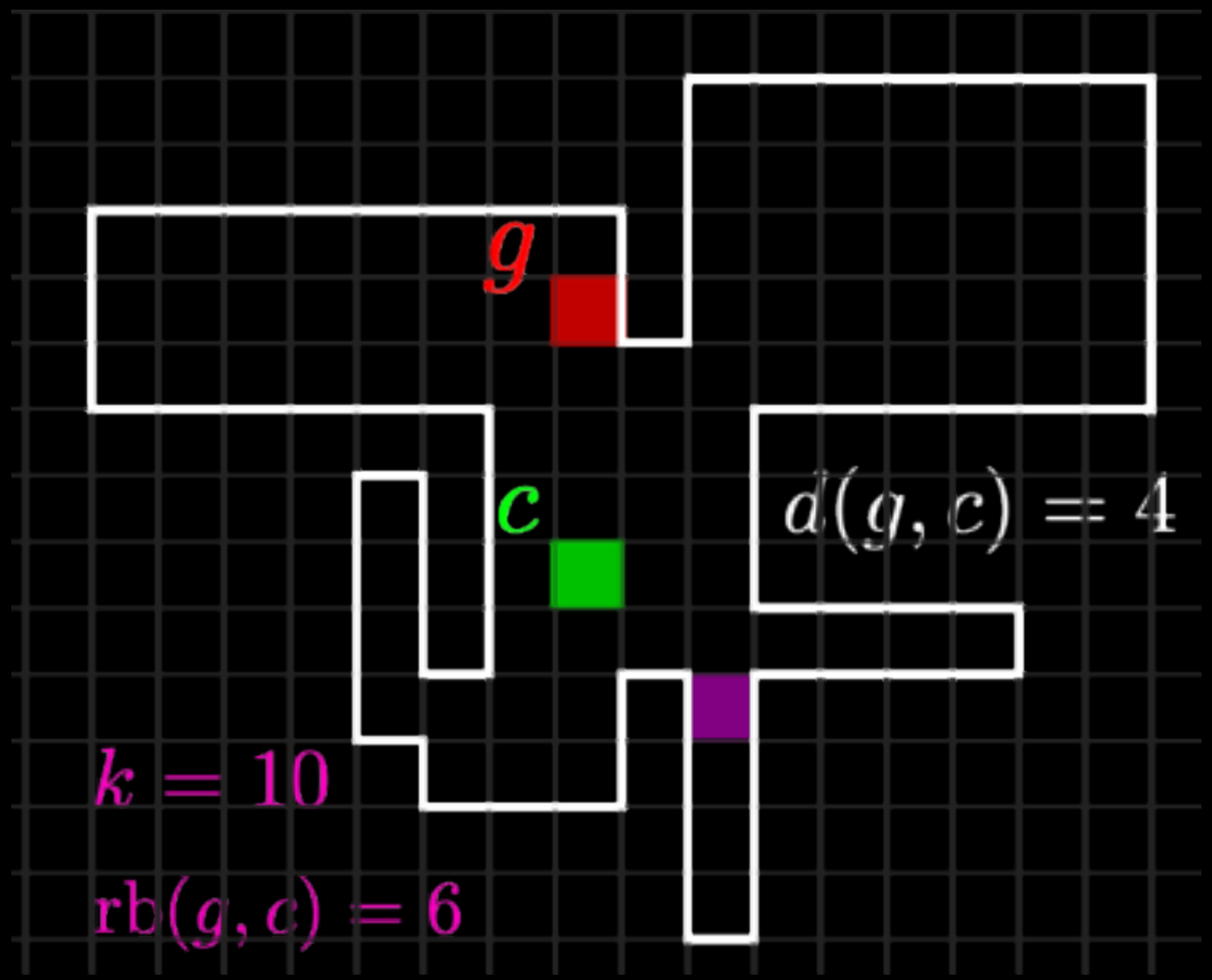
For Upper Bounds: Rest Budget

- *Rest budget* of unit square $g \in P$ at a unit square $c \in P$: $\mathrm{rb}(g,c)=\max\{k-d(g,c),0\}$



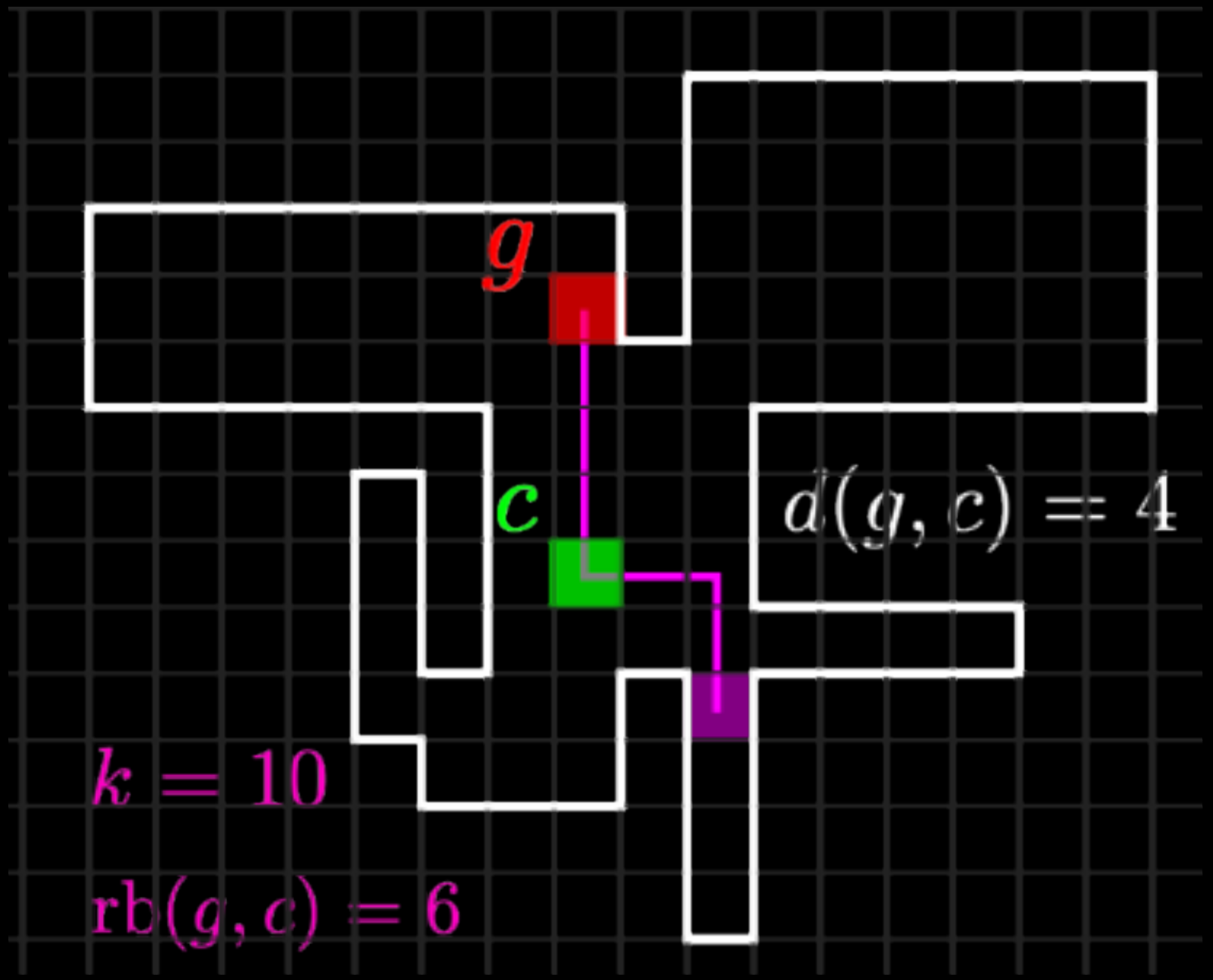
For Upper Bounds: Rest Budget

- *Rest budget* of unit square $g \in P$ at a unit square $c \in P$: $\mathrm{rb}(g,c)=\max\{k-d(g,c),0\}$



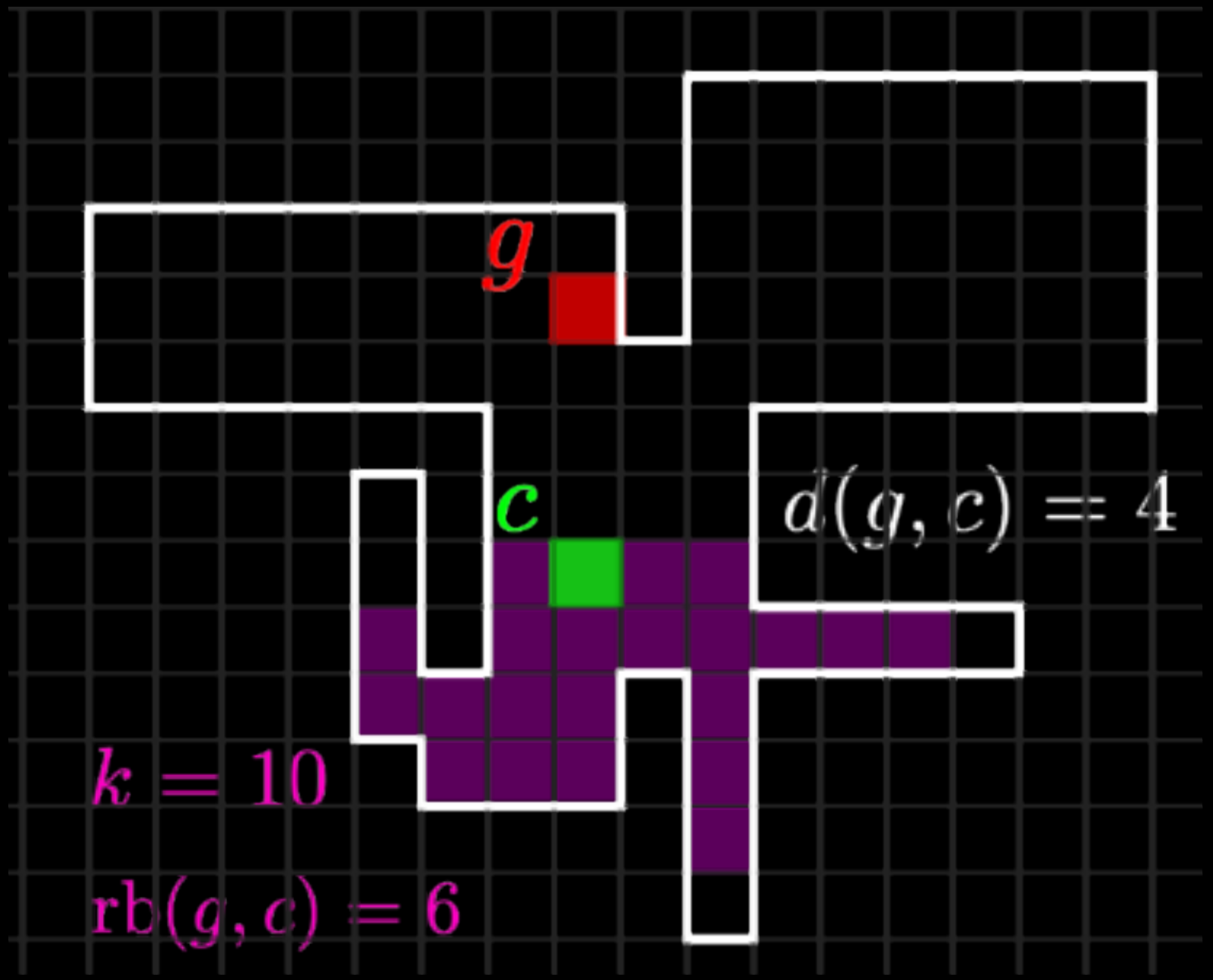
For Upper Bounds: Rest Budget

- *Rest budget* of unit square $g \in P$ at a unit square $c \in P$: $\mathrm{rb}(g,c) = \max\{k - d(g,c), 0\}$



For Upper Bounds: Rest Budget

- *Rest budget* of unit square $g \in P$ at a unit square $c \in P$: $\mathrm{rb}(g,c)=\max\{k-d(g,c),0\}$



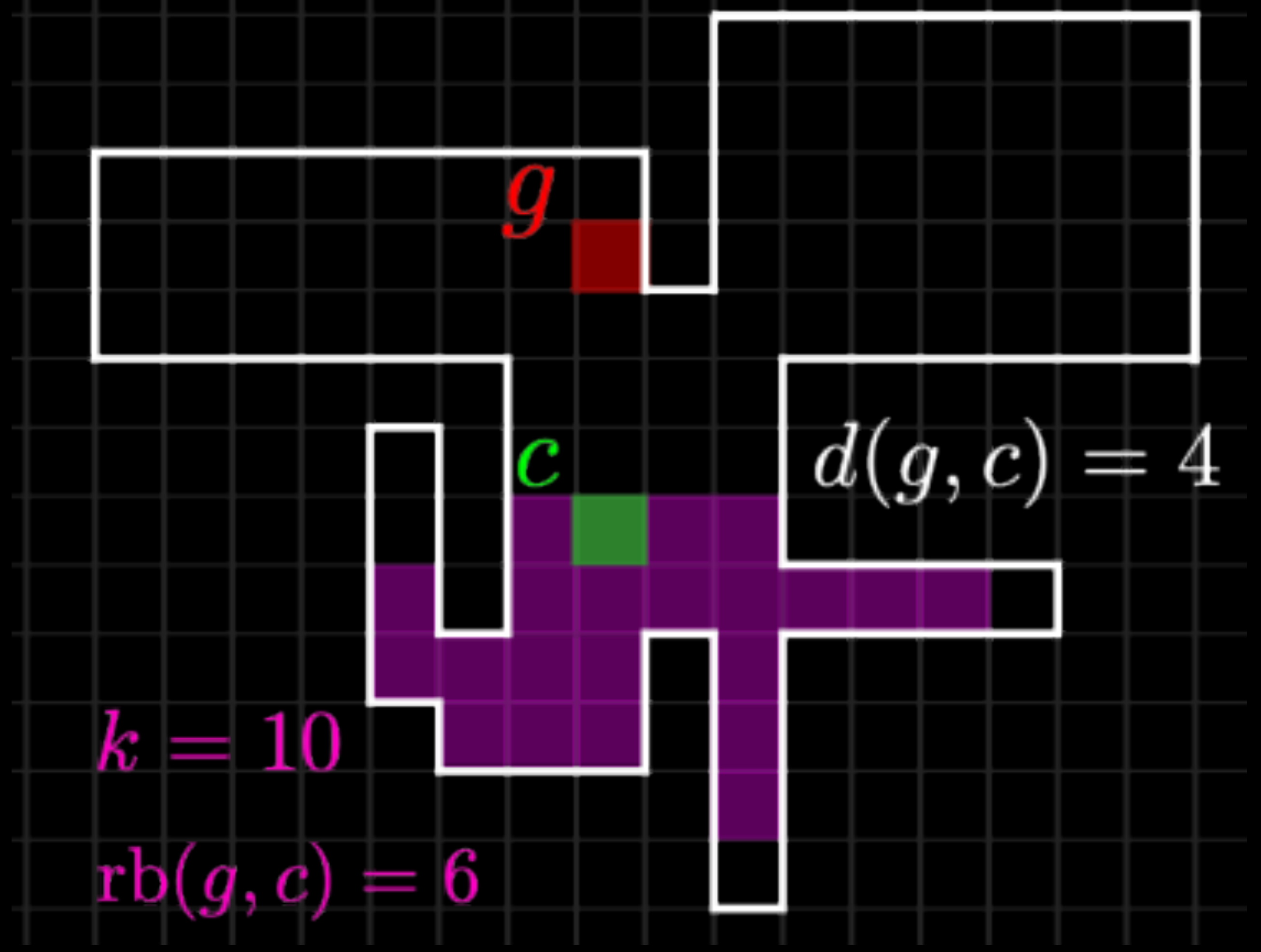
For Upper Bounds: Rest Budget

Rest-Budget Observation:

Let *P* be a polyomino, and let *g* and *u* be two unit squares in *P* such that a shortest path between them contains a unit square *c*. Then the following holds:

1. The unit square *g* covers *u*, if and only if *u* is within distance rb(*g*,*c*) from *c*.

2. For any unit square *g'* with rb(*g'*,*c*) > rb(*g*,*c*), if *g* covers *u*, then so does *g'*.



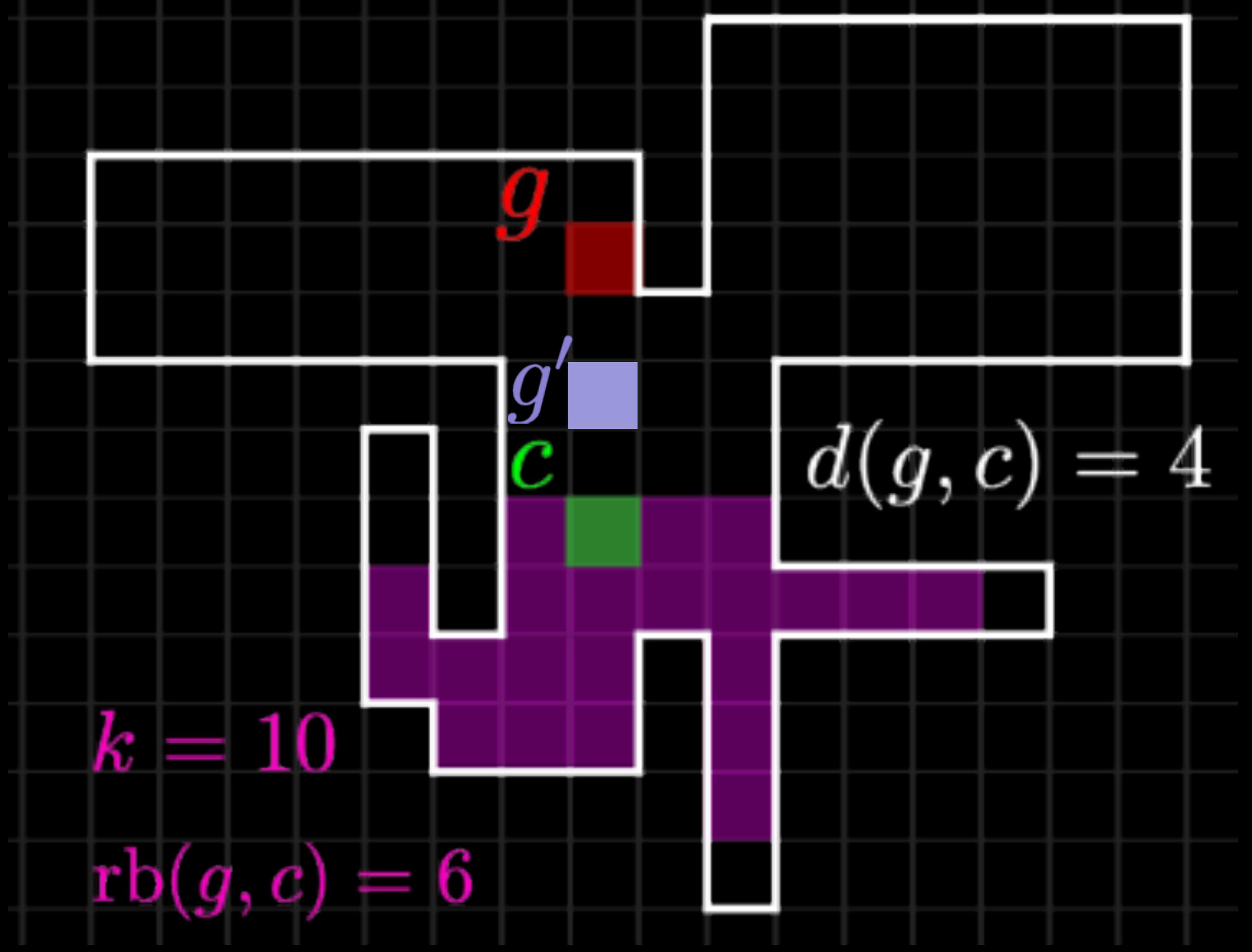
For Upper Bounds: Rest Budget

Rest-Budget Observation:

Let P be a polyomino, and let g and u be two unit squares in P such that a shortest path between them contains a unit square c. Then the following holds:

1. The unit square g covers u, if and only if u is within distance rb(g,c) from c.

2. For any unit square g' with rb(g',c) > rb(g,c), if g covers u, then so does g'.



For Upper Bounds: Rest Budget

Rest-Budget Observation:

Let P be a polyomino, and let g and u be two unit squares in P such that a shortest path between them contains a unit square c. Then the following holds:

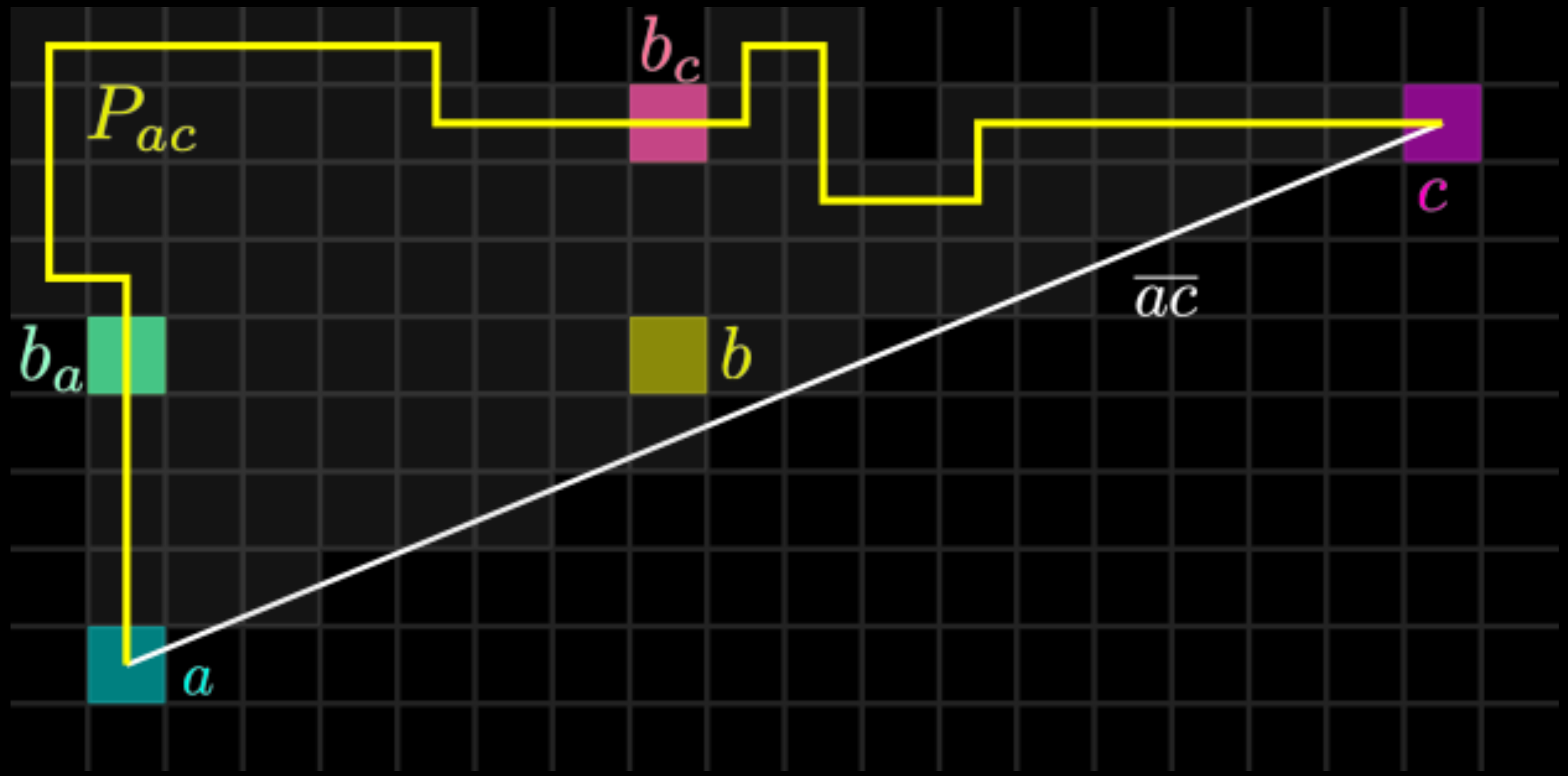
1. The unit square g covers u, if and only if u is within distance rb(g,c) from c.

2. For any unit square g' with rb(g',c) > rb(g,c), if g covers u, then so does g'.

For Upper Bounds: Rest Budget

Rest-Budget Lemma:

Let a,c be two unit squares in a simple polyomino P, such that P does not cross the line segment ac that connects their center points. Let P_{ac} be some path in the dual graph G_P between the center points of a and c, and let b be a unit square whose center point belongs to the area enclosed within $P_{ac} \circ ac$. Then there exists a unit square x on such that rb(b,x) ≥ rb(a,x) and rb(b,x) ≥ rb(c,x).



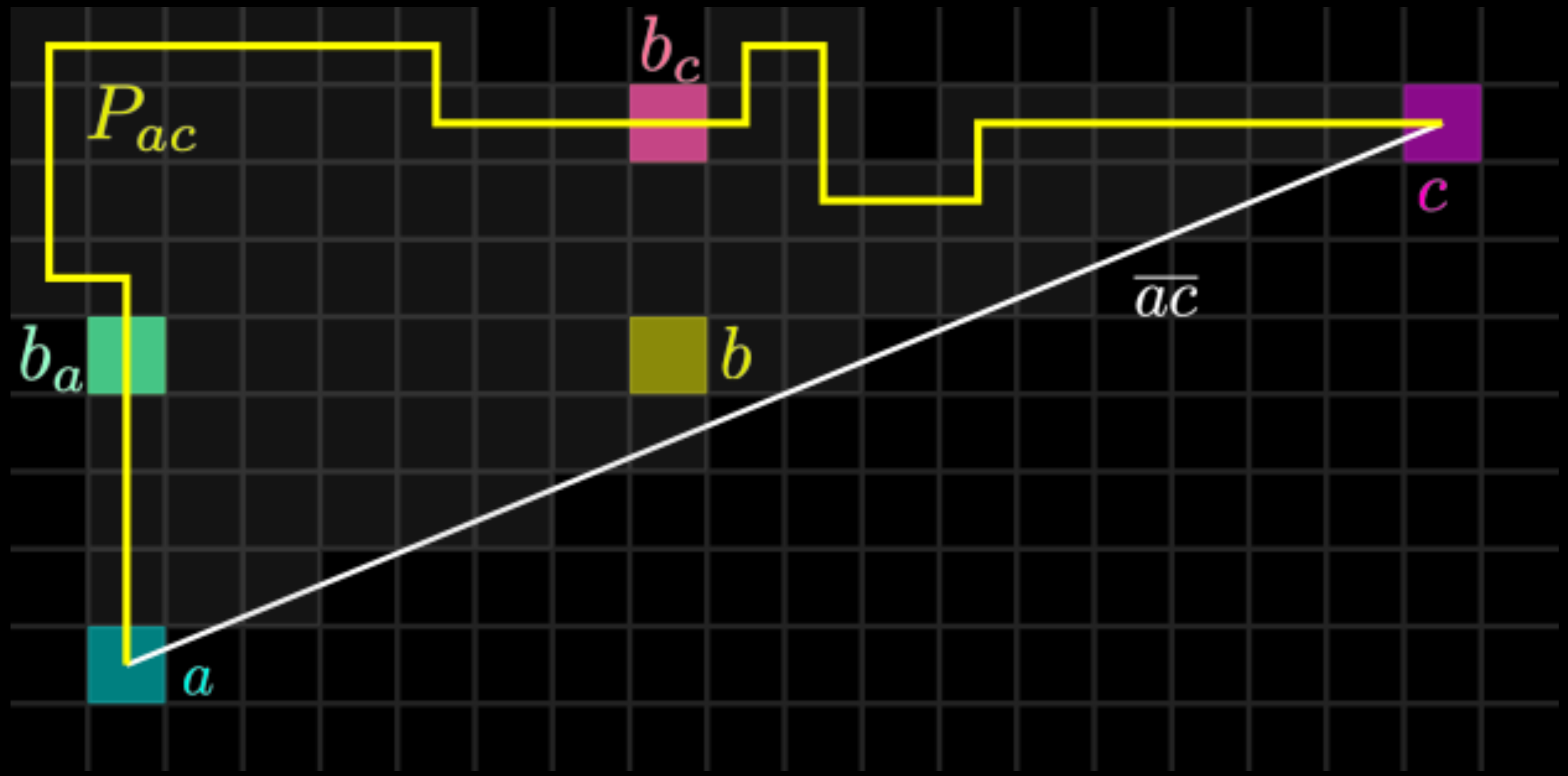
LINKÖPING UNIVERSITY

For Upper Bounds: Rest Budget

Rest-Budget Lemma:

Let a,c be two unit squares in a simple polyomino P, such that P does not cross the line segment ac that connects their center points. Let P_{ac} be some path in the dual graph G_P between the center points of a and c, and let b be a unit square whose center point belongs to the area enclosed within $P_{ac} \circ ac$. Then there exists a unit square x on such that rb(b,x) ≥ rb(a,x) and rb(b,x) ≥ rb(c,x).

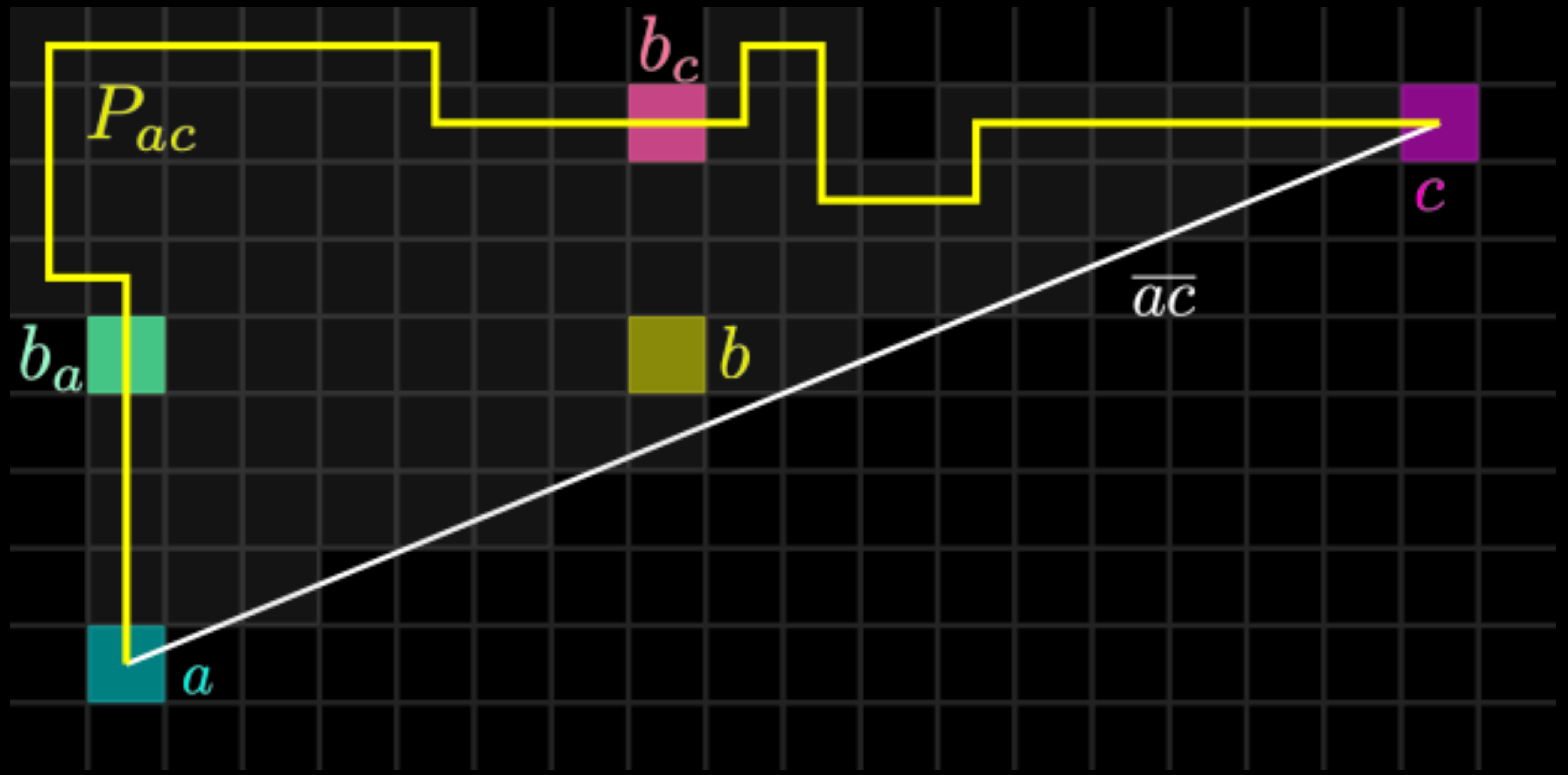
Proof cases:



For Upper Bounds: Rest Budget

Rest-Budget Lemma:

Let a,c be two unit squares in a simple polyomino P, such that P does not cross the line segment ac that connects their center points. Let P_{ac} be some path in the dual graph G_P between the center points of a and c, and let b be a unit square whose center point belongs to the area enclosed within $P_{ac} \circ ac$. Then there exists a unit square x on such that rb(b,x) ≥ rb(a,x) and rb(b,x) ≥ rb(c,x).



Proof cases:

- If b is above c: x is unit square on P_{ac} directly above b

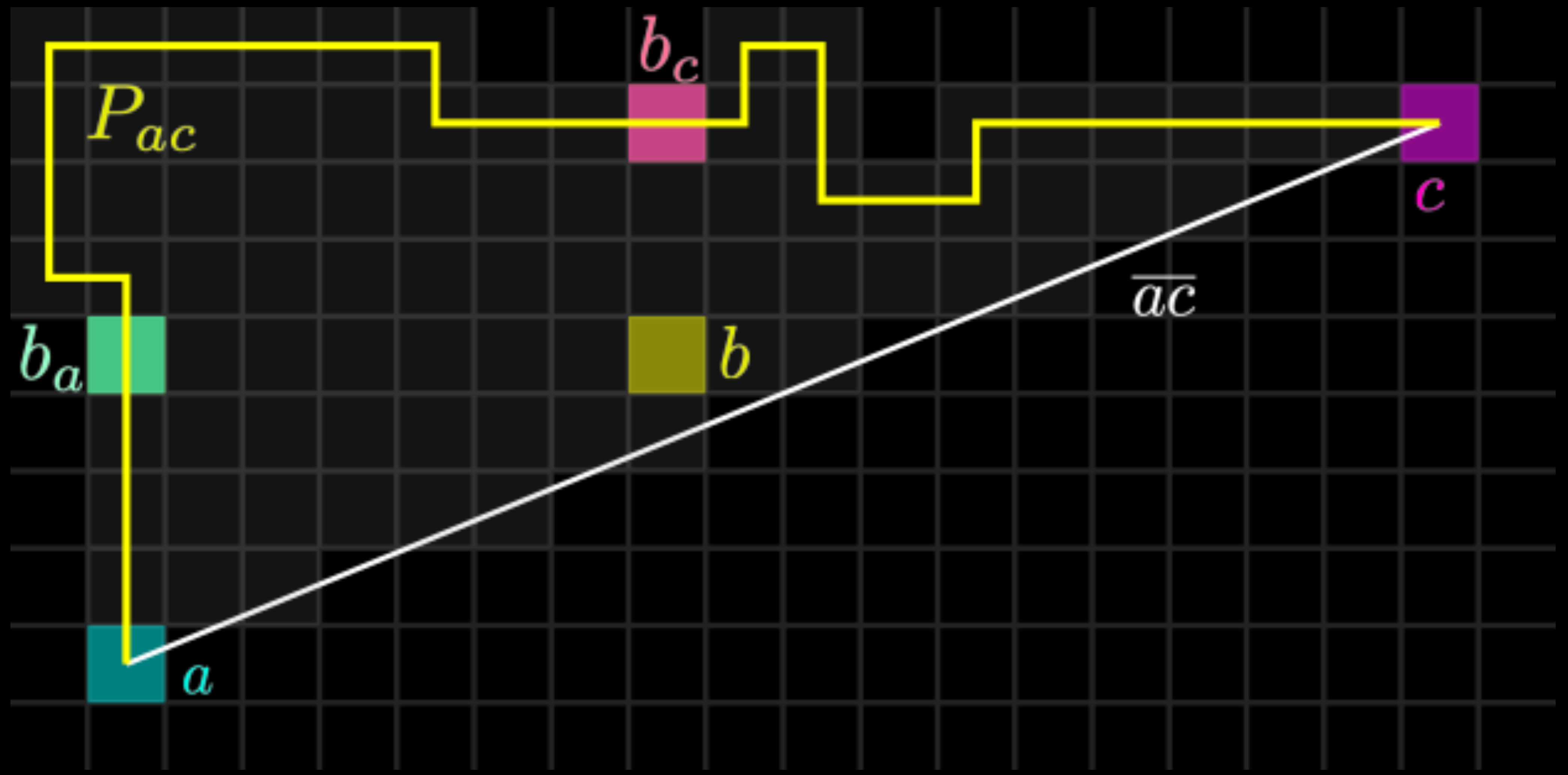
For Upper Bounds: Rest Budget

Rest-Budget Lemma:

Let a,c be two unit squares in a simple polyomino P, such that P does not cross the line segment ac that connects their center points. Let P_{ac} be some path in the dual graph G_P between the center points of a and c, and let b be a unit square whose center point belongs to the area enclosed within $P_{ac} \circ ac$. Then there exists a unit square x on such that rb(b,x) ≥ rb(a,x) and rb(b,x) ≥ rb(c,x).

Proof cases:

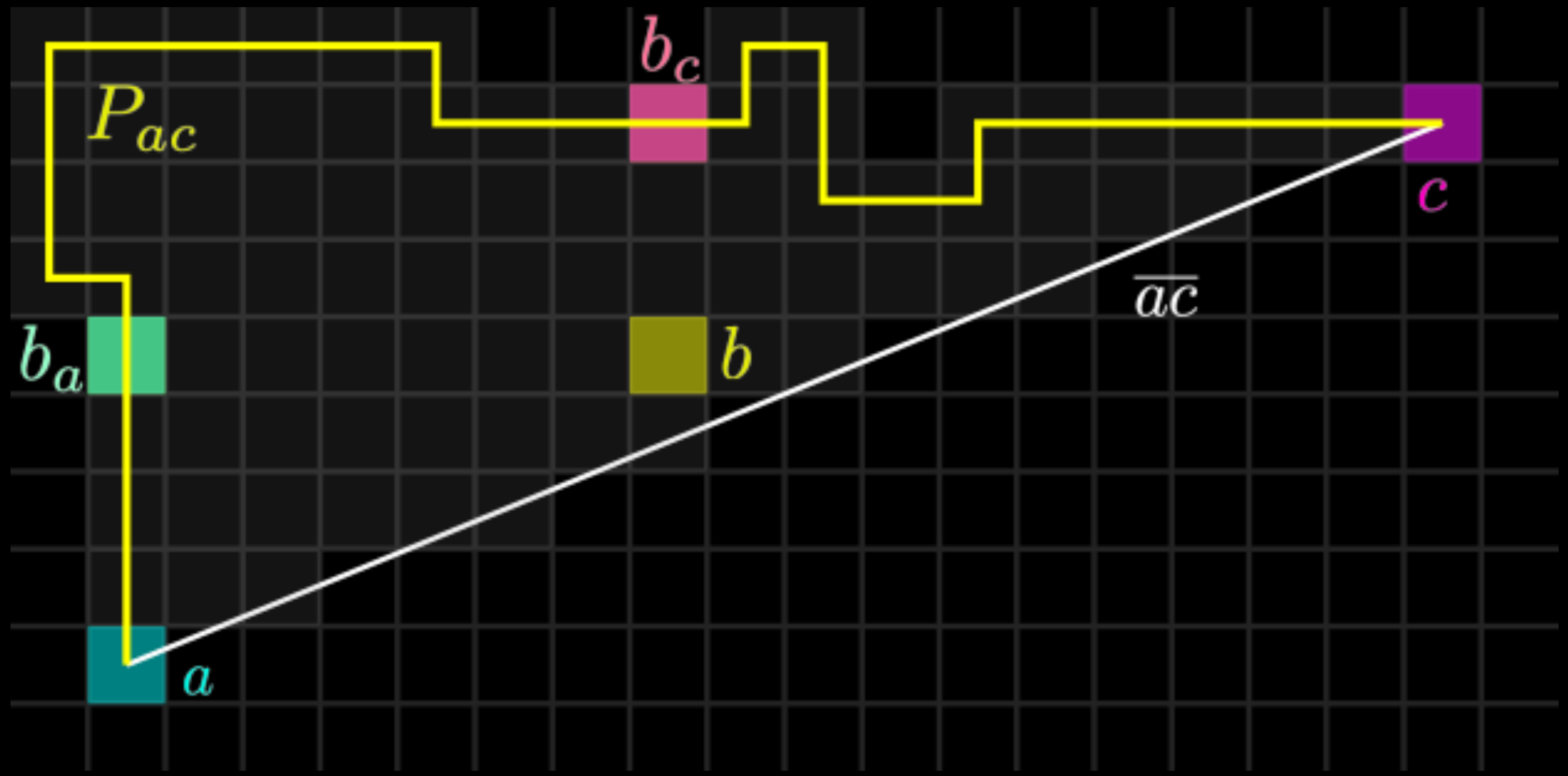
- If b is above c: x is unit square on P_{ac} directly above b
- If b is left of a: x is unit square on P_{ac} directly to left of b



For Upper Bounds: Rest Budget

Rest-Budget Lemma:

Let a,c be two unit squares in a simple polyomino P, such that P does not cross the line segment ac that connects their center points. Let P_{ac} be some path in the dual graph G_P between the center points of a and c, and let b be a unit square whose center point belongs to the area enclosed within $P_{ac} \circ ac$. Then there exists a unit square x on such that rb(b,x) ≥ rb(a,x) and rb(b,x) ≥ rb(c,x).



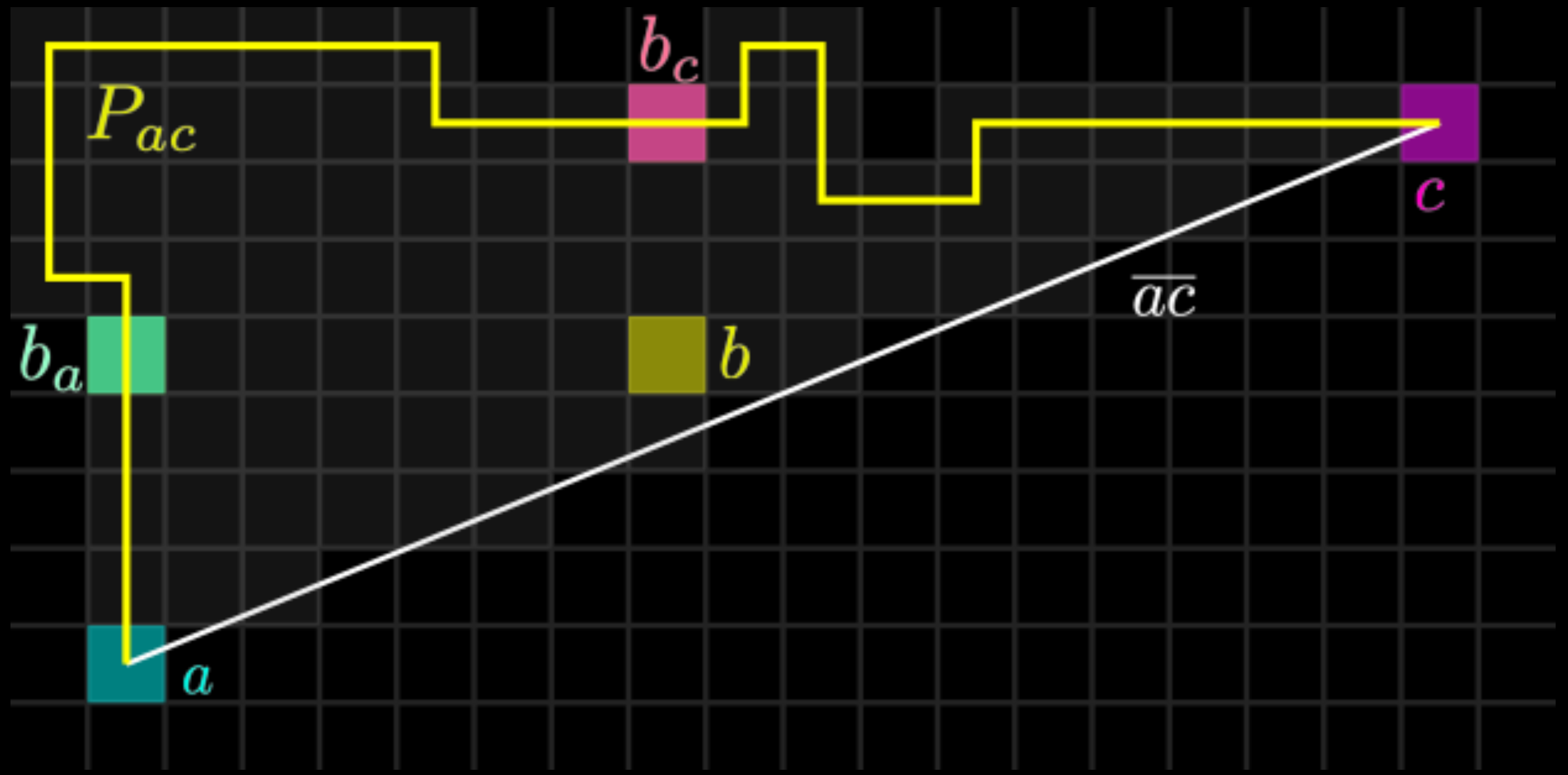
Proof cases:

- If b is above c: x is unit square on P_{ac} directly above b
- If b is left of a: x is unit square on P_{ac} directly to left of b
- If b is within axis-aligned bounding box of a and c

For Upper Bounds: Rest Budget

Rest-Budget Lemma:

Let a,c be two unit squares in a simple polyomino P, such that P does not cross the line segment ac that connects their center points. Let P_{ac} be some path in the dual graph G_P between the center points of a and c, and let b be a unit square whose center point belongs to the area enclosed within $P_{ac} \circ ac$. Then there exists a unit square x on such that rb(b,x) ≥ rb(a,x) and rb(b,x) ≥ rb(c,x).



Proof cases:

- If b is above c: x is unit square on P_{ac} directly above b
- If b is left of a: x is unit square on P_{ac} directly to left of b
- If b is within axis-aligned bounding box of a and c
  - Slope of ac is ≤1: b_c is our x (➔Figure)

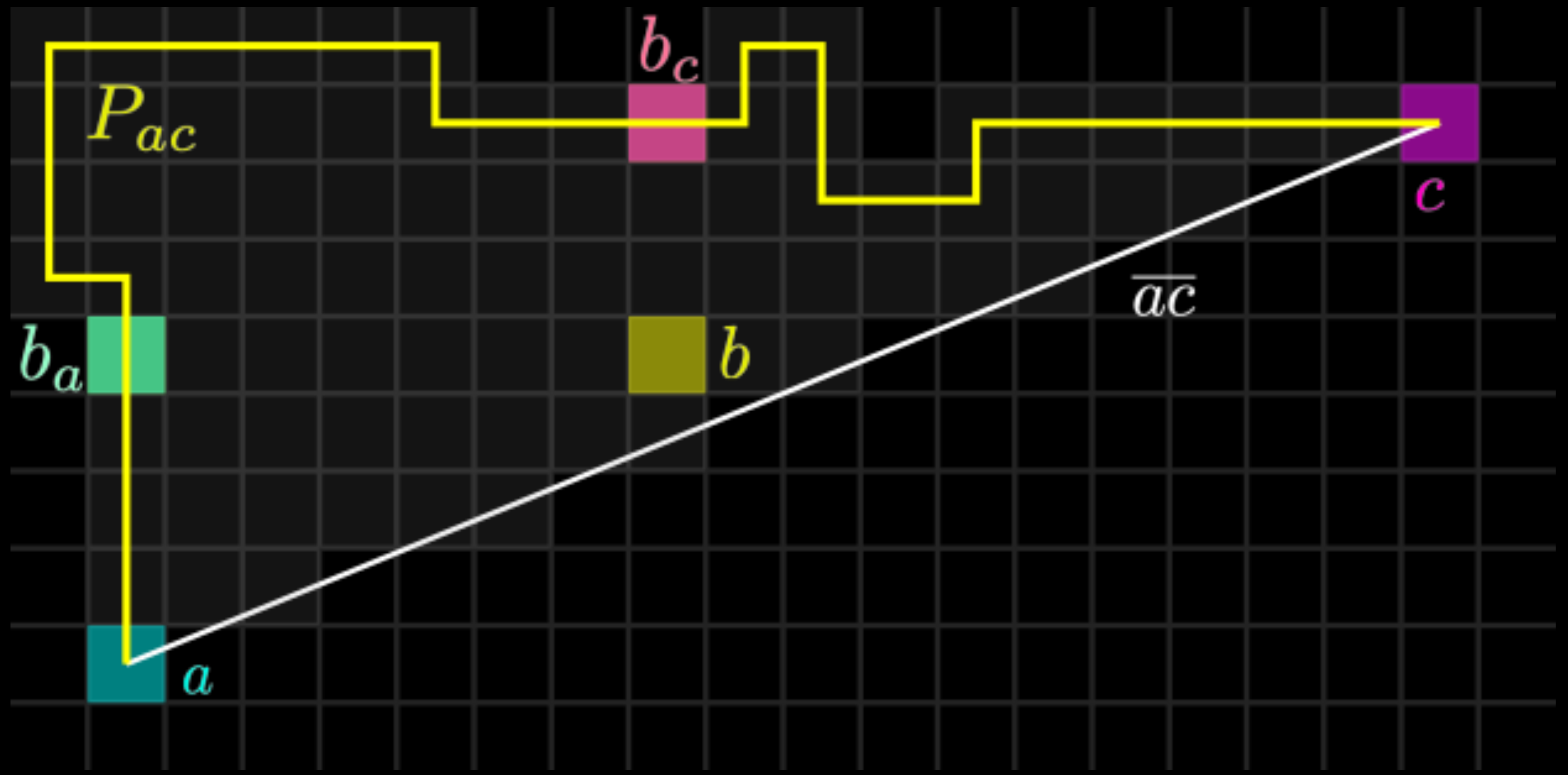
For Upper Bounds: Rest Budget

Rest-Budget Lemma:

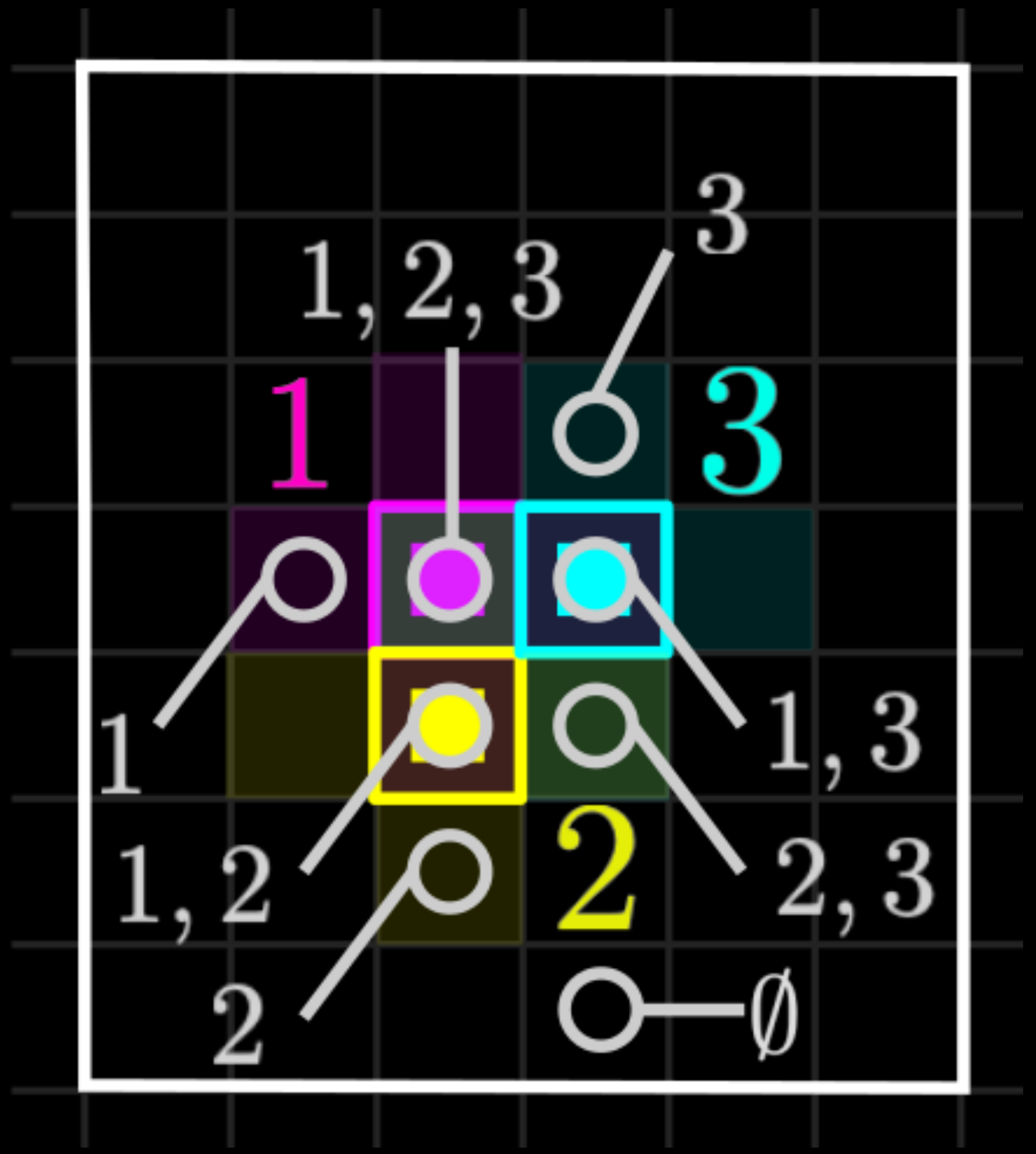
Let a,c be two unit squares in a simple polyomino P, such that P does not cross the line segment ac that connects their center points. Let P_{ac} be some path in the dual graph G_P between the center points of a and c, and let b be a unit square whose center point belongs to the area enclosed within $P_{ac} \circ ac$. Then there exists a unit square x on such that rb(b,x) ≥ rb(a,x) and rb(b,x) ≥ rb(c,x).

Proof cases:

- If b is above c: x is unit square on P_{ac} directly above b
- If b is left of a: x is unit square on P_{ac} directly to left of b
- If b is within axis-aligned bounding box of a and c
  - Slope of ac is ≤1: b_c is our x (➜Figure)
  - Slope of ac is >1: b_a is our x

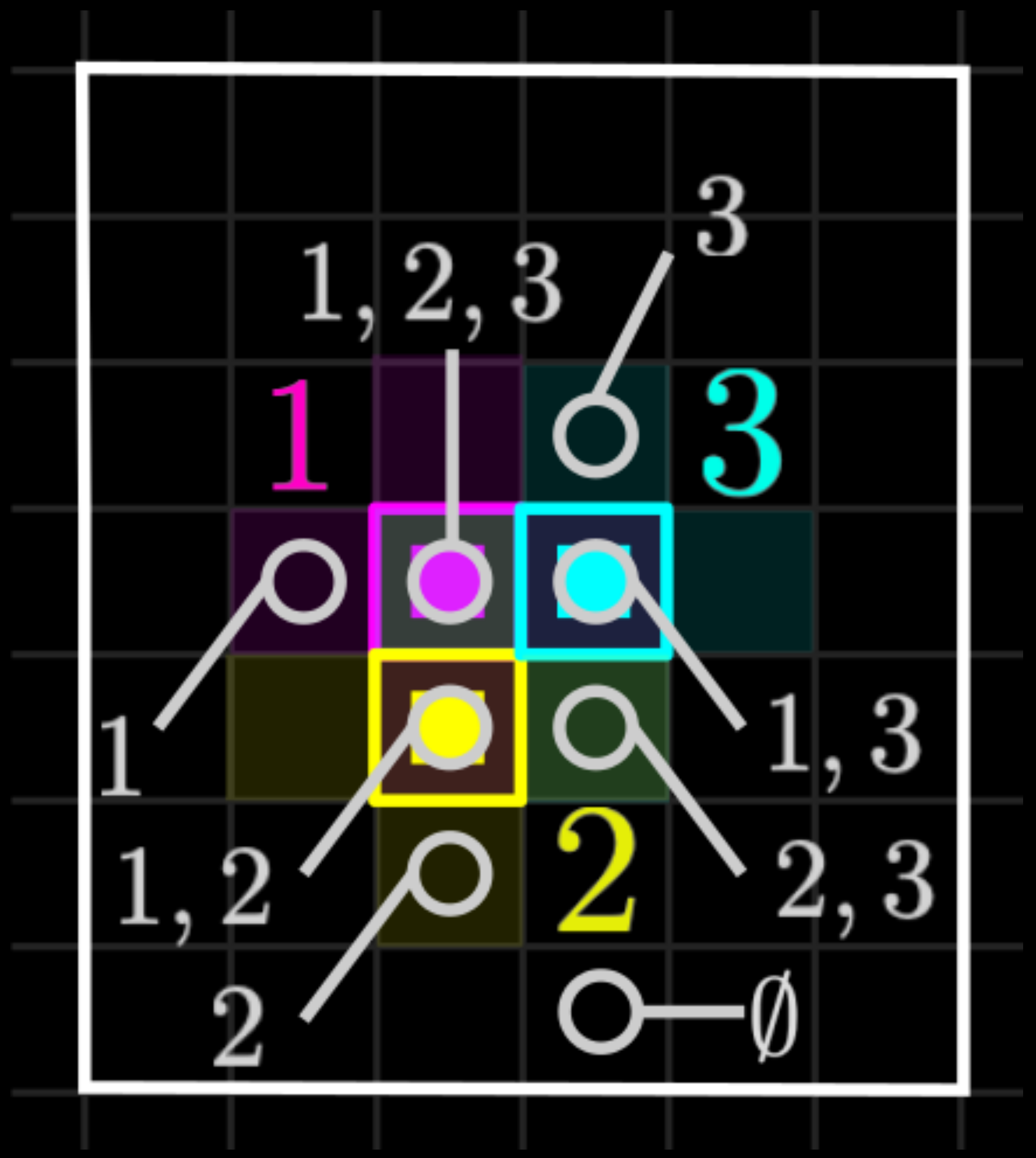


Simple Polyominoes: Lower Bound



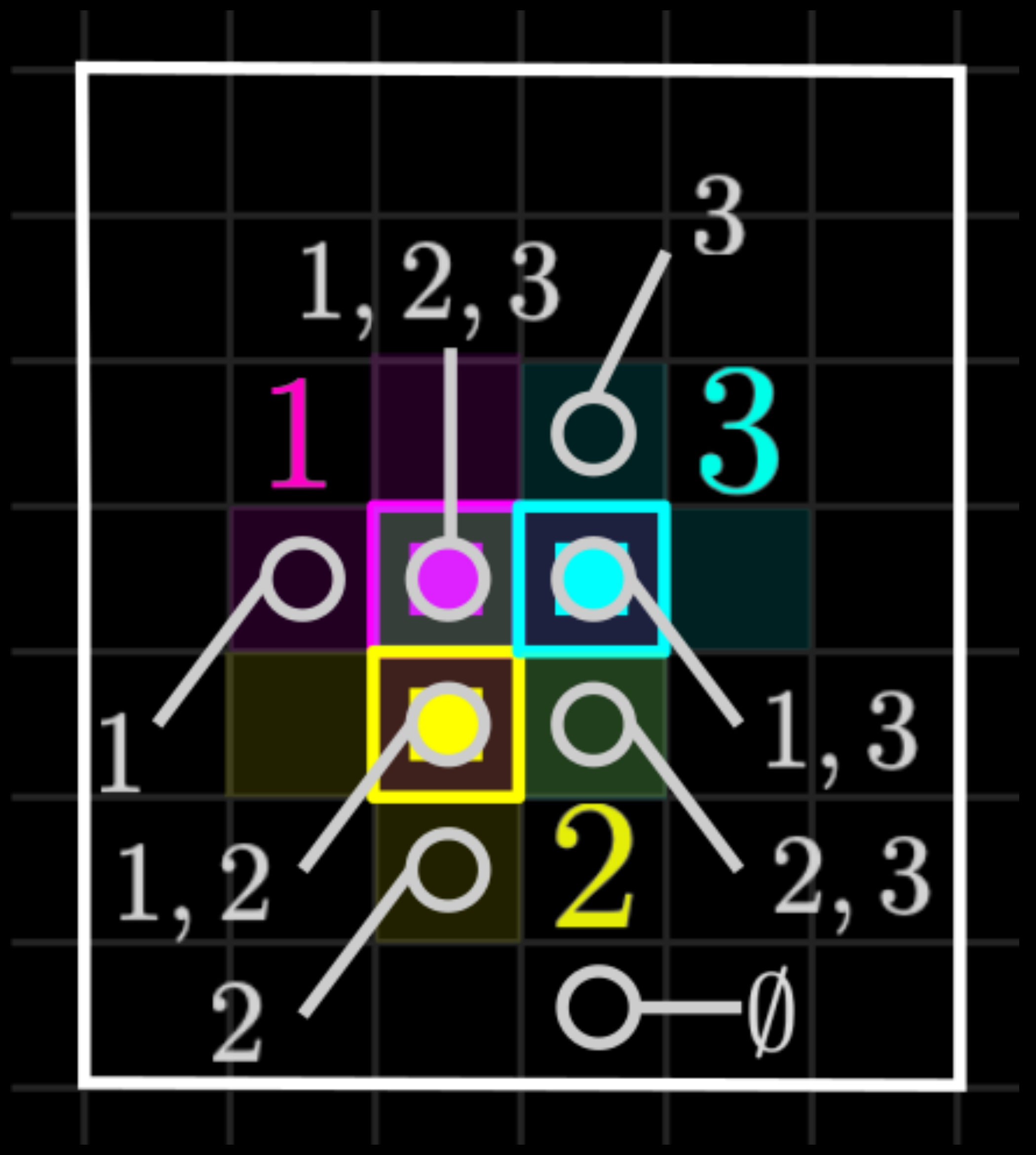
Simple Polyominoes: Lower Bound

3 guards (d=3)
k=1
2^3=8 viewpoints



Simple Polyominoes: Lower Bound

3 guards (*d*=3)
k=1
2^3=8 viewpoints



For larger *k*: keep placement of guards, but polyomino is a large rectangle that contains all *k*-hop visibility regions. Because of the relative position of the guards, all guards are still shattered.

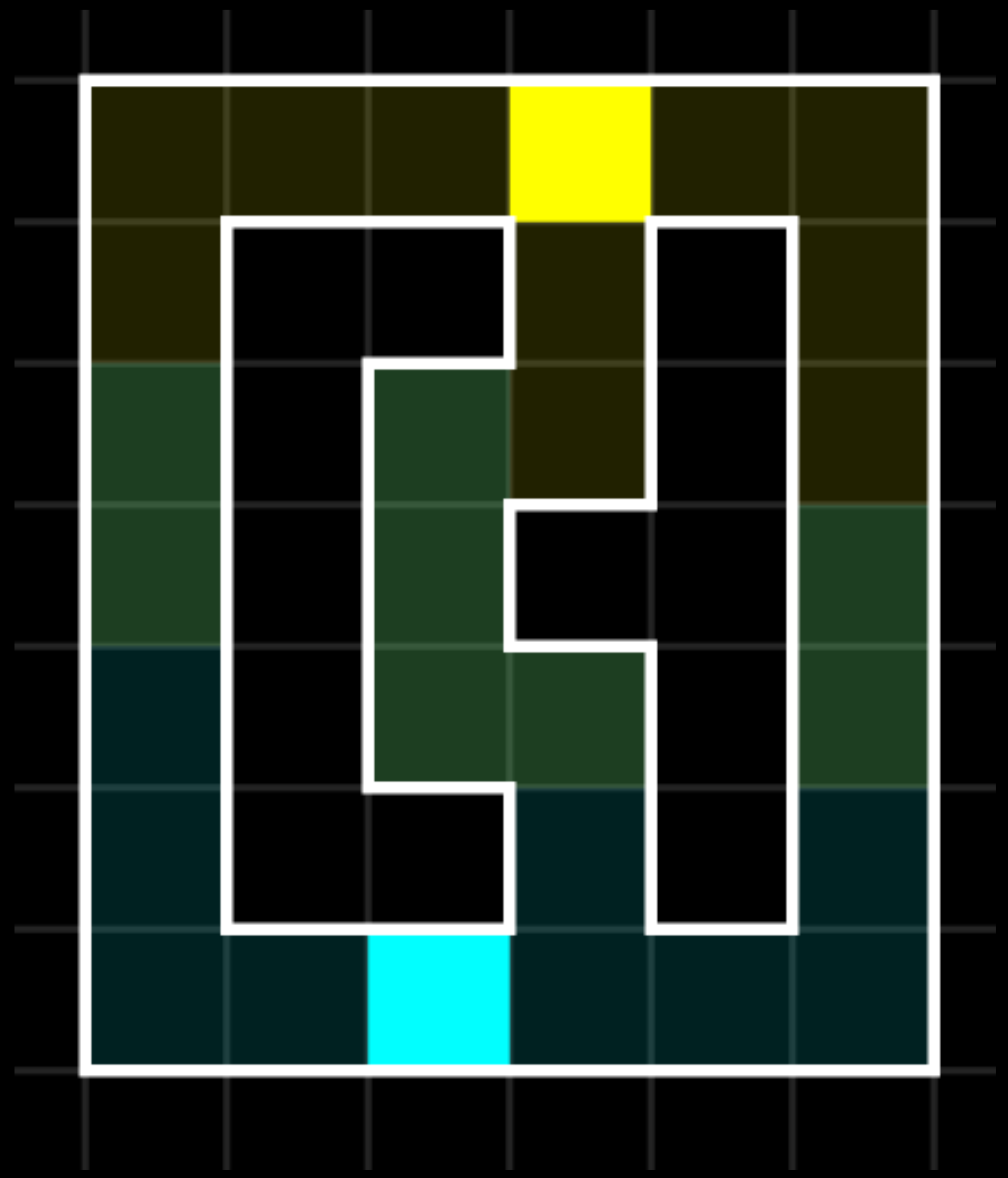
Polyominoes with Holes

- Upper bound of 4 for VC dimension in hypergraphs of pseudo disks [*]

[*] Boris Aronov, Anirudh Donakonda, Esther Ezra, and Rom Pinchasi. On pseudodisk hypergraphs. Computational Geometry, 92:101687, 2021

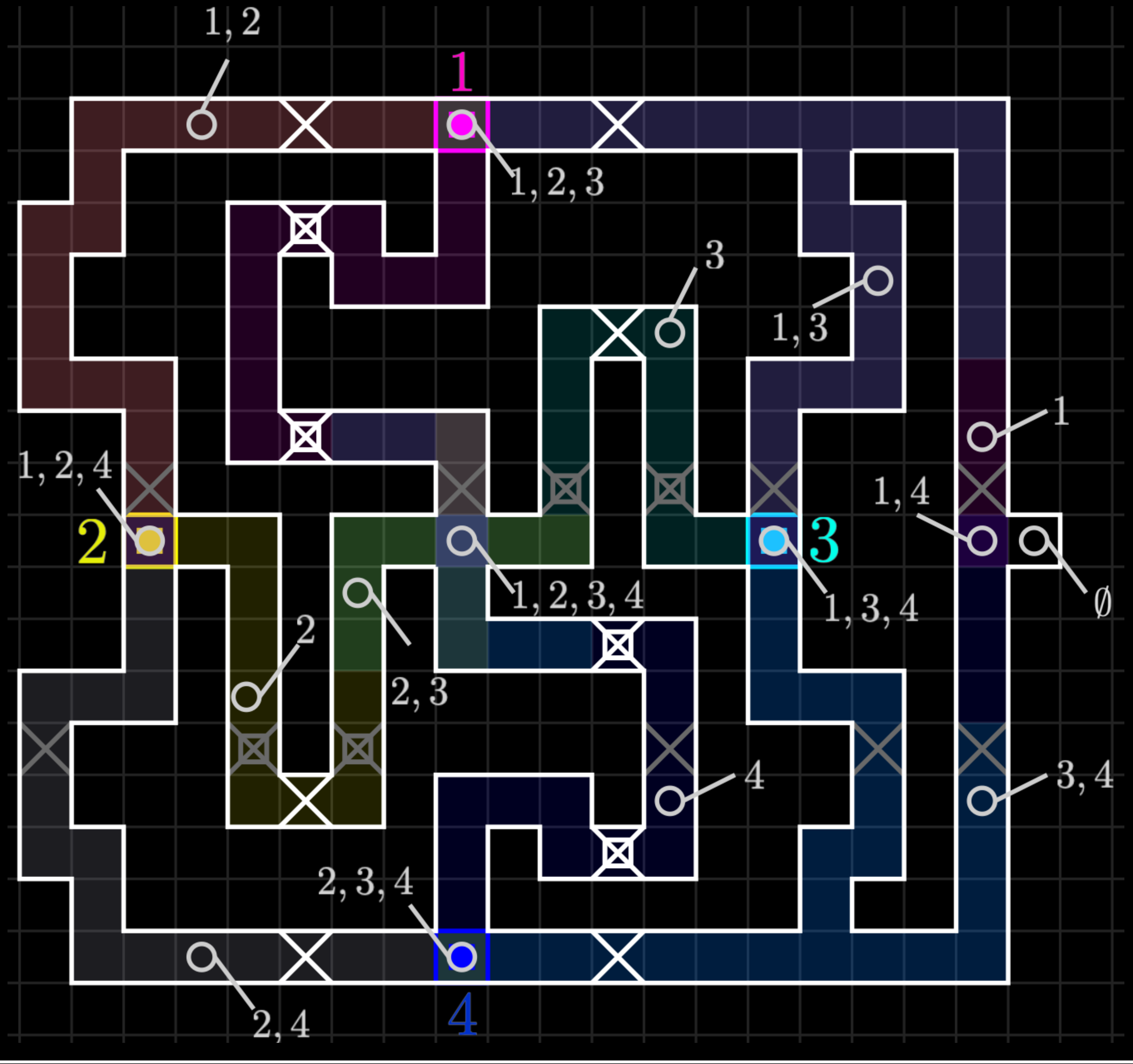
Polyominoes with Holes

- Upper bound of 4 for VC dimension in hypergraphs of pseudo disks [*]
- But: *k*-hop visibility regions of unit squares in polyominoes with holes are not pseudo disks



[*] Boris Aronov, Anirudh Donakonda, Esther Ezra, and Rom Pinchasi. On pseudodisk hypergraphs. Computational Geometry, 92:101687, 2021

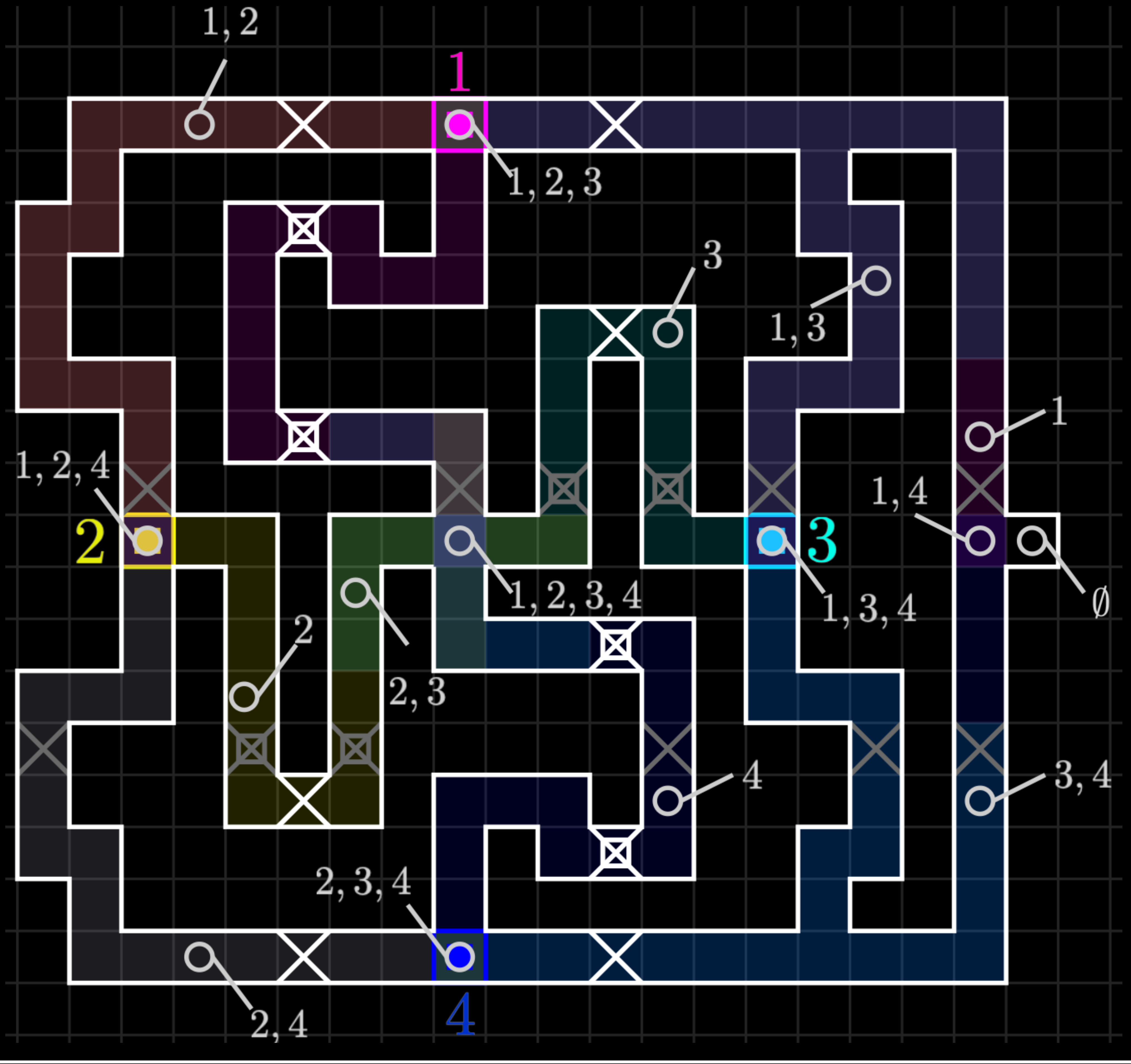
Polyominoes with Holes: Lower Bound



For k=18

d=4➔2^4=16 viewpoints (marked with the guard labels of guards that see them)

Polyominoes with Holes: Lower Bound

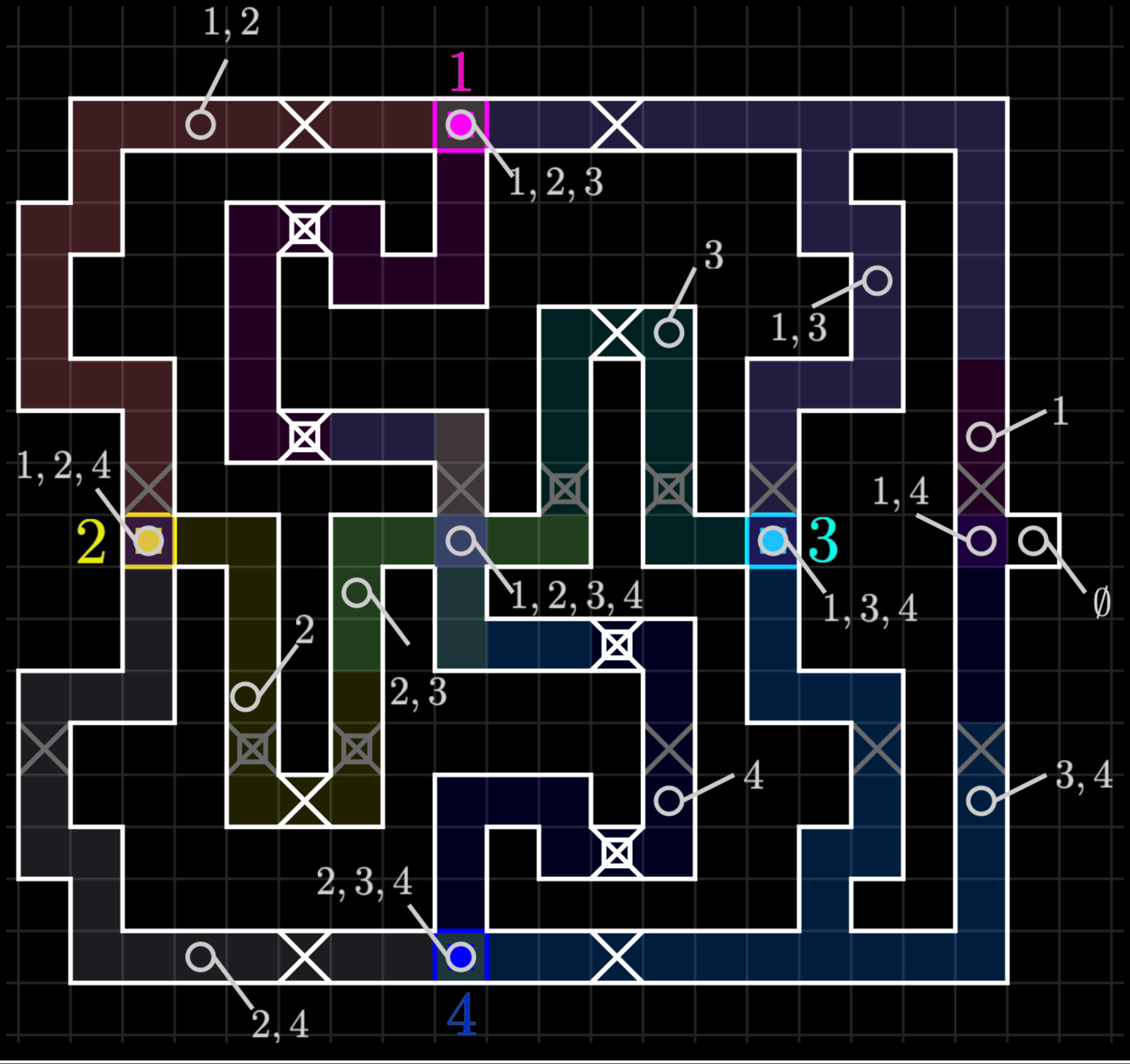


For k=18

d=4➔2^4=16 viewpoints (marked with the guard labels of guards that see them)

Larger even k?

Polyominoes with Holes: Lower Bound



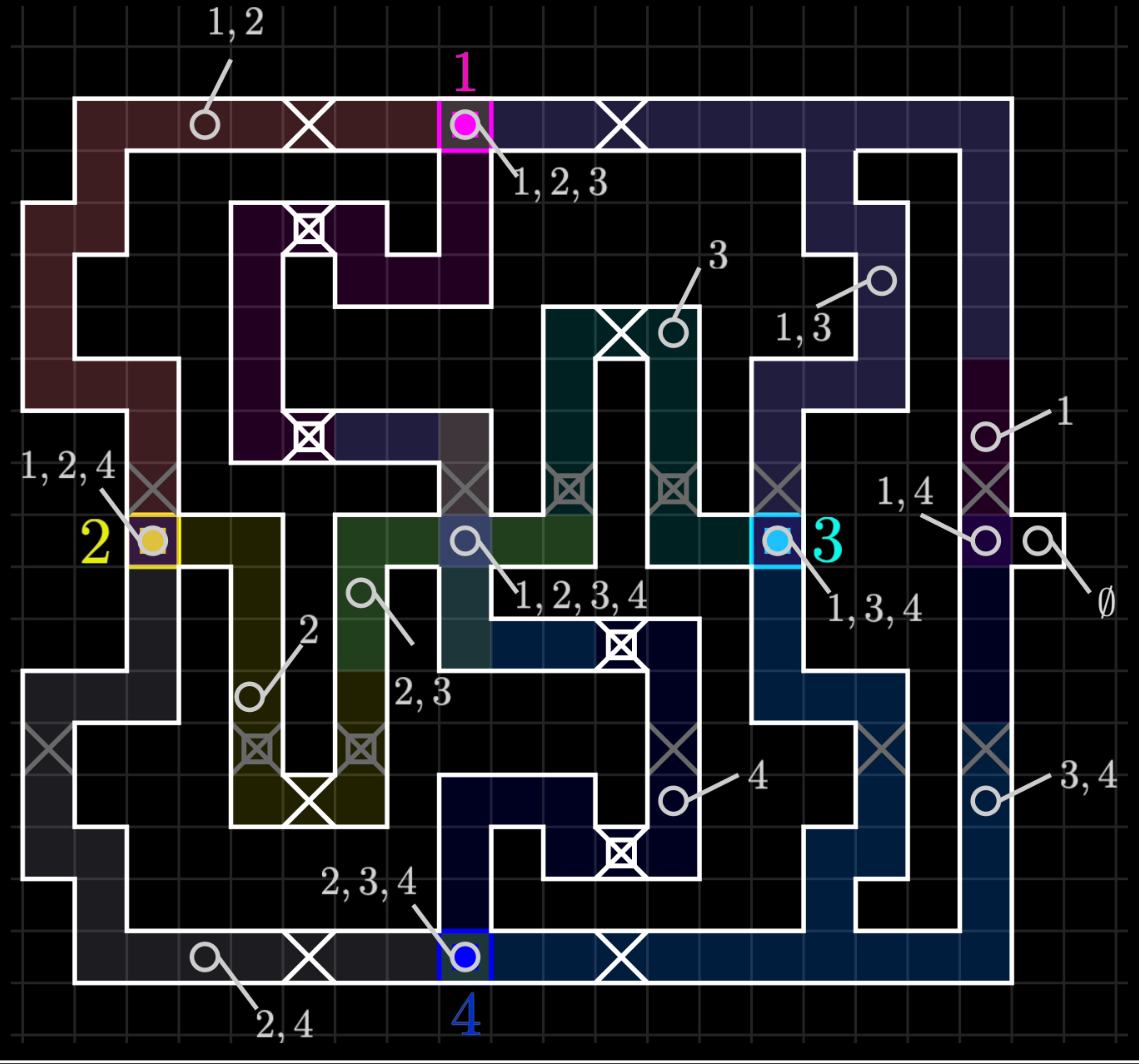
For k=18

d=4➔2^4=16 viewpoints (marked with the guard labels of guards that see them)

Larger even k?

Extend the edges at locations marked with ╳:

Polyominoes with Holes: Lower Bound



For k=18

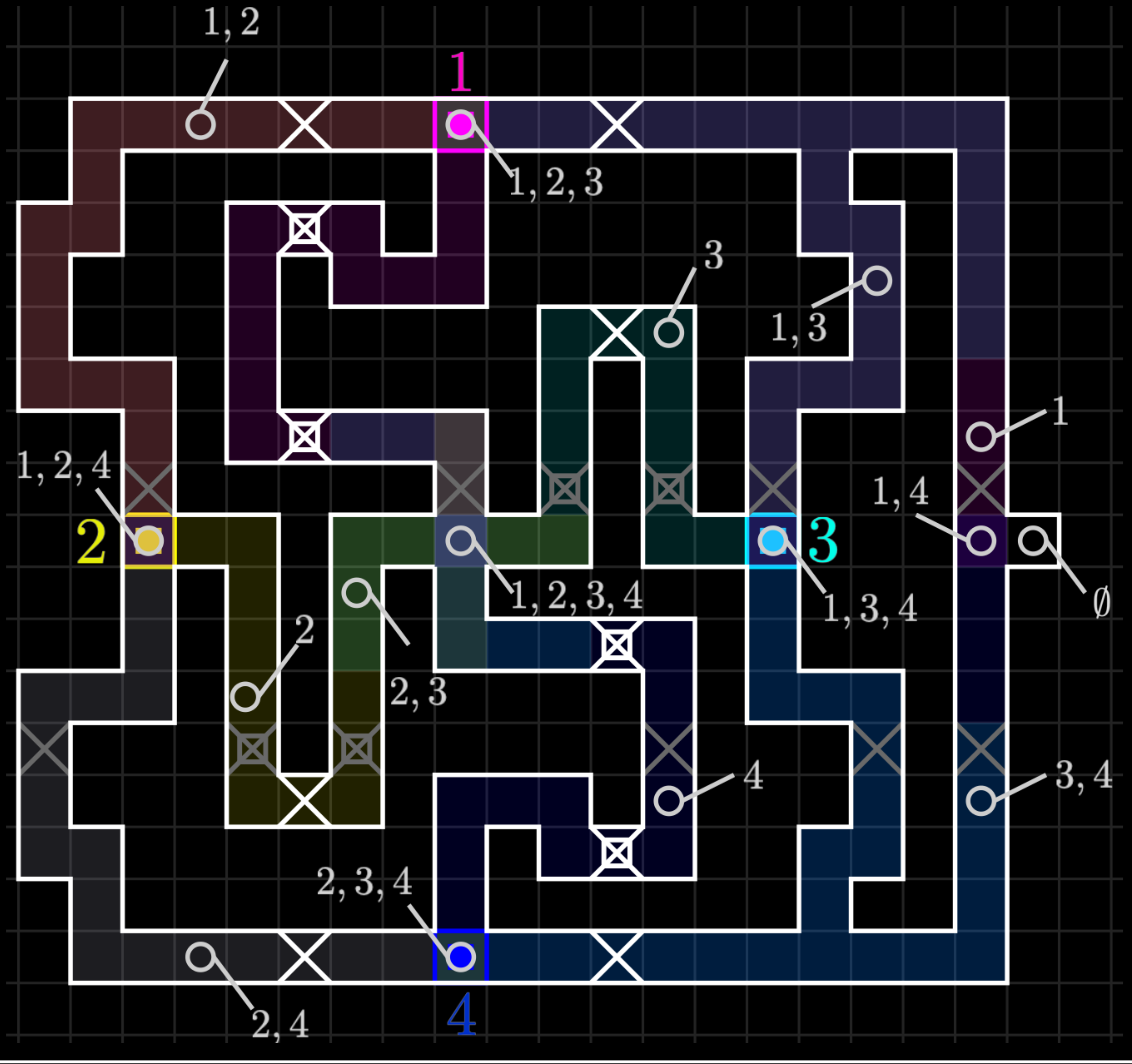
d=4➔2^4=16 viewpoints (marked with the guard labels of guards that see them)

Larger even k?

Extend the edges at locations marked with ╳:

- Alternate between the white and gray crosses

Polyominoes with Holes: Lower Bound



For k=18

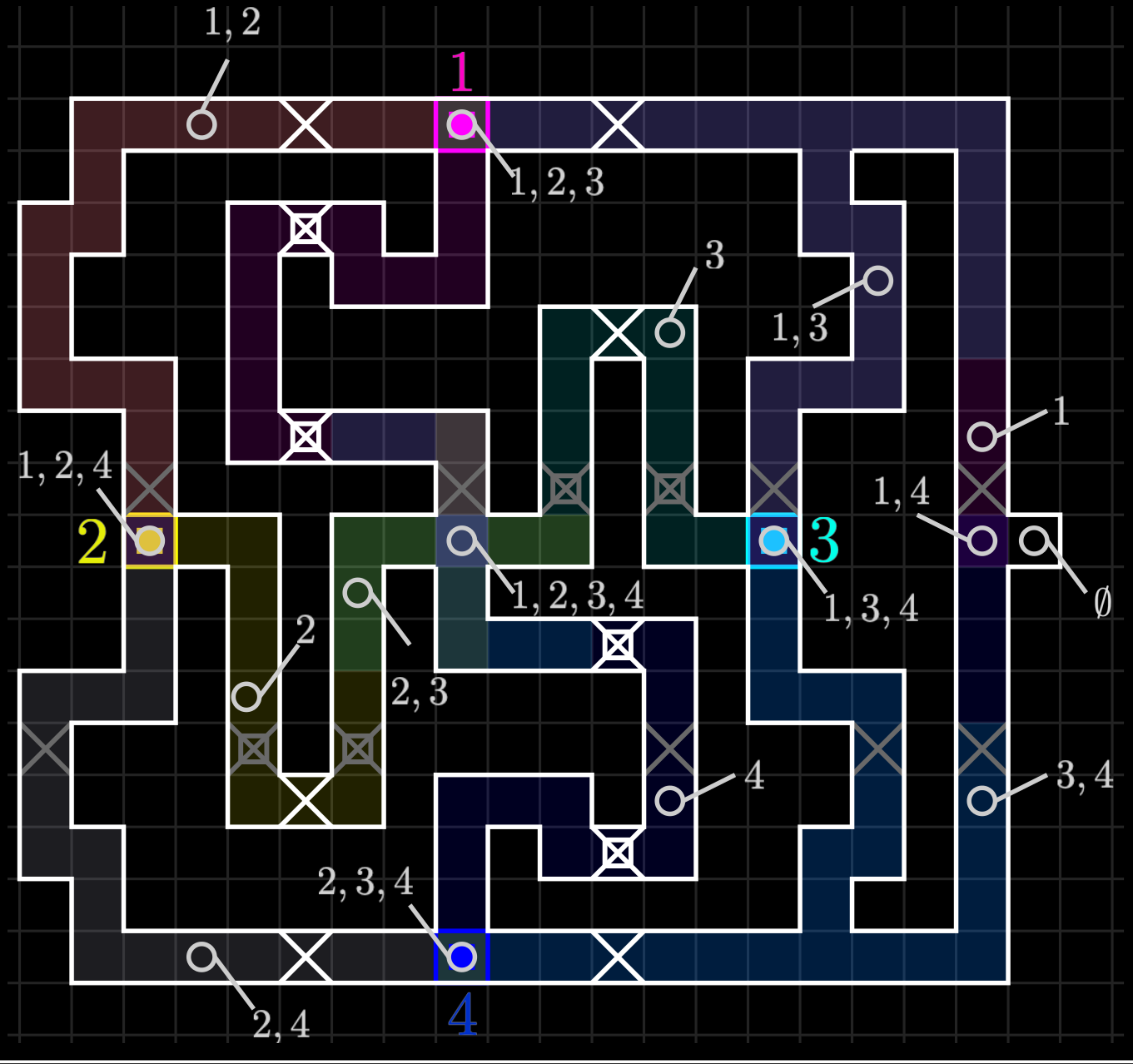
d=4➔2^4=16 viewpoints (marked with the guard labels of guards that see them)

Larger even k?

Extend the edges at locations marked with ╳:

- Alternate between the white and gray crosses
- At ╳: insert two unit squares

Polyominoes with Holes: Lower Bound



For k=18

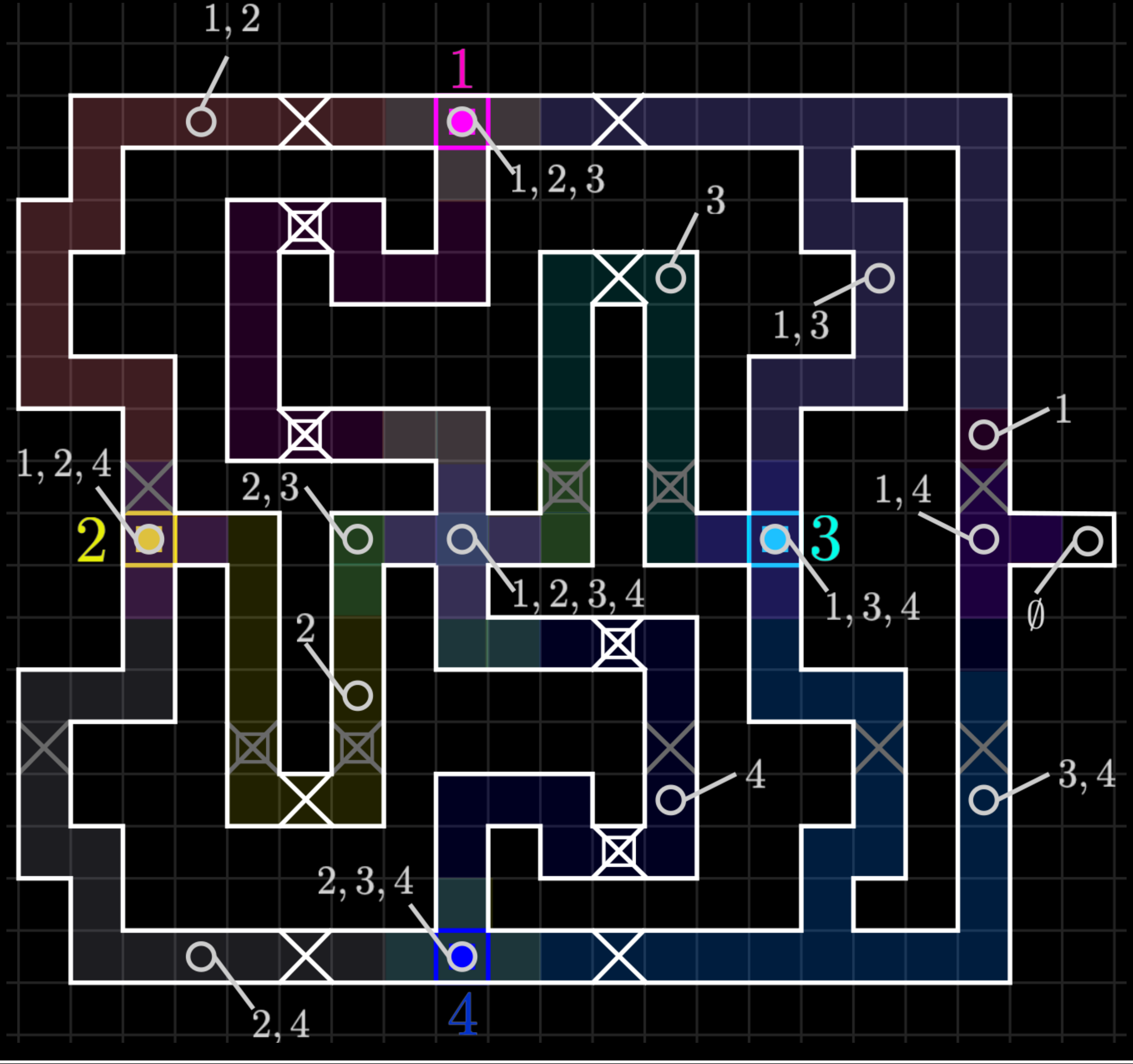
d=4➔2^4=16 viewpoints (marked with the guard labels of guards that see them)

Larger even k?

Extend the edges at locations marked with ╳:

- Alternate between the white and gray crosses
- At ╳: insert two unit squares
- At ⊠: insert one unit square

Polyominoes with Holes: Lower Bound



For k=19

d=4➔2^4=16 viewpoints (marked with the guard labels of guards that see them)

Larger **odd** k?

Extend the edges at locations marked with ╳:

- Alternate between the white and gray crosses
- At ╳: insert two unit squares
- At ☒: insert one unit square

Polyominoes with Holes

Theorem 2: For large enough $k \in \mathbb{N}$, the VC dimension of k-hop visibility in a polyomino with holes is 4.

Polyominoes with Holes

Theorem 2: For large enough $k \in \mathbb{N}$, the VC dimension of k-hop visibility in a polyomino with holes is 4.

Still to show: an upper bound of 4

LINKÖPING UNIVERSITY

Polyominoes with Holes

Theorem 2: For large enough $k \in \mathbb{N}$, the VC dimension of k-hop visibility in a polyomino with holes is 4.

Still to show: an upper bound of 4

Assume that we place five unit-square guards g_1, g_2, g_3, g_4, and g_5 that can be shattered.

Polyominoes with Holes

Theorem 2: For large enough $k \in \mathbb{N}$, the VC dimension of k-hop visibility in a polyomino with holes is 4.

Still to show: an upper bound of 4

Assume that we place five unit-square guards g_1, g_2, g_3, g_4, and g_5 that can be shattered.

$P_{i,j}$, $i \neq j \in \{1,\ldots,5\}$: shortest path from guard g_i to guard g_j along which viewpoint $v_{\{i,j\}}$ is located

Polyominoes with Holes

Theorem 2: For large enough $k \in \mathbb{N}$, the VC dimension of k-hop visibility in a polyomino with holes is 4.

Still to show: an upper bound of 4

Assume that we place five unit-square guards g_1, g_2, g_3, g_4, and g_5 that can be shattered.

$P_{i,j}$, $i \neq j \in \{1,\ldots,5\}$: shortest path from guard g_i to guard g_j along which viewpoint $v_{\{i,j\}}$ is located

➜ $P_{i,j}$ includes the shortest path from g_i to $v_{\{i,j\}}$ and the shortest path from g_j to $v_{\{i,j\}}$

Polyominoes with Holes

Let's start with placing four guards g_1, g_2, g_3, g_4

Polyominoes with Holes

Let's start with placing four guards g_1, g_2, g_3, g_4

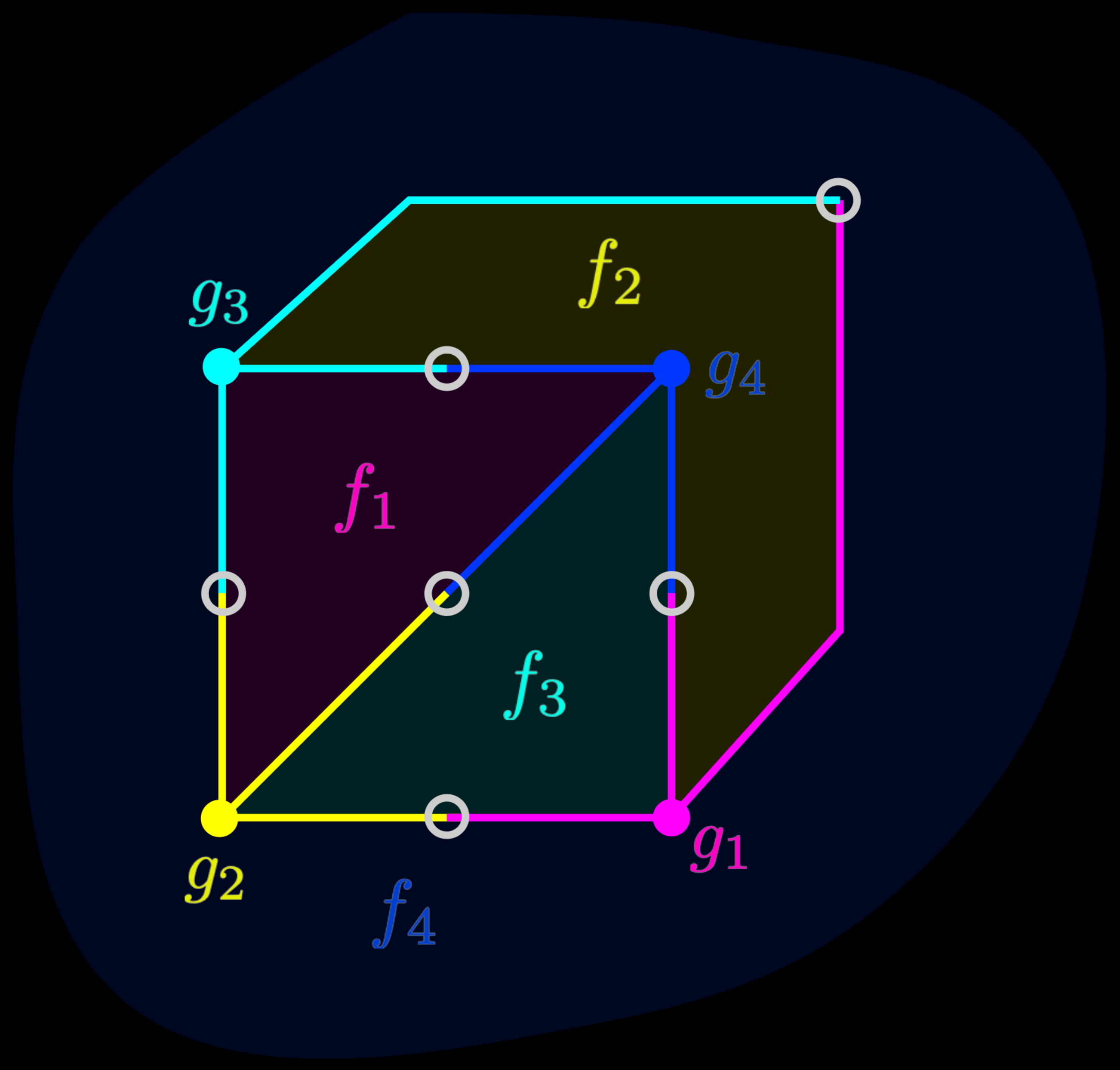
We need to generate all "pair" viewpoints $v_{\{i,j\}}$

Polyominoes with Holes

Let's start with placing four guards g_1, g_2, g_3, g_4

We need to generate all "pair" viewpoints $v_{\{i,j\}}$

➜ We need to embed the graph G_{4p}:



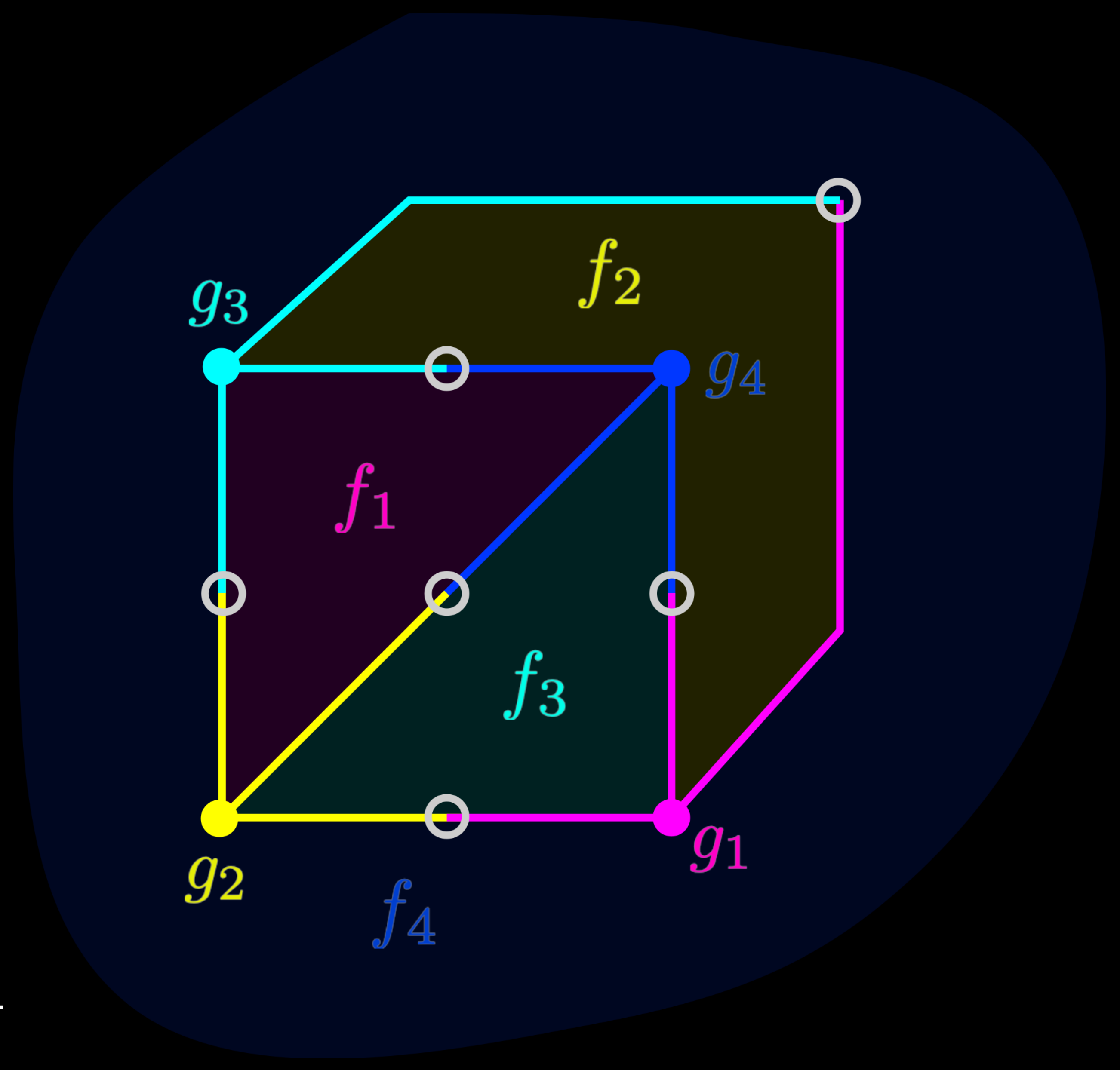
Polyominoes with Holes

Let's start with placing four guards g_1, g_2, g_3, g_4

We need to generate all "pair" viewpoints $v_{\{i,j\}}$

➜ We need to embed the graph G_{4p}:

G_{4p} contains edges representing paths $P_{1,2}$, $P_{1,3}$, $P_{1,4}$, $P_{2,3}$, $P_{2,4}$, and $P_{3,4}$



Polyominoes with Holes

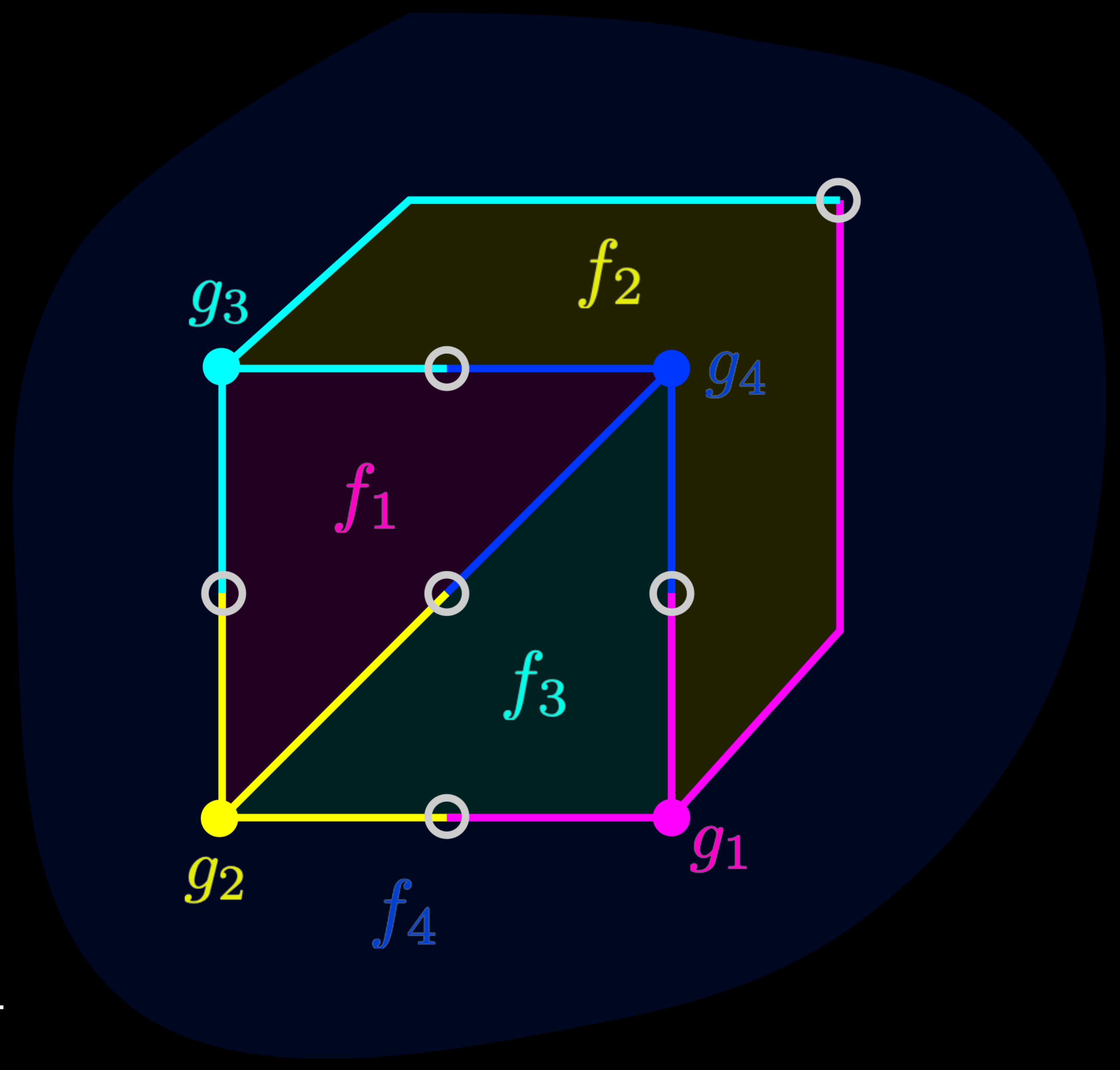
Let's start with placing four guards g_1, g_2, g_3, g_4

We need to generate all "pair" viewpoints $v_{\{i,j\}}$

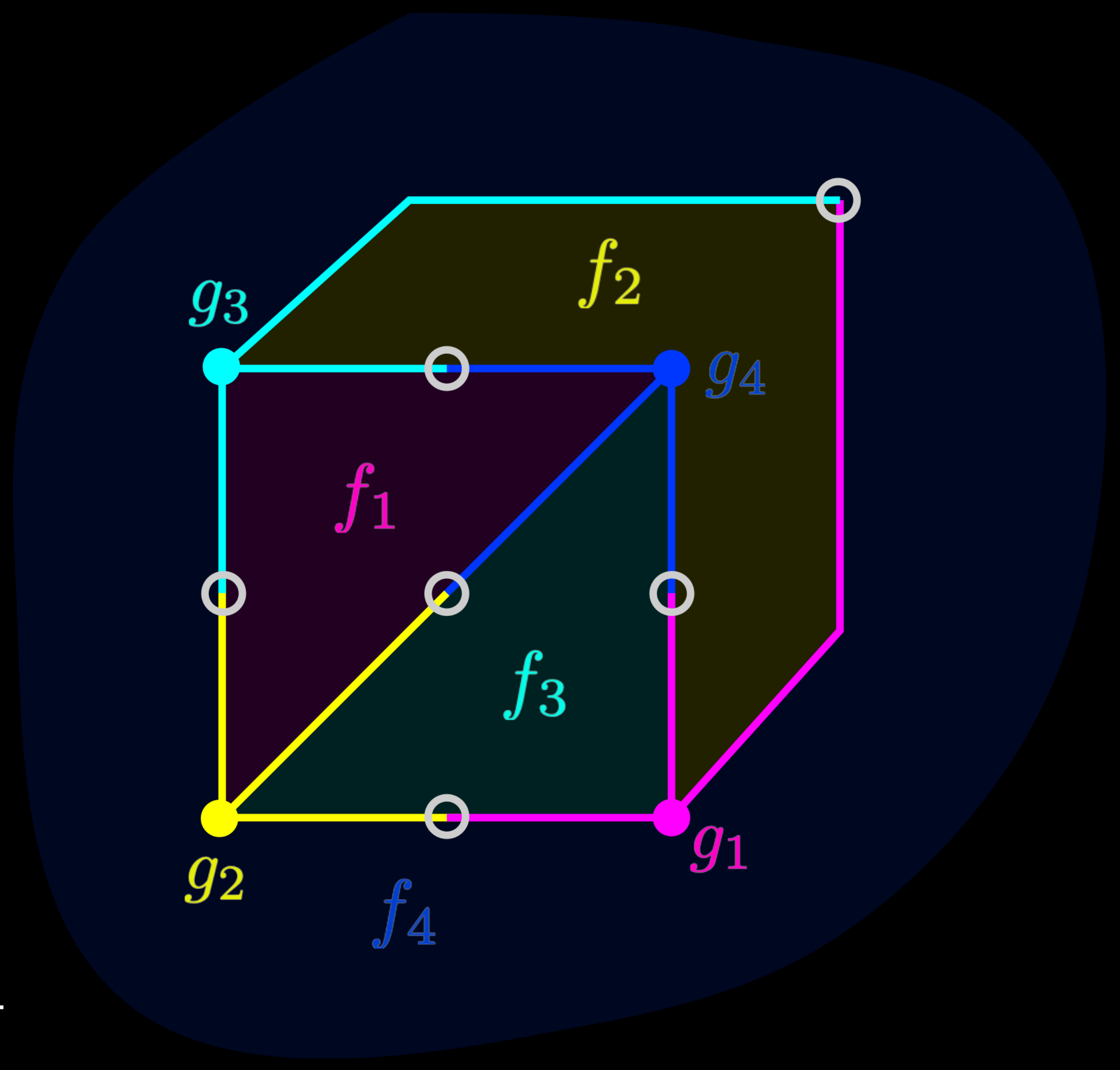
➜ We need to embed the graph G_{4p}:

G_{4p} contains edges representing paths $P_{1,2}$, $P_{1,3}$, $P_{1,4}$, $P_{2,3}$, $P_{2,4}$, and $P_{3,4}$

Face f_i is incident to all guards in $\{g_1, g_2, g_3, g_4\}$ except for g_i



Polyominoes with Holes



Let's start with placing four guards g_1, g_2, g_3, g_4

We need to generate all "pair" viewpoints $v_{\{i,j\}}$

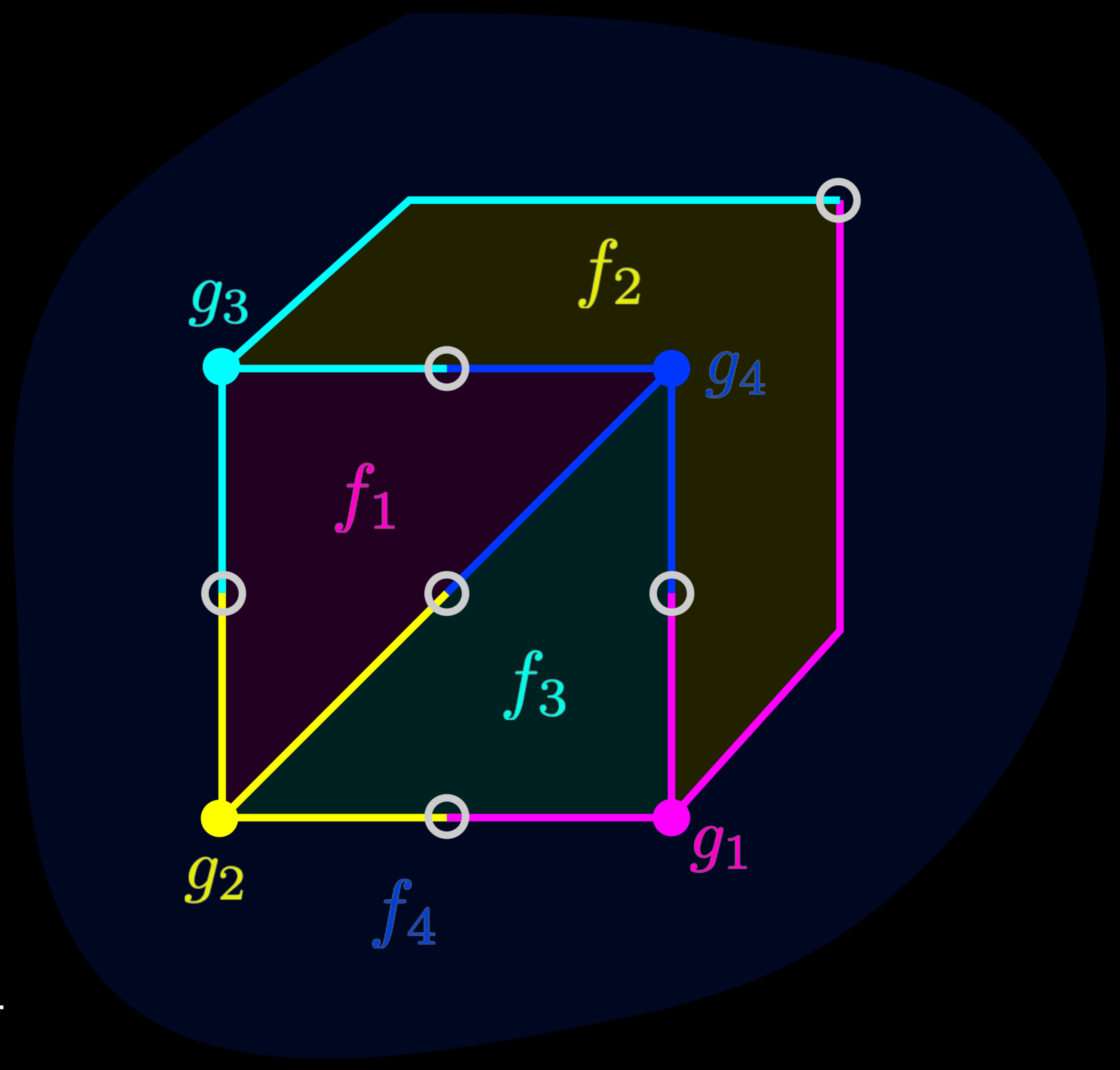
➜ We need to embed the graph G_{4p}:

G_{4p} contains edges representing paths $P_{1,2}$, $P_{1,3}$, $P_{1,4}$, $P_{2,3}$, $P_{2,4}$, and $P_{3,4}$

Face f_i is incident to all guards in $\{g_1, g_2, g_3, g_4\}$ except for g_i

Theorem for simple polyominoes ➜ one of the four faces (actually f_1, f_2, or f_3) contains a hole

Polyominoes with Holes



Let's start with placing four guards g_1, g_2, g_3, g_4

We need to generate all "pair" viewpoints $v_{\{i,j\}}$

➜ We need to embed the graph G_{4p}:

G_{4p} contains edges representing paths $P_{1,2}$, $P_{1,3}$, $P_{1,4}$, $P_{2,3}$, $P_{2,4}$, and $P_{3,4}$

Face f_i is incident to all guards in $\{g_1, g_2, g_3, g_4\}$ except for g_i

Theorem for simple polyominoes ➜ one of the four faces (actually f_1, f_2, or f_3) contains a hole

We need to place the fifth guard in one of the four faces, let it be f_i

Polyominoes with Holes

Let's start with placing four guards g_1, g_2, g_3, g_4

We need to generate all "pair" viewpoints $v_{\{i,j\}}$

➜ We need to embed the graph G_{4p}:

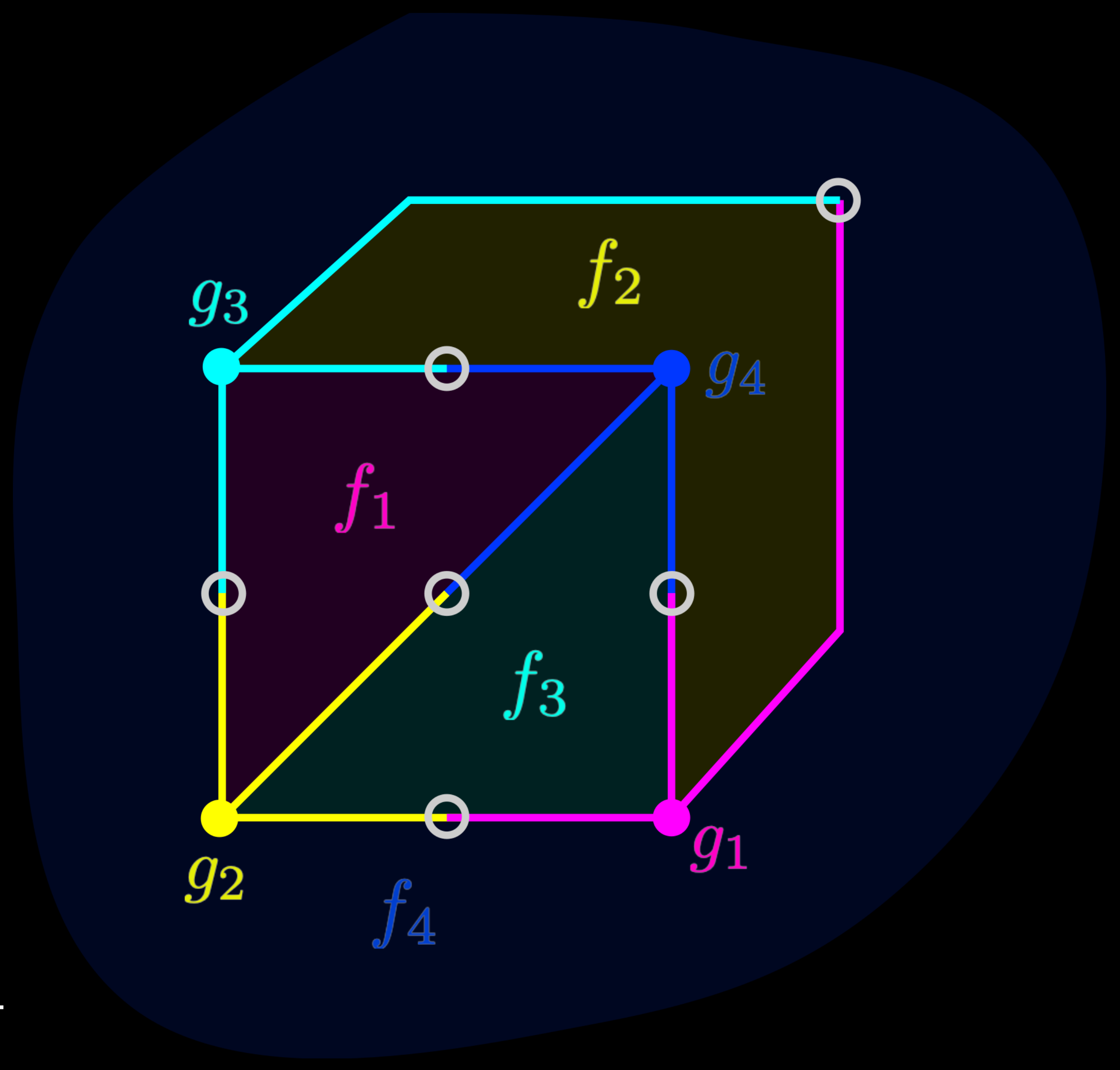
G_{4p} contains edges representing paths $P_{1,2}$, $P_{1,3}$, $P_{1,4}$, $P_{2,3}$, $P_{2,4}$, and $P_{3,4}$

Face f_i is incident to all guards in $\{g_1, g_2, g_3, g_4\}$ except for g_i

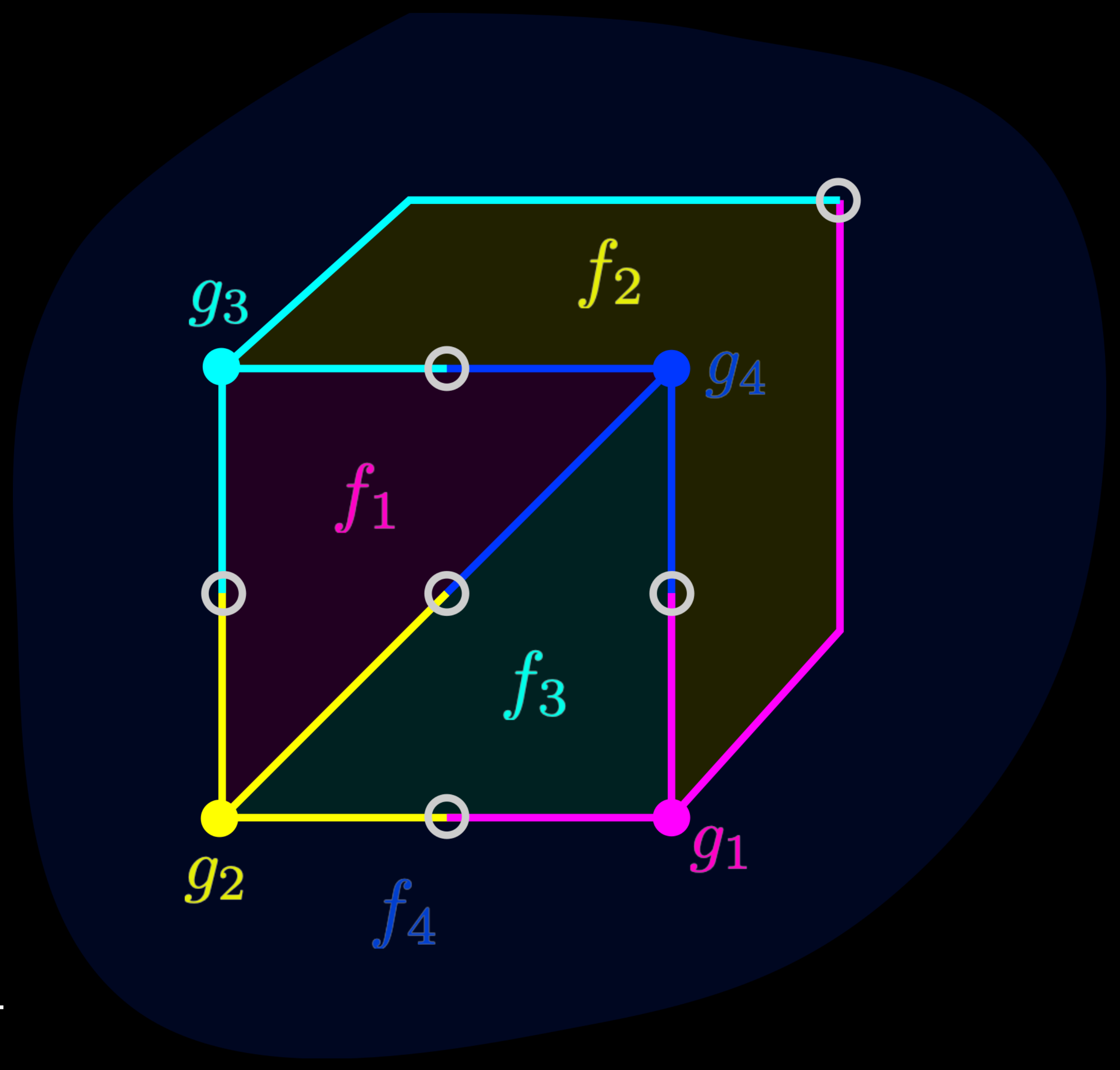
Theorem for simple polyominoes ➜ one of the four faces (actually f_1, f_2, or f_3) contains a hole

We need to place the fifth guard in one of the four faces, let it be f_i

Guard g_i is not incident to f_i



Polyominoes with Holes



Let's start with placing four guards g_1, g_2, g_3, g_4

We need to generate all "pair" viewpoints $v_{\{i,j\}}$

➜ We need to embed the graph G_{4p}:

G_{4p} contains edges representing paths $P_{1,2}$, $P_{1,3}$, $P_{1,4}$, $P_{2,3}$, $P_{2,4}$, and $P_{3,4}$

Face f_i is incident to all guards in $\{g_1, g_2, g_3, g_4\}$ except for g_i

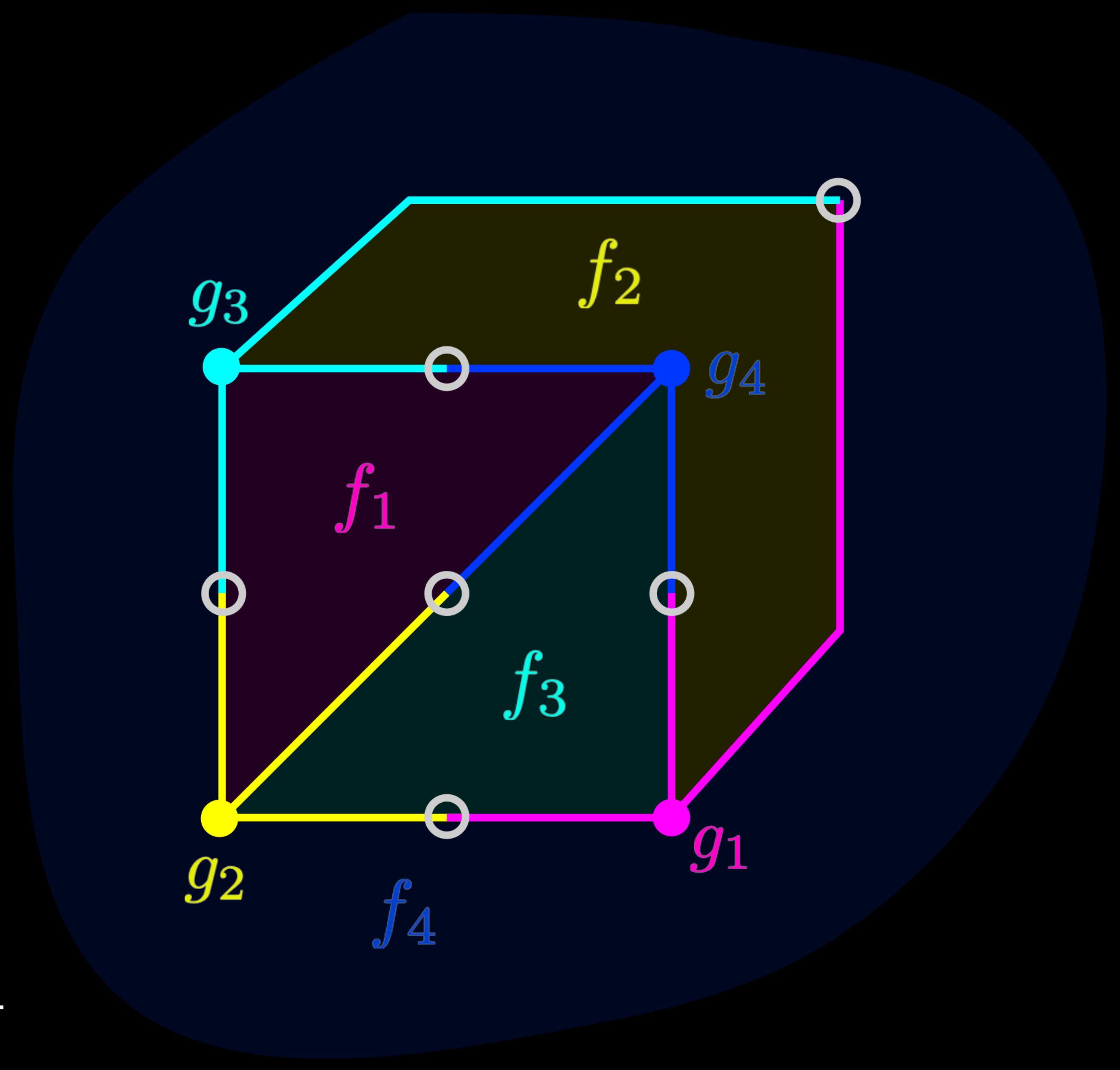
Theorem for simple polyominoes ➜ one of the four faces (actually f_1, f_2, or f_3) contains a hole

We need to place the fifth guard in one of the four faces, let it be f_i

Guard g_i is not incident to f_i

➜ Path $P_{i,5}$ must intersect at least one of the other paths represented by the edges in our graph: $P_{j,\ell}$

Polyominoes with Holes



Let's start with placing four guards g_1, g_2, g_3, g_4

We need to generate all "pair" viewpoints $v_{\{i,j\}}$

➜ We need to embed the graph G_{4p}:

G_{4p} contains edges representing paths $P_{1,2}$, $P_{1,3}$, $P_{1,4}$, $P_{2,3}$, $P_{2,4}$, and $P_{3,4}$

Face f_i is incident to all guards in $\{g_1, g_2, g_3, g_4\}$ except for g_i

Theorem for simple polyominoes ➜ one of the four faces (actually f_1, f_2, or f_3) contains a hole

We need to place the fifth guard in one of the four faces, let it be f_i

Guard g_i is not incident to f_i

➜ Path $P_{i,5}$ must intersect at least one of the other paths represented by the edges in our graph: $P_{j,\ell}$

Rest-budget observation

➜ One of the viewpoints $v_{\{i,5\}}$ and $v_{\{j,\ell\}}$ cannot be realized (a guard from the other pair will always see such a viewpoint) ↯

Computational Complexity: NP-Completeness for 1-Thin Polyominoes with Holes*

* Without holes: the dual is a tree, and we can solve it in linear time!

NP-Completeness for 1-Thin Polyominoes with Holes

- **Theorem 3:** For every $k \geq 2$, the decision version of the M*k*GP is NP-complete, even in 1-thin polyominoes with holes.

NP-Completeness for 1-Thin Polyominoes with Holes

- **Theorem 3:** For every $k \geq 2$, the decision version of the MkGP is NP-complete, even in 1-thin polyominoes with holes.
- Membership in NP✓

NP-Completeness for 1-Thin Polyominoes with Holes

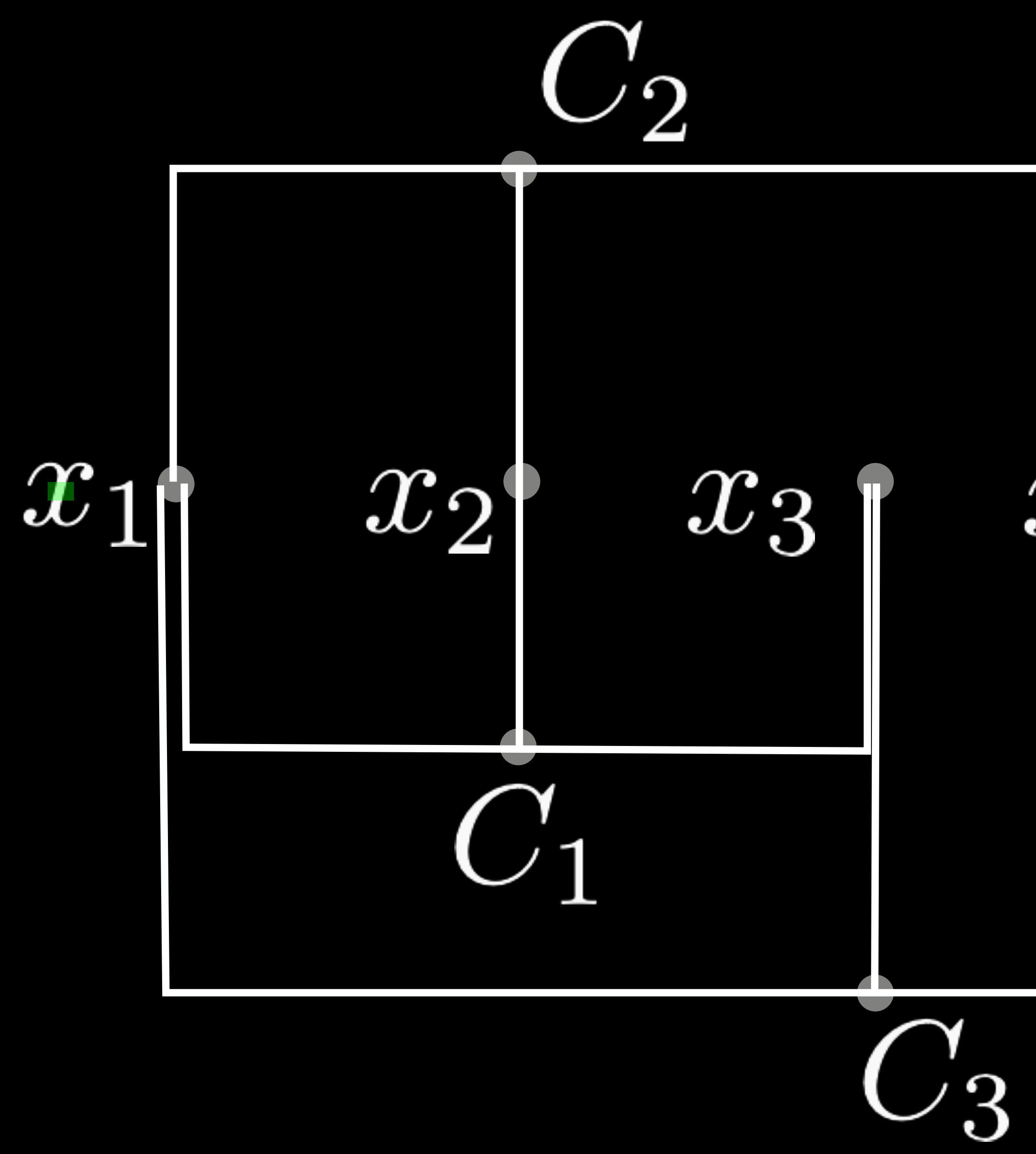
- **Theorem 3:** For every $k \geq 2$, the decision version of the MkGP is NP-complete, even in 1-thin polyominoes with holes.
- Membership in NP✓
- We reduce from PLANAR MONOTONE 3SAT

Global Construction (k=2)

$$\varphi = (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_1 \vee x_2 \vee x_4) \wedge (\overline{x_1} \vee \overline{x_3} \vee \overline{x_4})$$

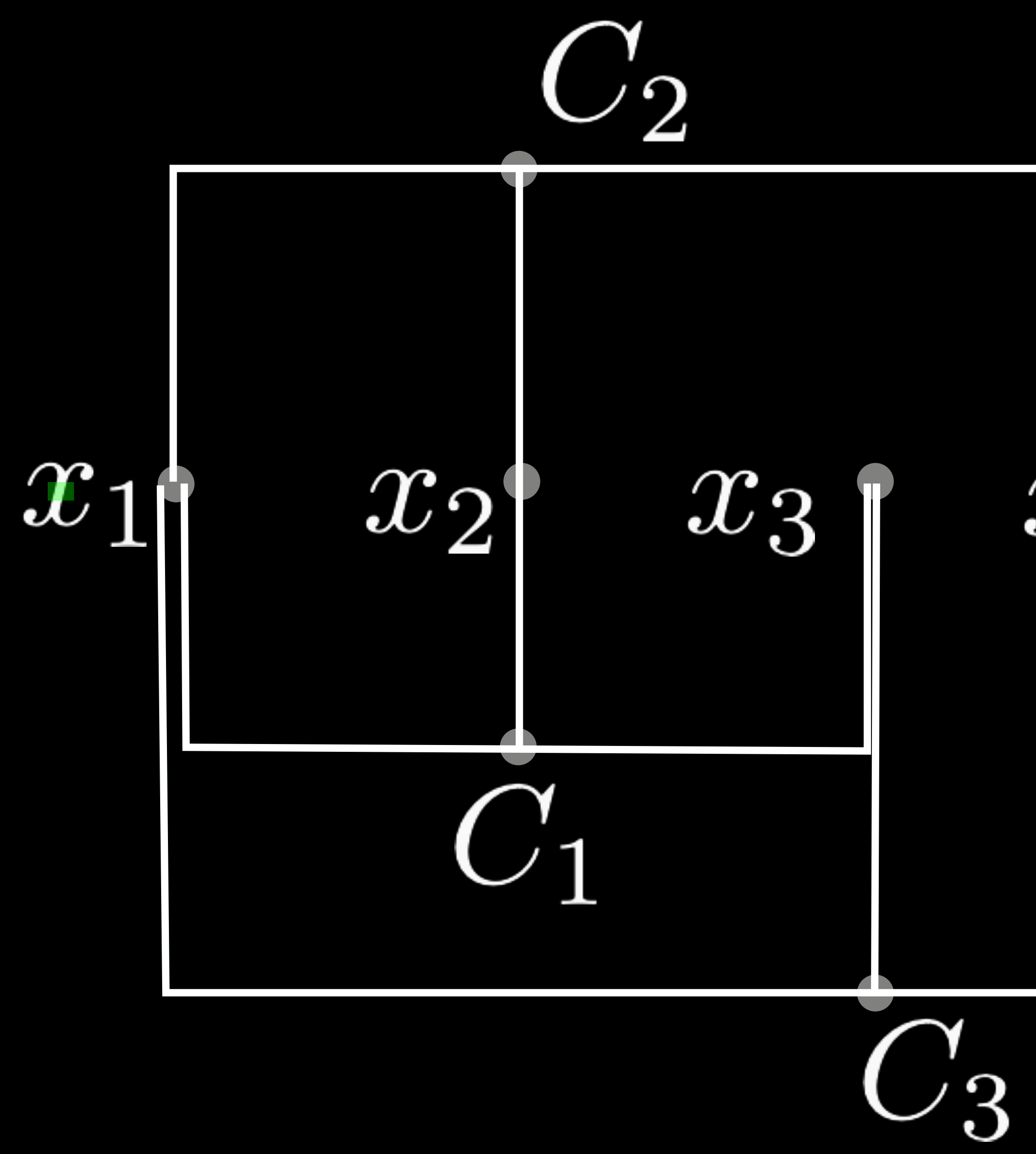
Global Construction (k=2)

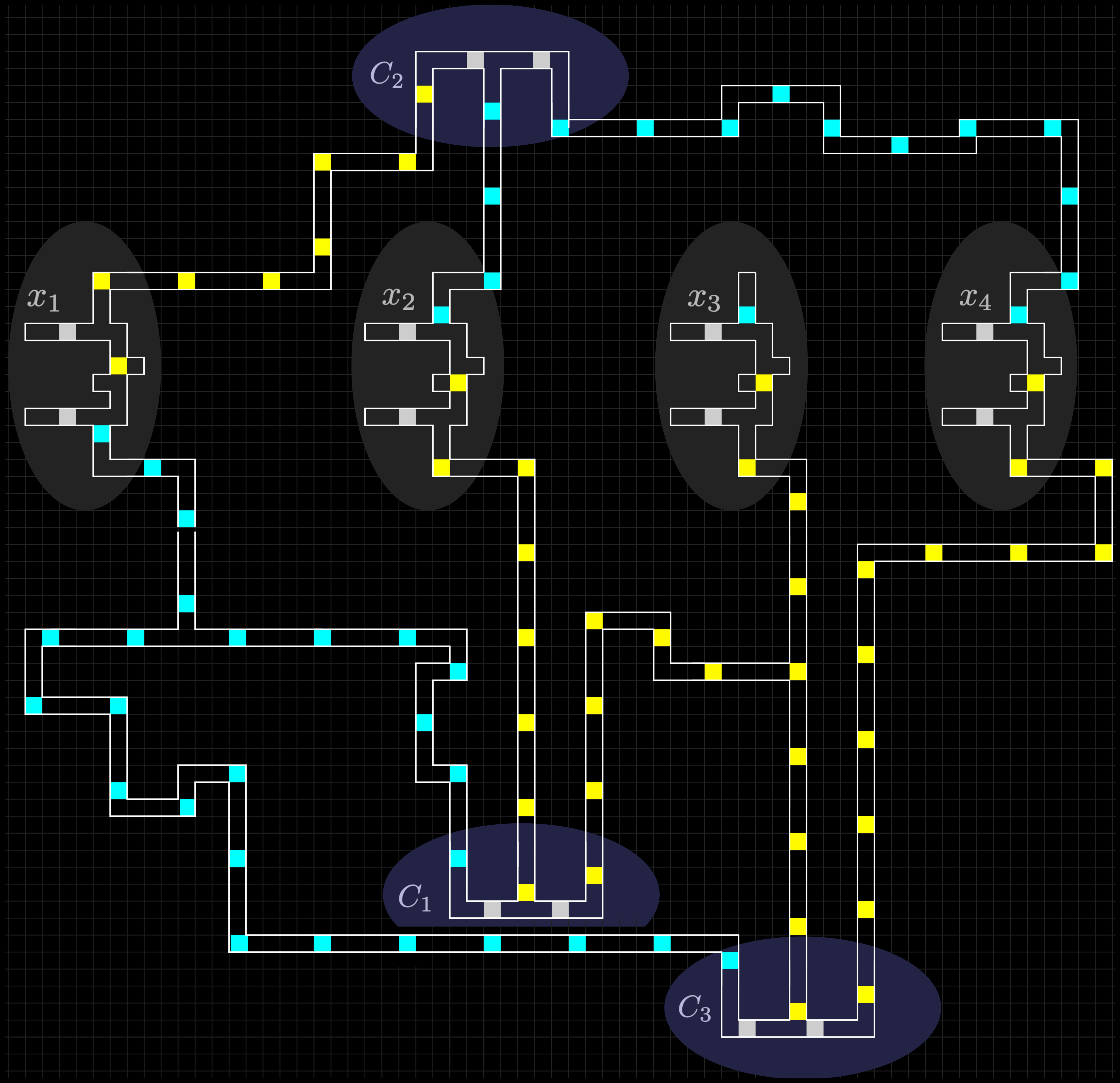
$$\varphi = (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_1 \vee x_2 \vee x_4) \wedge (\overline{x_1} \vee \overline{x_3} \vee \overline{x_4})$$



Global Construction (k=2)

$$\varphi = (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_1 \vee x_2 \vee x_4) \wedge (\overline{x_1} \vee \overline{x_3} \vee \overline{x_4})$$





A Linear-Time 4-Approximation for Simple 2-Thin Polyominoes

Linear-Time 4-Appx for Simple 2-Thin Polyominoes

- There is a PTAS for *k*-hop domination in H-minor free graphs based on local search [*], but runtime may be infeasible for extremely large *n*

[*] Arnold Filtser and Hung Le. Clan embeddings into trees, and low treewidth graphs. In Symposium on Theory of Computing, pages 342–355, 2021.
[#] Glencora Borradaile and Hung Le. Optimal dynamic program for r-domination problems over tree decompositions. In International Symposium on Parameterized and Exact Computation (IPEC), pages 8:1–8:23, 2017.
[+] A. Karim Abu-Affash, Paz Carmi, and Adi Krasin. A linear-time algorithm for minimum k-hop dominating set of a cactus graph. Discrete Applied Mathematics, 320:488–499, 2022

Linear-Time 4-Appx for Simple 2-Thin Polyominoes

- There is a PTAS for *k*-hop domination in H-minor free graphs based on local search [*], but runtime may be infeasible for extremely large *n*
- Exact algorithm for graphs with treewidth *tw* has runtime $O((2k+1)^{tw}\ n)$

[*] Arnold Filtser and Hung Le. Clan embeddings into trees, and low treewidth graphs. In Symposium on Theory of Computing, pages 342–355, 2021.
[#] Glencora Borradaile and Hung Le. Optimal dynamic program for r-domination problems over tree decompositions. In International Symposium on Parameterized and Exact Computation (IPEC), pages 8:1–8:23, 2017.
[+] A. Karim Abu-Affash, Paz Carmi, and Adi Krasin. A linear-time algorithm for minimum k-hop dominating set of a cactus graph. Discrete Applied Mathematics, 320:488–499, 2022

Linear-Time 4-Appx for Simple 2-Thin Polyominoes

- There is a PTAS for *k*-hop domination in H-minor free graphs based on local search [*], but runtime may be infeasible for extremely large *n*
- Exact algorithm for graphs with treewidth *tw* has runtime $O((2k+1)^{tw}\, n)$
  - 2-thin polyominoes have *tw*=2, as K_4 is not a minor

[*] Arnold Filtser and Hung Le. Clan embeddings into trees, and low treewidth graphs. In Symposium on Theory of Computing, pages 342–355, 2021.
[#] Glencora Borradaile and Hung Le. Optimal dynamic program for r-domination problems over tree decompositions. In International Symposium on Parameterized and Exact Computation (IPEC), pages 8:1–8:23, 2017.
[+] A. Karim Abu-Affash, Paz Carmi, and Adi Krasin. A linear-time algorithm for minimum k-hop dominating set of a cactus graph. Discrete Applied Mathematics, 320:488–499, 2022

Linear-Time 4-Appx for Simple 2-Thin Polyominoes

- There is a PTAS for *k*-hop domination in H-minor free graphs based on local search [*], but runtime may be infeasible for extremely large *n*
- Exact algorithm for graphs with treewidth *tw* has runtime $O((2k+1)^{tw}\, n)$
  - 2-thin polyominoes have $tw=2$, as K_4 is not a minor
  - But runtime may be too high if $k=\Omega(n)$

[*] Arnold Filtser and Hung Le. Clan embeddings into trees, and low treewidth graphs. In Symposium on Theory of Computing, pages 342–355, 2021.
[#] Glencora Borradaile and Hung Le. Optimal dynamic program for r-domination problems over tree decompositions. In International Symposium on Parameterized and Exact Computation (IPEC), pages 8:1–8:23, 2017.
[+] A. Karim Abu-Affash, Paz Carmi, and Adi Krasin. A linear-time algorithm for minimum k-hop dominating set of a cactus graph. Discrete Applied Mathematics, 320:488–499, 2022

Linear-Time 4-Appx for Simple 2-Thin Polyominoes

- There is a PTAS for k-hop domination in H-minor free graphs based on local search [*], but runtime may be infeasible for extremely large n
- Exact algorithm for graphs with treewidth tw has runtime $O((2k+1)^{tw}\, n)$
  - 2-thin polyominoes have $tw=2$, as K_4 is not a minor
  - But runtime may be too high if $k=\Omega(n)$
- Runtime of our algorithm does not depend on k

[*] Arnold Filtser and Hung Le. Clan embeddings into trees, and low treewidth graphs. In Symposium on Theory of Computing, pages 342–355, 2021.
[#] Glencora Borradaile and Hung Le. Optimal dynamic program for r-domination problems over tree decompositions. In International Symposium on Parameterized and Exact Computation (IPEC), pages 8:1–8:23, 2017.
[+] A. Karim Abu-Affash, Paz Carmi, and Adi Krasin. A linear-time algorithm for minimum k-hop dominating set of a cactus graph. Discrete Applied Mathematics, 320:488–499, 2022

Linear-Time 4-Appx for Simple 2-Thin Polyominoes

- There is a PTAS for *k*-hop domination in H-minor free graphs based on local search [*], but runtime may be infeasible for extremely large *n*
- Exact algorithm for graphs with treewidth *tw* has runtime $O((2k+1)^{tw}\, n)$
  - 2-thin polyominoes have *tw*=2, as K_4 is not a minor
  - But runtime may be too high if $k=\Omega(n)$
- Runtime of our algorithm does not depend on *k*
- Idea: construct a tree *T* on the polyomino *P* and let *T* guide us in placing guards

[*] Arnold Filtser and Hung Le. Clan embeddings into trees, and low treewidth graphs. In Symposium on Theory of Computing, pages 342–355, 2021.
[#] Glencora Borradaile and Hung Le. Optimal dynamic program for r-domination problems over tree decompositions. In International Symposium on Parameterized and Exact Computation (IPEC), pages 8:1–8:23, 2017.
[+] A. Karim Abu-Affash, Paz Carmi, and Adi Krasin. A linear-time algorithm for minimum k-hop dominating set of a cactus graph. Discrete Applied Mathematics, 320:488–499, 2022

Linear-Time 4-Appx for Simple 2-Thin Polyominoes

- There is a PTAS for k-hop domination in H-minor free graphs based on local search [*], but runtime may be infeasible for extremely large n
- Exact algorithm for graphs with treewidth tw has runtime $O((2k+1)^{tw}\, n)$
  - 2-thin polyominoes have $tw=2$, as K_4 is not a minor
  - But runtime may be too high if $k=\Omega(n)$
- Runtime of our algorithm does not depend on k
- Idea: construct a tree T on the polyomino P and let T guide us in placing guards
- Inspired by the linear-time algorithm for trees [+]

[*] Arnold Filtser and Hung Le. Clan embeddings into trees, and low treewidth graphs. In Symposium on Theory of Computing, pages 342–355, 2021.
[#] Glencora Borradaile and Hung Le. Optimal dynamic program for r-domination problems over tree decompositions. In International Symposium on Parameterized and Exact Computation (IPEC), pages 8:1–8:23, 2017.
[+] A. Karim Abu-Affash, Paz Carmi, and Adi Krasin. A linear-time algorithm for minimum k-hop dominating set of a cactus graph. Discrete Applied Mathematics, 320:488–499, 2022

Linear-Time 4-Appx for Simple 2-Thin Polyominoes

- There is a PTAS for *k*-hop domination in H-minor free graphs based on local search [*], but runtime may be infeasible for extremely large *n*
- Exact algorithm for graphs with treewidth *tw* has runtime $O((2k+1)^{tw}\, n)$
  - 2-thin polyominoes have $tw=2$, as K_4 is not a minor
  - But runtime may be too high if $k=\Omega(n)$
- Runtime of our algorithm does not depend on *k*
- Idea: construct a tree *T* on the polyomino *P* and let *T* guide us in placing guards
- Inspired by the linear-time algorithm for trees [+]
- In each iteration, we place 1, 2, or 4 guards and 1 witness

[*] Arnold Filtser and Hung Le. Clan embeddings into trees, and low treewidth graphs. In Symposium on Theory of Computing, pages 342–355, 2021.
[#] Glencora Borradaile and Hung Le. Optimal dynamic program for r-domination problems over tree decompositions. In International Symposium on Parameterized and Exact Computation (IPEC), pages 8:1–8:23, 2017.
[+] A. Karim Abu-Affash, Paz Carmi, and Adi Krasin. A linear-time algorithm for minimum k-hop dominating set of a cactus graph. Discrete Applied Mathematics, 320:488–499, 2022

Linear-Time 4-Appx for Simple 2-Thin Polyominoes

- There is a PTAS for *k*-hop domination in H-minor free graphs based on local search [*], but runtime may be infeasible for extremely large *n*
- Exact algorithm for graphs with treewidth *tw* has runtime $O((2k+1)^{tw}\, n)$
  - 2-thin polyominoes have $tw=2$, as K_4 is not a minor
  - But runtime may be too high if $k=\Omega(n)$
- Runtime of our algorithm does not depend on *k*
- Idea: construct a tree *T* on the polyomino *P* and let *T* guide us in placing guards
- Inspired by the linear-time algorithm for trees [+]
- In each iteration, we place 1, 2, or 4 guards and 1 witness
- Cardinality of witness set is a lower bound

[*] Arnold Filtser and Hung Le. Clan embeddings into trees, and low treewidth graphs. In Symposium on Theory of Computing, pages 342–355, 2021.
[#] Glencora Borradaile and Hung Le. Optimal dynamic program for r-domination problems over tree decompositions. In International Symposium on Parameterized and Exact Computation (IPEC), pages 8:1–8:23, 2017.
[+] A. Karim Abu-Affash, Paz Carmi, and Adi Krasin. A linear-time algorithm for minimum k-hop dominating set of a cactus graph. Discrete Applied Mathematics, 320:488–499, 2022

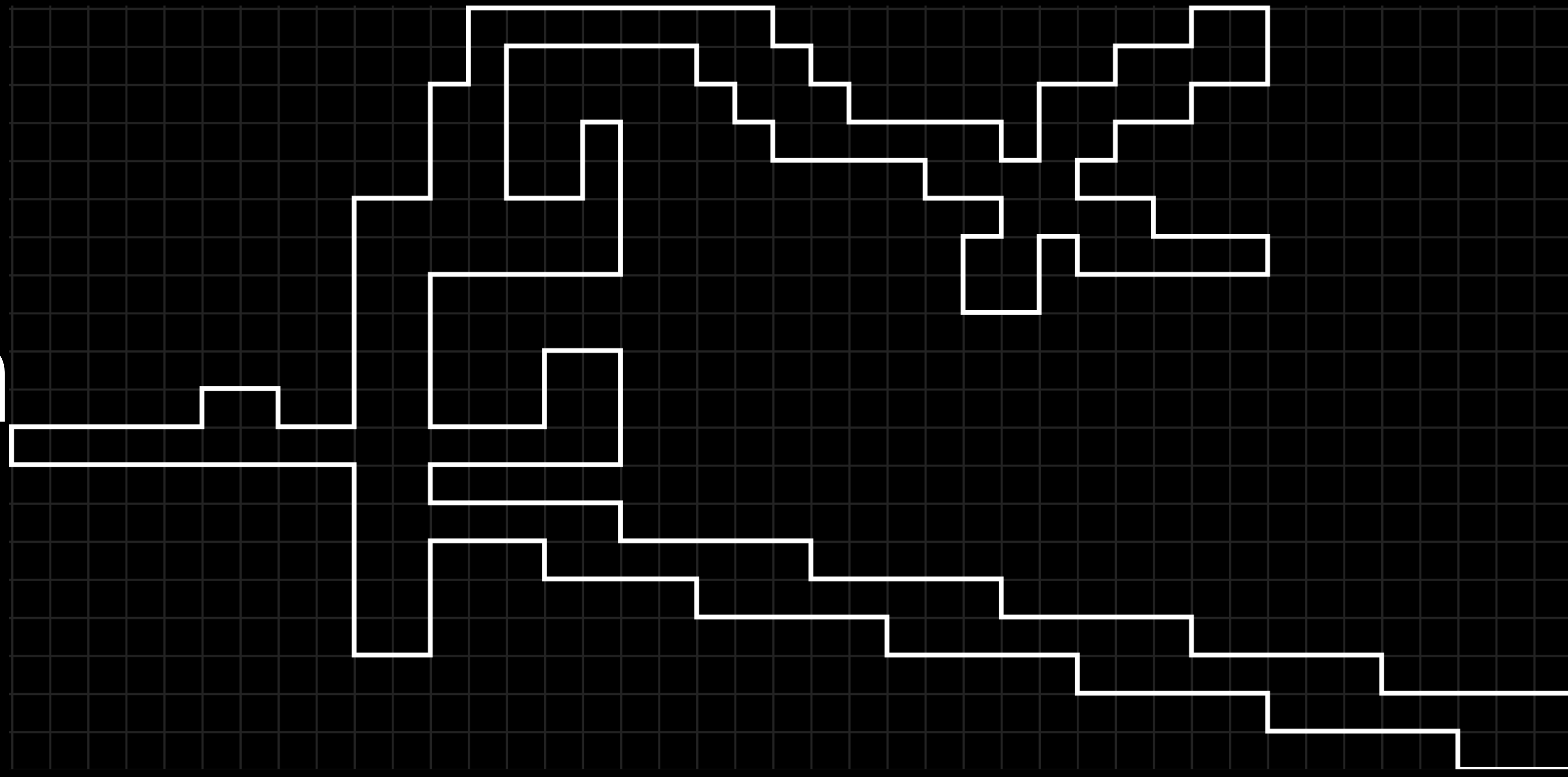
Linear-Time 4-Appx for Simple 2-Thin Polyominoes

- There is a PTAS for k-hop domination in H-minor free graphs based on local search [*], but runtime may be infeasible for extremely large n
- Exact algorithm for graphs with treewidth tw has runtime $O((2k+1)^{tw}\, n)$
  - 2-thin polyominoes have $tw=2$, as K_4 is not a minor
  - But runtime may be too high if $k=\Omega(n)$
- Runtime of our algorithm does not depend on k
- Idea: construct a tree T on the polyomino P and let T guide us in placing guards
- Inspired by the linear-time algorithm for trees [+]
- In each iteration, we place 1, 2, or 4 guards and 1 witness
- Cardinality of witness set is a lower bound

➡ 4-approximation

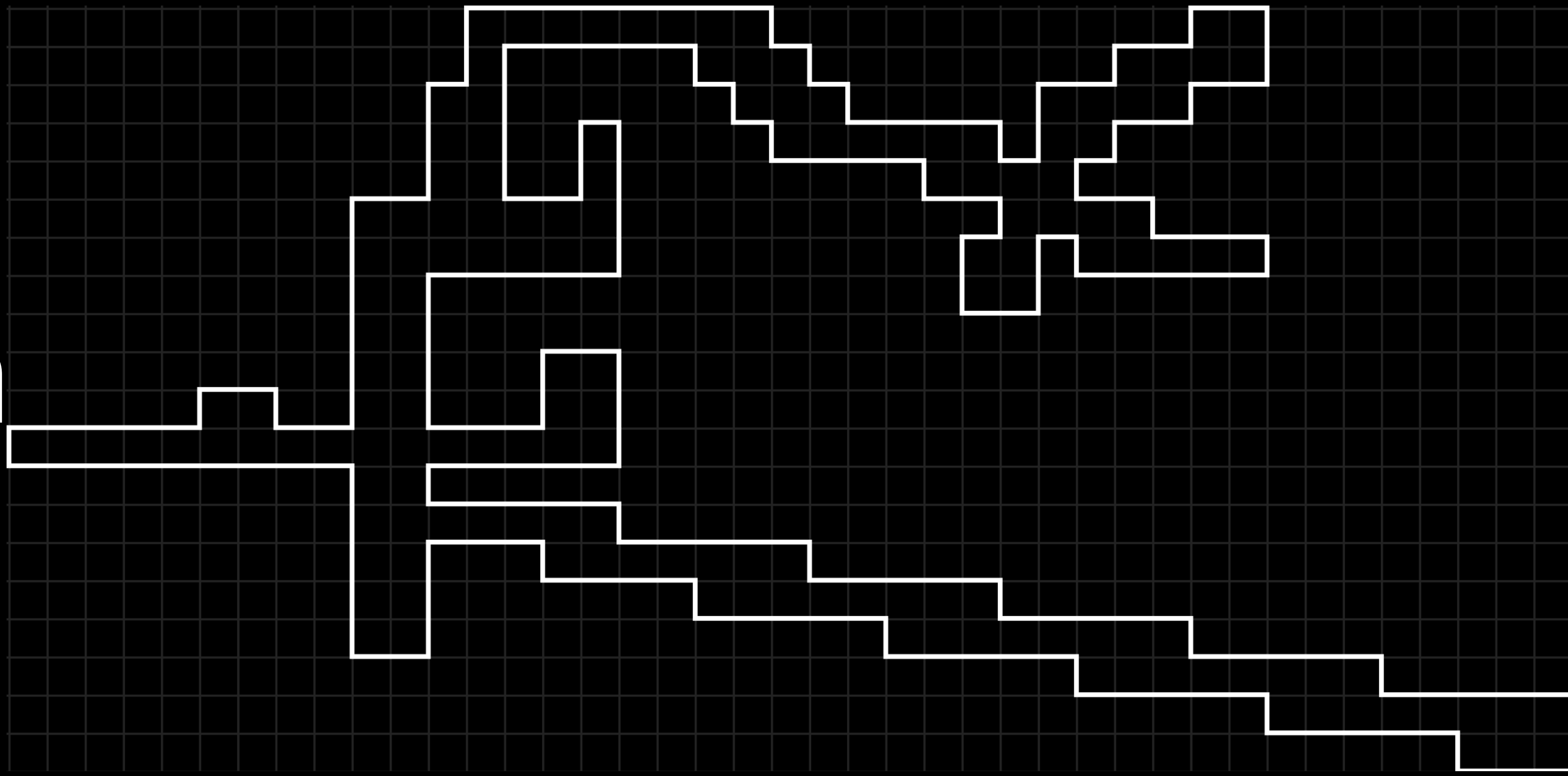
[*] Arnold Filtser and Hung Le. Clan embeddings into trees, and low treewidth graphs. In Symposium on Theory of Computing, pages 342–355, 2021.
[#] Glencora Borradaile and Hung Le. Optimal dynamic program for r-domination problems over tree decompositions. In International Symposium on Parameterized and Exact Computation (IPEC), pages 8:1–8:23, 2017.
[+] A. Karim Abu-Affash, Paz Carmi, and Adi Krasin. A linear-time algorithm for minimum k-hop dominating set of a cactus graph. Discrete Applied Mathematics, 320:488–499, 2022

Constructing a Skeleton Graph

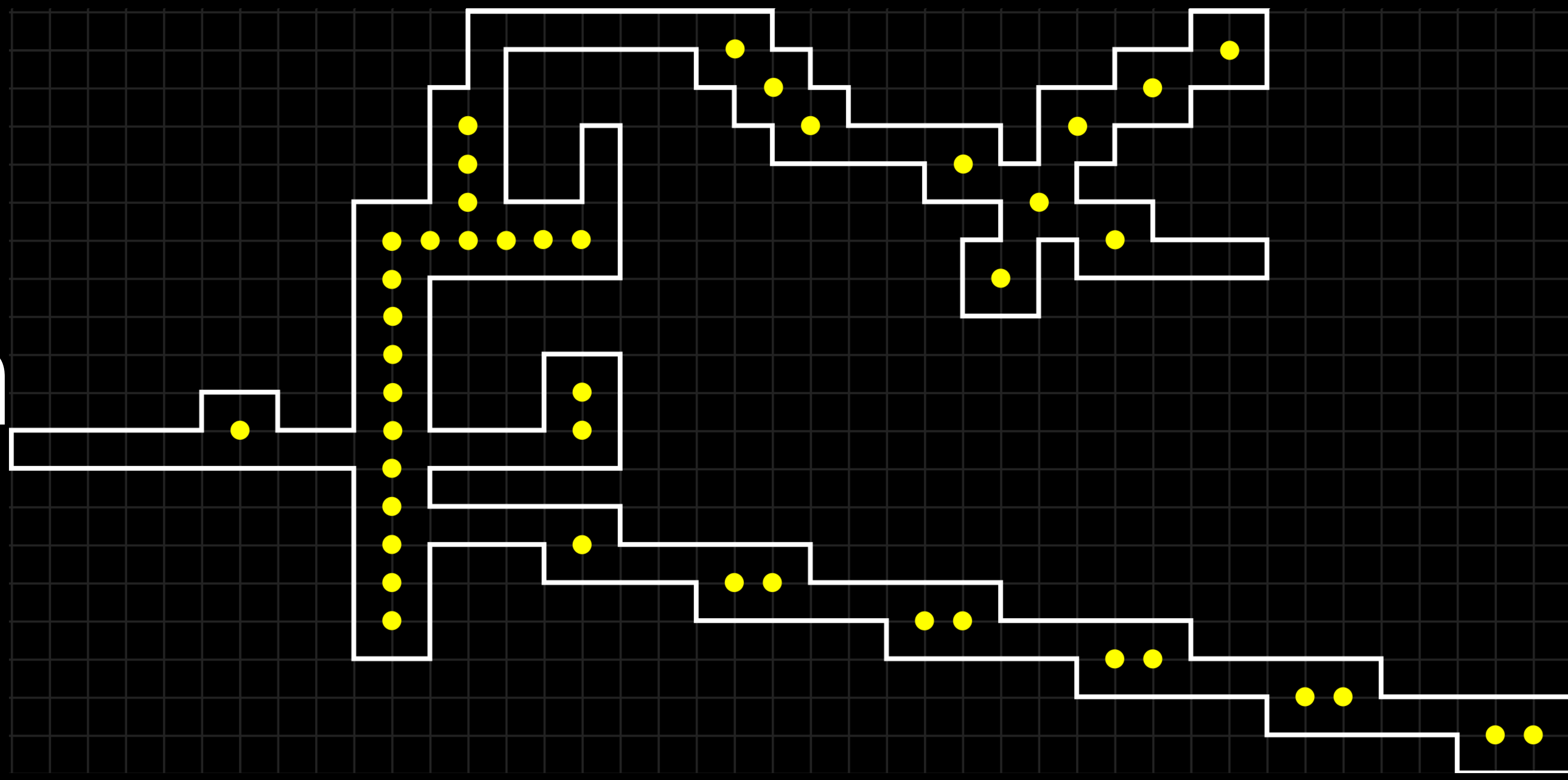


- P: 2-thin polyomino

Constructing a Skeleton Graph

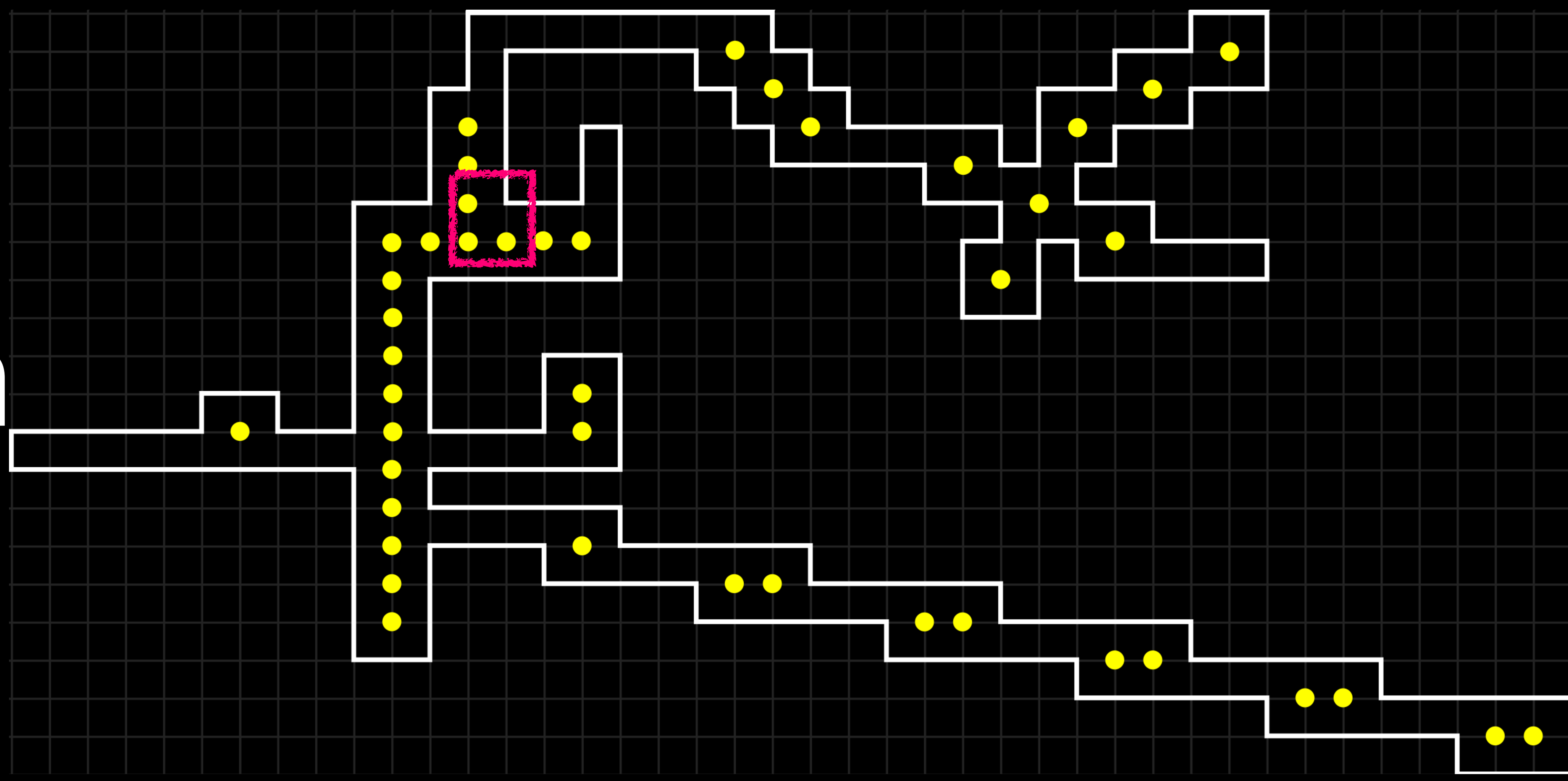


- P: 2-thin polyomino
- Each unit square/cell is incident to four vertices



Constructing a Skeleton Graph

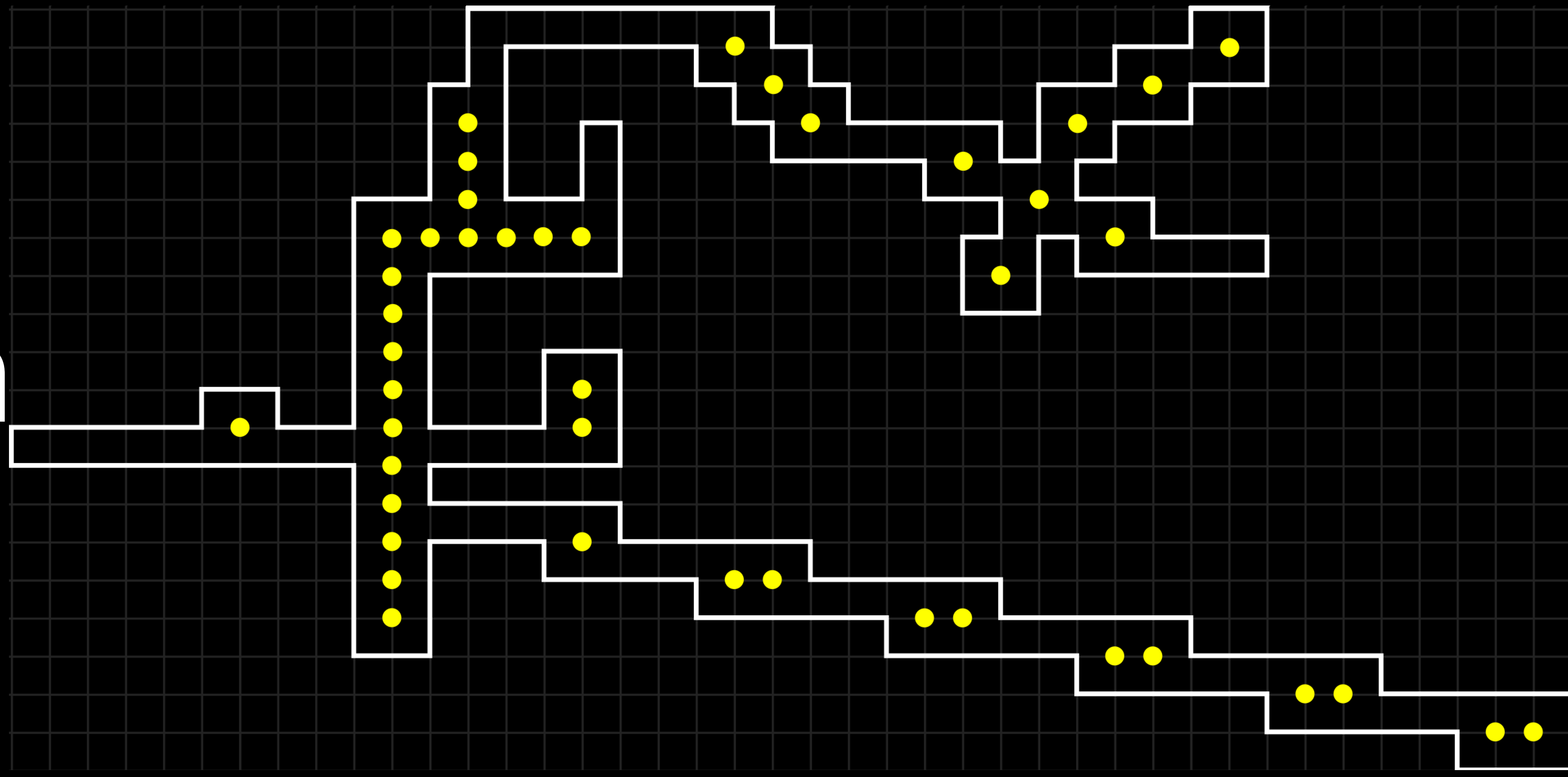
- P: 2-thin polyomino
- Each unit square/cell is incident to four vertices
- Such a vertex is *internal* if it does not lie on P's boundary



Constructing a Skeleton Graph

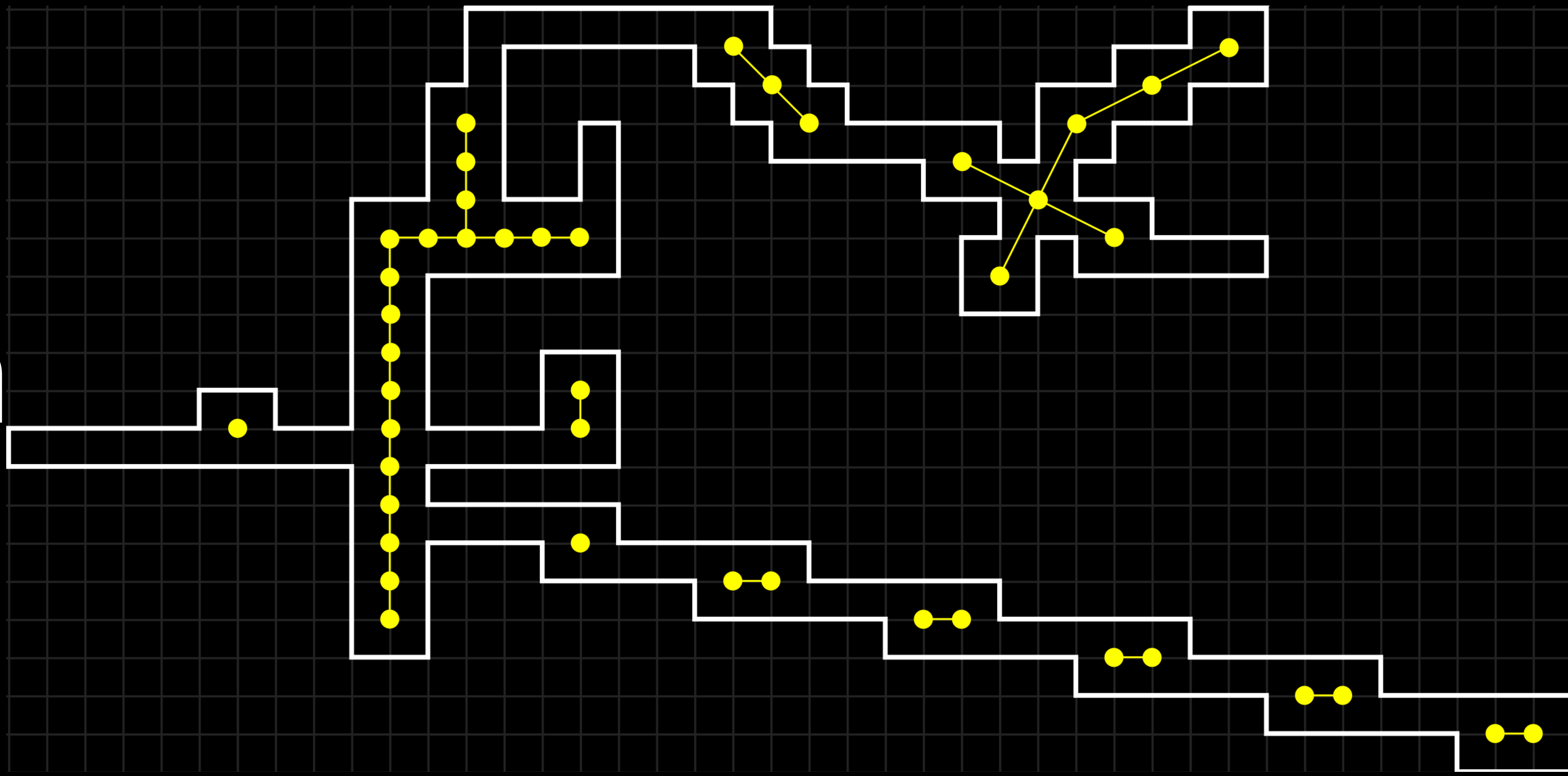
- P: 2-thin polyomino
- Each unit square/cell is incident to four vertices
- Such a vertex is *internal* if it does not lie on P's boundary
- P 2-thin ➜ a unit square can have at most 3 internal vertices

Constructing a Skeleton Graph

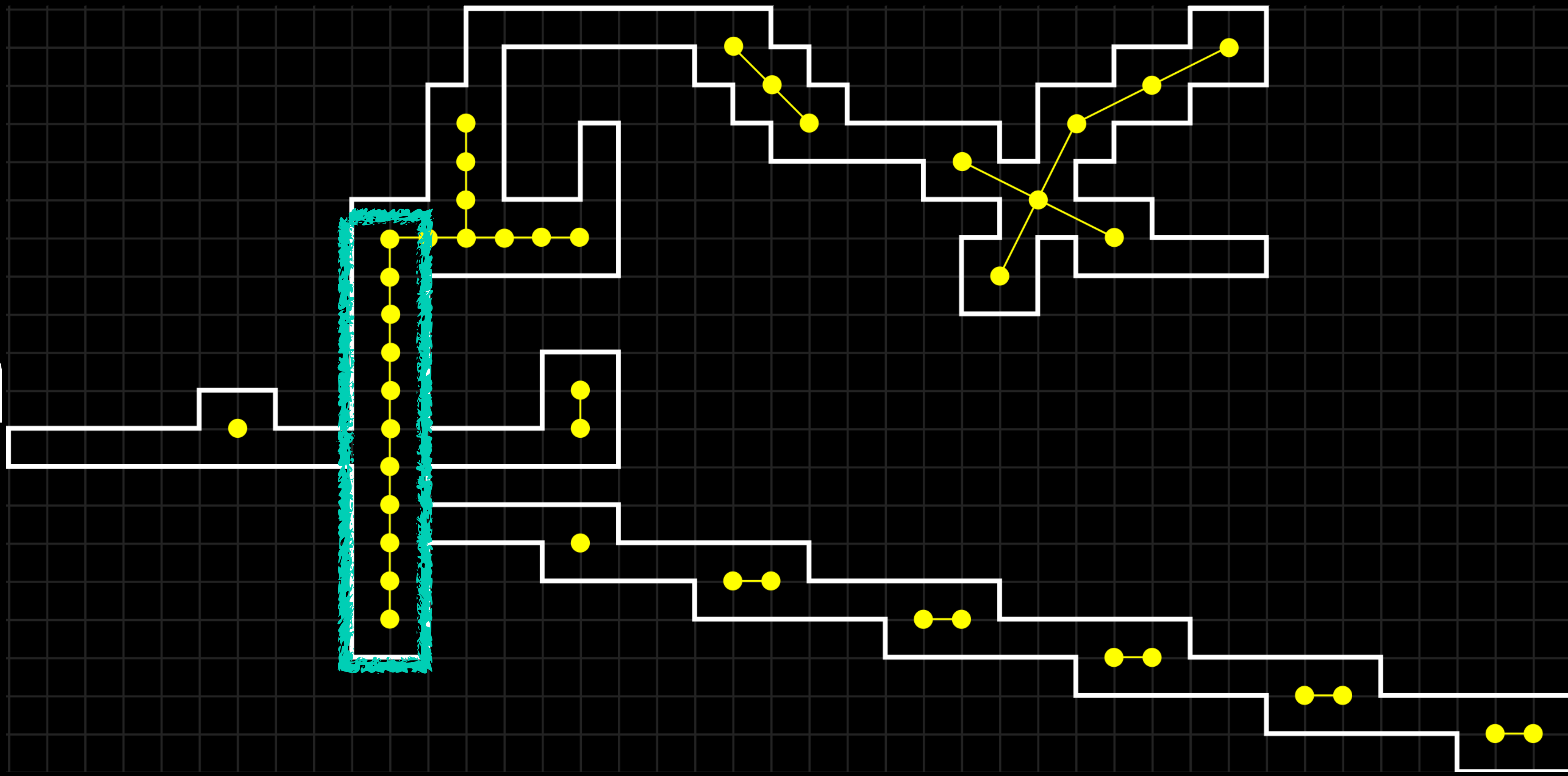


- P: 2-thin polyomino
- Each unit square/cell is incident to four vertices
- Such a vertex is *internal* if it does not lie on P’s boundary
- P 2-thin ➜ a unit square can have at most 3 internal vertices
- I= set of internal vertices of the unit squares in P

Constructing a Skeleton Graph

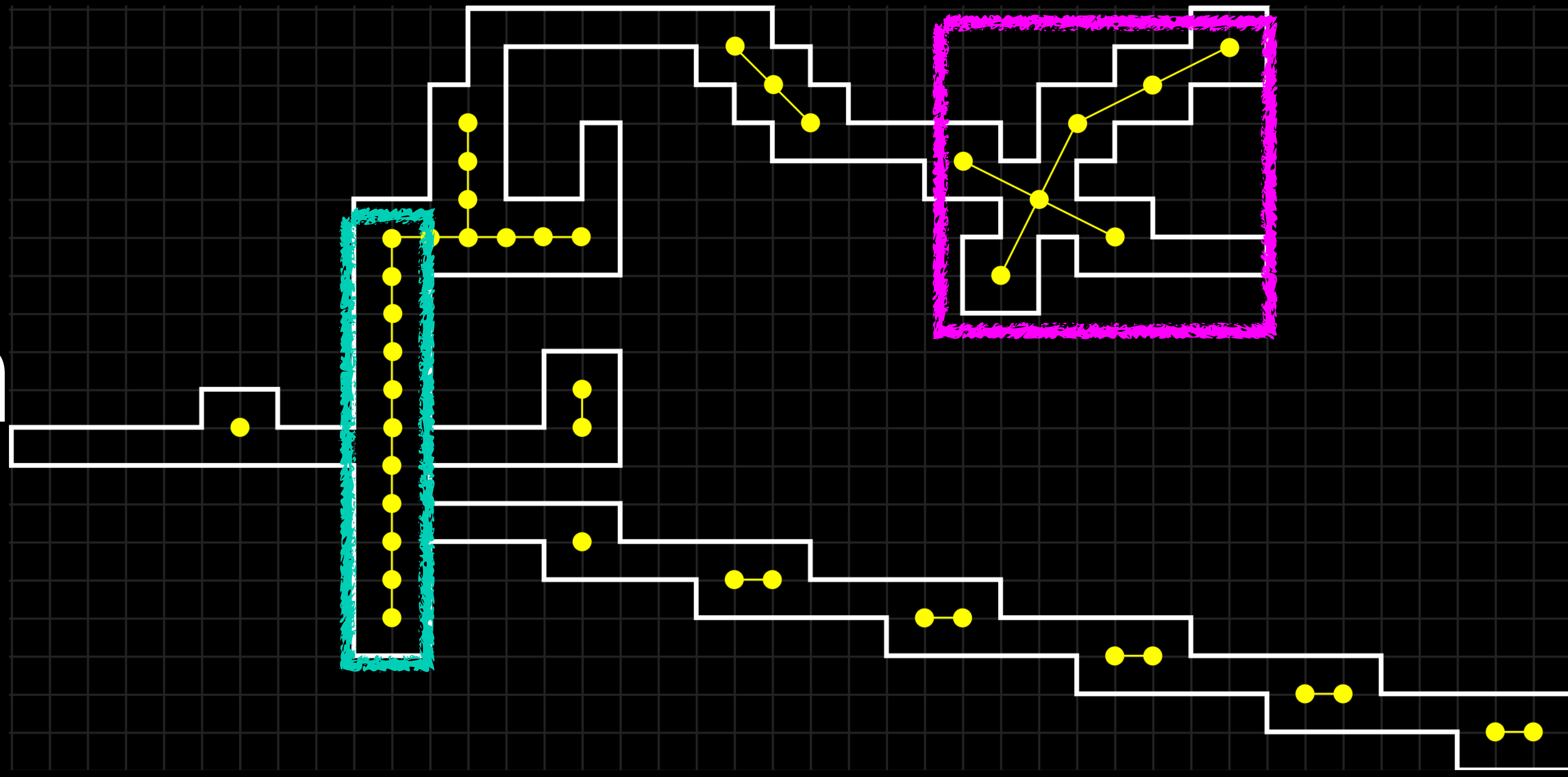


- P: 2-thin polyomino
- Each unit square/cell is incident to four vertices
- Such a vertex is *internal* if it does not lie on P's boundary
- P 2-thin ➜ a unit square can have at most 3 internal vertices
- I= set of internal vertices of the unit squares in P
- For any u,v∈I, we add edges {u,v} to E_I if:



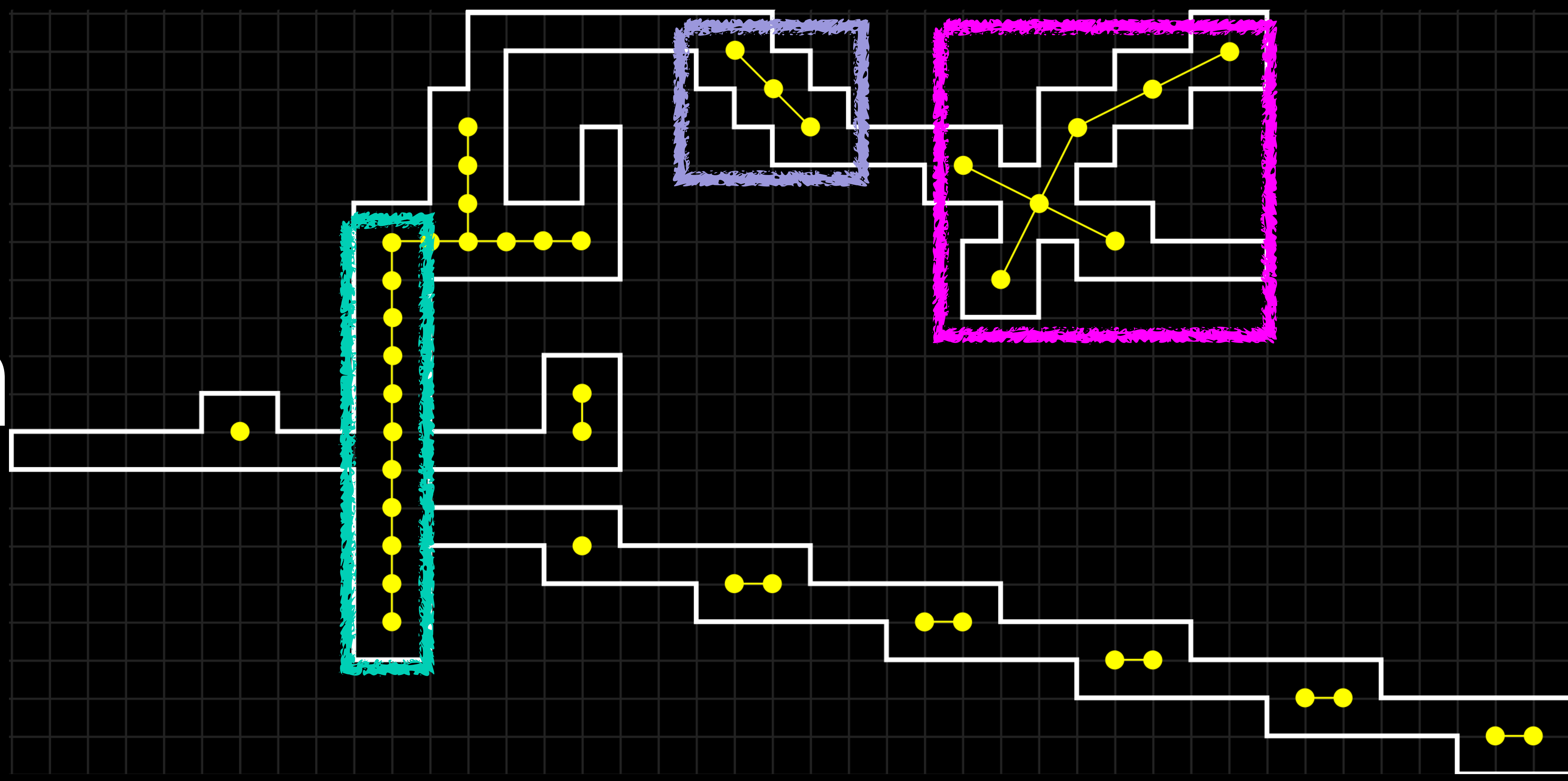
Constructing a Skeleton Graph

- P: 2-thin polyomino
- Each unit square/cell is incident to four vertices
- Such a vertex is *internal* if it does not lie on P's boundary
- P 2-thin ➜ a unit square can have at most 3 internal vertices
- I= set of internal vertices of the unit squares in P
- For any u,v∈I, we add edges {u,v} to E_I if:
  1. u and v belong to the same cell and ||u-v||=1



Constructing a Skeleton Graph

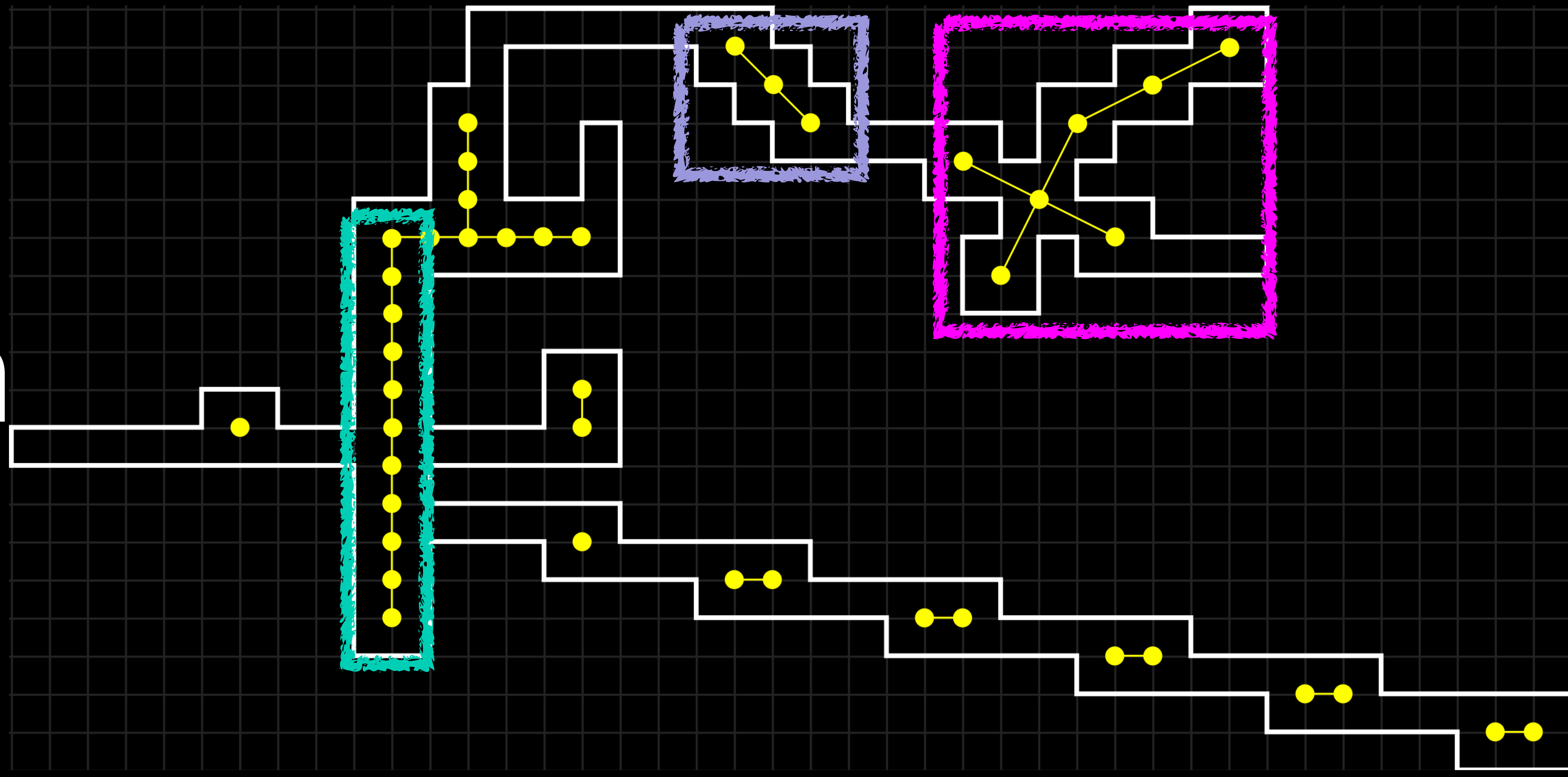
- P: 2-thin polyomino
- Each unit square/cell is incident to four vertices
- Such a vertex is *internal* if it does not lie on P's boundary
- P 2-thin ➜ a unit square can have at most 3 internal vertices
- I= set of internal vertices of the unit squares in P
- For any u,v∈I, we add edges {u,v} to E_I if:
  1. u and v belong to the same cell and ||u-v||=1
  2. u and v belong to two different cells that share an edge and both vertices of this edge are not internal



Constructing a Skeleton Graph

- P: 2-thin polyomino
- Each unit square/cell is incident to four vertices
- Such a vertex is *internal* if it does not lie on P's boundary
- P 2-thin ➜ a unit square can have at most 3 internal vertices
- I= set of internal vertices of the unit squares in P
- For any u,v∈I, we add edges {u,v} to E_I if:

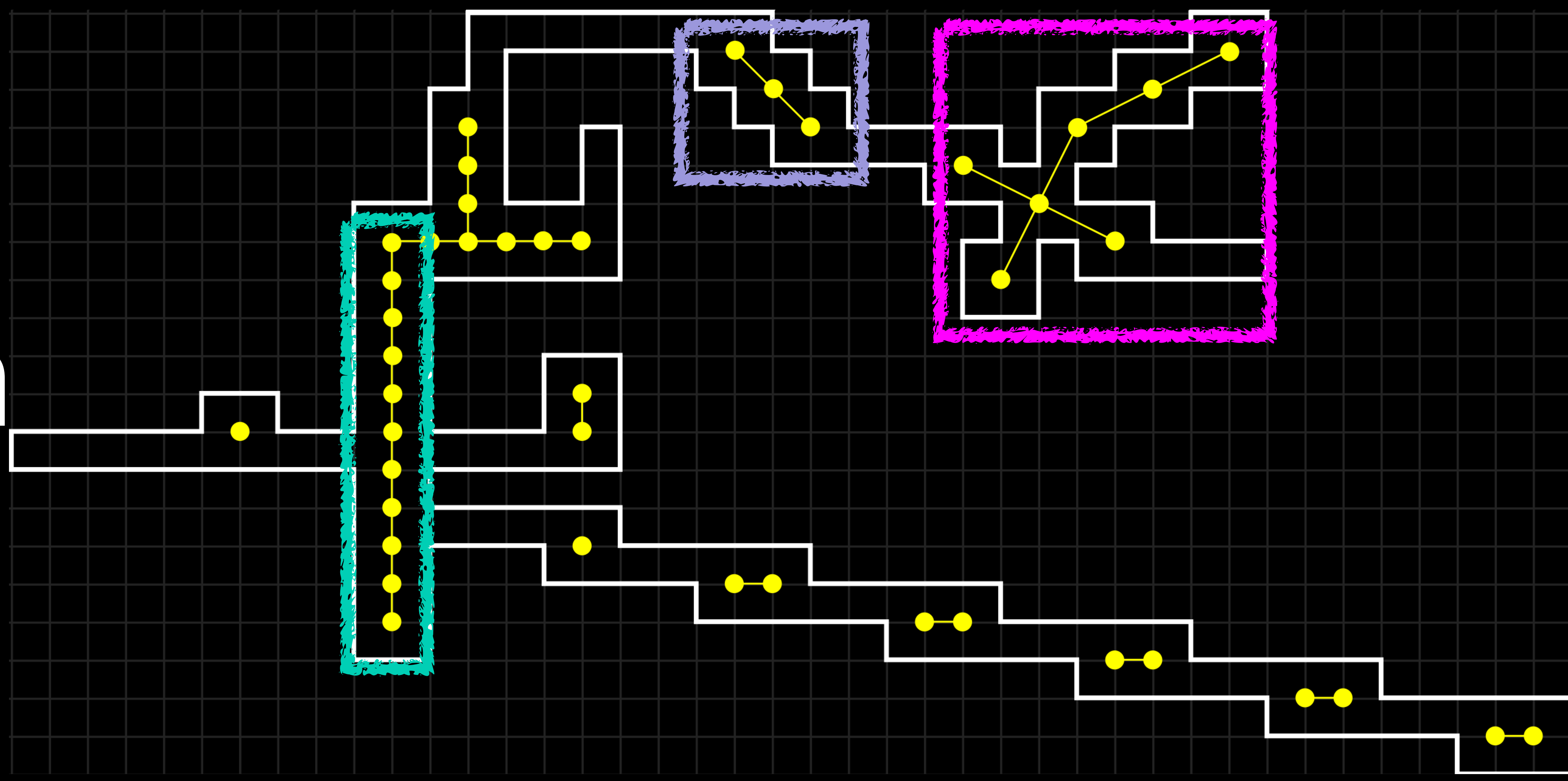
  1. u and v belong to the same cell and ||u-v||=1
  2. u and v belong to two different cells that share an edge and both vertices of this edge are not internal
  3. u and v belong to the same cell s and both other vertices belonging to s are not internal



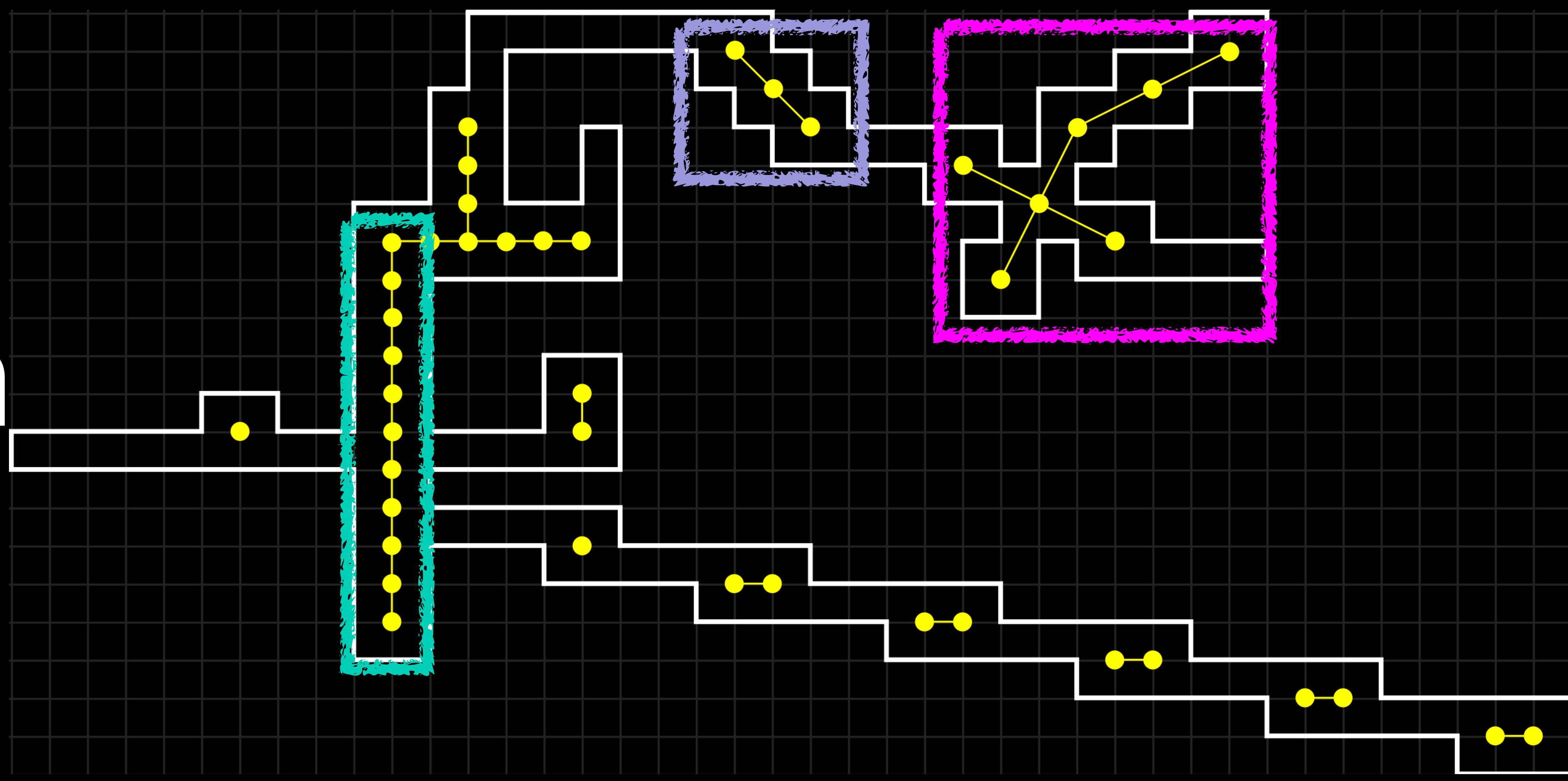
Constructing a Skeleton Graph

- P: 2-thin polyomino
- Each unit square/cell is incident to four vertices
- Such a vertex is *internal* if it does not lie on P's boundary
- P 2-thin ➜ a unit square can have at most 3 internal vertices
- I= set of internal vertices of the unit squares in P
- For any u,v∈I, we add edges {u,v} to E_I if:
  1. u and v belong to the same cell and ||u-v||=1
  2. u and v belong to two different cells that share an edge and both vertices of this edge are not internal
  3. u and v belong to the same cell s and both other vertices belonging to s are not internal
- P 2-thin ➜ with edges from 1. we cannot create a four-cycle

Constructing a Skeleton Graph



- P: 2-thin polyomino
- Each unit square/cell is incident to four vertices
- Such a vertex is *internal* if it does not lie on P’s boundary
- P 2-thin ➜ a unit square can have at most 3 internal vertices
- I= set of internal vertices of the unit squares in P
- For any u,v∈I, we add edges {u,v} to E_I if:
  1. u and v belong to the same cell and ||u-v||=1
  2. u and v belong to two different cells that share an edge and both vertices of this edge are not internal
  3. u and v belong to the same cell s and both other vertices belonging to s are not internal
- P 2-thin ➜ with edges from 1. we cannot create a four-cycle
- Cycle with edges from 2. or 3. ➜ P would have a hole

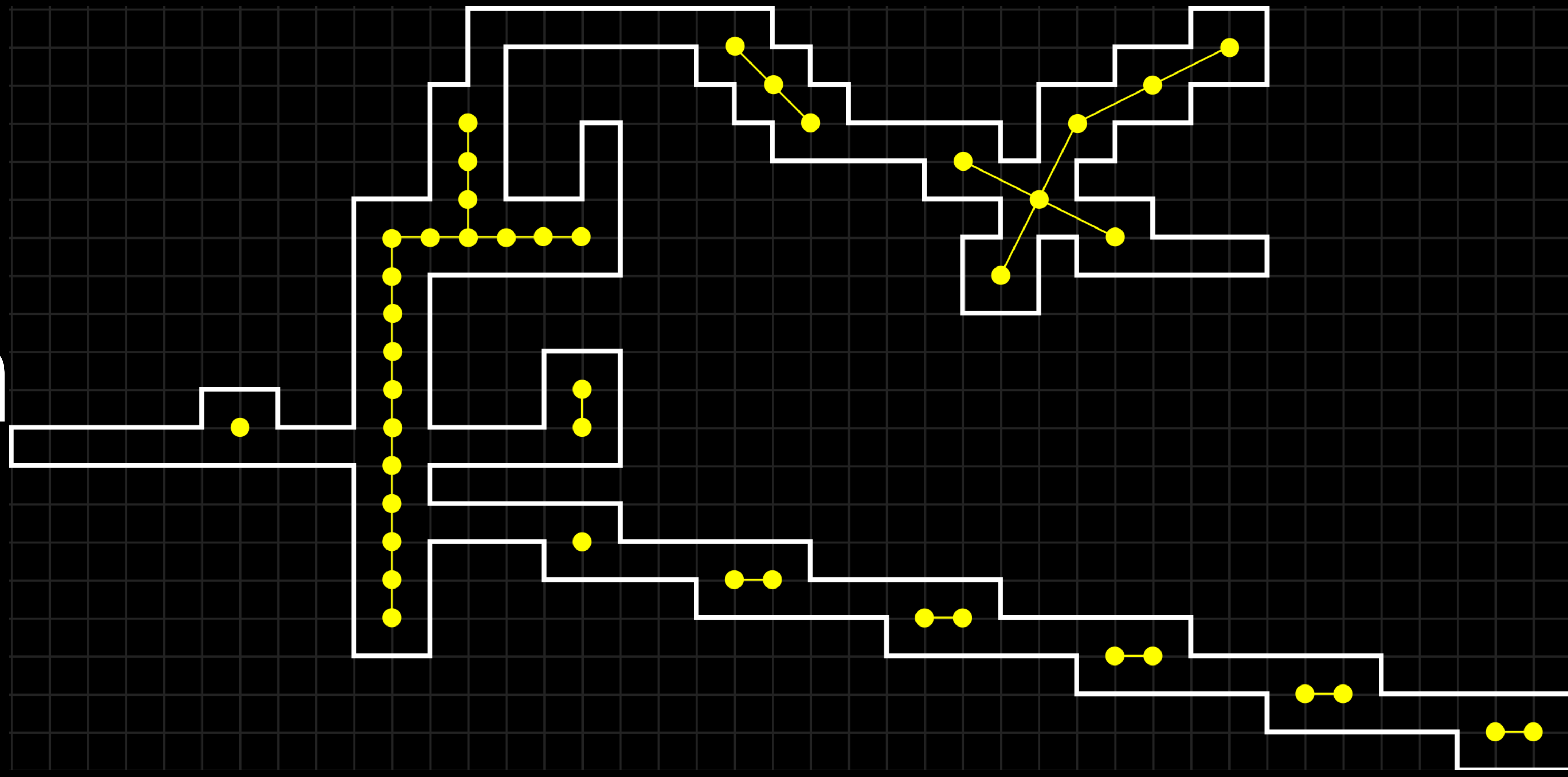


Constructing a Skeleton Graph

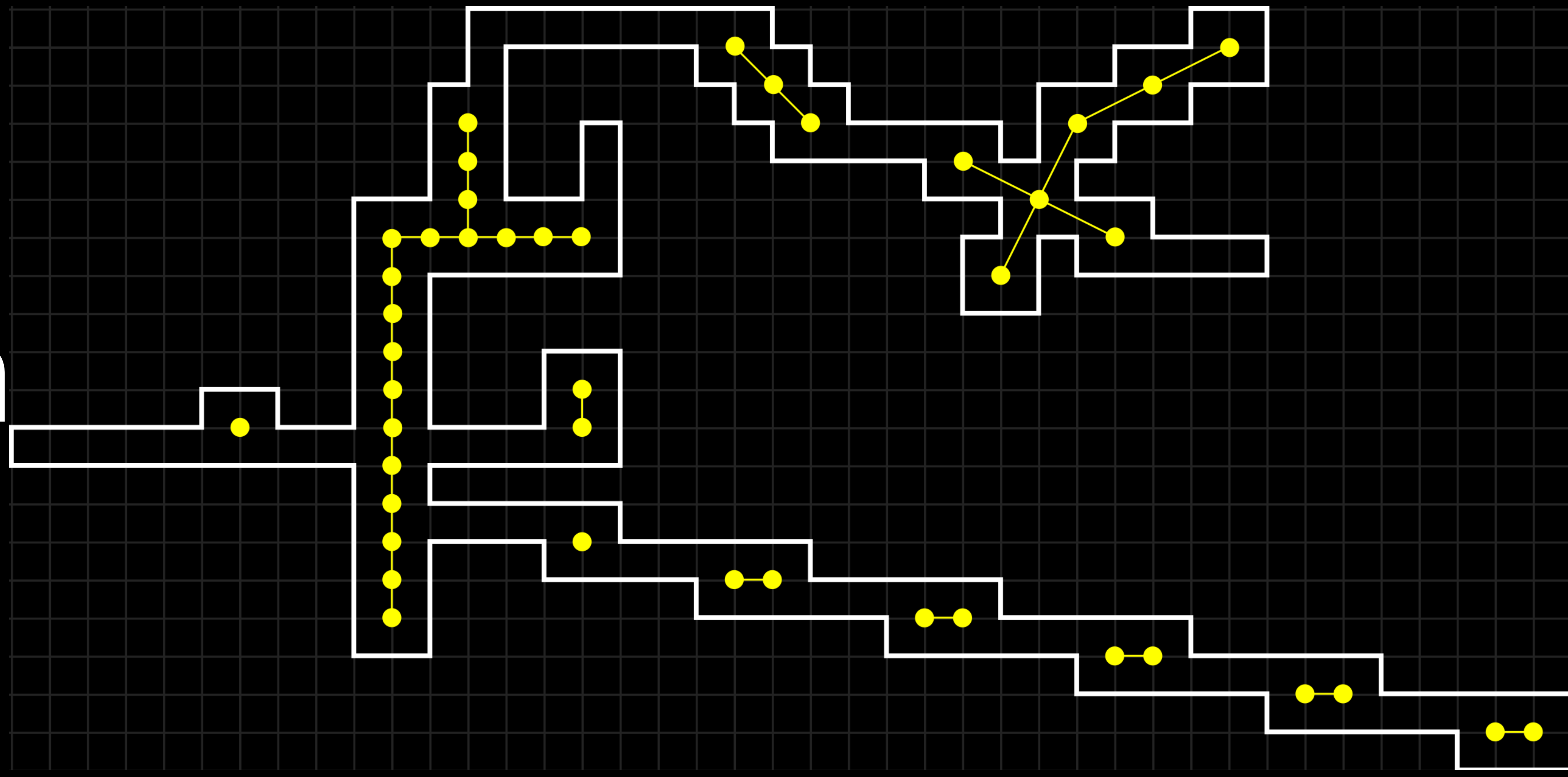
- P: 2-thin polyomino
- Each unit square/cell is incident to four vertices
- Such a vertex is *internal* if it does not lie on P's boundary
- P 2-thin ➜ a unit square can have at most 3 internal vertices
- I= set of internal vertices of the unit squares in P
- For any u,v∈I, we add edges {u,v} to E_I if:
  1. u and v belong to the same cell and ||u-v||=1
  2. u and v belong to two different cells that share an edge and both vertices of this edge are not internal
  3. u and v belong to the same cell s and both other vertices belonging to s are not internal
- P 2-thin ➜ with edges from 1. we cannot create a four-cycle
- Cycle with edges from 2. or 3. ➜ P would have a hole

➡ Edges from E_I form a forest T_I on I

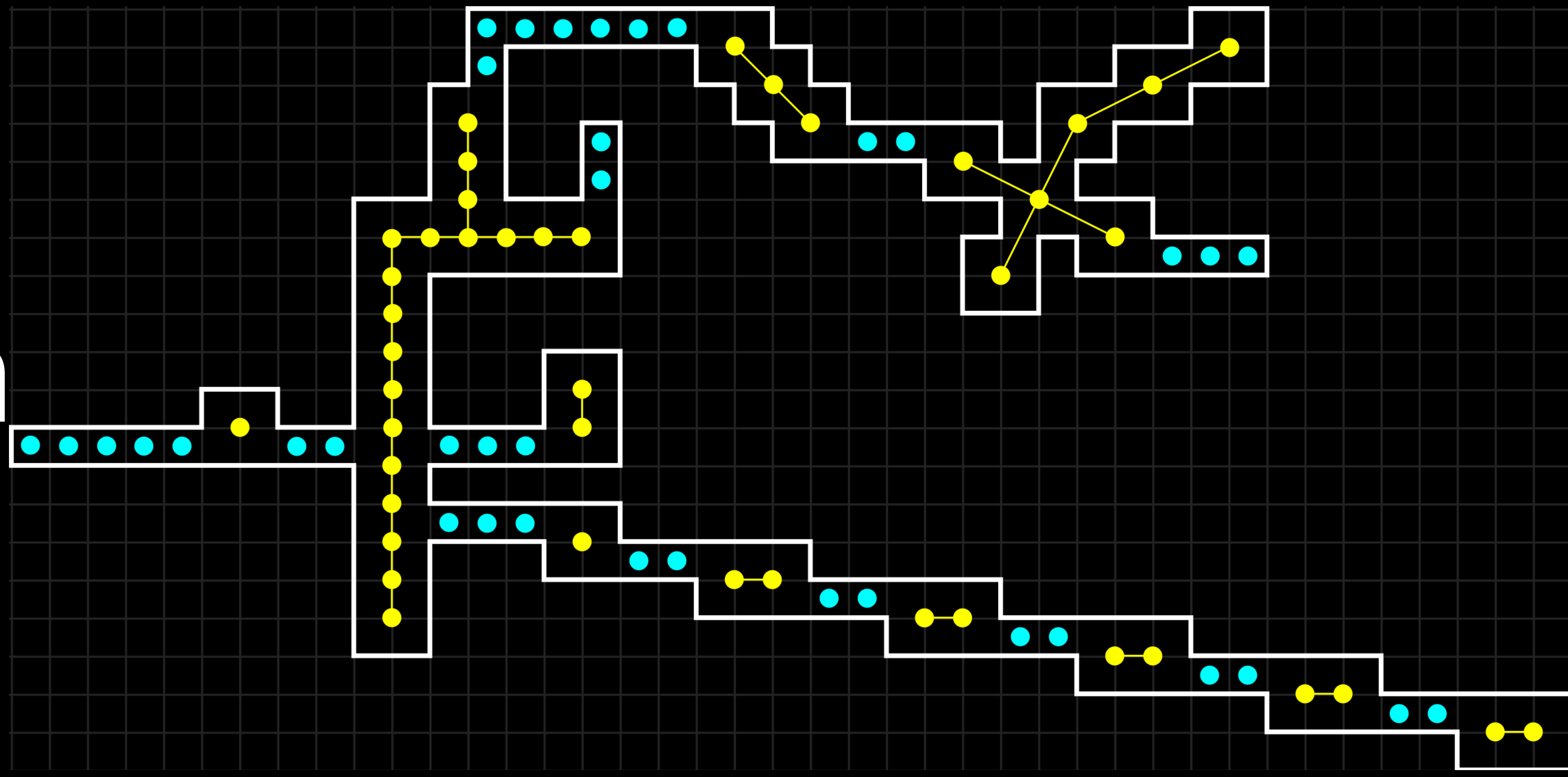
Constructing a Skeleton Graph



Constructing a Skeleton Graph



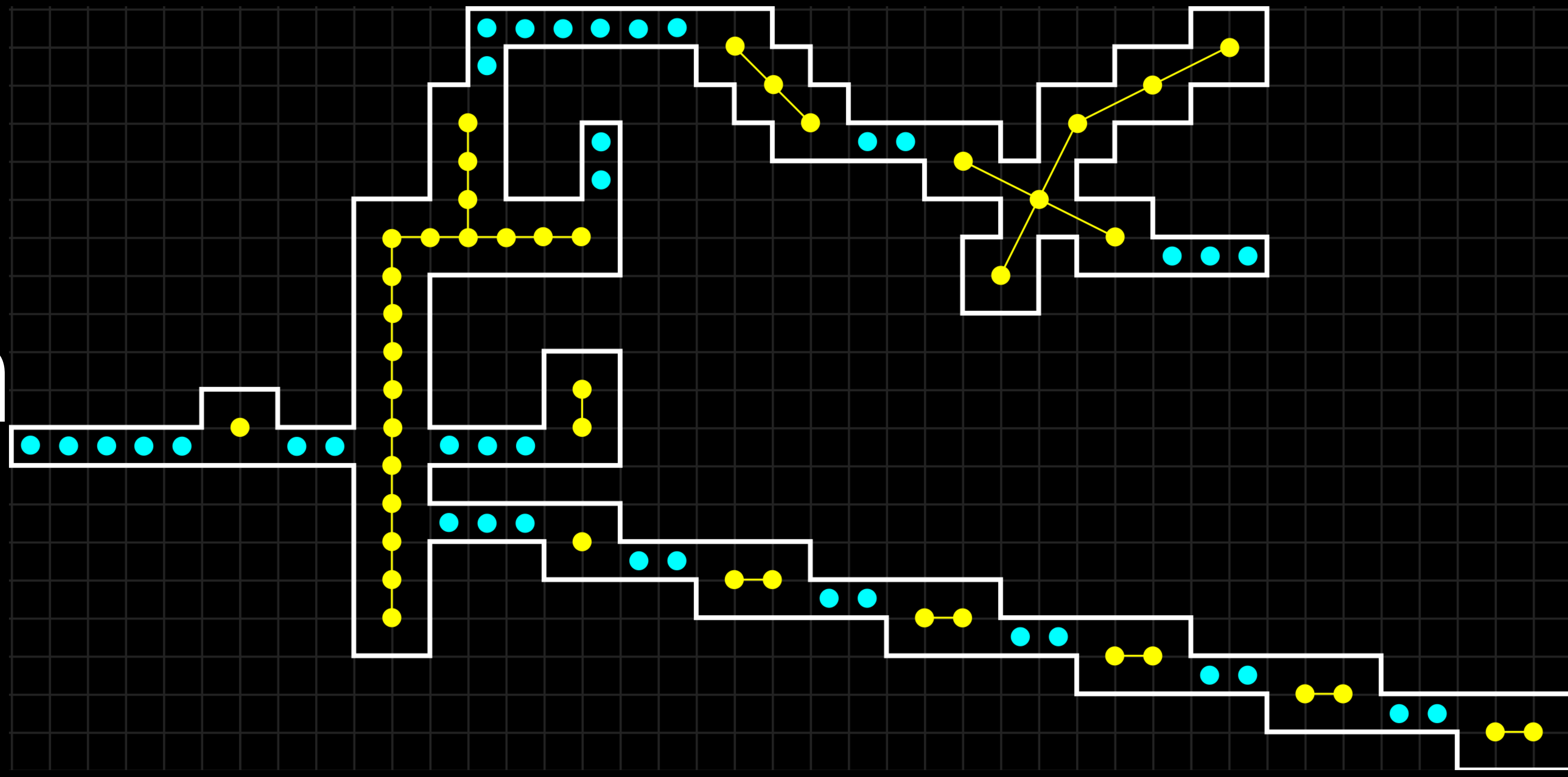
- Unit square s without internal vertex:



Constructing a Skeleton Graph

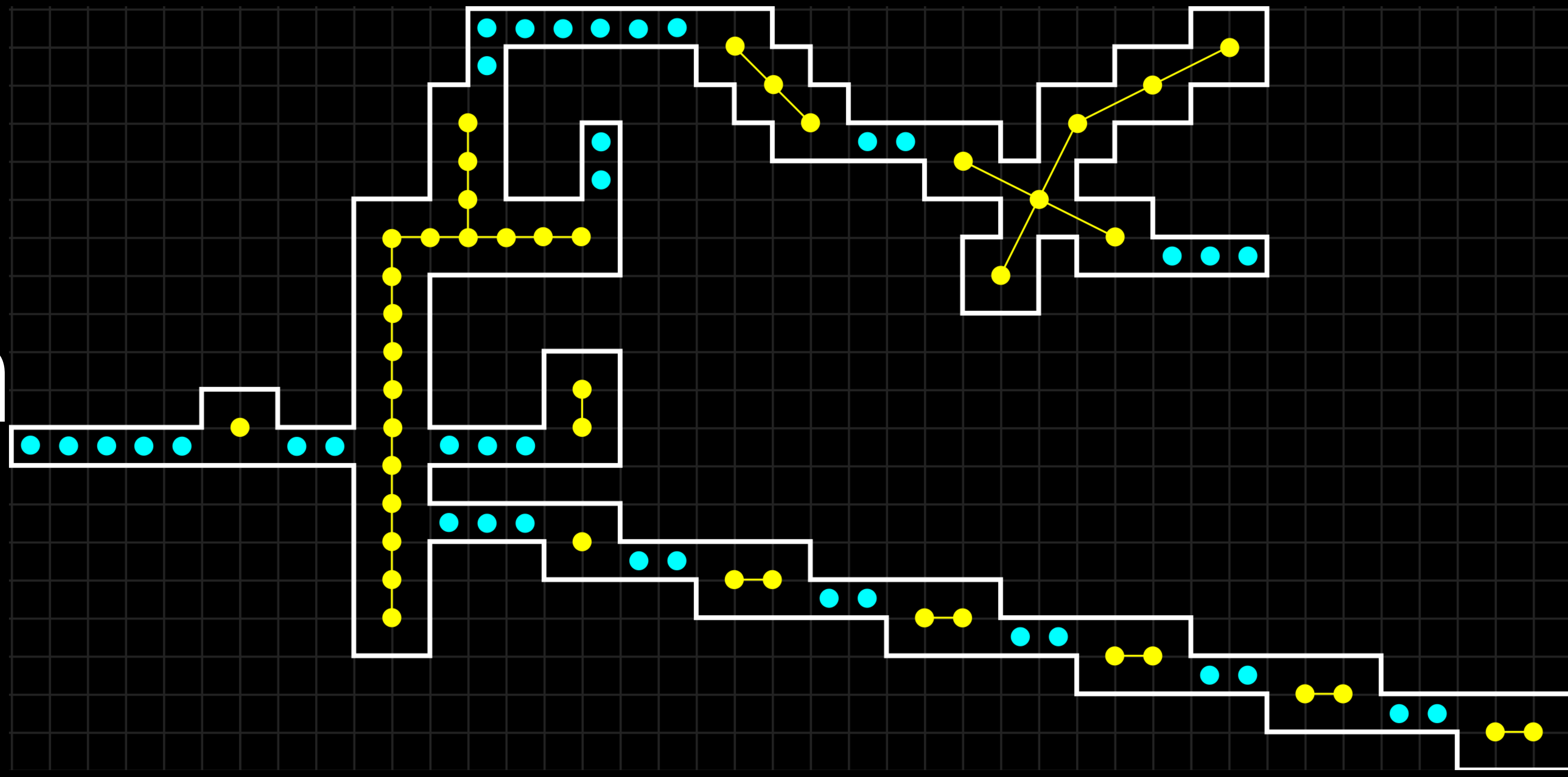
- Unit square s without internal vertex:
  - Vertex b_s in center of s

Constructing a Skeleton Graph

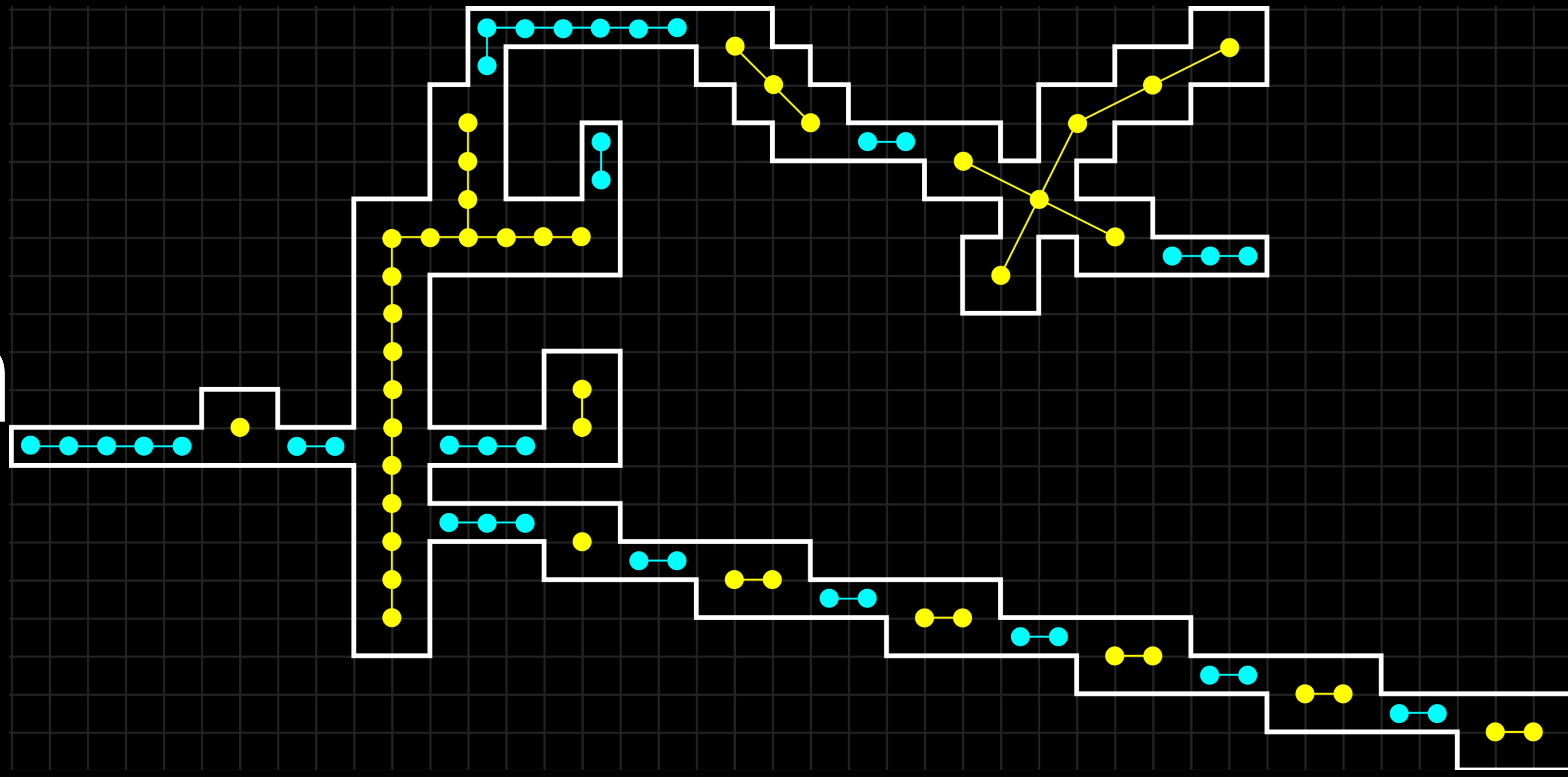


- Unit square s without internal vertex:
  - Vertex b_s in center of s
  - s: boundary square, b_s: boundary node

Constructing a Skeleton Graph

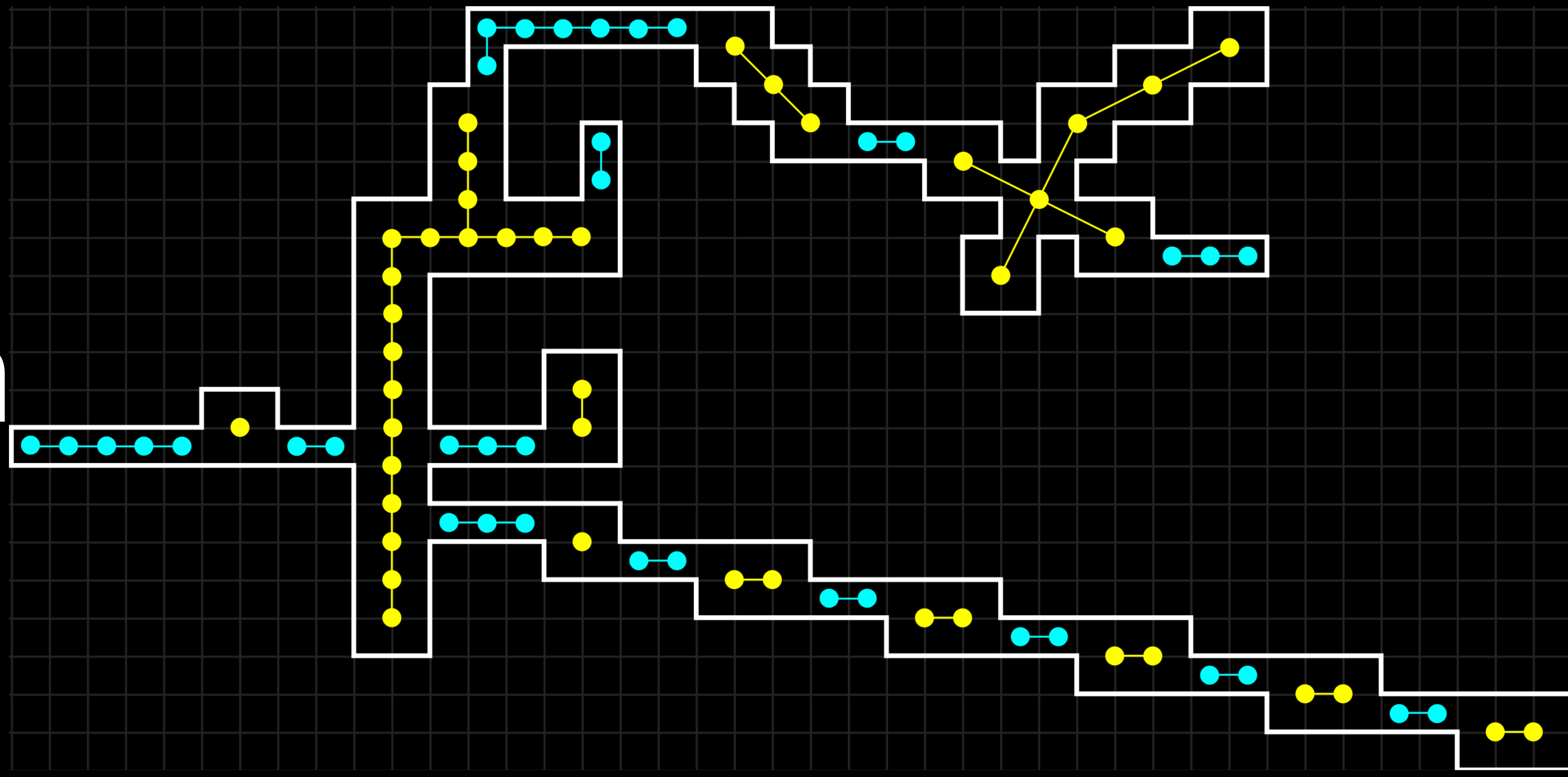


- Unit square s without internal vertex:
  - Vertex b_s in center of s
  - s: boundary square, b_s: boundary node
  - B= set of all boundary nodes



Constructing a Skeleton Graph

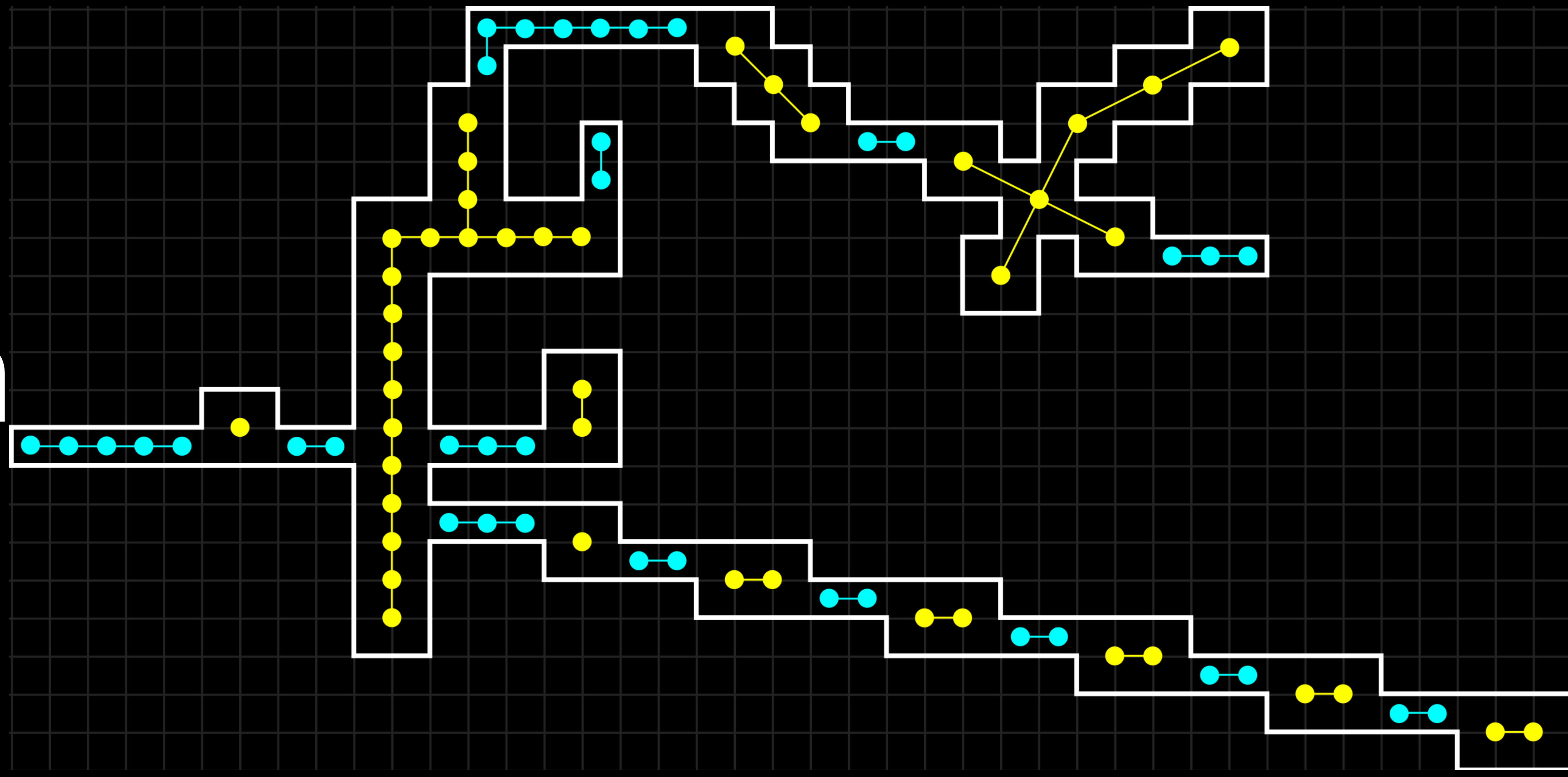
- Unit square s without internal vertex:
  - Vertex b_s in center of s
  - s: boundary square, b_s: boundary node
  - B= set of all boundary nodes
- Any b_s, $b_{s'}$ in B with s and s' sharing an edge: add $\{b_s, b_{s'}\}$ to E_B



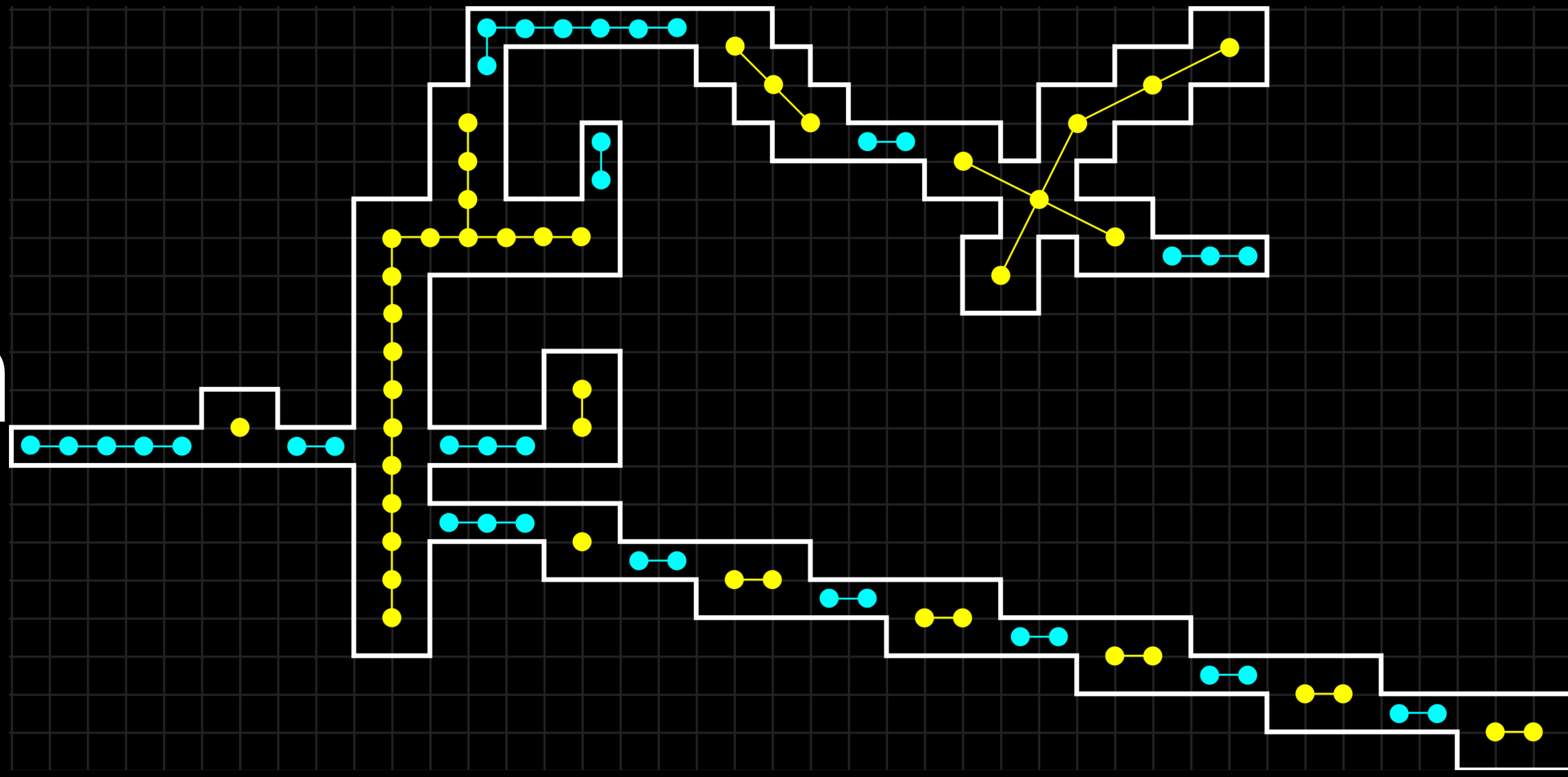
Constructing a Skeleton Graph

- Unit square s without internal vertex:
  - Vertex b_s in center of s
  - s: boundary square, b_s: boundary node
  - B= set of all boundary nodes
- Any b_s, $b_{s'}$ in B with s and s' sharing an edge: add $\{b_s, b_{s'}\}$ to E_B
- Edges from E_B form a forest T_B on B

Constructing a Skeleton Graph

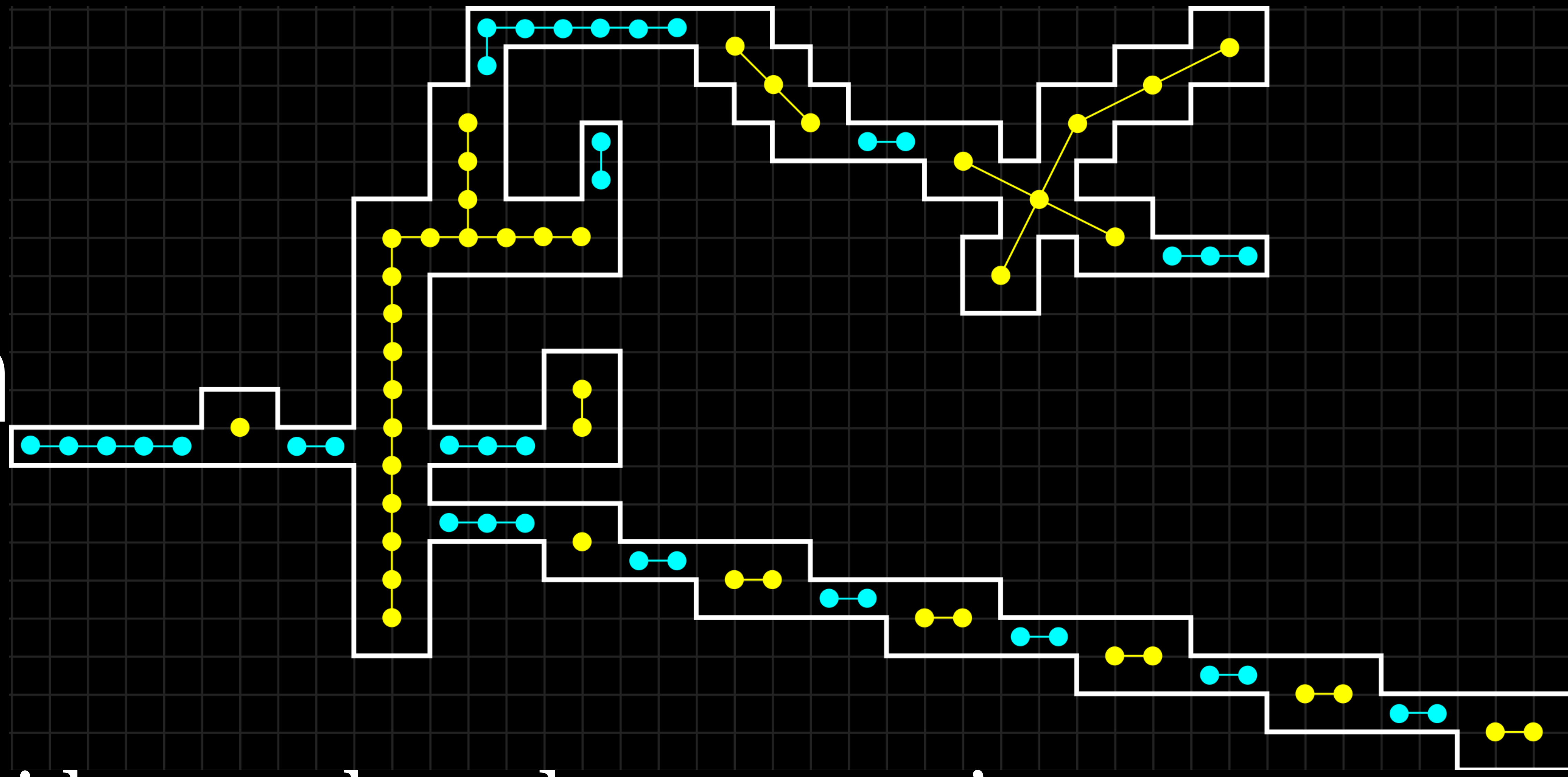


Constructing a Skeleton Graph

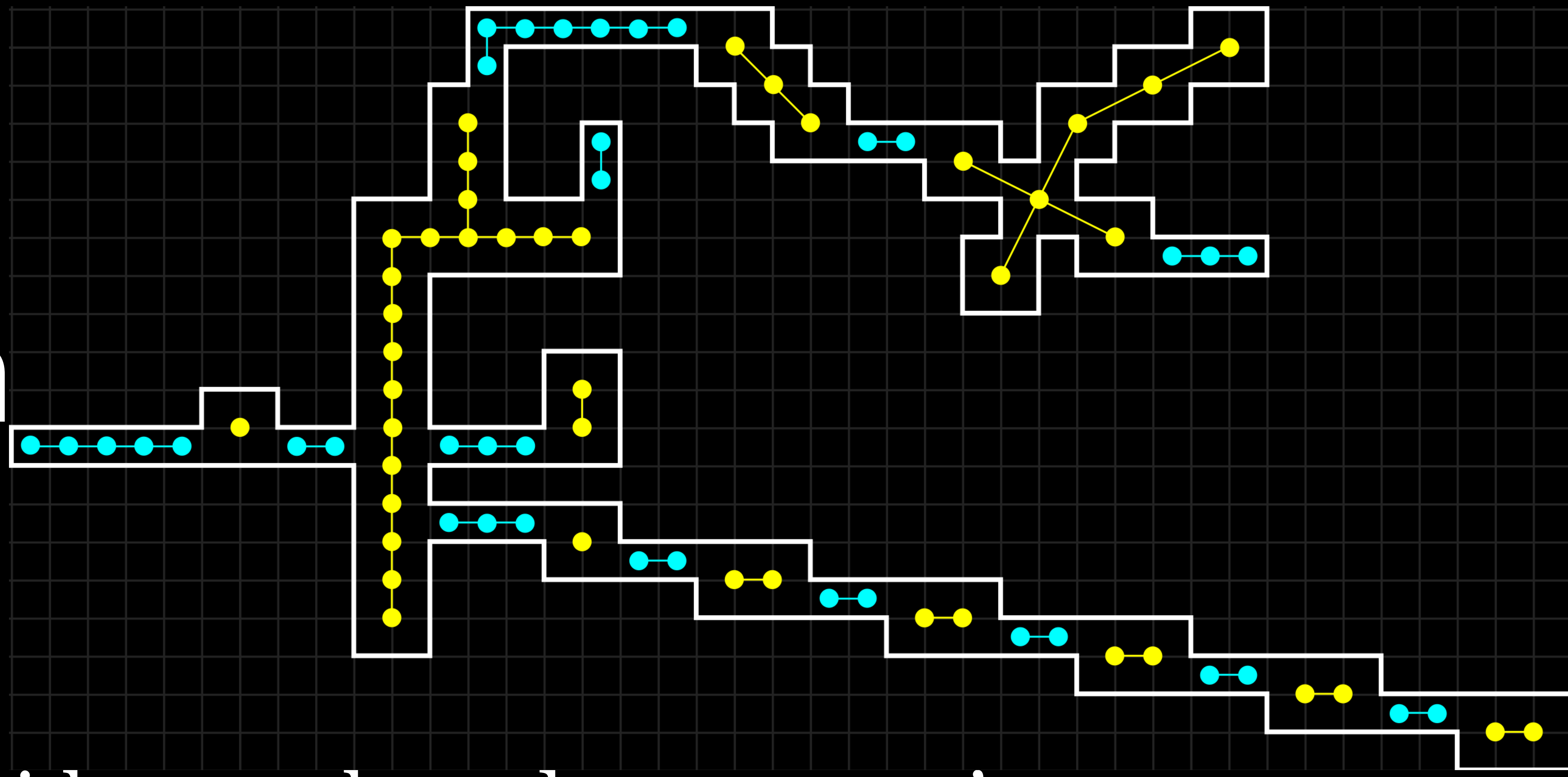


- Connect T_I and T_B

Constructing a Skeleton Graph



- Connect T_I and T_B
- Let s be a boundary square that shares an edge with a non-boundary square s'

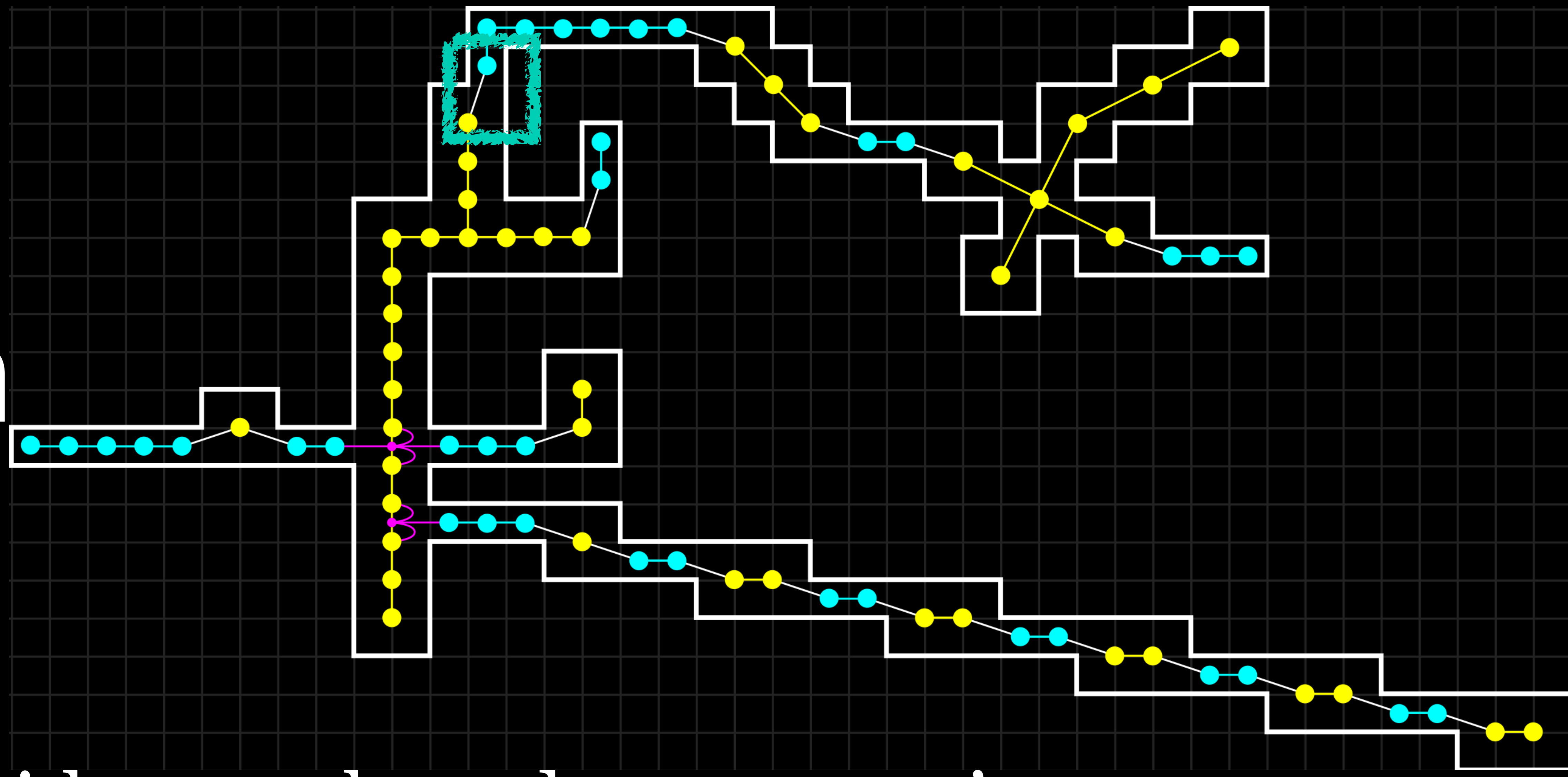


Constructing a Skeleton Graph

- Connect T_I and T_B
- Let s be a boundary square that shares an edge with a non-boundary square s’

➡ s’ has at most two internal vertices

Constructing a Skeleton Graph

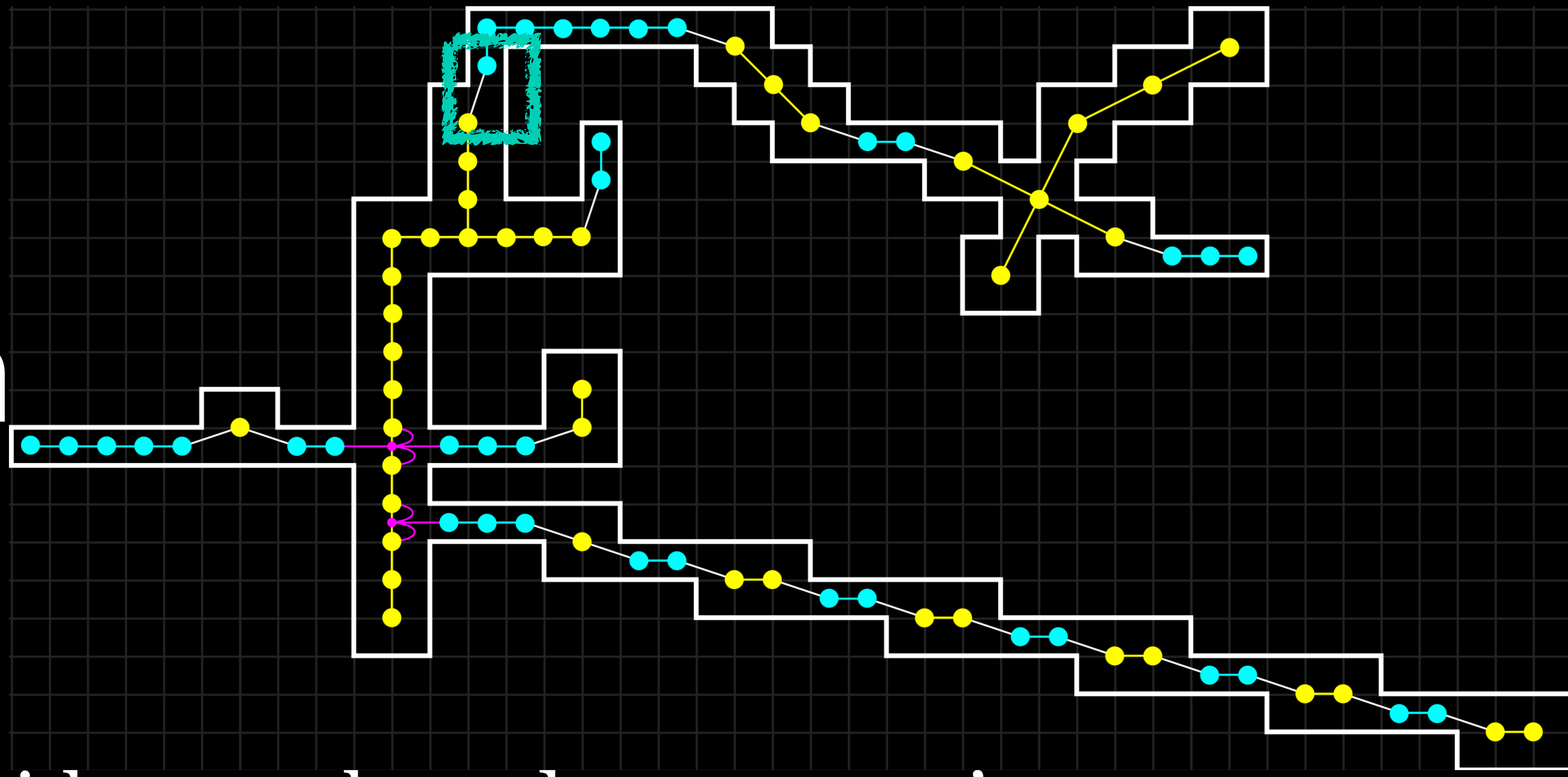


- Connect T_I and T_B
- Let s be a boundary square that shares an edge with a non-boundary square s'

➡ s' has at most two internal vertices

  - If s' has a single internal vertex v: add $\{b_s, v\}$ to E_{con}

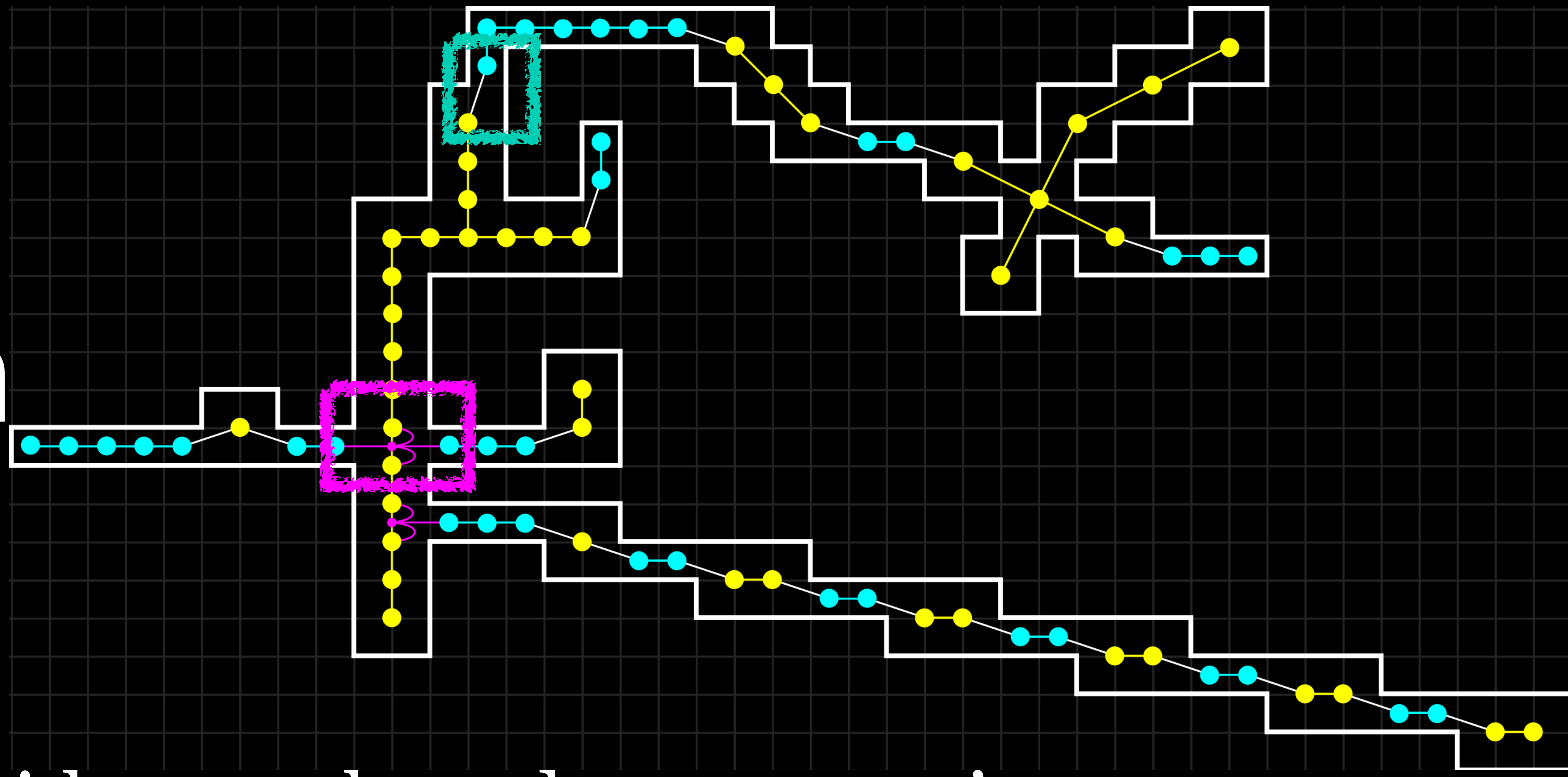
Constructing a Skeleton Graph



- Connect T_I and T_B
- Let s be a boundary square that shares an edge with a non-boundary square s'

➡ s' has at most two internal vertices

  - If s' has a single internal vertex v: add $\{b_s, v\}$ to E_{con}
  - If s' has two internal vertices v and u:



Constructing a Skeleton Graph

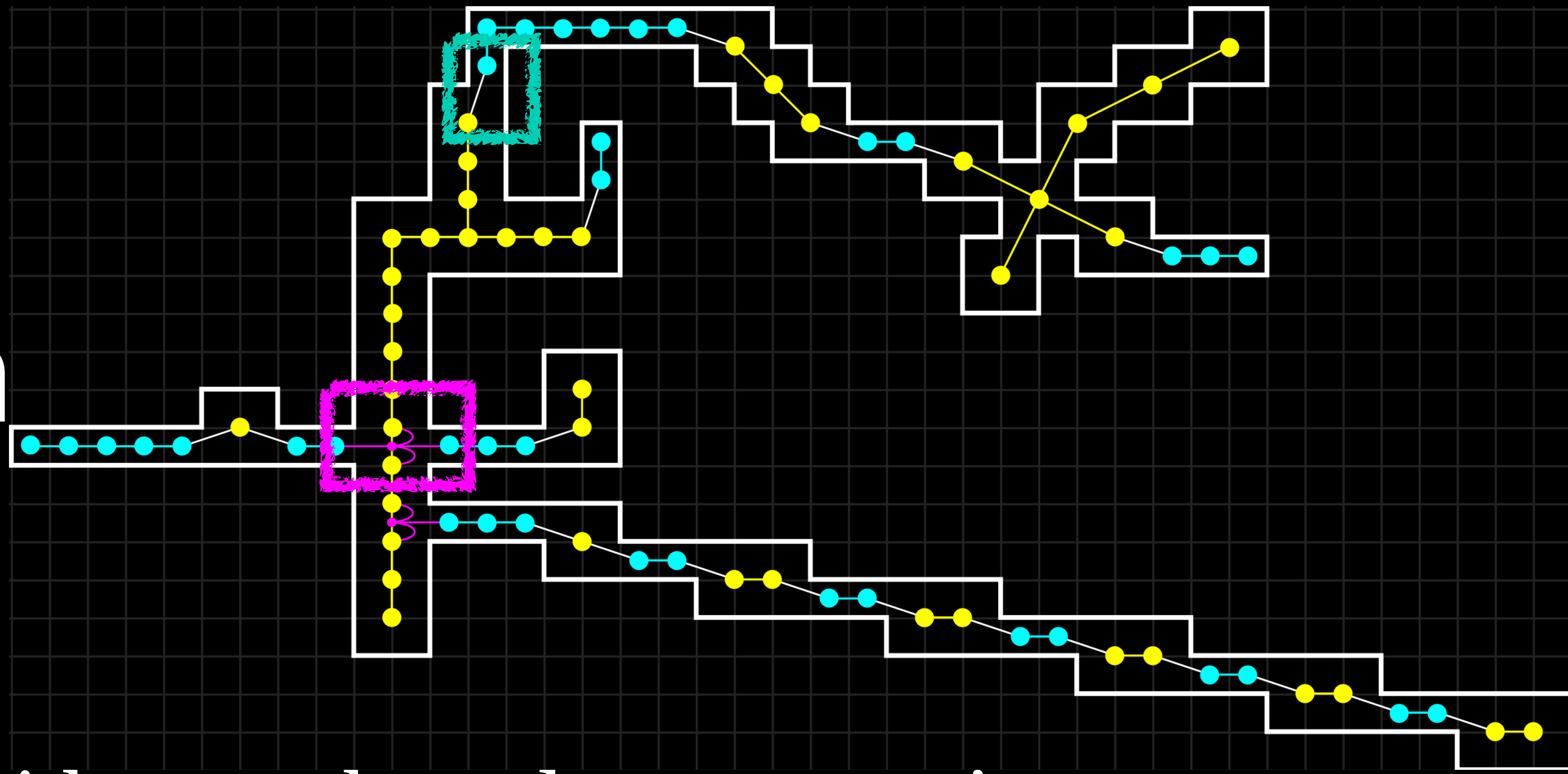
- Connect T_I and T_B
- Let s be a boundary square that shares an edge with a non-boundary square s'

➡ s' has at most two internal vertices

  - If s' has a single internal vertex v: add $\{b_s, v\}$ to E_{con}
  - If s' has two internal vertices v and u:
    - ✳ Add artificial node $x_{v,u}$ to set X

LINKÖPING UNIVERSITY

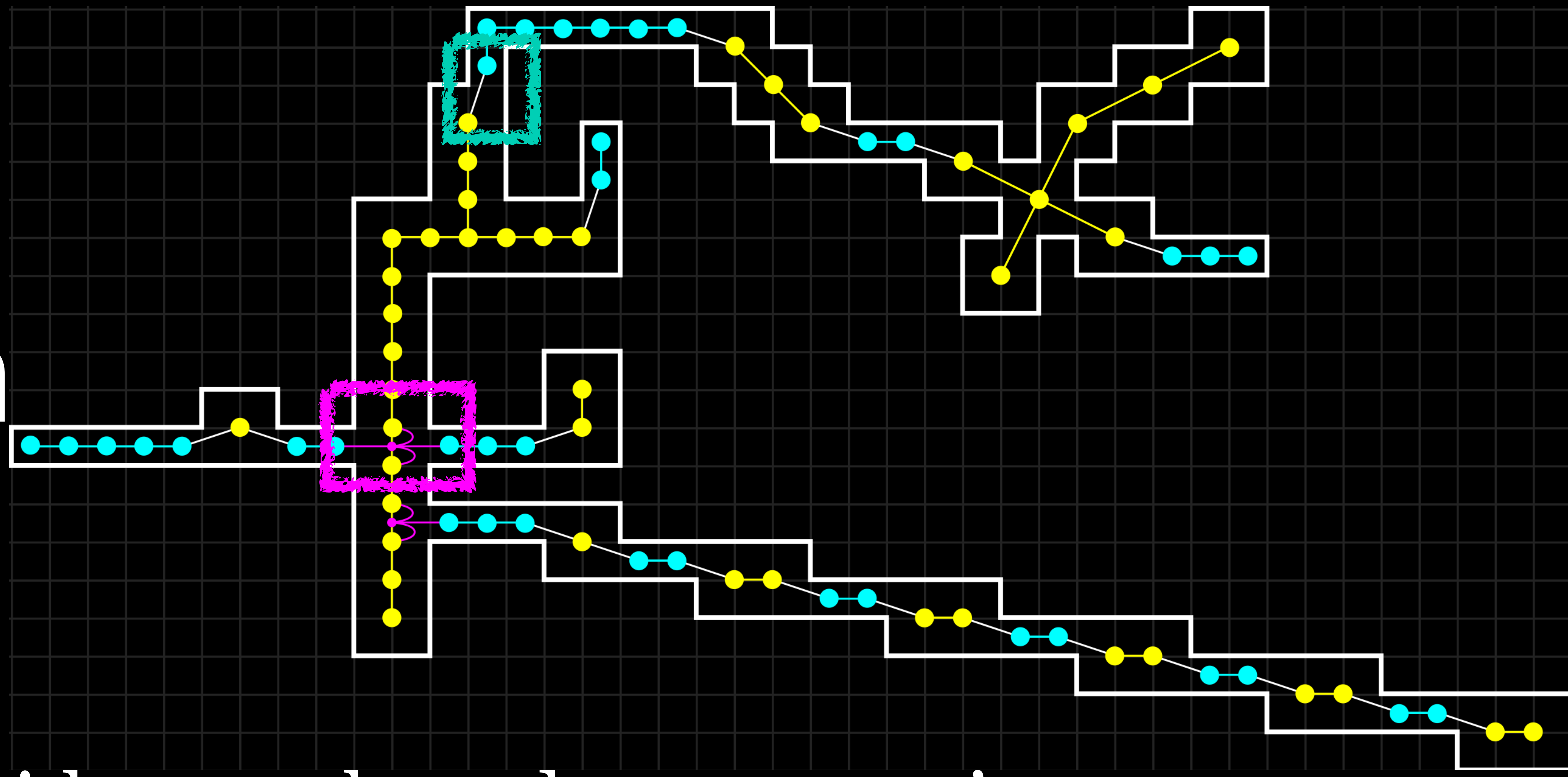
Constructing a Skeleton Graph



- Connect T_I and T_B
- Let s be a boundary square that shares an edge with a non-boundary square s'

➡ s' has at most two internal vertices

  - If s' has a single internal vertex v: add $\{b_s, v\}$ to E_{con}
  - If s' has two internal vertices v and u:
    - ✳ Add artificial node $x_{v,u}$ to set X
    - ✳ Add edges $\{b_s, x_{v,u}\}$, $\{u, x_{v,u}\}$, and $\{v, x_{v,u}\}$ to E_{con}



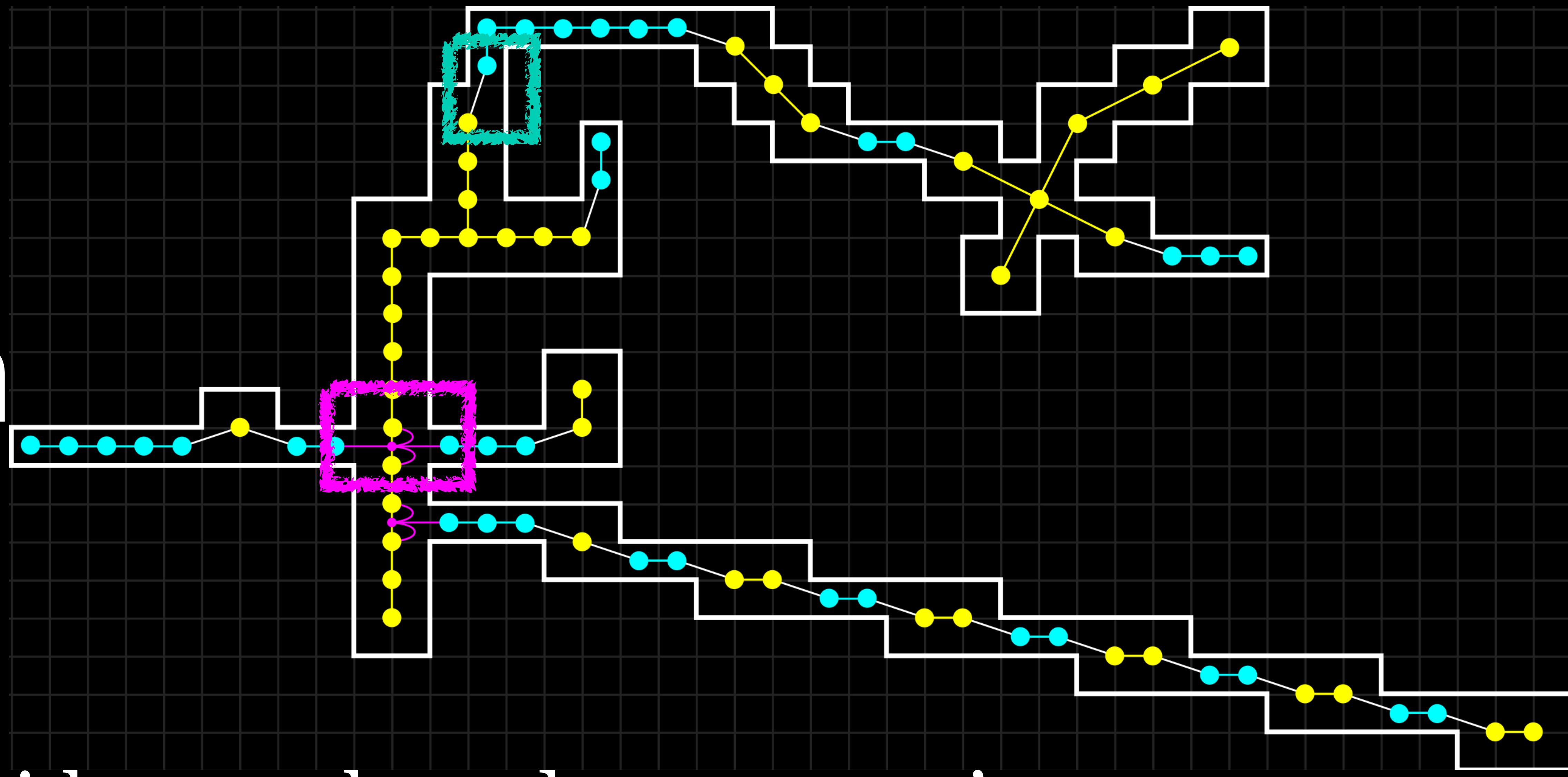
Constructing a Skeleton Graph

- Connect T_I and T_B
- Let s be a boundary square that shares an edge with a non-boundary square s'

➡ s' has at most two internal vertices

- If s' has a single internal vertex v: add $\{b_s, v\}$ to E_{con}
- If s' has two internal vertices v and u:
  - ✳ Add artificial node $x_{v,u}$ to set X
  - ✳ Add edges $\{b_s, x_{v,u}\}$, $\{u, x_{v,u}\}$, and $\{v, x_{v,u}\}$ to E_{con}
  - ✳ Remove edge $\{u,v\}$ from E_I

Constructing a Skeleton Graph

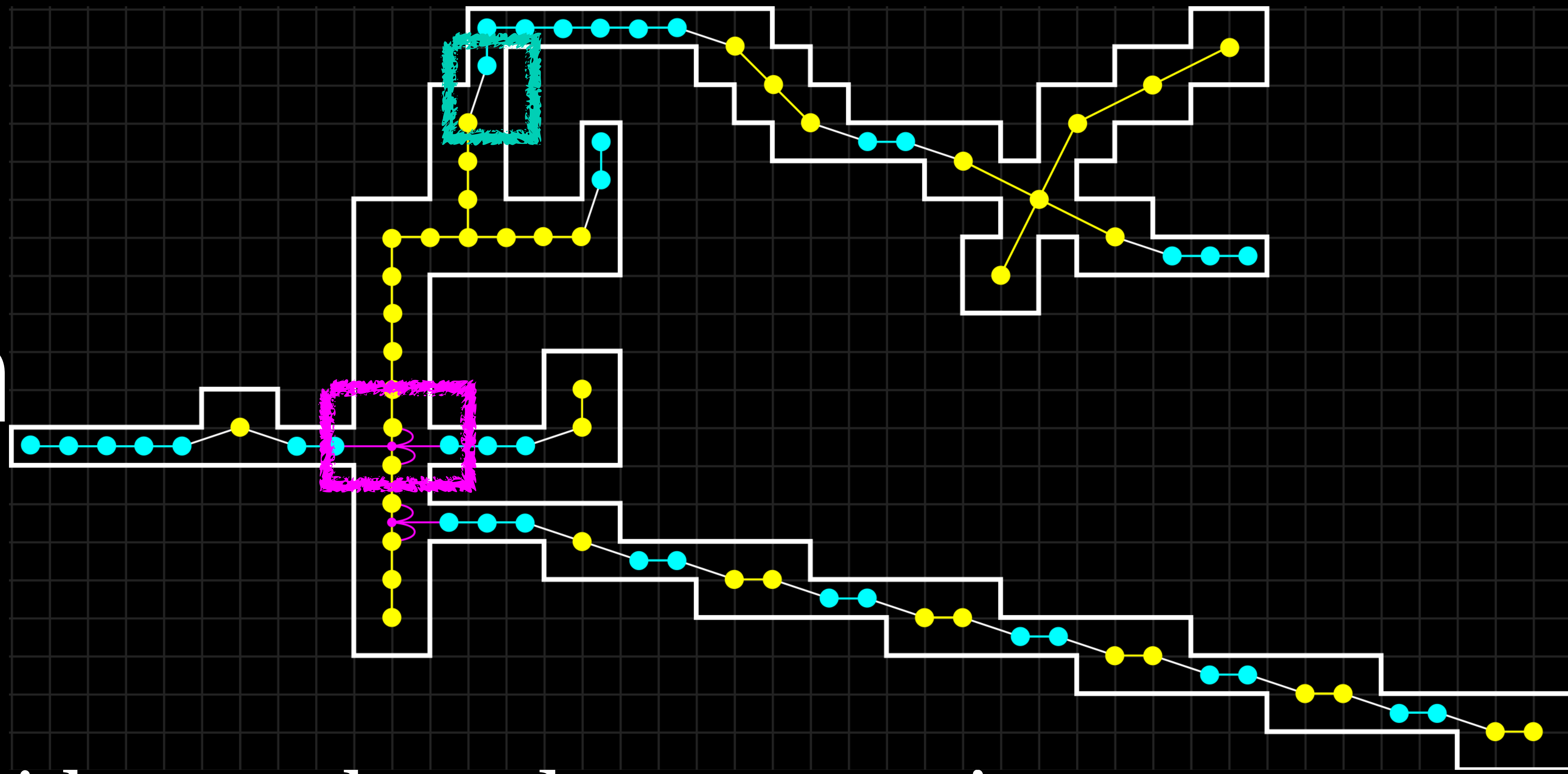


- Connect T_I and T_B
- Let s be a boundary square that shares an edge with a non-boundary square s'

➡ s' has at most two internal vertices

  - If s' has a single internal vertex v: add $\{b_s, v\}$ to E_{con}
  - If s' has two internal vertices v and u:
    - ✳ Add artificial node $x_{v,u}$ to set X
    - ✳ Add edges $\{b_s, x_{v,u}\}$, $\{u, x_{v,u}\}$, and $\{v, x_{v,u}\}$ to E_{con}
    - ✳ Remove edge $\{u,v\}$ from E_I

- T graph on vertex set $V=I \cup B \cup X$ and edge set $E=E_I \cup E_B \cup E_{con}$



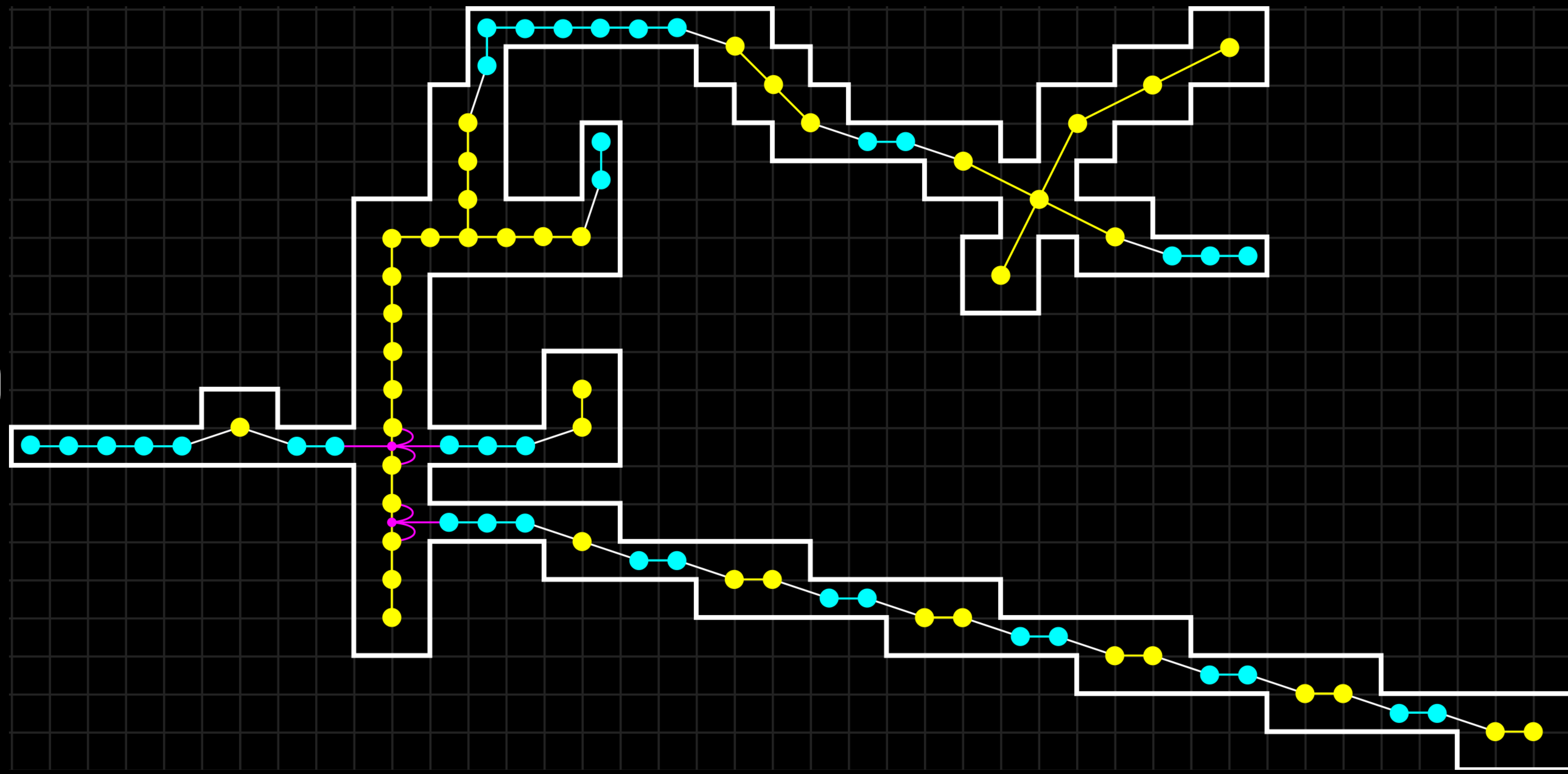
Constructing a Skeleton Graph

- Connect T_I and T_B
- Let s be a boundary square that shares an edge with a non-boundary square s'

➡ s' has at most two internal vertices

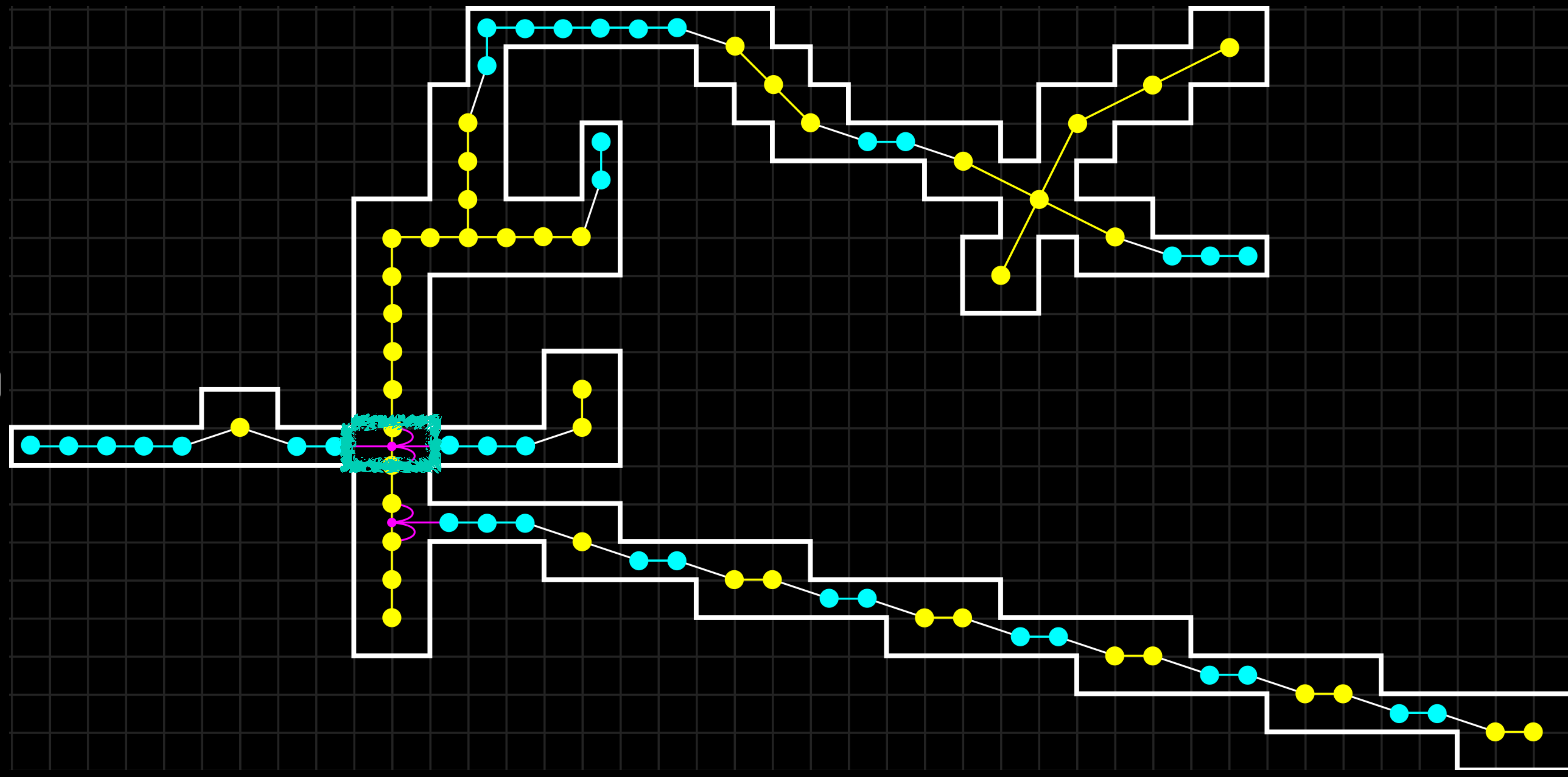
  - If s' has a single internal vertex v: add $\{b_s, v\}$ to E_{con}
  - If s' has two internal vertices v and u:
    - ✳ Add artificial node $x_{v,u}$ to set X
    - ✳ Add edges $\{b_s, x_{v,u}\}$, $\{u, x_{v,u}\}$, and $\{v, x_{v,u}\}$ to E_{con}
    - ✳ Remove edge $\{u,v\}$ from E_I
- T graph on vertex set $V=I \cup B \cup X$ and edge set $E=E_I \cup E_B \cup E_{con}$
- When we connect T_I and T_B no cyles are created ➜ T is a tree and max degree is 4

Associated Squares for the No



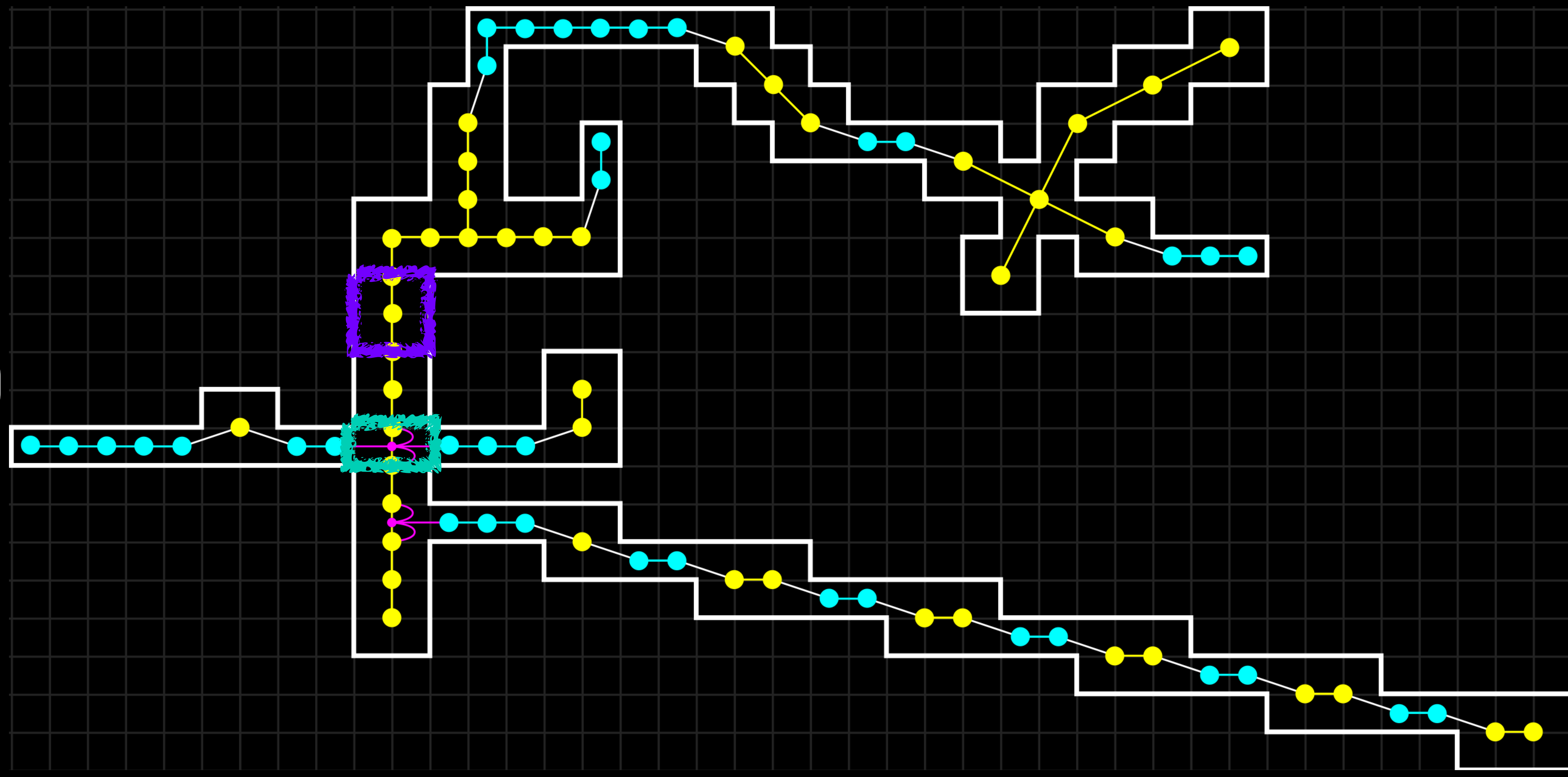
- With each node v∈T, we associate a block S(v) of unit squares:

Associated Squares for the No



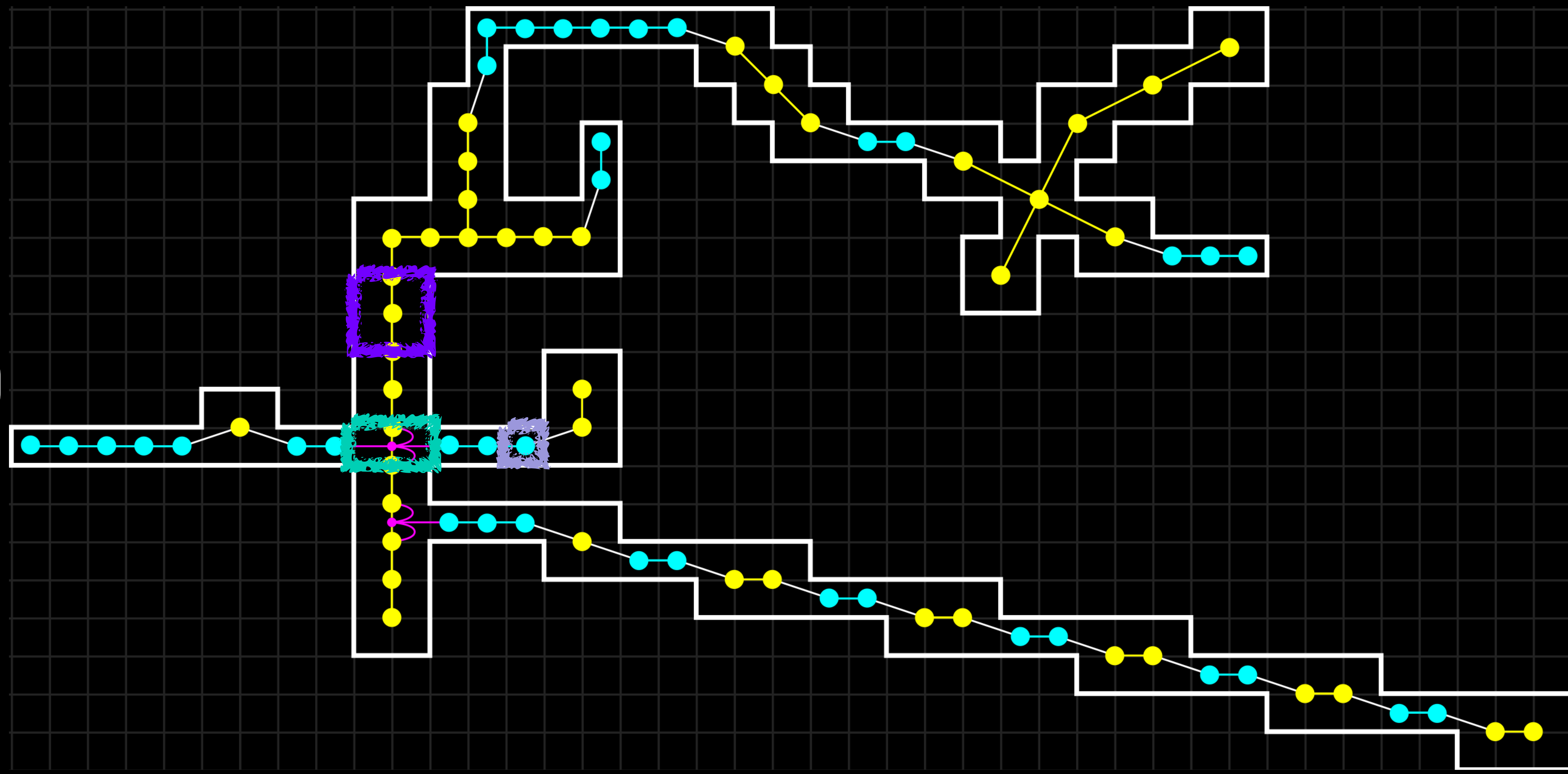
- With each node $v \in T$, we associate a block S(v) of unit squares:
  - $v = x_{u,v} \in X$: S(v) are the two unit squares with the edge {u,v}

Associated Squares for the No



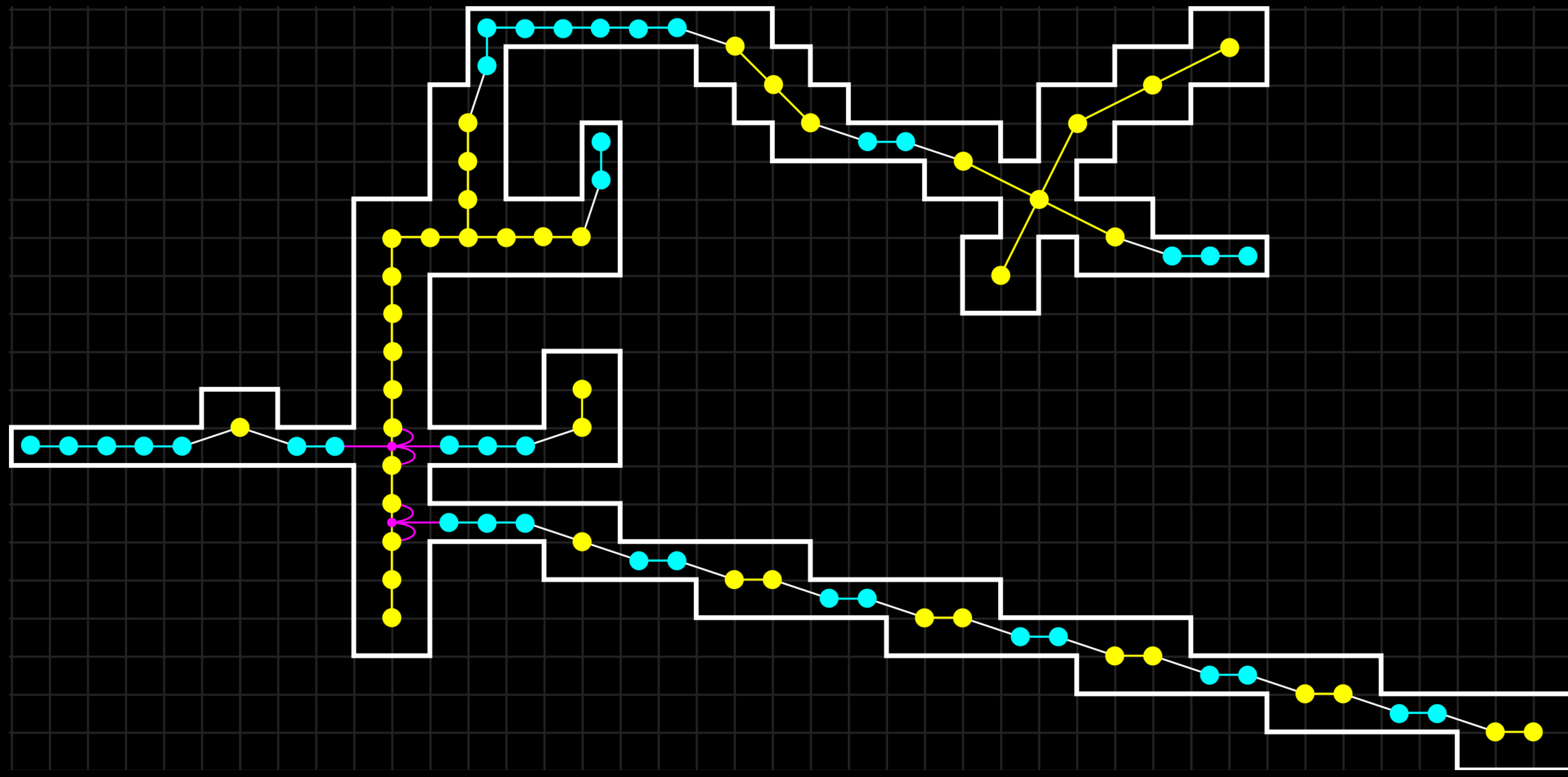
- With each node $v \in T$, we associate a block S(v) of unit squares:
  - $v = x_{u,v} \in X$: S(v) are the two unit squares with the edge {u,v}
  - $v \in I$: S(v) is the 2x2 block with internal vertex v

Associated Squares for the No



- With each node v∈T, we associate a block S(v) of unit squares:
  - $v=x_{u,v}\in X$: S(v) are the two unit squares with the edge {u,v}
  - v∈I: S(v) is the 2x2 block with internal vertex v
  - $v=b_s\in B$: S(v)={s}

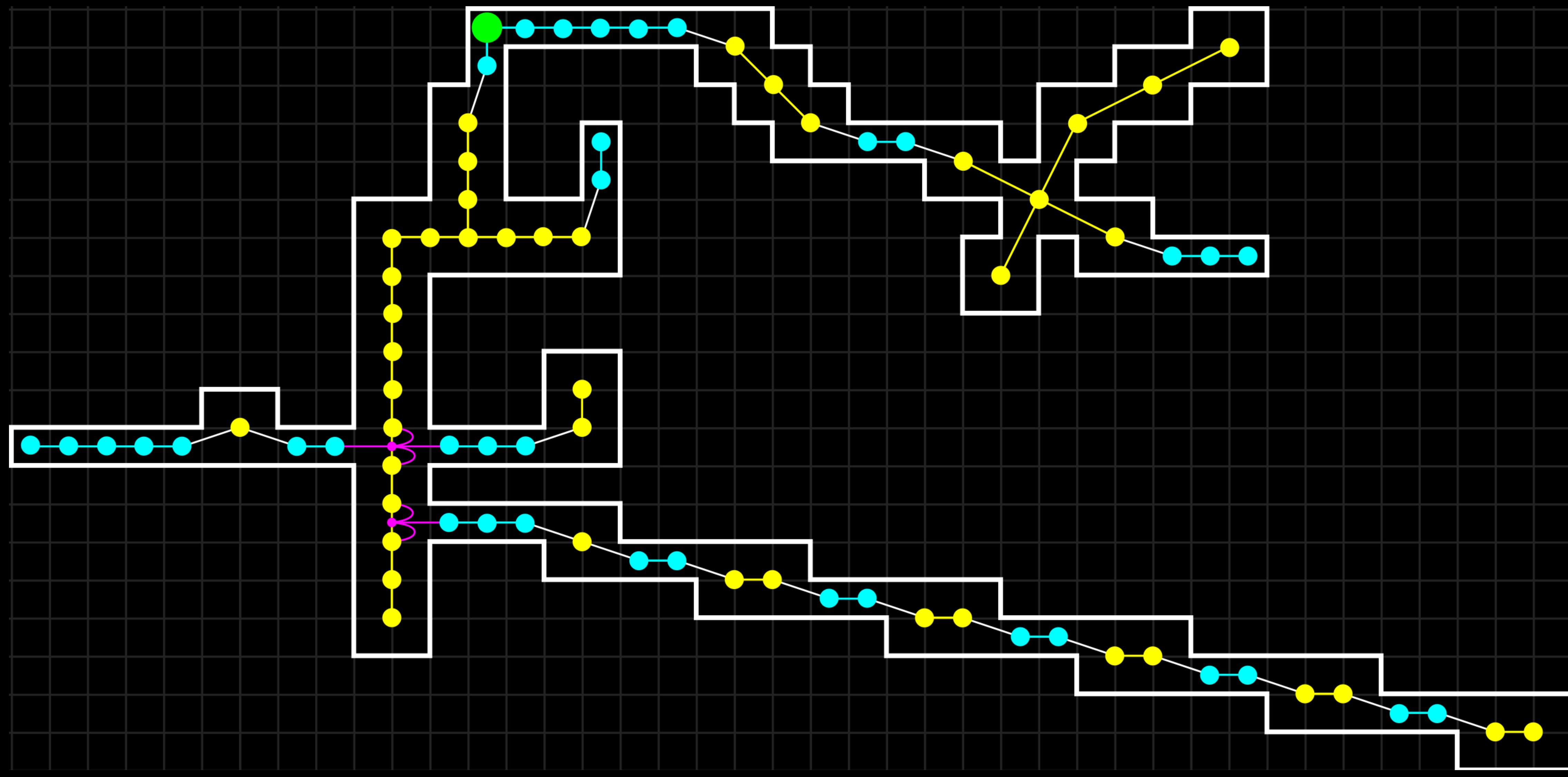
Our Approximation Algorithm



- Along the lines of algorithm from [+]

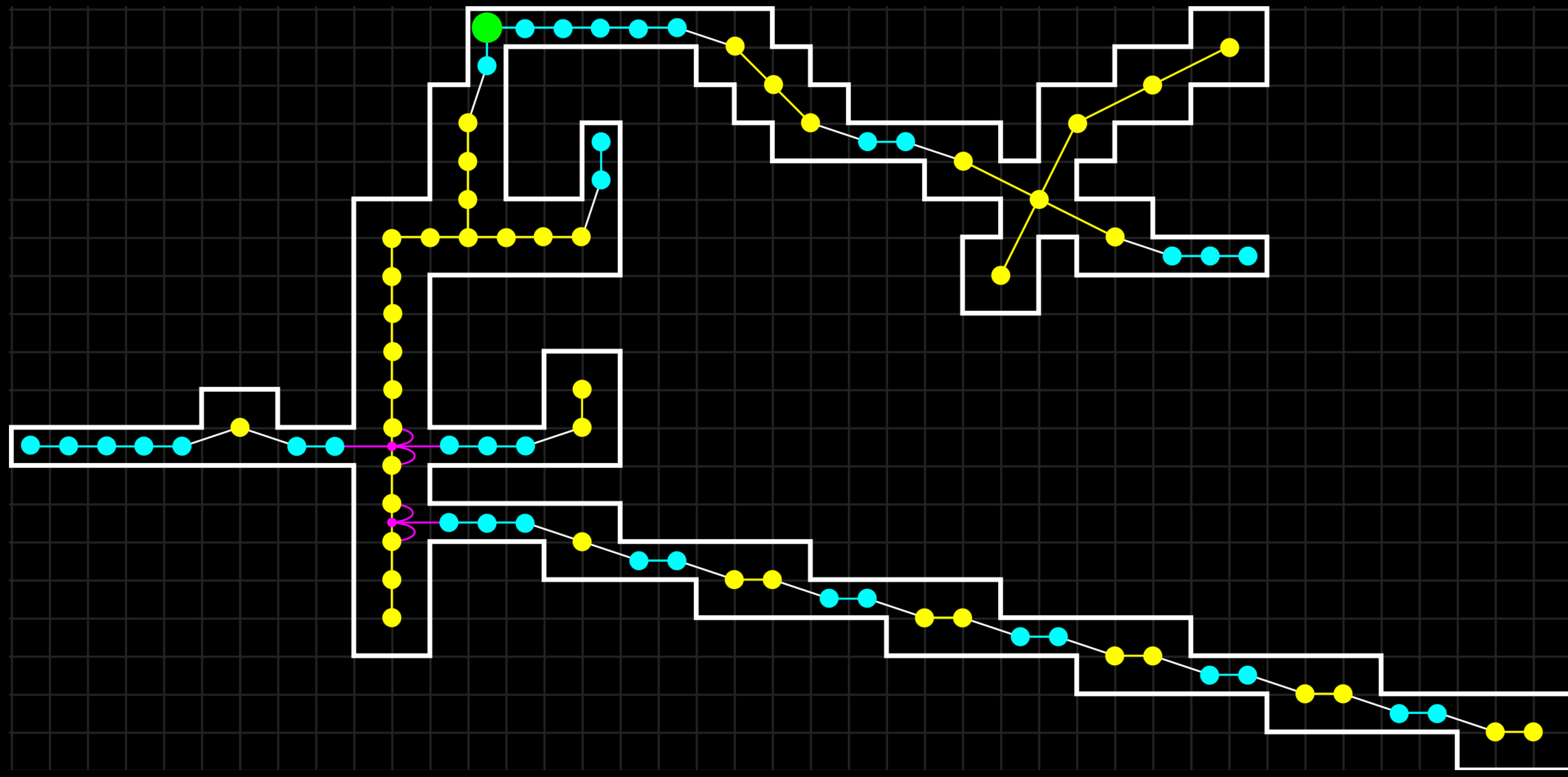
[+] A. Karim Abu-Affash, Paz Carmi, and Adi Krasin. A linear-time algorithm for minimum k-hop dominating set of a cactus graph. Discrete Applied Mathematics, 320:488–499, 2022

Our Approximation Algorithm



- Along the lines of algorithm from [+]
- Pick arbitrary node r from T as root

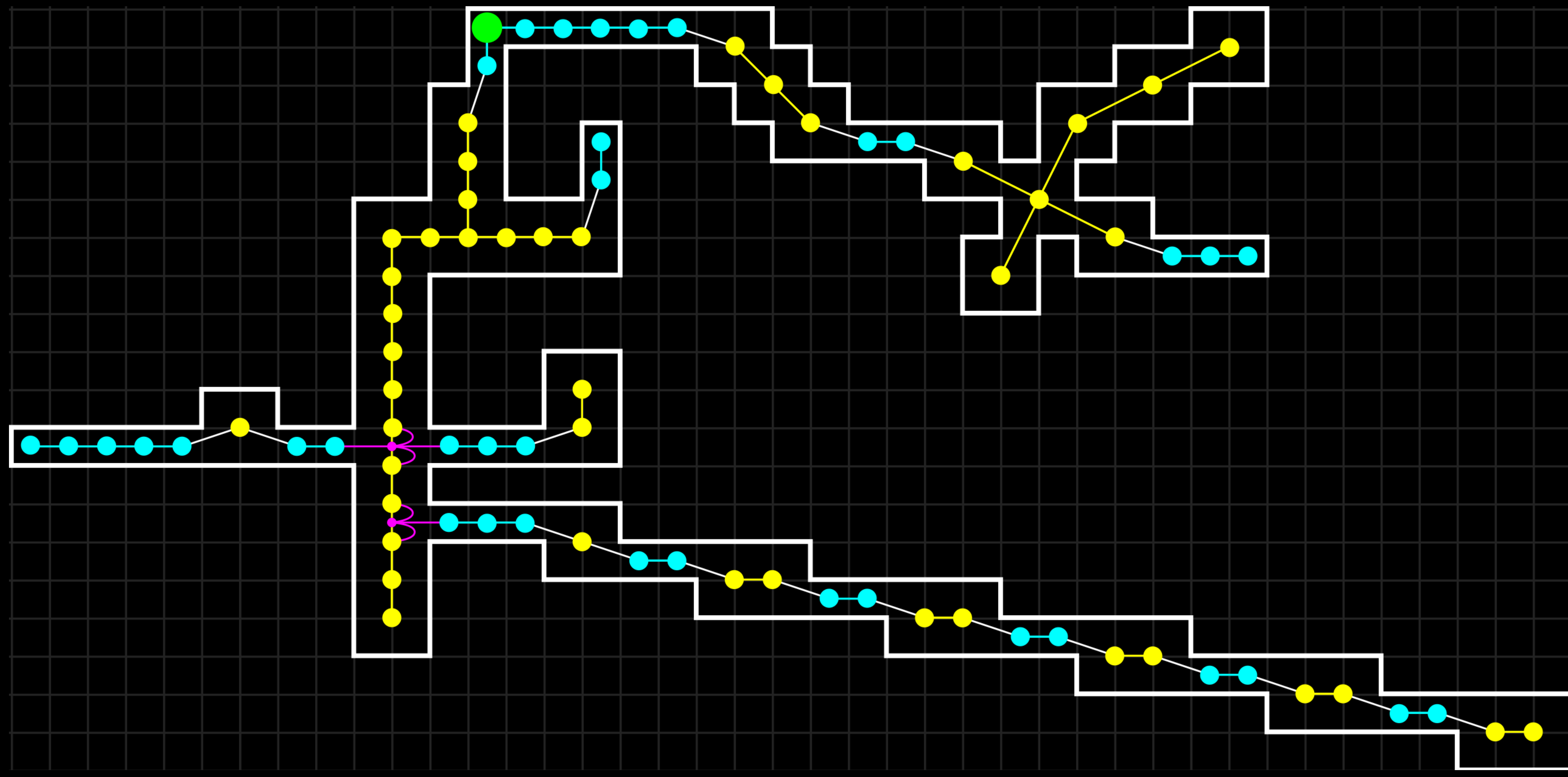
[+] A. Karim Abu-Affash, Paz Carmi, and Adi Krasin. A linear-time algorithm for minimum k-hop dominating set of a cactus graph. Discrete Applied Mathematics, 320:488–499, 2022



Our Approximation Algorithm

- Along the lines of algorithm from [+]
- Pick arbitrary node r from T as root
- For u∈T: T_u is subtree of T rooted at u

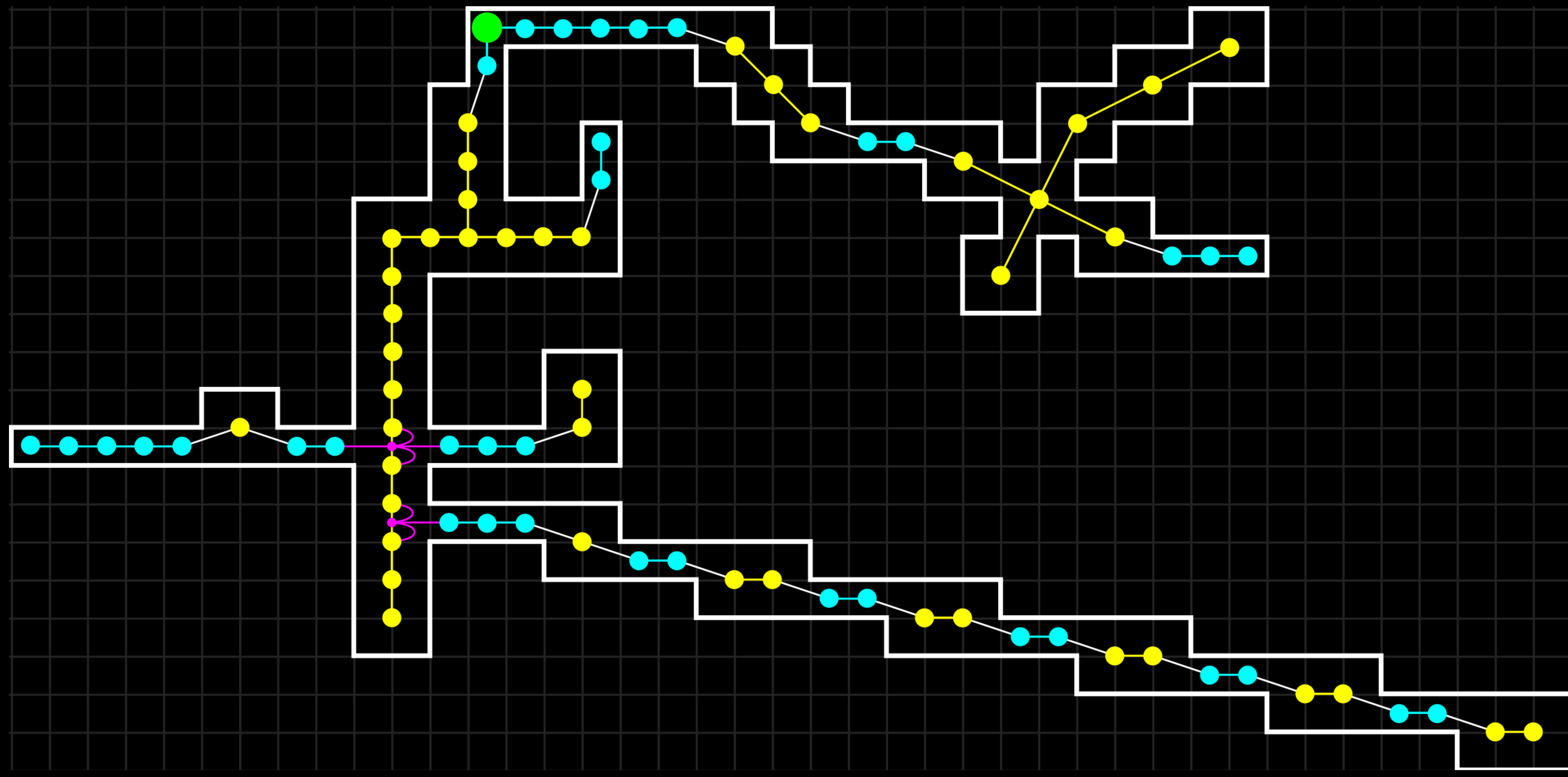
[+] A. Karim Abu-Affash, Paz Carmi, and Adi Krasin. A linear-time algorithm for minimum k-hop dominating set of a cactus graph. Discrete Applied Mathematics, 320:488–499, 2022



Our Approximation Algorithm

- Along the lines of algorithm from [+]
- Pick arbitrary node r from T as root
- For u∈T: T_u is subtree of T rooted at u
- Any path between unit square associated with a node in T_u and a unit square associated with a node in $T \setminus T_u$ includes a unit square from S(u)

[+] A. Karim Abu-Affash, Paz Carmi, and Adi Krasin. A linear-time algorithm for minimum k-hop dominating set of a cactus graph. Discrete Applied Mathematics, 320:488–499, 2022

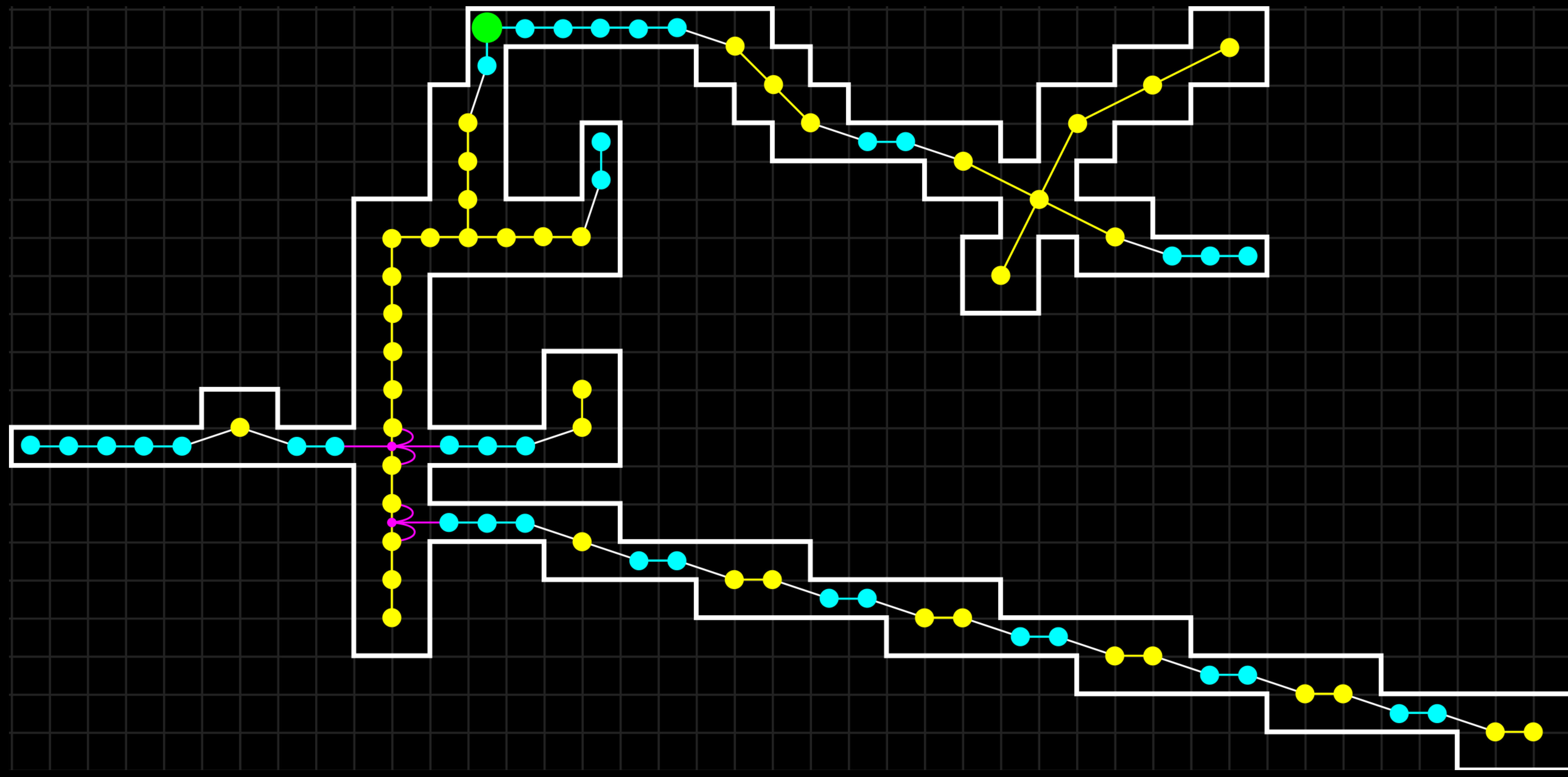


Our Approximation Algorithm

- Along the lines of algorithm from [+]
- Pick arbitrary node r from T as root
- For $u \in T$: T_u is subtree of T rooted at u
- Any path between unit square associated with a node in T_u and a unit square associated with a node in $T \setminus T_u$ includes a unit square from S(u)
- For $u \in T$: $h(T_u) = \max_{v \in T\_u.\ s \in S(v)} \min_{s' \in S(u)} d_P(s,s')$ (hop distance between s and s')
- $h(T_u)$ is the largest hop distance from a unit square in $\cup_{v \cup T\_u} S(v)$ to its closest unit square from S(u)

[+] A. Karim Abu-Affash, Paz Carmi, and Adi Krasin. A linear-time algorithm for minimum k-hop dominating set of a cactus graph. Discrete Applied Mathematics, 320:488–499, 2022

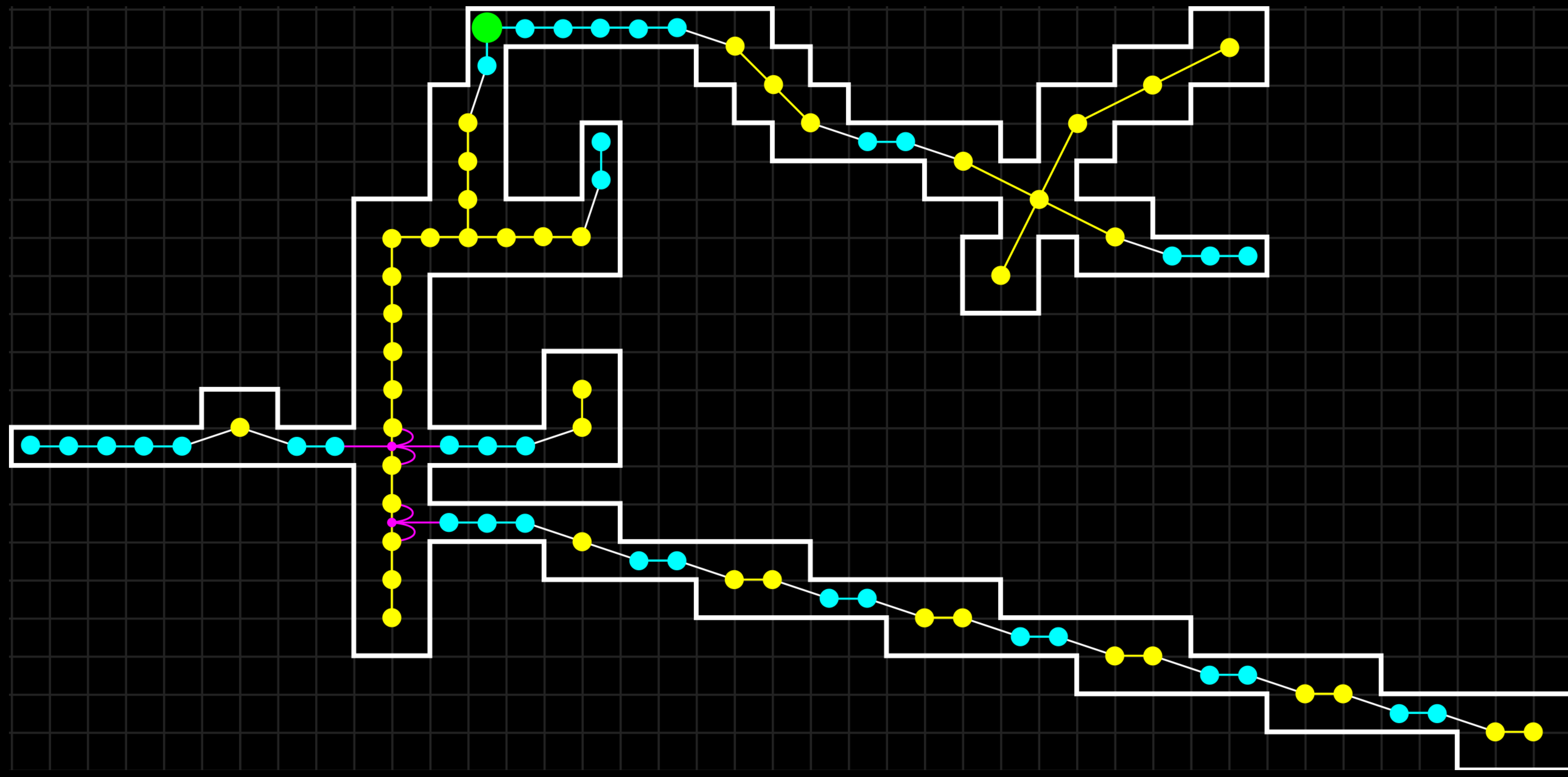
LINKÖPING UNIVERSITY



Our Approximation Algorithm

- Along the lines of algorithm from [+]
- Pick arbitrary node r from T as root
- For u∈T: T_u is subtree of T rooted at u
- Any path between unit square associated with a node in T_u and a unit square associated with a node in $T \setminus T_u$ includes a unit square from S(u)
- For u∈T: $h(T_u)= \max_{v \in T\_u,\ s \in S(v)} \min_{s' \in S(u)} d_P(s,s')$ (hop distance between s and s')
- $h(T_u)$ is the largest hop distance from a unit square in $\cup_{v \cup T\_u}$ S(v) to its closest unit square from S(u)
- For each s'∈ $\cup_{v \in T\_u}$ S(v) the minimum distance $\min_{s' \in S(u)} d_P(s,s')$ is assumed at a unit square s— M(s)=set of all these cells for which the distance is assumed for s

[+] A. Karim Abu-Affash, Paz Carmi, and Adi Krasin. A linear-time algorithm for minimum k-hop dominating set of a cactus graph. Discrete Applied Mathematics, 320:488–499, 2022

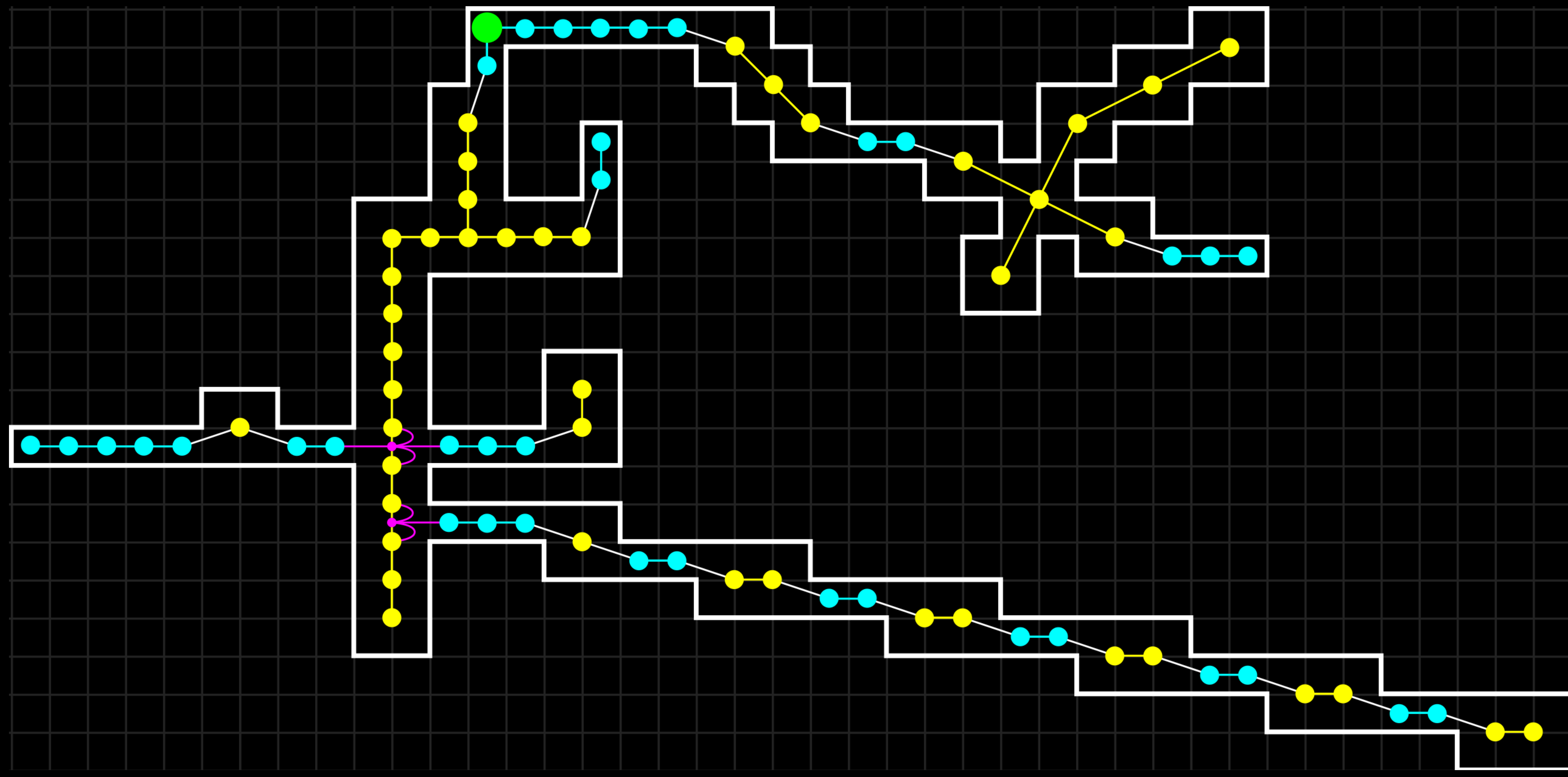


Our Approximation Algorithm

- Along the lines of algorithm from [+]
- Pick arbitrary node r from T as root
- For u∈T: T_u is subtree of T rooted at u
- Any path between unit square associated with a node in T_u and a unit square associated with a node in $T \setminus T_u$ includes a unit square from S(u)
- For u∈T: $h(T_u)= \max_{v\in T\_u.\ s\in S(v)} \min_{s'\in S(u)} d_P(s,s')$ (hop distance between s and s')
- $h(T_u)$ is the largest hop distance from a unit square in $\cup_{v\cup T\_u}$ S(v) to its closest unit square from S(u)
- For each $s' \in \cup_{v\in T\_u}$ S(v) the minimum distance $\min_{s'\in S(u)} d_P(s,s')$ is assumed at a unit square s— M(s)=set of all these cells for which the distance is assumed for s
- We set $h_s(T_u)= \max_{s'\in M(s)} d_P(s,s')$

[+] A. Karim Abu-Affash, Paz Carmi, and Adi Krasin. A linear-time algorithm for minimum k-hop dominating set of a cactus graph. Discrete Applied Mathematics, 320:488–499, 2022

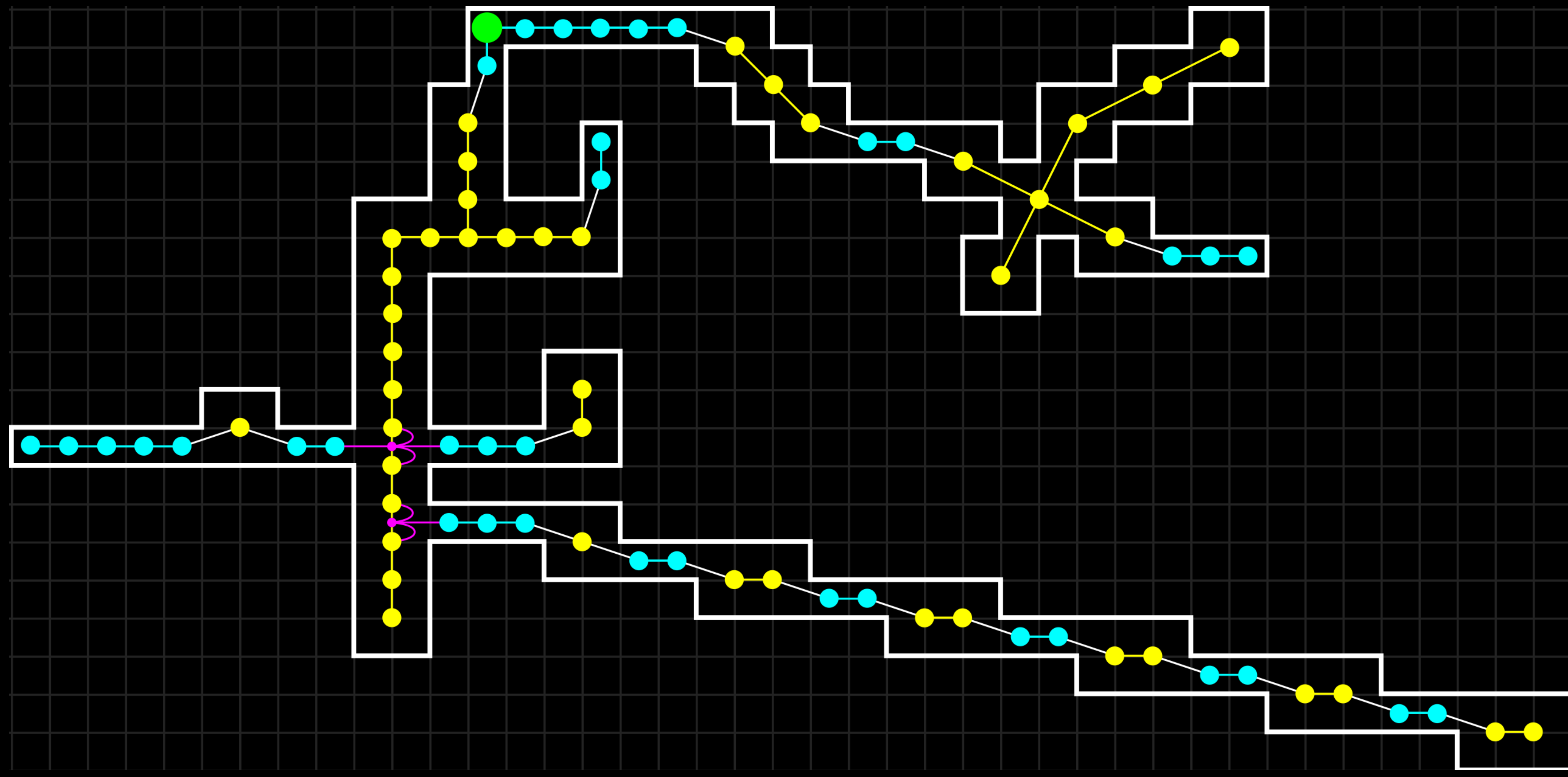
Our Approximation Algorithm



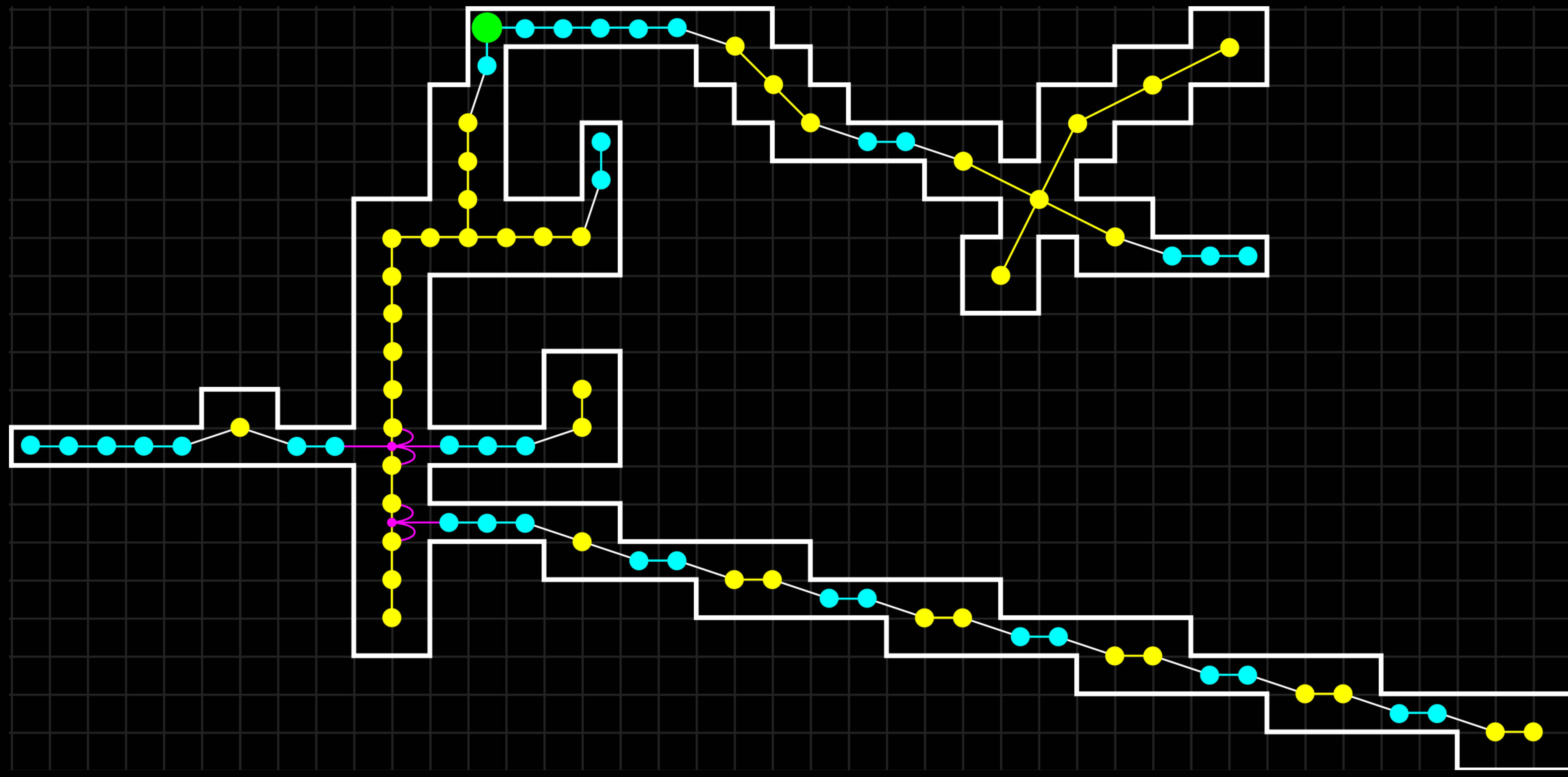
- Along the lines of algorithm from [+]
- Pick arbitrary node r from T as root
- For u∈T: T_u is subtree of T rooted at u
- Any path between unit square associated with a node in T_u and a unit square associated with a node in $T \setminus T_u$ includes a unit square from S(u)
- For u∈T: $h(T_u)= \max_{v\in T\_u.\ s\in S(v)} \min_{s'\in S(u)} d_P(s,s')$ (hop distance between s and s')
- $h(T_u)$ is the largest hop distance from a unit square in $\cup_{v\cup T\_u}$ S(v) to its closest unit square from S(u)
- For each s'∈ $\cup_{v\in T\_u}$ S(v) the minimum distance $\min_{s'\in S(u)} d_P(s,s')$ is assumed at a unit square s— M(s)=set of all these cells for which the distance is assumed for s
- We set $h_s(T_u)= \max_{s'\in M(s)} d_P(s,s')$
- If $h(T_u)$=k and we pick S(u) for the guard set ➜ every unit square associated with a node in T_u guarded

[+] A. Karim Abu-Affash, Paz Carmi, and Adi Krasin. A linear-time algorithm for minimum k-hop dominating set of a cactus graph. Discrete Applied Mathematics, 320:488–499, 2022

Our Approximation Algorithm



- D=∅ (guard set),
- Compute $h(T_u)$ for every $u \in T$
- Compute $h_s(T_u)$ for every $s \in S(u)$

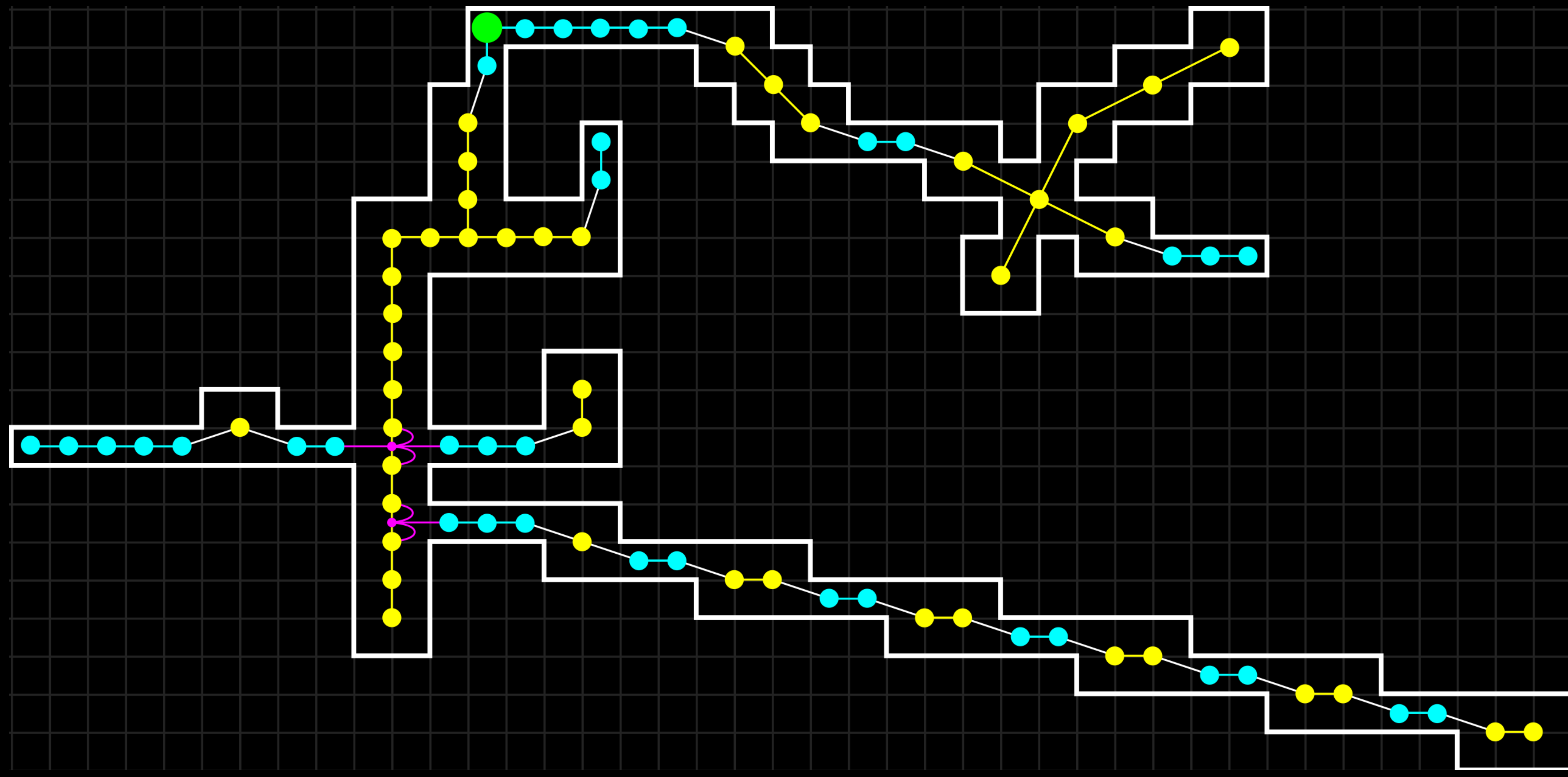


Our Approximation Algorithm

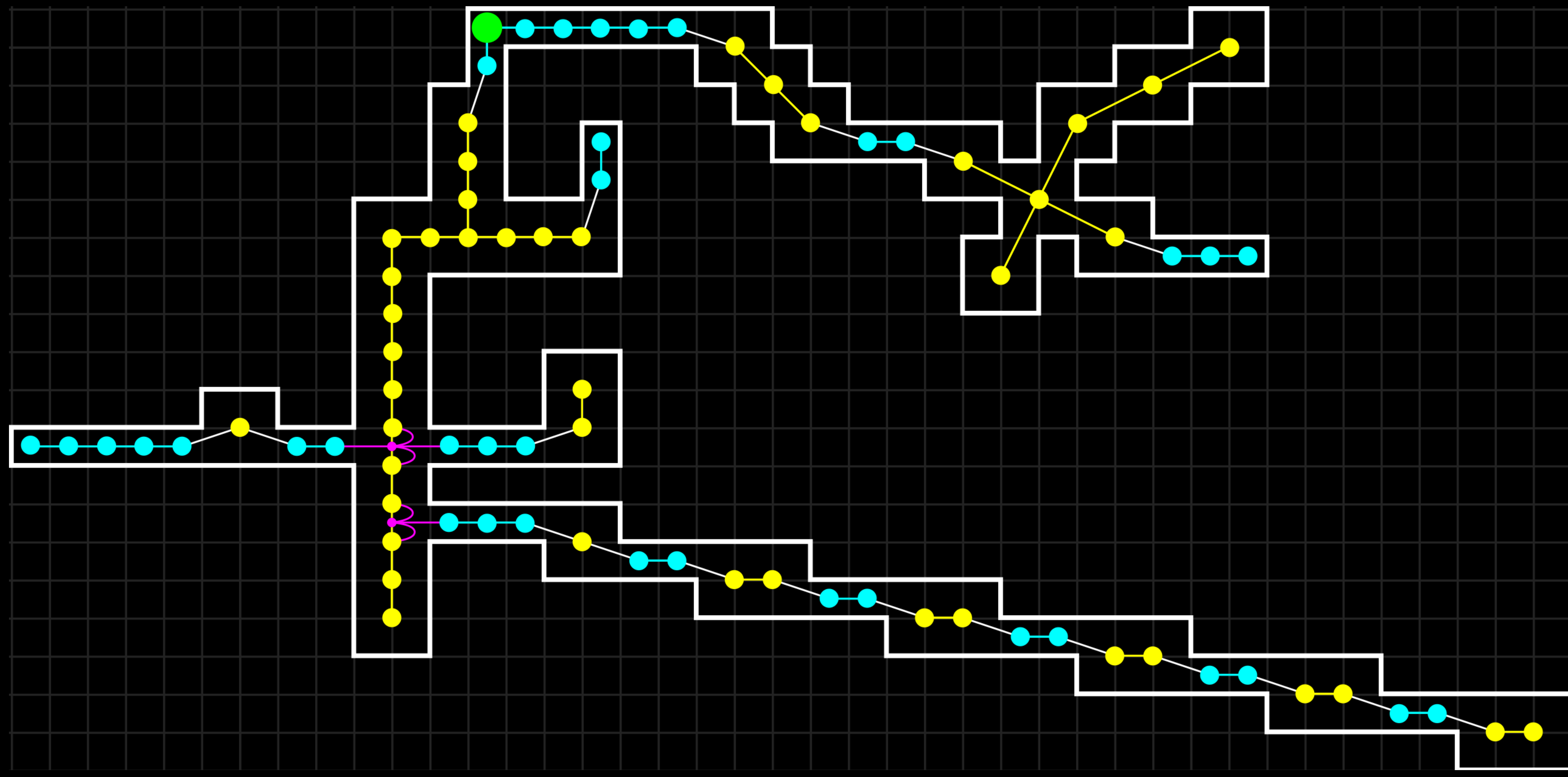
- $D = \emptyset$ (guard set),
- Compute $h(T_u)$ for every $u \in T$
- Compute $h_s(T_u)$ for every $s \in S(u)$
- $\forall s \in P$: set $rb_D = -1$, the maximum rest budget of the unit square s over all squares in guard set D (up to rest budget of 0, s is *k*-hop visible to the nodes in D)

LINKÖPING UNIVERSITY

Our Approximation Algorithm



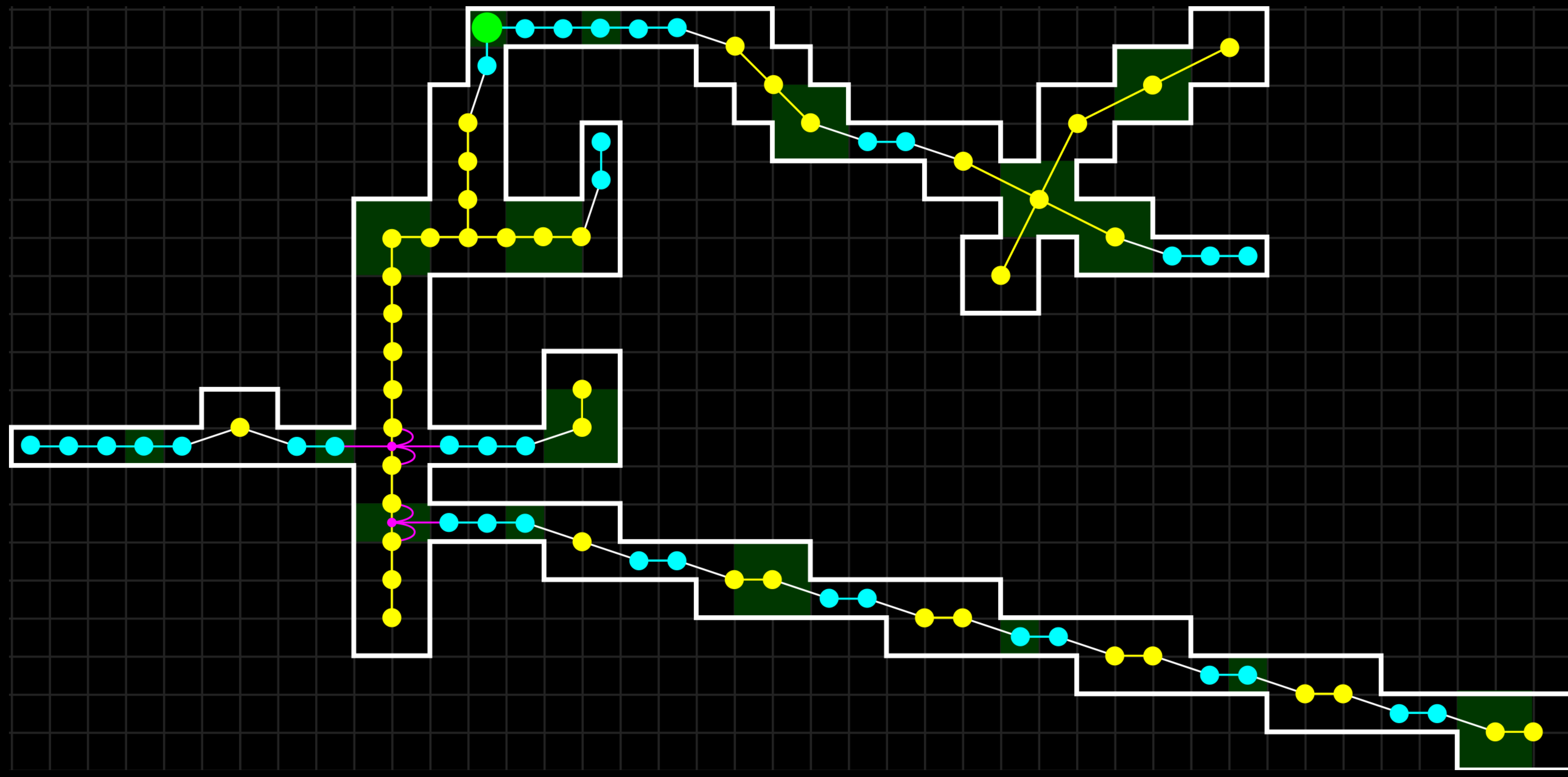
- D=∅ (guard set),
- Compute $h(T_u)$ for every u∈T
- Compute $h_s(T_u)$ for every s∈S(u)
- ∀s∈P: set rb_D=-1, the maximum rest budget of the unit square s over all squares in guard set D (up to rest budget of 0, s is *k*-hop visible to the nodes in D)
- Run DFS starting from root r



Our Approximation Algorithm

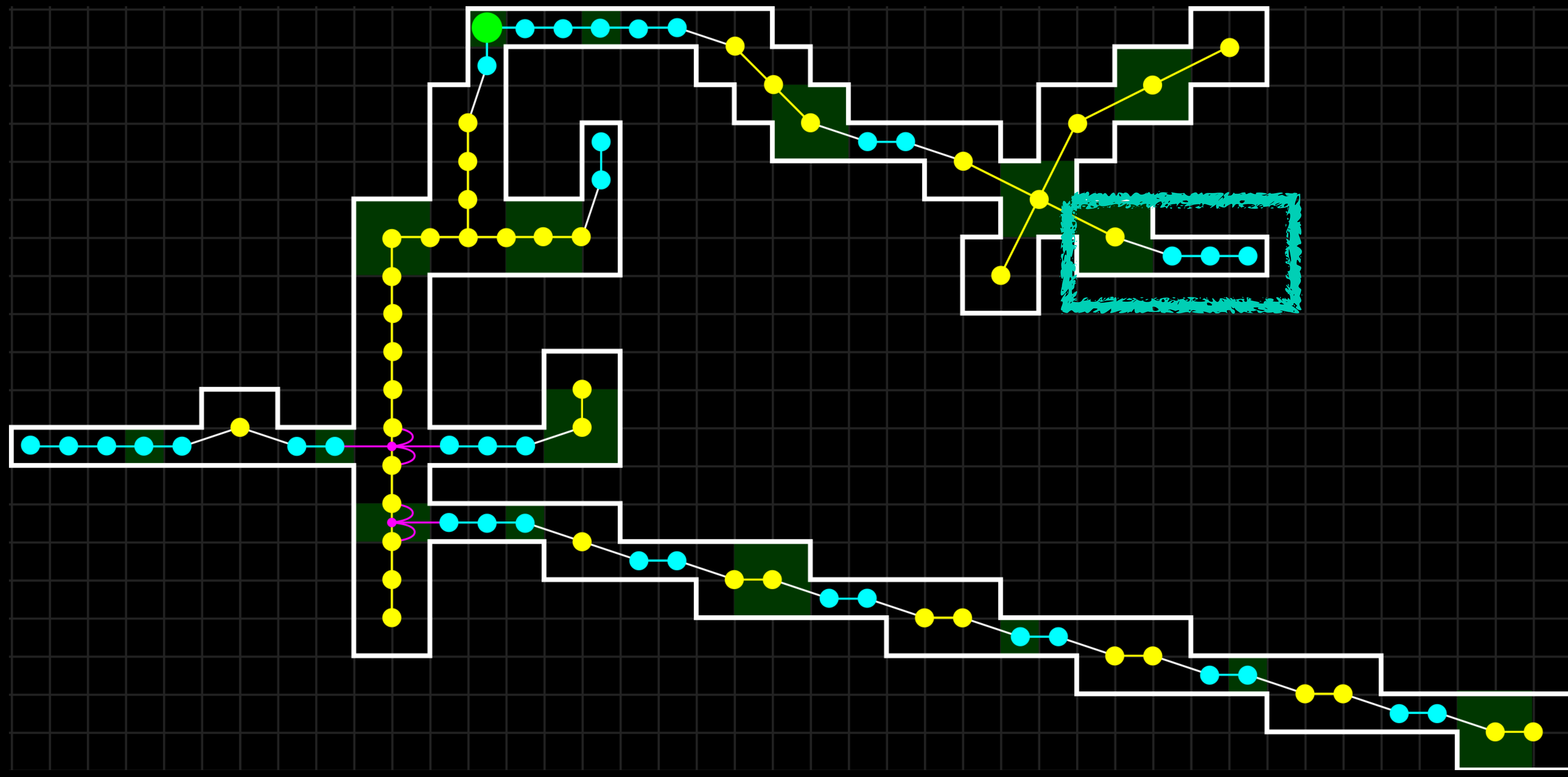
- D=∅ (guard set),
- Compute $h(T_u)$ for every u∈T
- Compute $h_s(T_u)$ for every s∈S(u)
- ∀s∈P: set rb_D=-1, the maximum rest budget of the unit square s over all squares in guard set D (up to rest budget of 0, s is *k*-hop visible to the nodes in D)
- Run DFS starting from root r
  - If $h(T_r) \leq k$, we output S(r) as guard set

D for k=3



Our Approximation Algorithm

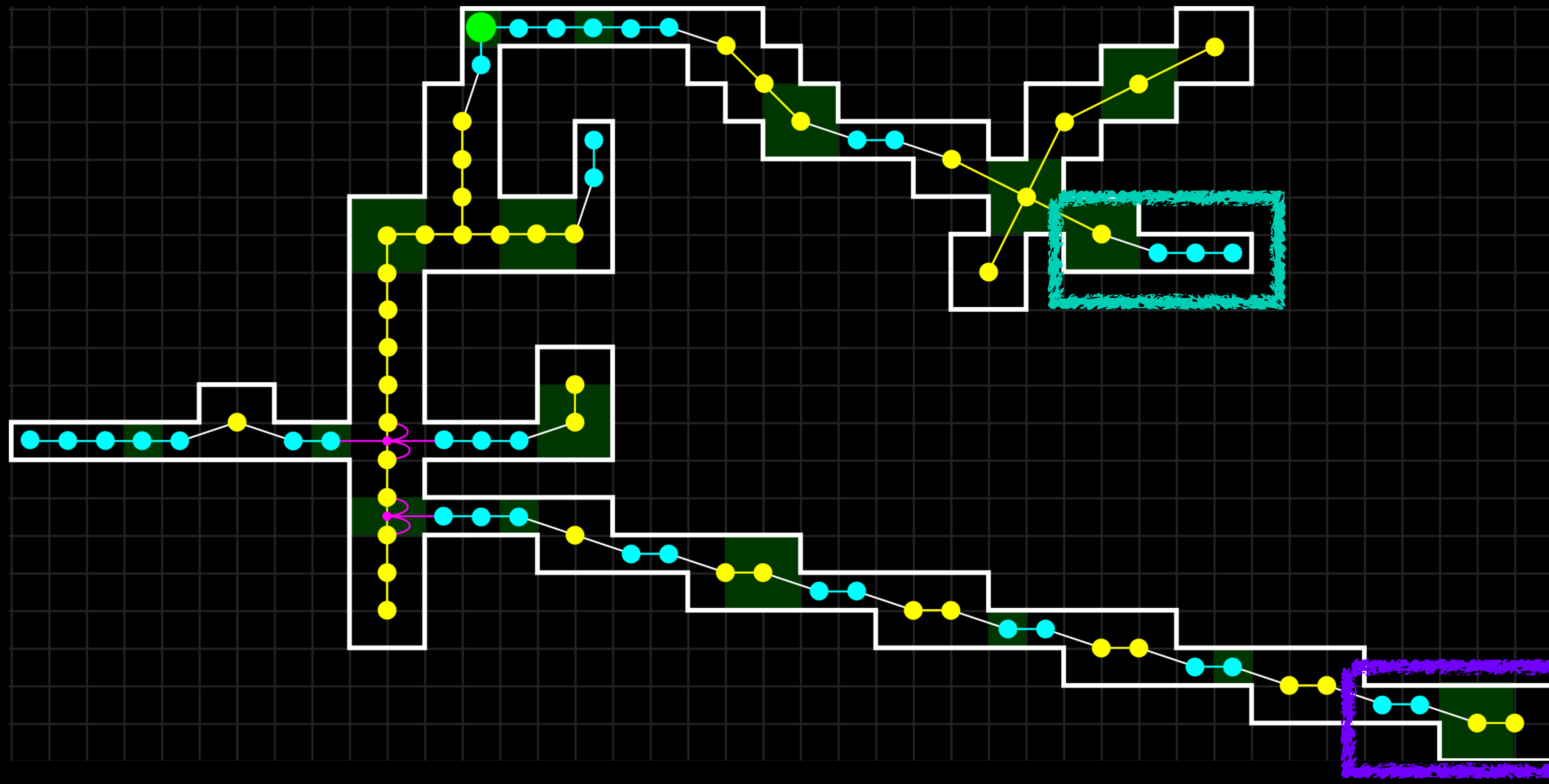
- D=∅ (guard set),
- Compute $h(T_u)$ for every $u \in T$
- Compute $h_s(T_u)$ for every $s \in S(u)$
- $\forall s \in P$: set rb_D=-1, the maximum rest budget of the unit square s over all squares in guard set D (up to rest budget of 0, s is *k*-hop visible to the nodes in D)
- Run DFS starting from root r
  - If $h(T_r) \leq k$, we output S(r) as guard set
  - u=current node in DFS call



Our Approximation Algorithm

- D=∅ (guard set),
- Compute $h(T_u)$ for every u∈T
- Compute $h_s(T_u)$ for every s∈S(u)
- ∀s∈P: set rb_D=-1, the maximum rest budget of the unit square s over all squares in guard set D (up to rest budget of 0, s is *k*-hop visible to the nodes in D)
- Run DFS starting from root r
  - If $h(T_r) \leq k$, we output S(r) as guard set
  - u=current node in DFS call
    1. If $h(T_u)=k$: add S(u) to D, remove T_u from T, set $rb_D=k$ ∀s∈S(u)

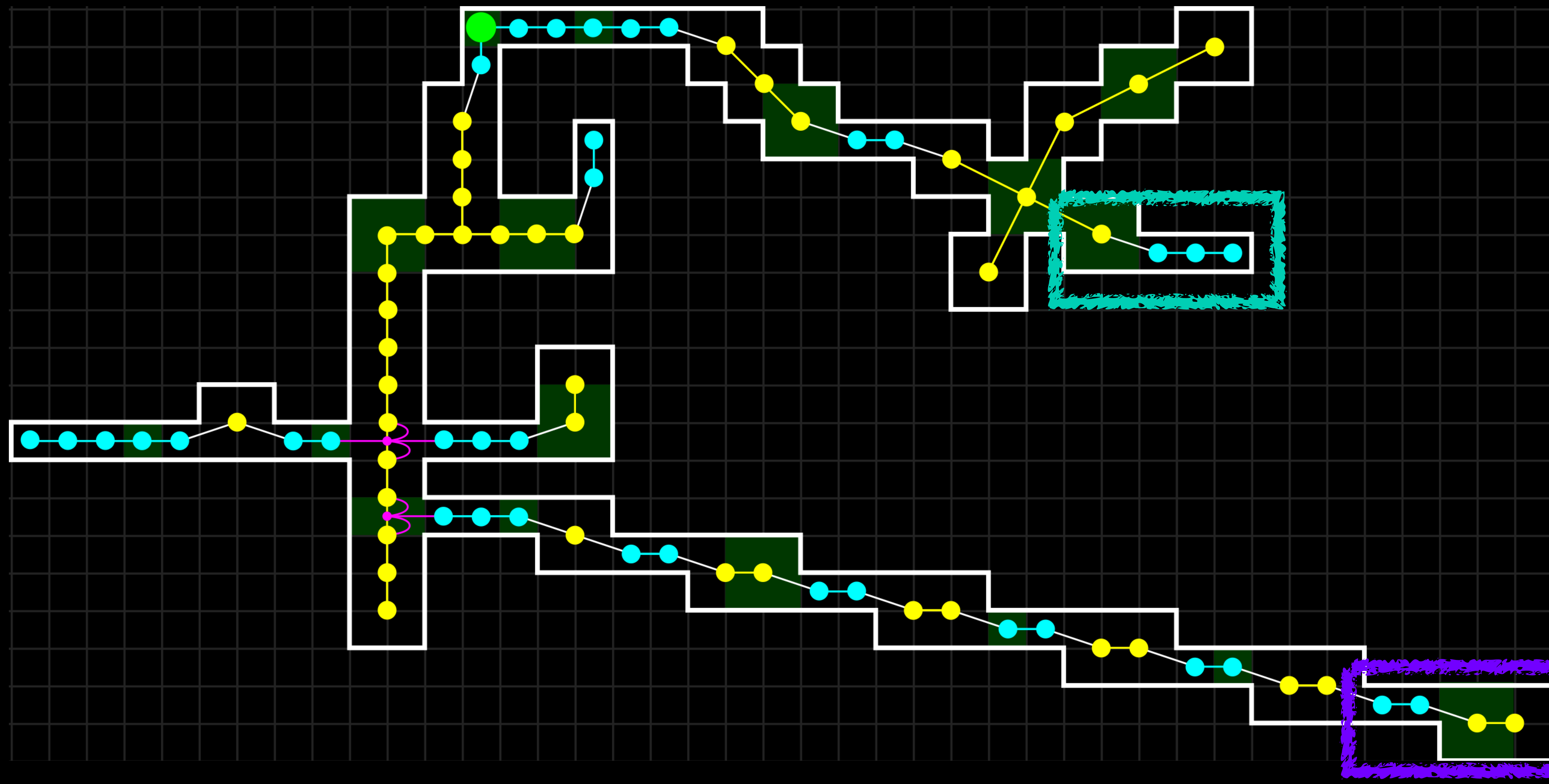
LINKÖPING UNIVERSITY



Our Approximation Algorithm

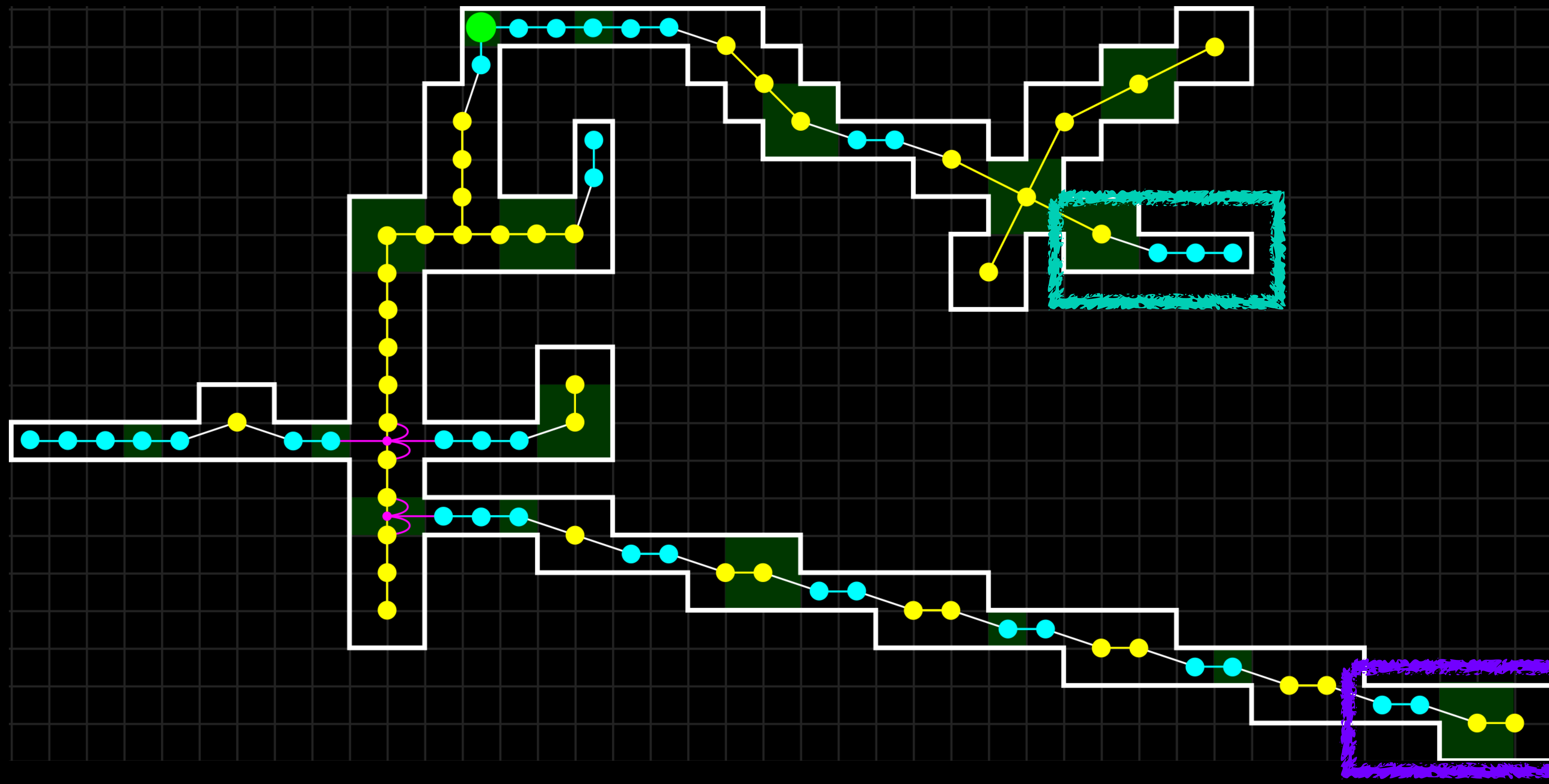
- D=∅ (guard set),
- Compute $h(T_u)$ for every u∈T
- Compute $h_s(T_u)$ for every s∈S(u)
- ∀s∈P: set rb_D=-1, the maximum rest budget of the unit square s over all squares in guard set D (up to rest budget of 0, s is *k*-hop visible to the nodes in D)
- Run DFS starting from root r
  - If $h(T_r) \leq k$, we output S(r) as guard set
  - u=current node in DFS call
    1. If $h(T_u)=k$: add S(u) to D, remove T_u from T, set $rb_D=k$ ∀s∈S(u)
    2. Else, if $k-1 \geq h(T_u) \geq k-2$ and $\min_{s' \in S(p(u=)} d_P(s,s') > k$ for the parent p(u) of u and $s' \in T_u$ the unit square that realizes $h(T_u)$: add S(u) to D, remove T_u from T, set $rb_D=k$ ∀s∈S(u)

D for k=3



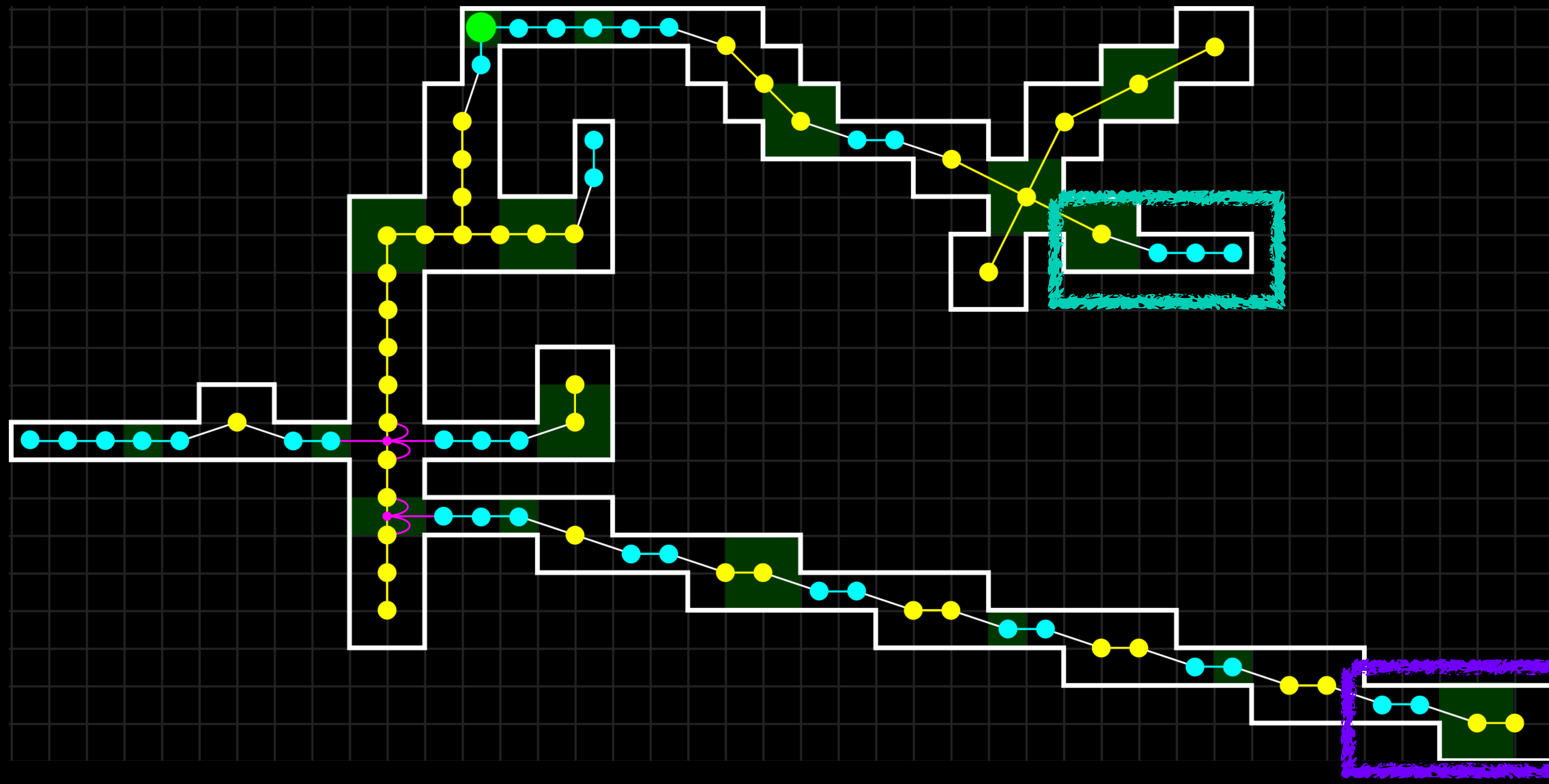
Our Approximation Algorithm

- D=∅ (guard set),
- Compute $h(T_u)$ for every u∈T
- Compute $h_s(T_u)$ for every s∈S(u)
- ∀s∈P: set rb_D=-1, the maximum rest budget of the unit square s over all squares in guard set D (up to rest budget of 0, s is k-hop visible to the nodes in D)
- Run DFS starting from root r
  - If $h(T_r) \leq k$, we output S(r) as guard set
  - u=current node in DFS call
    1. If $h(T_u)=k$: add S(u) to D, remove T_u from T, set $rb_D=k$ ∀s∈S(u)
    2. Else, if $k-1 \geq h(T_u) \geq k-2$ and $\min_{s' \in S(p(u=)} d_P(s,s') > k$ for the parent p(u) of u and $s' \in T_u$ the unit square that realizes $h(T_u)$: add S(u) to D, remove T_u from T, set $rb_D=k$ ∀s∈S(u)
    3. Else, for each child v of u with $h(T_v) \geq k-2$, run DFS on v. Update $h(T_u)$ and $h_s(T_u)$, $rb_D(s)$ for s∈S(u)



Our Approximation Algorithm

- D=∅ (guard set),
- Compute $h(T_u)$ for every u∈T
- Compute $h_s(T_u)$ for every s∈S(u)
- ∀s∈P: set rb_D=-1, the maximum rest budget of the unit square s over all squares in guard set D (up to rest budget of 0, s is *k*-hop visible to the nodes in D)
- Run DFS starting from root r
  - If $h(T_r) \leq k$, we output S(r) as guard set
  - u=current node in DFS call
    1. If $h(T_u)$=k: add S(u) to D, remove T_u from T, set rb_D=k ∀s∈S(u)
    2. Else, if $k-1 \geq h(T_u) \geq k-2$ and $\min_{s' \in S(p(u=)} d_P(s,s') > k$ for the parent p(u) of u and $s' \in T_u$ the unit square that realizes $h(T_u)$: add S(u) to D, remove T_u from T, set rb_D=k ∀s∈S(u)
    3. Else, for each child v of u with $h(T_v) \geq k-2$, run DFS on v. Update $h(T_u)$ and $h_s(T_u)$, $rb_D(s)$ for s∈S(u)
       (a) Check if remaining T_u is already guarded by D (only consider associates squares with negative rest budget)



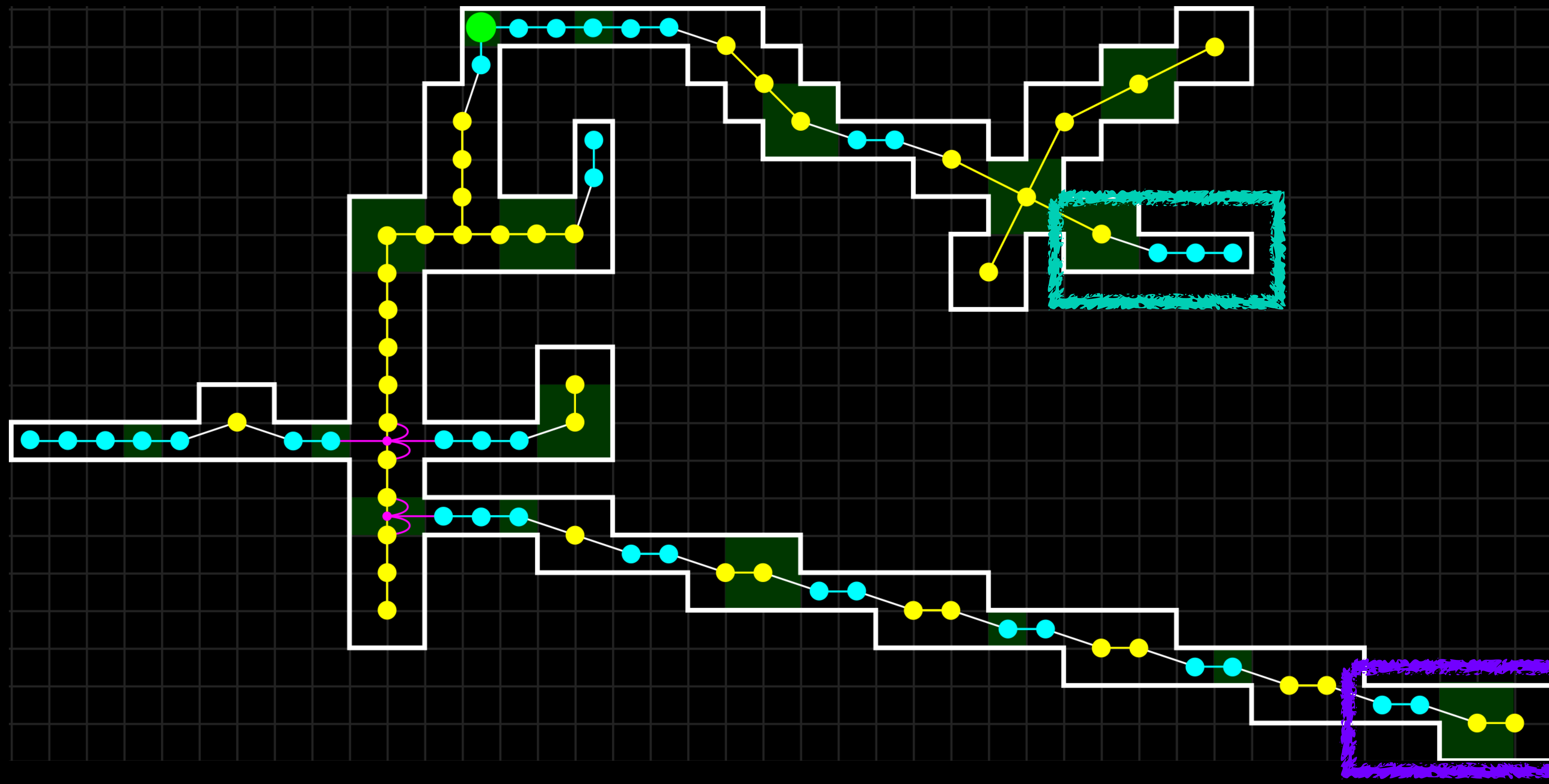
D for k=3

Our Approximation Algorithm

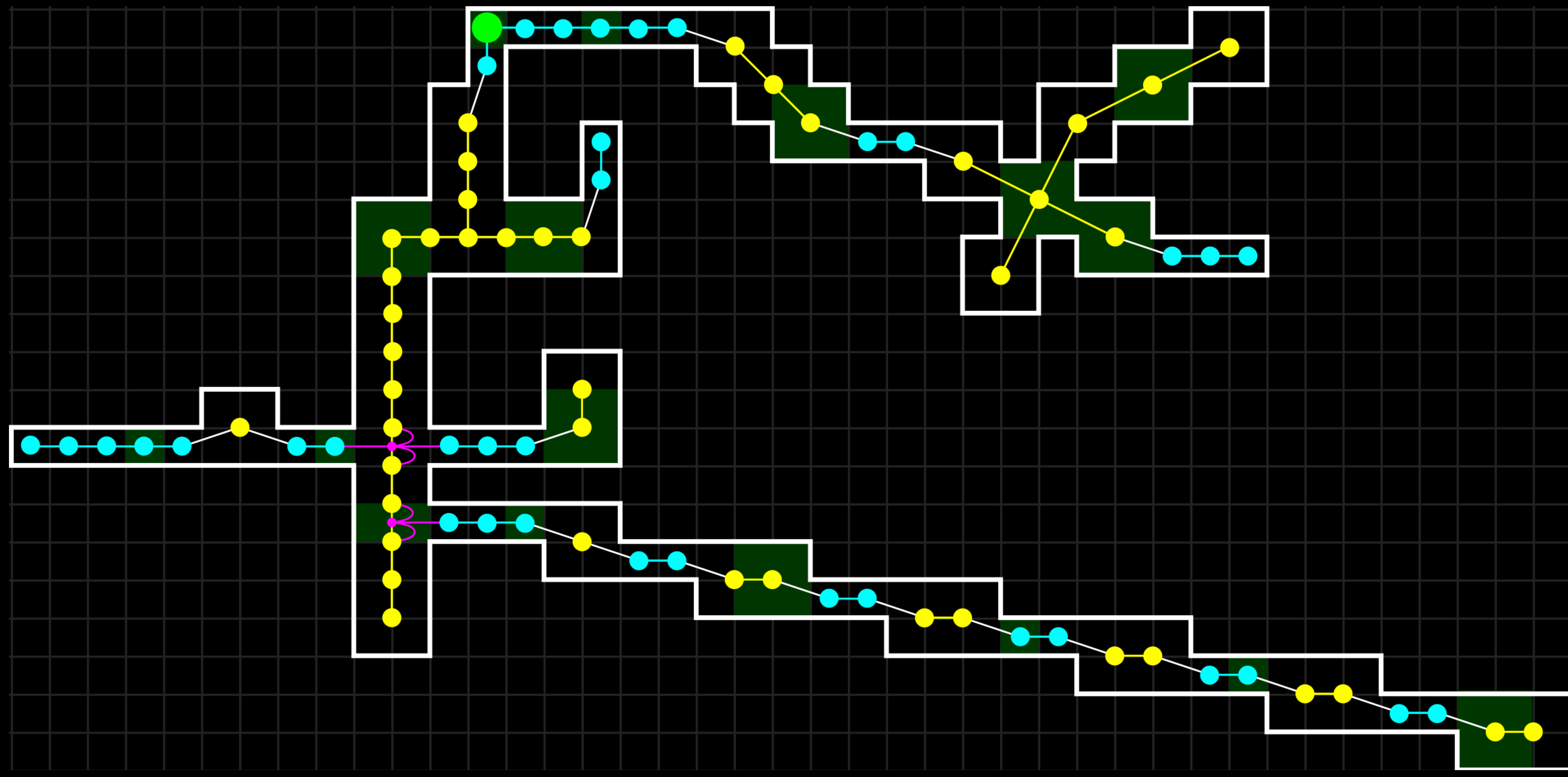
- D=∅ (guard set),
- Compute $h(T_u)$ for every u∈T
- Compute $h_s(T_u)$ for every s∈S(u)
- ∀s∈P: set rb_D=-1, the maximum rest budget of the unit square s over all squares in guard set D (up to rest budget of 0, s is k-hop visible to the nodes in D)
- Run DFS starting from root r
  - If $h(T_r)\leq k$, we output S(r) as guard set
  - u=current node in DFS call
    1. If $h(T_u)=k$: add S(u) to D, remove T_u from T, set $rb_D=k$ ∀s∈S(u)
    2. Else, if $k-1\geq h(T_u)\geq k-2$ and $\min_{s'\in S(p(u=)} d_P(s,s') > k$ for the parent p(u) of u and $s'\in T_u$ the unit square that realizes $h(T_u)$: add S(u) to D, remove T_u from T, set $rb_D=k$ ∀s∈S(u)
    3. Else, for each child v of u with $h(T_v)\geq k-2$, run DFS on v. Update $h(T_u)$ and $h_s(T_u)$, $rb_D(s)$ for s∈S(u)
       (a) Check if remaining T_u is already guarded by D (only consider associates squares with negative rest budget)
       (b) Else, if new $h(T_u)=k$ or if condition 2. holds: add S(u) to D, remove T_u from T, set $rb_D=k$ ∀s∈S(u)

D for k=3

Our Approximation Algorithm

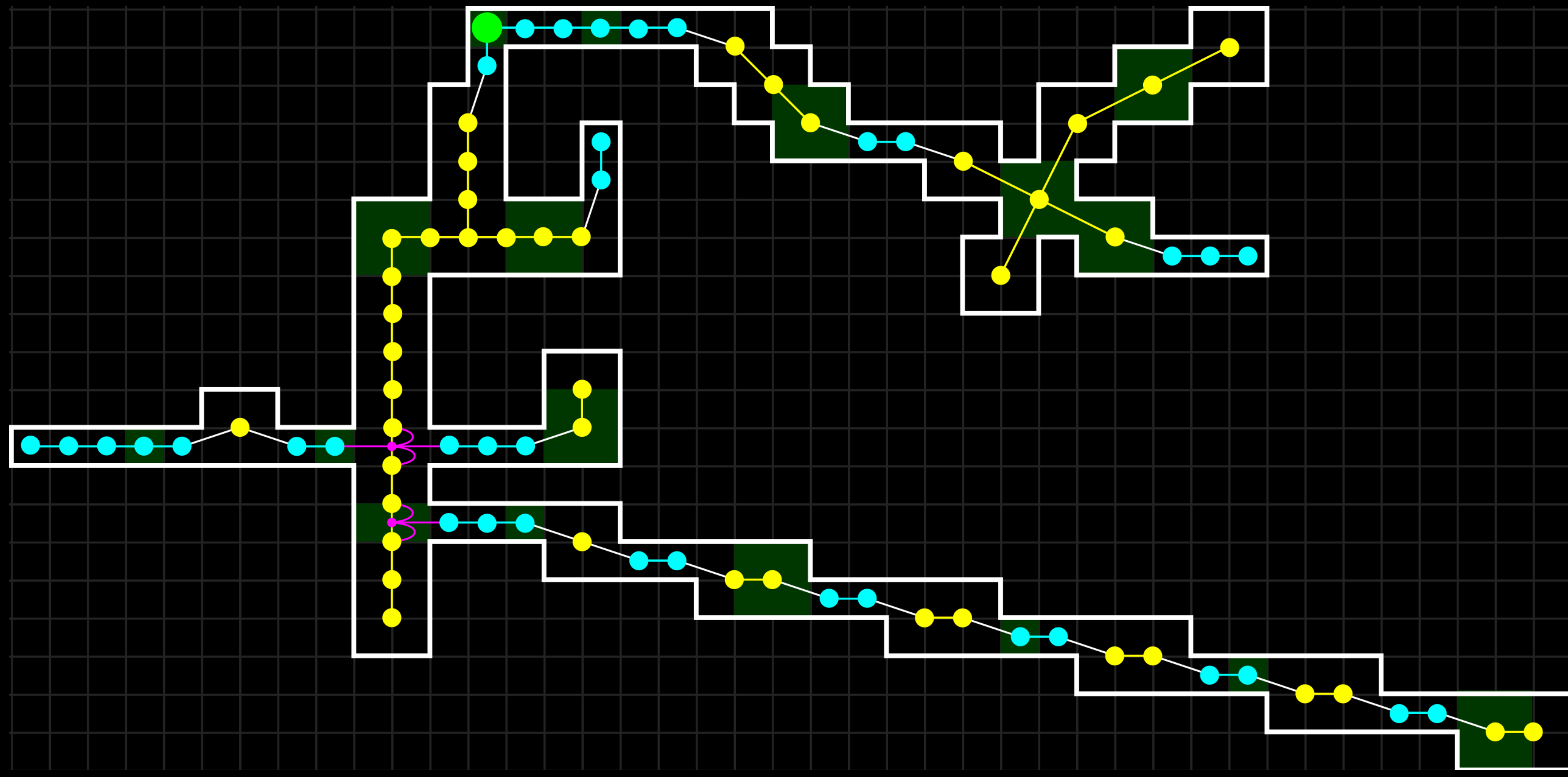


- D=∅ (guard set),
- Compute $h(T_u)$ for every u∈T
- Compute $h_s(T_u)$ for every s∈S(u)
- ∀s∈P: set rb_D=-1, the maximum rest budget of the unit square s over all squares in guard set D (up to rest budget of 0, s is k-hop visible to the nodes in D)
- Run DFS starting from root r
  - If $h(T_r) \leq k$, we output S(r) as guard set
  - u=current node in DFS call
    1. If $h(T_u)=k$: add S(u) to D, remove T_u from T, set $rb_D=k$ ∀s∈S(u)
    2. Else, if $k-1 \geq h(T_u) \geq k-2$ and $\min_{s' \in S(p(u=)} d_P(s,s') > k$ for the parent p(u) of u and $s' \in T_u$ the unit square that realizes $h(T_u)$: add S(u) to D, remove T_u from T, set $rb_D=k$ ∀s∈S(u)
    3. Else, for each child v of u with $h(T_v) \geq k-2$, run DFS on v. Update $h(T_u)$ and $h_s(T_u)$, $rb_D(s)$ for s∈S(u)
       (a) Check if remaining T_u is already guarded by D (only consider associates squares with negative rest budget)
       (b) Else, if new $h(T_u)=k$ or if condition 2. holds: add S(u) to D, remove T_u from T, set $rb_D=k$ ∀s∈S(u)
  - If at the end we have $rb_D(s)=-1$ for some s∈S(r): add S(r) to D



Yields Approx Factor of 4!

Theorem 4: There is a linear-time algorithm computing a 4-approximation for the M*k*GP in simple 2-thin polyominoes.

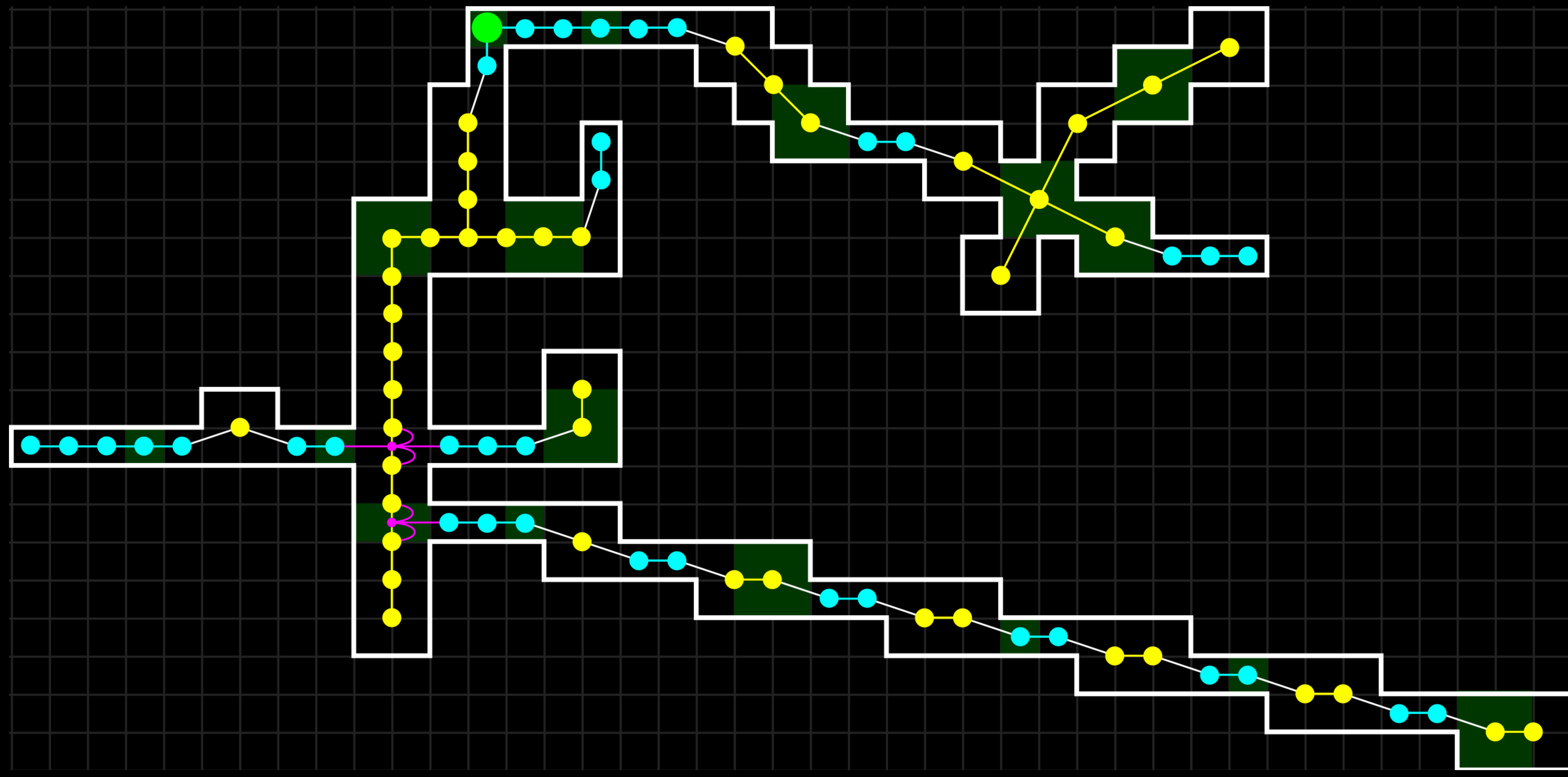


Yields Approx Factor of 4!

Theorem 4: There is a linear-time algorithm computing a 4-approximation for the M*k*GP in simple 2-thin polyominoes.

Proof:

We only remove a node v from T if S(v) is covered by cells in D.



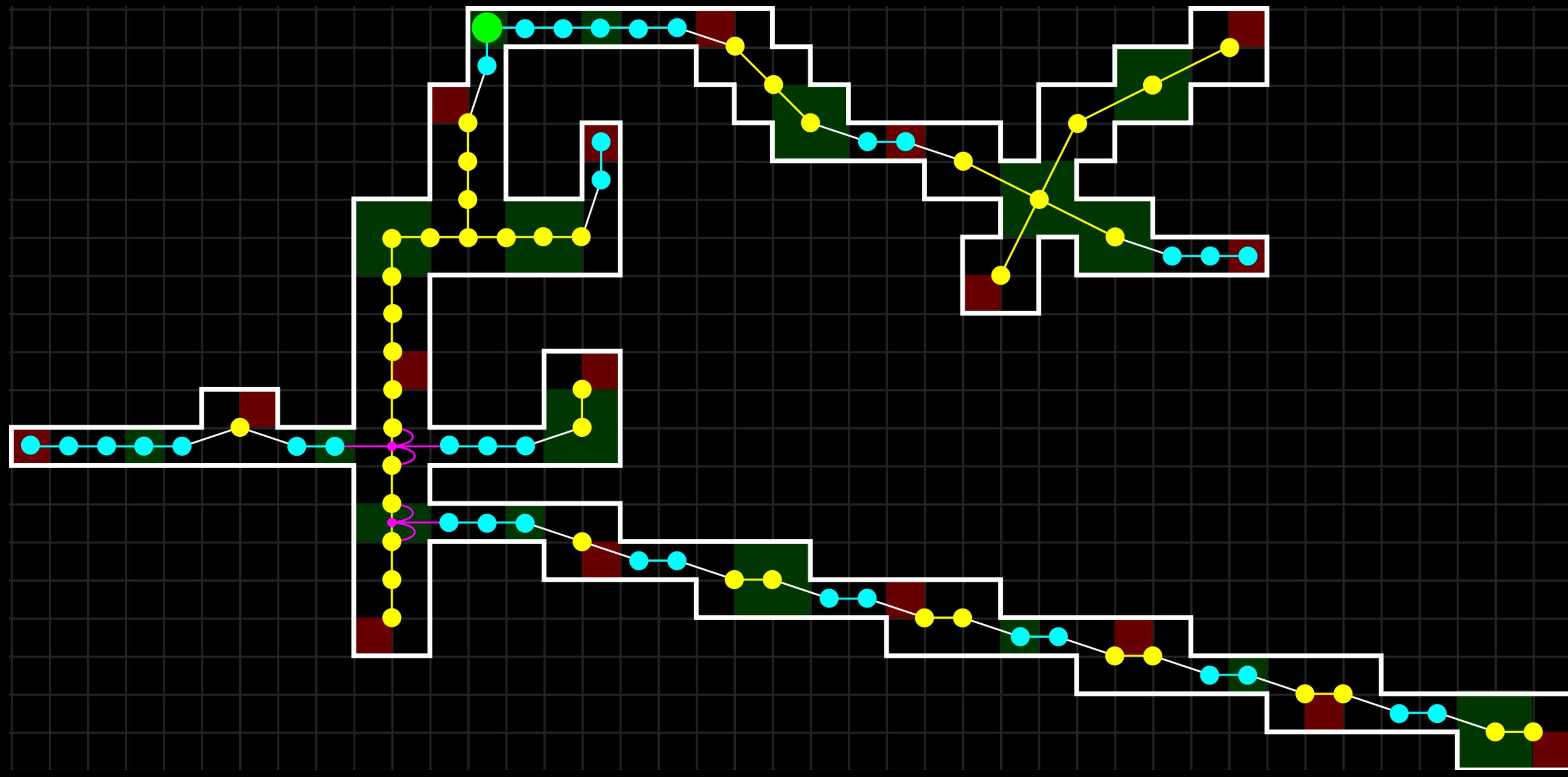
Yields Approx Factor of 4!

Theorem 4: There is a linear-time algorithm computing a 4-approximation for the M*k*GP in simple 2-thin polyominoes.

Proof:

We only remove a node v from T if S(v) is covered by cells in D.

$\cup_{v \in T} S(v) = P$ ➜ D is a *k*-hop visibility guard set



Yields Approx Factor of 4!

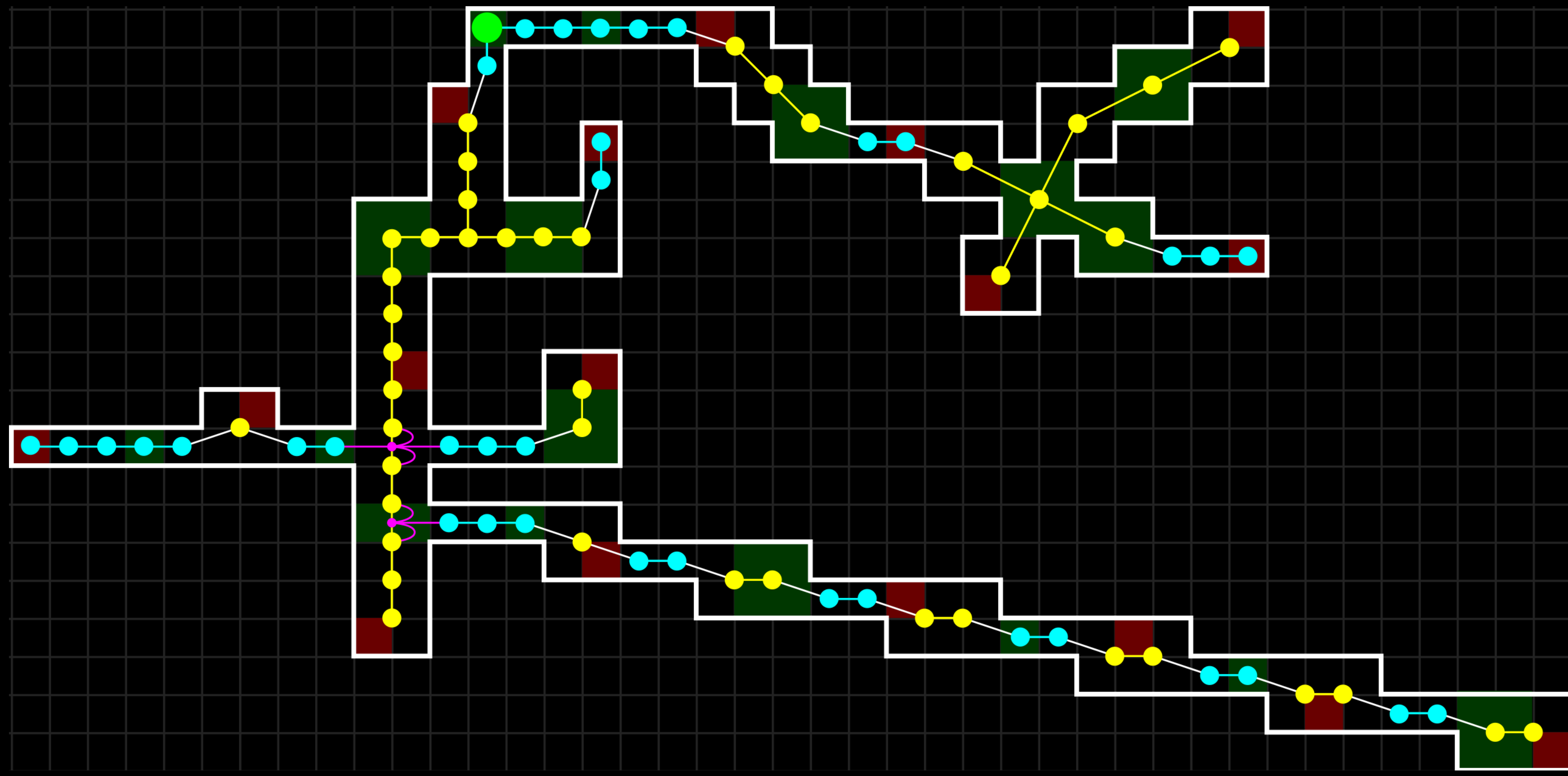
Theorem 4: There is a linear-time algorithm computing a 4-approximation for the M*k*GP in simple 2-thin polyominoes.

Proof:

We only remove a node v from T if S(v) is covered by cells in D.

$\bigcup_{v \in T} S(v) = P$ ➜ D is a *k*-hop visibility guard set

We show: for each node in sequence $u_1,\ldots u_\ell$ of T for which we added $S(u_i)$ to D, we can find a witness square s_i in T_{u_i}, such that no two witness squares $s_i \neq s_j$ have a single unit square in P within hop distance k from both



Yields Approx Factor of 4!

Theorem 4: There is a linear-time algorithm computing a 4-approximation for the M*k*GP in simple 2-thin polyominoes.

Proof:

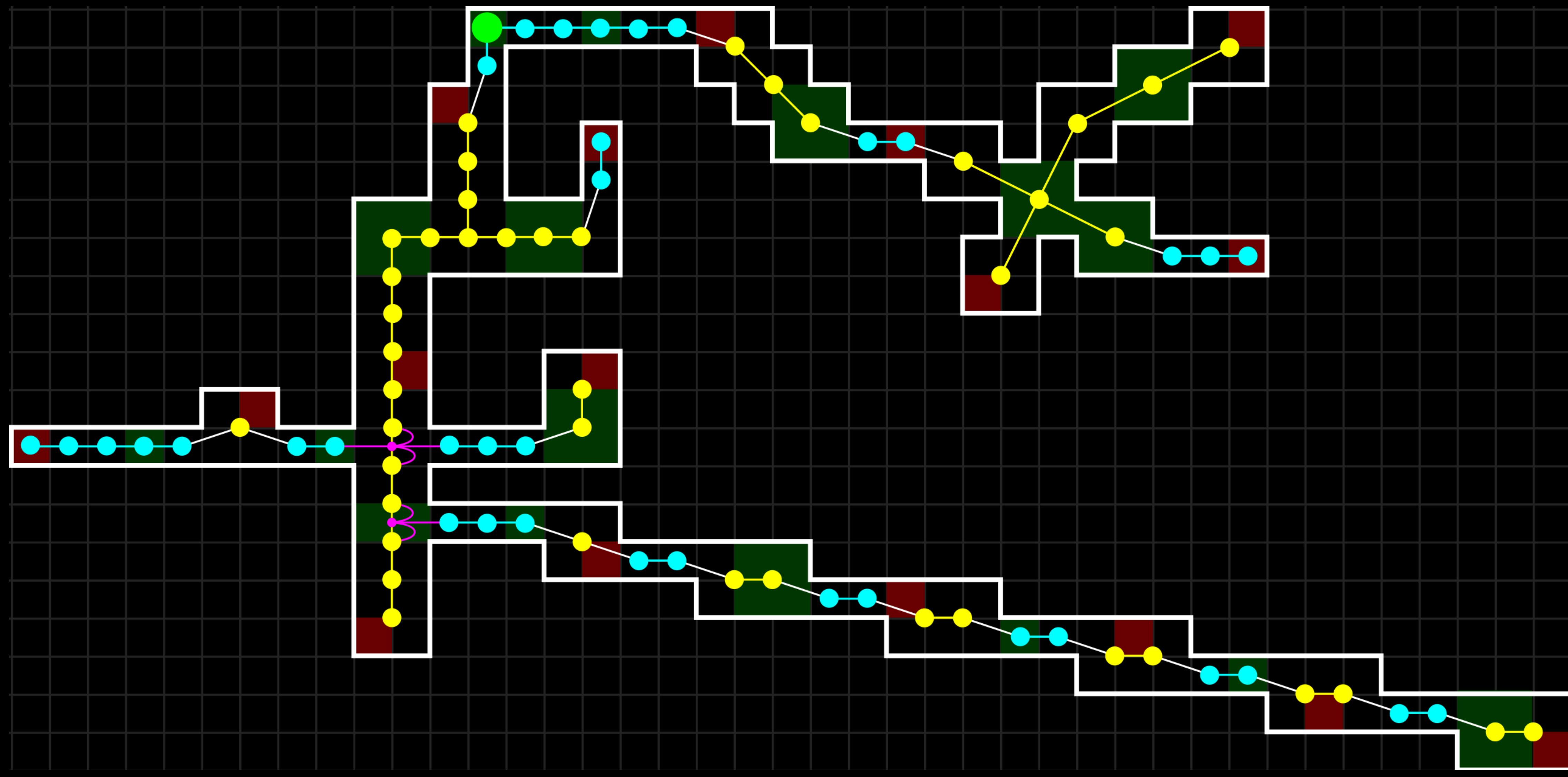
We only remove a node v from T if S(v) is covered by cells in D.

$\bigcup_{v \in T} S(v) = P$ ➜ D is a *k*-hop visibility guard set

We show: for each node in sequence $u_1,\ldots u_\ell$ of T for which we added $S(u_i)$ to D, we can find a witness square s_i in T_{u_i}, such that no two witness squares $s_i \neq s_j$ have a single unit square in P within hop distance k from both

➜ OPT $\geq \ell$

Yields Approx Factor of 4!



Theorem 4: There is a linear-time algorithm computing a 4-approximation for the M*k*GP in simple 2-thin polyominoes.

Proof:

We only remove a node v from T if S(v) is covered by cells in D.

$\cup_{v \in T} S(v) = P$ ➜ D is a *k*-hop visibility guard set

We show: for each node in sequence $u_1, \ldots u_\ell$ of T for which we added $S(u_i)$ to D, we can find a witness square s_i in T_{u_i}, such that no two witness squares $s_i \neq s_j$ have a single unit square in P within hop distance k from both

➜ OPT $\geq \ell$

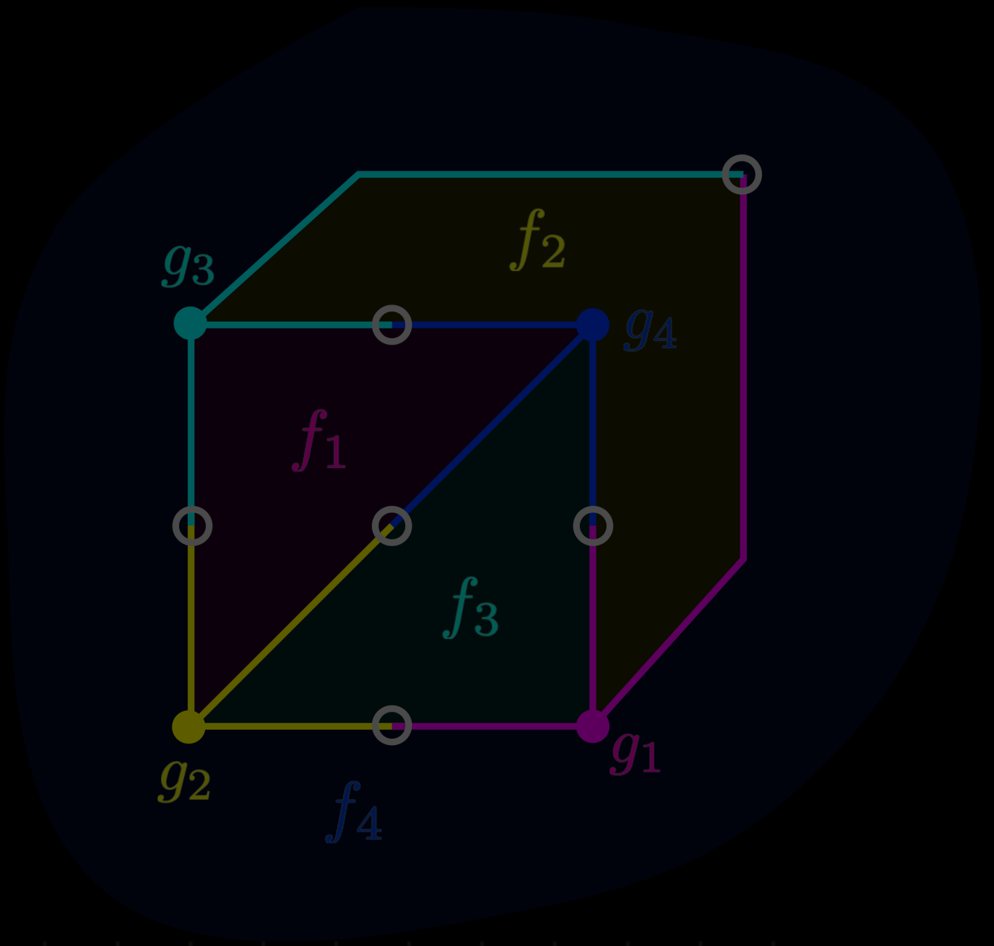
We add at most 4 unit squares to D in each step: $|D| \leq 4\ell$

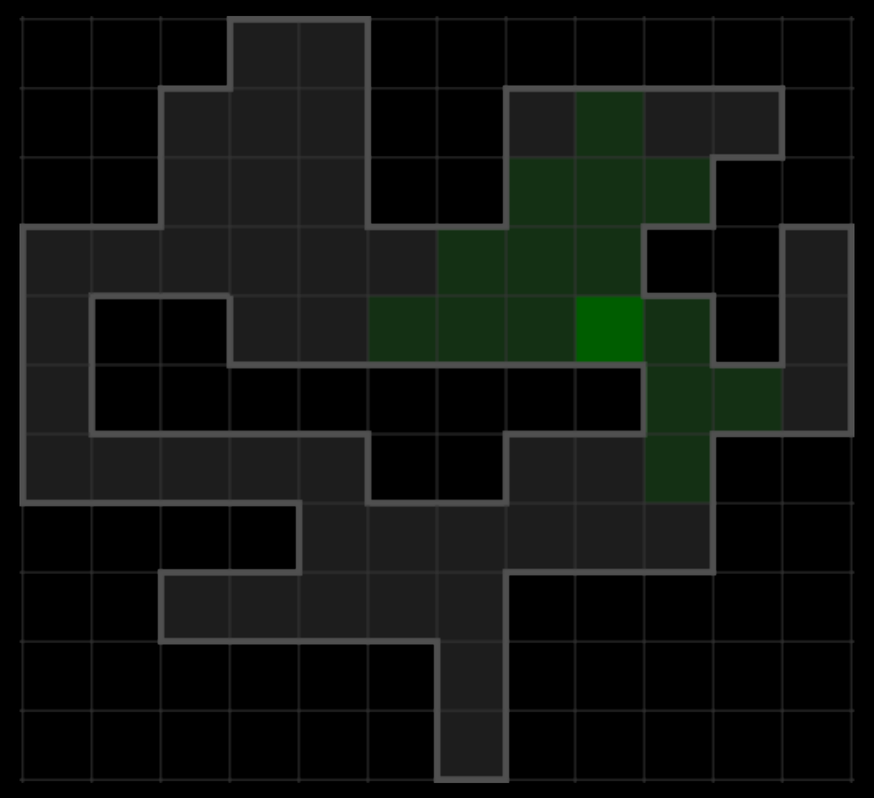
Outlook

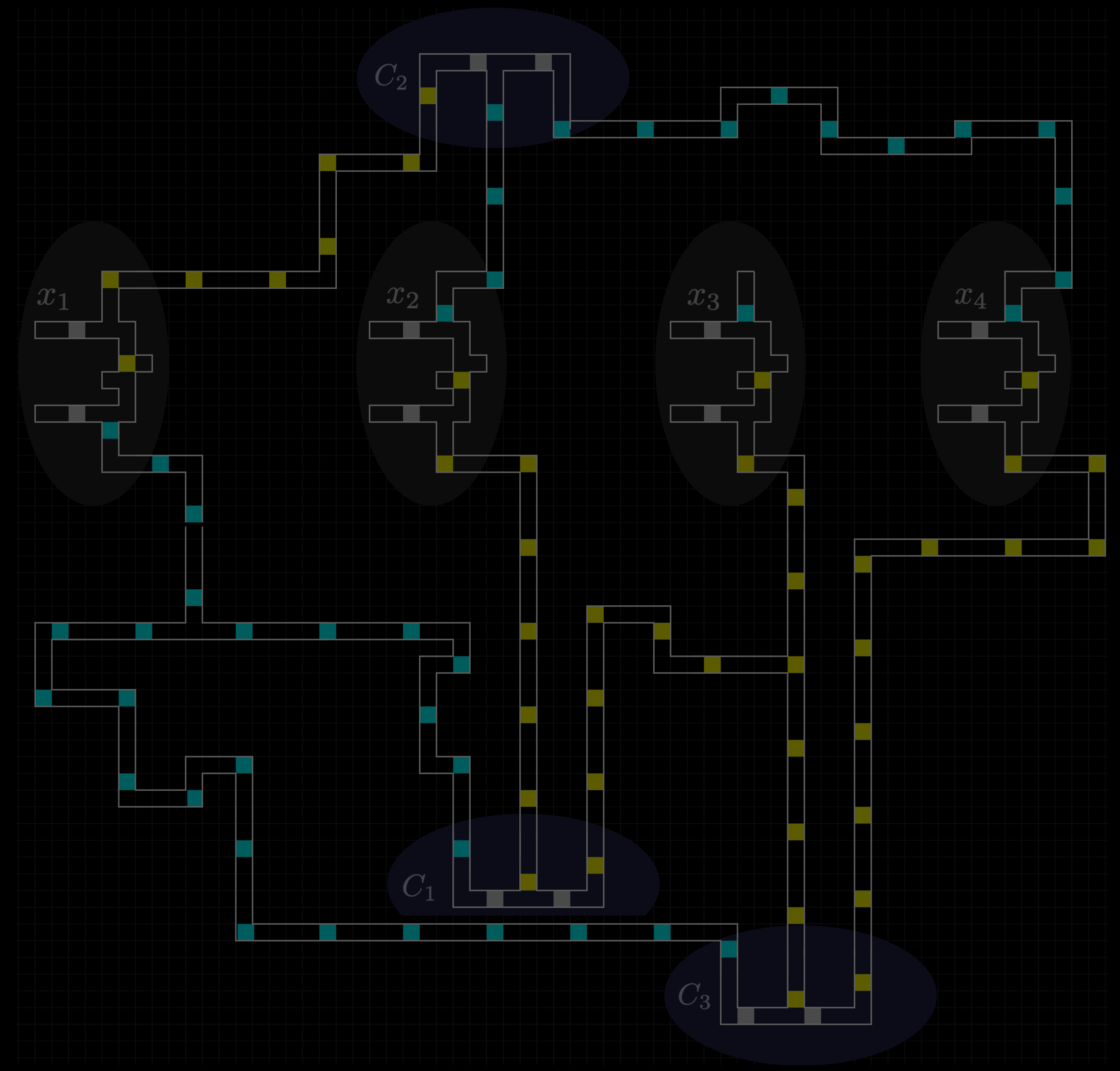
Outlook

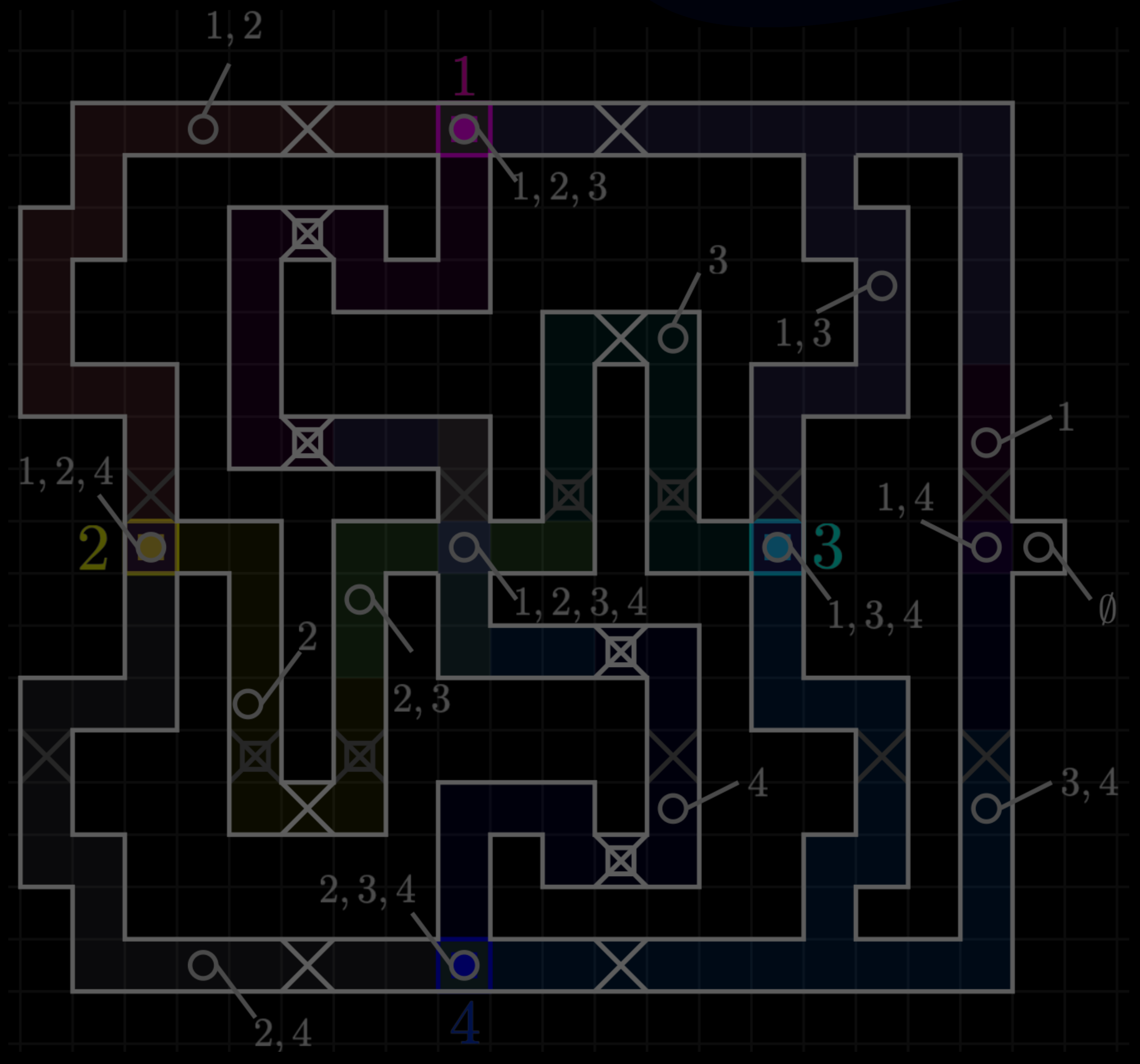
- Complexity in arbitrary simple polyominoes?
- More general approximation framework for arbitrary *k* in arbitrary simple polyominoes? (Or polytime algorithm??)
- Constant-factor approximation for polyominoes with holes?
- CS: city center requires higher density of carsharing stations, lower density in outskirts—what if we have various classes of guards with different values of *k*?

OUR PAPER

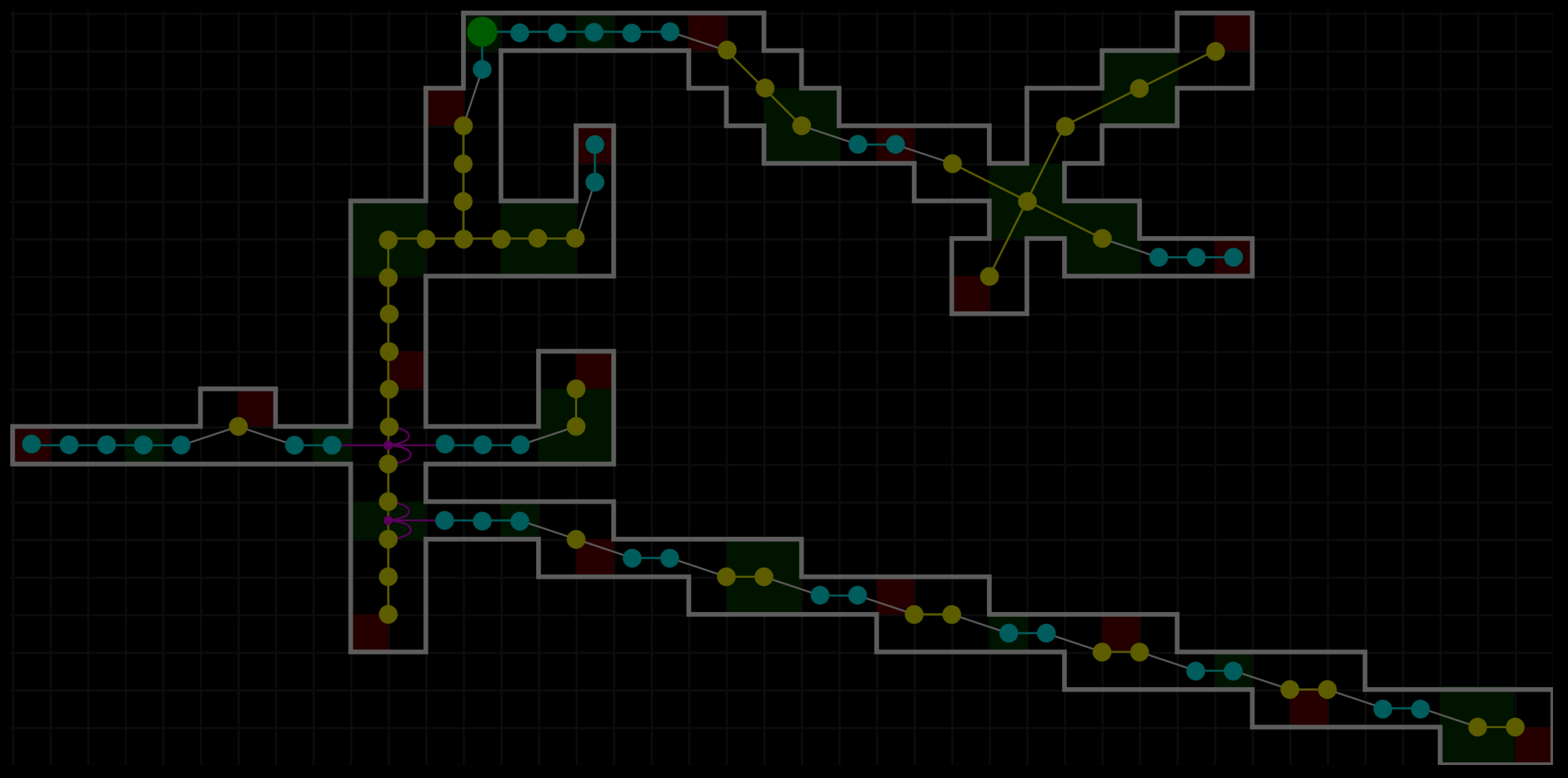








Thank you.



SCAN ME
christiane.schmidt@liu.se
https://www.itn.liu.se/~chrsc91/