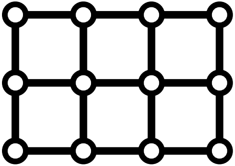
Detecting a Machine Failure in a Network, a.k.a. Vertex Identifying Codes

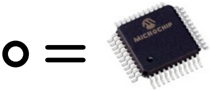
Daniel W. Cranston

Virginia Commonwealth University
dcranston@vcu.edu
Joint with Gexin Yu

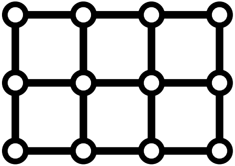
Applications of Graph Theory
Joint Math Meetings, San Francisco
13 January 2010

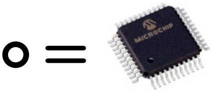
Definitions and Motivation





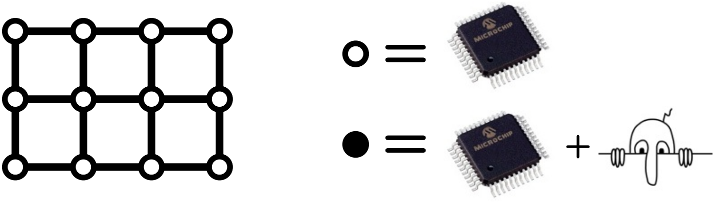
Definitions and Motivation





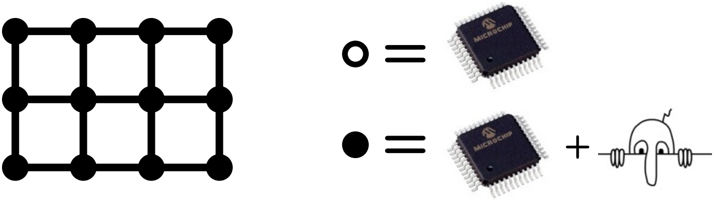
Goal: put sensors in the network to detect which machine failed

Definitions and Motivation



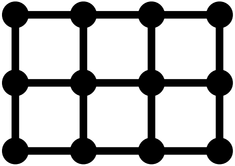
Goal: put sensors in the network to detect which machine failed

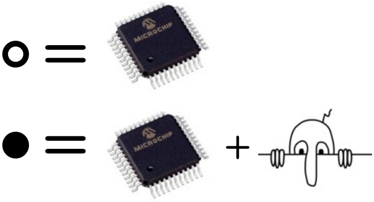
Definitions and Motivation



Goal: put sensors in the network to detect which machine failed

Definitions and Motivation

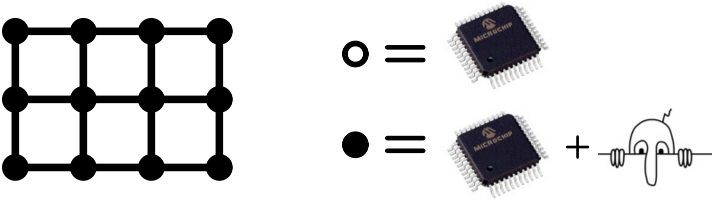




Bad Solution: too much $$$ and bandwidth

Goal: put sensors in the network to detect which machine failed

Definitions and Motivation

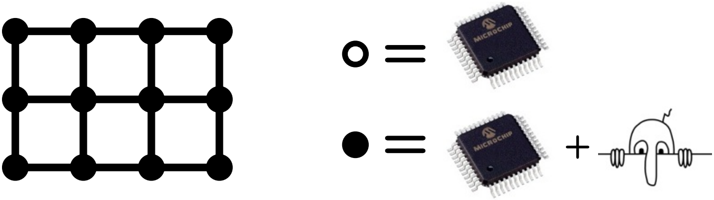


Bad Solution: too much $$$ and bandwidth

Goal: put sensors in the network to detect which machine failed

Assumptions: - machines fail one at a time

Definitions and Motivation



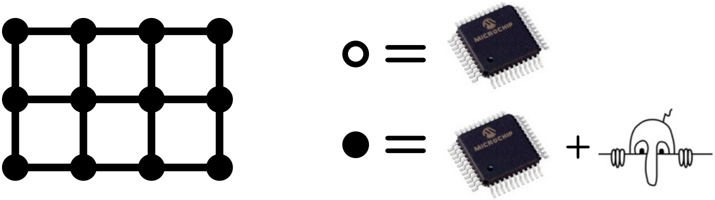
Bad Solution: too much $$$ and bandwidth

Goal: put sensors in the network to detect which machine failed

Assumptions: - machines fail one at a time

- each sensor only sends one bit

Definitions and Motivation



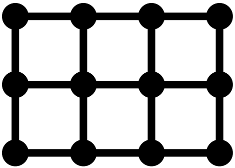
Bad Solution: too much $$$ and bandwidth

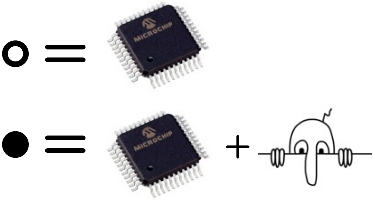
Goal: put sensors in the network to detect which machine failed

Assumptions: - machines fail one at a time

- each sensor only sends one bit
- a sensor at v can see v and its neighbors

Definitions and Motivation





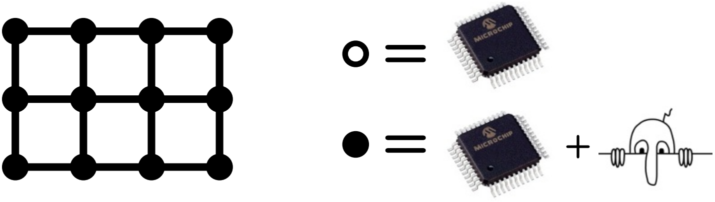
Bad Solution: too much $$$ and bandwidth

Goal: put sensors in the network to detect which machine failed

Assumptions: - machines fail one at a time
- each sensor only sends one bit
- a sensor at v can see v and its neighbors

Find a subset $\mathcal{C} \subset V(G)$ s.t. for all $v \in V(G)$ $N[v] \cap \mathcal{C} \neq \emptyset$ and $\forall u, v \in V(G)$ if $u \neq v$ then $N[u] \cap \mathcal{C} \neq N[v] \cap \mathcal{C}$.

Definitions and Motivation



Bad Solution: too much $$$ and bandwidth

Goal: put sensors in the network to detect which machine failed

Assumptions:
- machines fail one at a time
- each sensor only sends one bit
- a sensor at v can see v and its neighbors

Find a subset $\mathcal{C} \subset V(G)$ s.t. for all $v \in V(G)$ $N[v] \cap \mathcal{C} \neq \emptyset$ and $\forall u, v \in V(G)$ if $u \neq v$ then $N[u] \cap \mathcal{C} \neq N[v] \cap \mathcal{C}$.

Definition: We call such a set $\mathcal{C}$ a (vertex identifying) code.

Codes: Examples and Non-examples

Codes: Examples and Non-examples

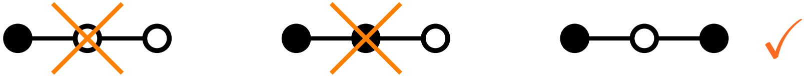
Codes: Examples and Non-examples

Codes: Examples and Non-examples

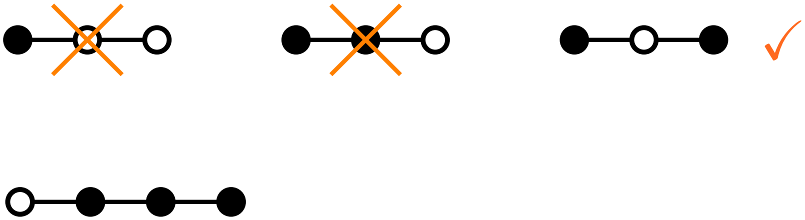
Codes: Examples and Non-examples



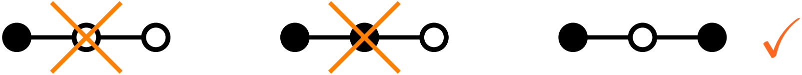
Codes: Examples and Non-examples

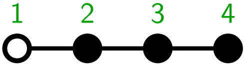


Codes: Examples and Non-examples

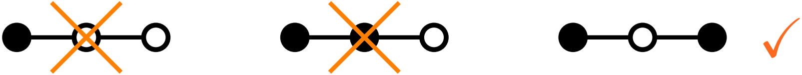


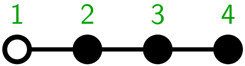
Codes: Examples and Non-examples





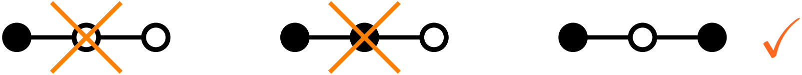
Codes: Examples and Non-examples

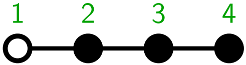




$1: \{2\}$

Codes: Examples and Non-examples

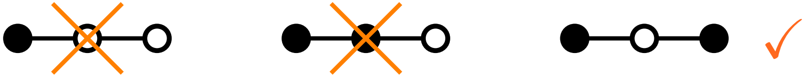


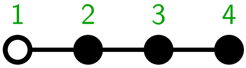


$1: \{2\}$

$2: \{2, 3\}$

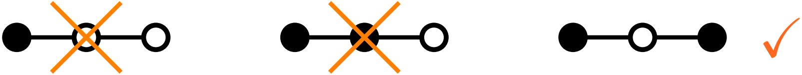
Codes: Examples and Non-examples

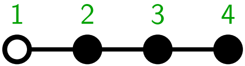




$1: \{2\}$
$2: \{2,3\}$
$3: \{2,3,4\}$

Codes: Examples and Non-examples





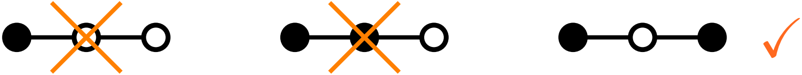
$1: \{2\}$

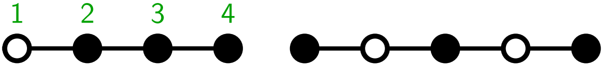
$2: \{2, 3\}$

$3: \{2, 3, 4\}$

$4: \{3, 4\}$

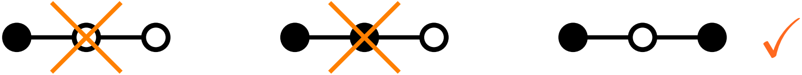
Codes: Examples and Non-examples

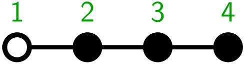


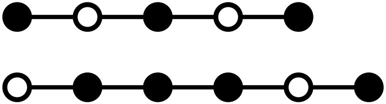


1 : $\{2\}$
2 : $\{2, 3\}$
3 : $\{2, 3, 4\}$
4 : $\{3, 4\}$

Codes: Examples and Non-examples

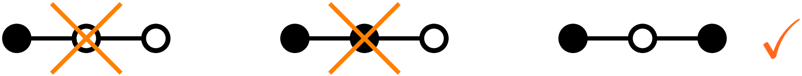


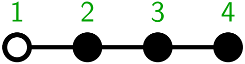




1 : $\{2\}$
2 : $\{2, 3\}$
3 : $\{2, 3, 4\}$
4 : $\{3, 4\}$

Codes: Examples and Non-examples



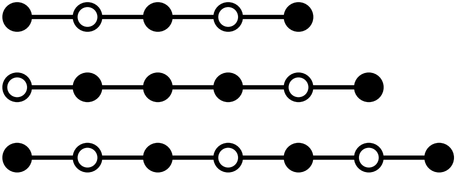


1 : $\{2\}$

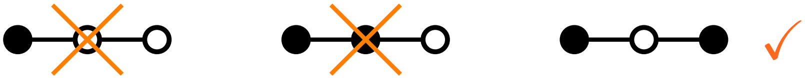
2 : $\{2, 3\}$

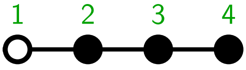
3 : $\{2, 3, 4\}$

4 : $\{3, 4\}$

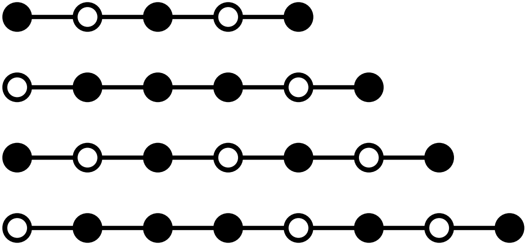


Codes: Examples and Non-examples

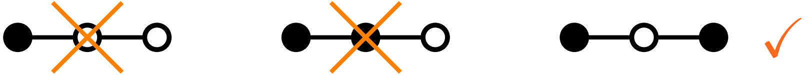


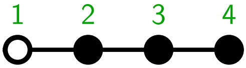


1 : $\{2\}$
2 : $\{2, 3\}$
3 : $\{2, 3, 4\}$
4 : $\{3, 4\}$

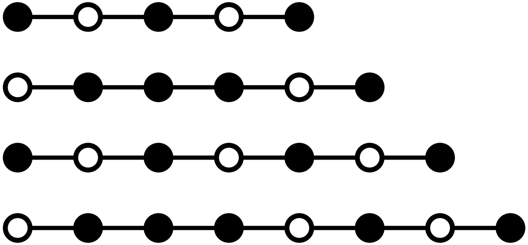


Codes: Examples and Non-examples



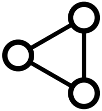


$1: \{2\}$
$2: \{2,3\}$
$3: \{2,3,4\}$
$4: \{3,4\}$

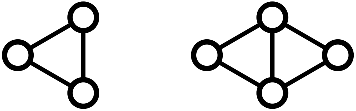


Observation: Every path has a code.

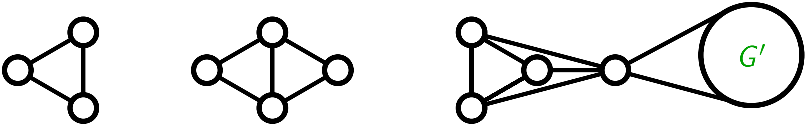
Finding the Right Problem



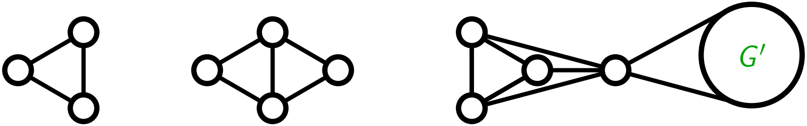
Finding the Right Problem



Finding the Right Problem

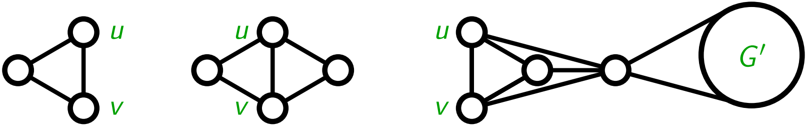


Finding the Right Problem



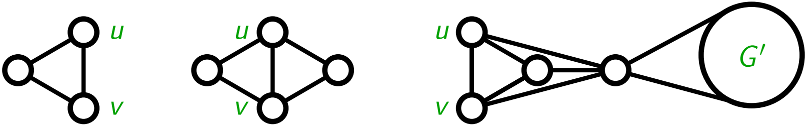
Difficulty:

Finding the Right Problem



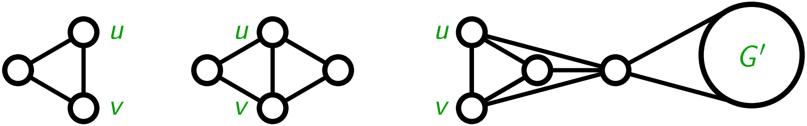
Difficulty: $N[u] = N[v]$, so for any $\mathcal{C}$ we get $N[u] \cap \mathcal{C} = N[v] \cap \mathcal{C}$.

Finding the Right Problem



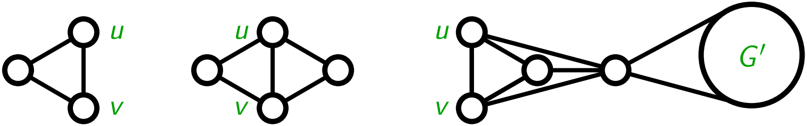
Difficulty: $N[u] = N[v]$, so for any $\mathcal{C}$ we get $N[u] \cap \mathcal{C} = N[v] \cap \mathcal{C}$.
Observation: G has a code iff for all $u \neq v$ we have $N[u] \neq N[v]$.

Finding the Right Problem



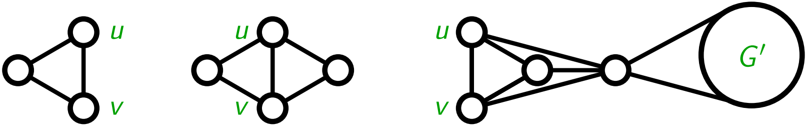
Difficulty: $N[u] = N[v]$, so for any $\mathcal{C}$ we get $N[u] \cap \mathcal{C} = N[v] \cap \mathcal{C}$.
Observation: G has a code iff for all $u \neq v$ we have $N[u] \neq N[v]$.
Definition: We call such a graph twin-free.

Finding the Right Problem

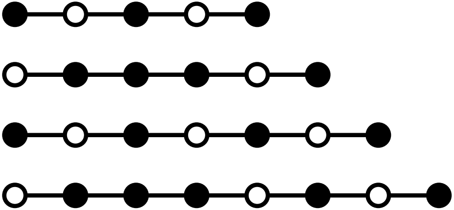


Difficulty: $N[u] = N[v]$, so for any $\mathcal{C}$ we get $N[u] \cap \mathcal{C} = N[v] \cap \mathcal{C}$.
Observation: G has a code iff for all $u \neq v$ we have $N[u] \neq N[v]$.
Definition: We call such a graph twin-free.
New problem: If G is twin-free, find a smallest code.

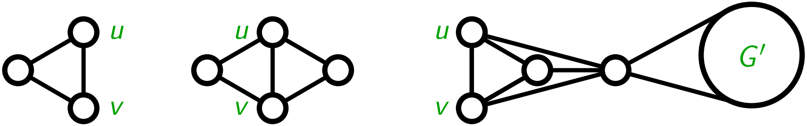
Finding the Right Problem



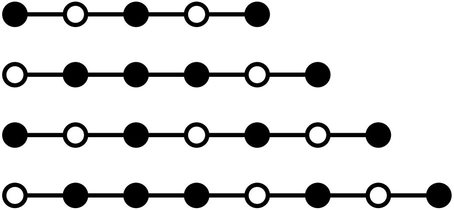
Difficulty: $N[u] = N[v]$, so for any $\mathcal{C}$ we get $N[u] \cap \mathcal{C} = N[v] \cap \mathcal{C}$.
Observation: G has a code iff for all $u \neq v$ we have $N[u] \neq N[v]$.
Definition: We call such a graph twin-free.
New problem: If G is twin-free, find a smallest code.



Finding the Right Problem



Difficulty: $N[u] = N[v]$, so for any $\mathcal{C}$ we get $N[u] \cap \mathcal{C} = N[v] \cap \mathcal{C}$.
Observation: G has a code iff for all $u \neq v$ we have $N[u] \neq N[v]$.
Definition: We call such a graph twin-free.
New problem: If G is twin-free, find a smallest code.



Exer. Show that min size of code for path on k nodes is $\lceil \frac{k+1}{2} \rceil$.

Infinite Graphs

Infinite Graphs

We need the following properties:

Infinite Graphs

We need the following properties:

- twin-free

Infinite Graphs

We need the following properties:

- twin-free
- locally finite (every vertex has finite degree)

Infinite Graphs

We need the following properties:

- twin-free
- locally finite (every vertex has finite degree)
- vertex transitive (graph looks the same from every vertex)

Infinite Graphs

We need the following properties:

- twin-free
- locally finite (every vertex has finite degree)
- vertex transitive (graph looks the same from every vertex)

Ex. $V(G_{\mathbb{Z}}) = \mathbb{Z}$ and $u \leftrightarrow v$ if $|u - v| = 1$

Infinite Graphs

We need the following properties:

- twin-free
- locally finite (every vertex has finite degree)
- vertex transitive (graph looks the same from every vertex)

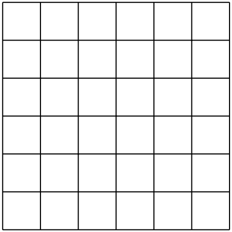
Ex. $V(G_{\mathbb{Z}}) = \mathbb{Z}$ and $u \leftrightarrow v$ if $|u - v| = 1$ (infinite path)

Infinite Graphs

We need the following properties:

- twin-free
- locally finite (every vertex has finite degree)
- vertex transitive (graph looks the same from every vertex)

Ex. $V(G_{\mathbb{Z}}) = \mathbb{Z}$ and $u \leftrightarrow v$ if $|u - v| = 1$ (infinite path)

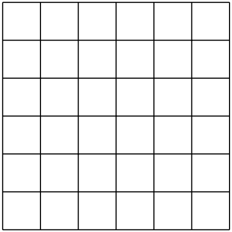


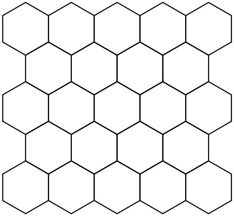
Infinite Graphs

We need the following properties:

- twin-free
- locally finite (every vertex has finite degree)
- vertex transitive (graph looks the same from every vertex)

Ex. $V(G_{\mathbb{Z}}) = \mathbb{Z}$ and $u \leftrightarrow v$ if $|u - v| = 1$ (infinite path)



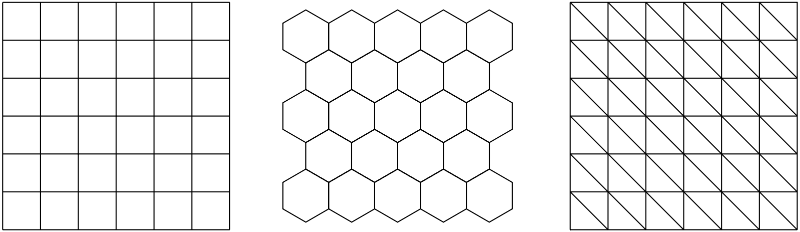


Infinite Graphs

We need the following properties:

- twin-free
- locally finite (every vertex has finite degree)
- vertex transitive (graph looks the same from every vertex)

Ex. $V(G_{\mathbb{Z}}) = \mathbb{Z}$ and $u \leftrightarrow v$ if $|u - v| = 1$ (infinite path)

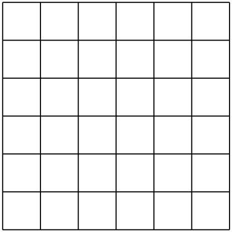


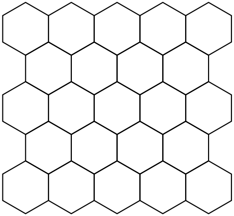
Infinite Graphs

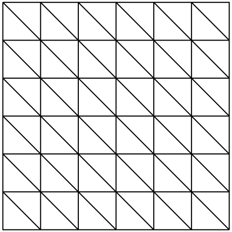
We need the following properties:

- twin-free
- locally finite (every vertex has finite degree)
- vertex transitive (graph looks the same from every vertex)

Ex. $V(G_{\mathbb{Z}}) = \mathbb{Z}$ and $u \leftrightarrow v$ if $|u - v| = 1$ (infinite path)







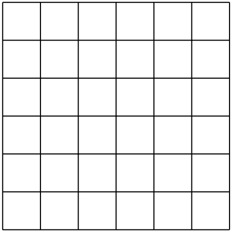
Definition: Rather than the smallest size code, we want the lowest density (fraction) code.

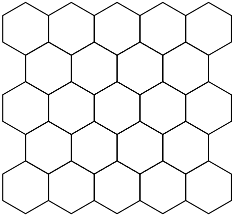
Infinite Graphs

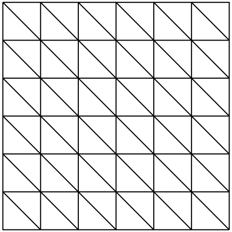
We need the following properties:

- twin-free
- locally finite (every vertex has finite degree)
- vertex transitive (graph looks the same from every vertex)

Ex. $V(G_{\mathbb{Z}}) = \mathbb{Z}$ and $u \leftrightarrow v$ if $|u - v| = 1$ (infinite path)







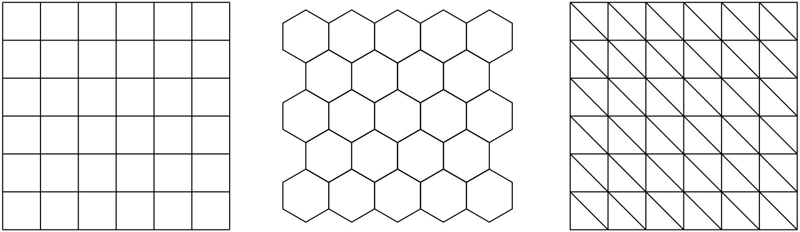
Definition: Rather than the smallest size code, we want the lowest density (fraction) code. We call this the density of G, $\tau(G)$.

Infinite Graphs

We need the following properties:

- twin-free
- locally finite (every vertex has finite degree)
- vertex transitive (graph looks the same from every vertex)

Ex. $V(G_{\mathbb{Z}}) = \mathbb{Z}$ and $u \leftrightarrow v$ if $|u - v| = 1$ (infinite path)



Definition: Rather than the smallest size code, we want the lowest density (fraction) code. We call this the density of G, $\tau(G)$.

Question: What is $\tau(G_{\mathbb{Z}})$?

Proving a Lower Bound (sketch)

Proving a Lower Bound (sketch)

Forget infinite for now.

Proving a Lower Bound (sketch)

Forget infinite for now.

Suppose $\mathcal{C}$ is a code for G.

Proving a Lower Bound (sketch)

Forget infinite for now.

Suppose $\mathcal{C}$ is a code for G. Put a cake at each $v \in \mathcal{C}$ and redistribute so each $u \in V(G)$ gets at least k cake. ($0 < k < 1$)

Proving a Lower Bound (sketch)

Forget infinite for now.

Suppose $\mathcal{C}$ is a code for G. Put a cake at each $v \in \mathcal{C}$ and redistribute so each $u \in V(G)$ gets at least k cake. ($0 < k < 1$)

Thus $k|V(G)| \leq |\mathcal{C}|$.

Proving a Lower Bound (sketch)

Forget infinite for now.

Suppose $\mathcal{C}$ is a code for G. Put a cake at each $v \in \mathcal{C}$ and redistribute so each $u \in V(G)$ gets at least k cake. ($0 < k < 1$)

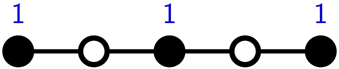
Thus $k|V(G)| \leq |\mathcal{C}|$. Hence, $k \leq \frac{|\mathcal{C}|}{|V(G)|} = \tau(G)$.

Proving a Lower Bound (sketch)

Forget infinite for now.

Suppose $\mathcal{C}$ is a code for G. Put a cake at each $v \in \mathcal{C}$ and redistribute so each $u \in V(G)$ gets at least k cake. $(0 < k < 1)$

Thus $k|V(G)| \leq |\mathcal{C}|$. Hence, $k \leq \frac{|\mathcal{C}|}{|V(G)|} = \tau(G)$.

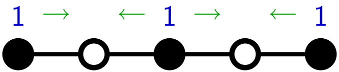


Proving a Lower Bound (sketch)

Forget infinite for now.

Suppose $\mathcal{C}$ is a code for G. Put a cake at each $v \in \mathcal{C}$ and redistribute so each $u \in V(G)$ gets at least k cake. ($0 < k < 1$)

Thus $k|V(G)| \leq |\mathcal{C}|$. Hence, $k \leq \frac{|\mathcal{C}|}{|V(G)|} = \tau(G)$.

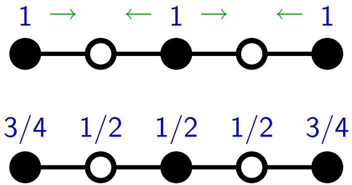


Proving a Lower Bound (sketch)

Forget infinite for now.

Suppose $\mathcal{C}$ is a code for G. Put a cake at each $v \in \mathcal{C}$ and redistribute so each $u \in V(G)$ gets at least k cake. ($0 < k < 1$)

Thus $k|V(G)| \leq |\mathcal{C}|$. Hence, $k \leq \frac{|\mathcal{C}|}{|V(G)|} = \tau(G)$.

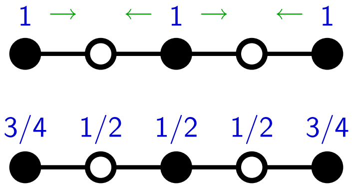


Proving a Lower Bound (sketch)

Forget infinite for now.

Suppose $\mathcal{C}$ is a code for G. Put a cake at each $v \in \mathcal{C}$ and redistribute so each $u \in V(G)$ gets at least k cake. ($0 < k < 1$)

Thus $k|V(G)| \leq |\mathcal{C}|$. Hence, $k \leq \frac{|\mathcal{C}|}{|V(G)|} = \tau(G)$.



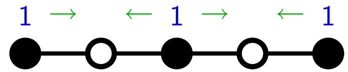
$\frac{1}{2} \leq \tau(P_5) = \frac{3}{5}$

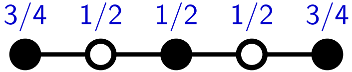
Proving a Lower Bound (sketch)

Forget infinite for now.

Suppose $\mathcal{C}$ is a code for G. Put a cake at each $v \in \mathcal{C}$ and redistribute so each $u \in V(G)$ gets at least k cake. ($0 < k < 1$)

Thus $k|V(G)| \leq |\mathcal{C}|$. Hence, $k \leq \frac{|\mathcal{C}|}{|V(G)|} = \tau(G)$.





$\frac{1}{2} \leq \tau(P_5) = \frac{3}{5}$

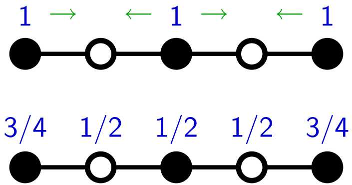
Observation: The same idea works for infinite graphs.

Proving a Lower Bound (sketch)

Forget infinite for now.

Suppose $\mathcal{C}$ is a code for G. Put a cake at each $v \in \mathcal{C}$ and redistribute so each $u \in V(G)$ gets at least k cake. ($0 < k < 1$)

Thus $k|V(G)| \leq |\mathcal{C}|$. Hence, $k \leq \frac{|\mathcal{C}|}{|V(G)|} = \tau(G)$.



$\frac{1}{2} \leq \tau(P_5) = \frac{3}{5}$

Observation: The same idea works for infinite graphs.

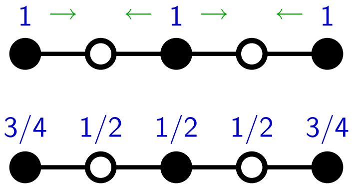
Question: How should we share the cake?

Proving a Lower Bound (sketch)

Forget infinite for now.

Suppose $\mathcal{C}$ is a code for G. Put a cake at each $v \in \mathcal{C}$ and redistribute so each $u \in V(G)$ gets at least k cake. ($0 < k < 1$)

Thus $k|V(G)| \leq |\mathcal{C}|$. Hence, $k \leq \frac{|\mathcal{C}|}{|V(G)|} = \tau(G)$.



$\frac{1}{2} \leq \tau(P_5) = \frac{3}{5}$

Observation: The same idea works for infinite graphs.

Question: How should we share the cake?

Question: How much cake can each vertex have?

Proving a Lower Bound (real proof)

Proving a Lower Bound (real proof)

Theorem: For the hex grid, $\tau \geq \frac{2}{5}$.

Proof. Each vertex in $\mathcal{C}$ gives $\frac{2}{5k}$ cake to each neighbor not in $\mathcal{C}$ that has k neighbors in $\mathcal{C}$.

Proving a Lower Bound (real proof)

Theorem: For the hex grid, $\tau \geq \frac{2}{5}$.

Proof. Each vertex in $\mathcal{C}$ gives $\frac{2}{5k}$ cake to each neighbor not in $\mathcal{C}$ that has k neighbors in $\mathcal{C}$.

We must show that each vertex v gets at least $\frac{2}{5}$ of a cake.

Proving a Lower Bound (real proof)

Theorem: For the hex grid, $\tau \geq \frac{2}{5}$.

Proof. Each vertex in $\mathcal{C}$ gives $\frac{2}{5k}$ cake to each neighbor not in $\mathcal{C}$ that has k neighbors in $\mathcal{C}$.

We must show that each vertex v gets at least $\frac{2}{5}$ of a cake.
We consider cases, based on what size cluster contains v.

Proving a Lower Bound (real proof)

Theorem: For the hex grid, $\tau \geq \frac{2}{5}$.

Proof. Each vertex in $\mathcal{C}$ gives $\frac{2}{5k}$ cake to each neighbor not in $\mathcal{C}$ that has k neighbors in $\mathcal{C}$.

We must show that each vertex v gets at least $\frac{2}{5}$ of a cake. We consider cases, based on what size cluster contains v.

- $v \notin \mathcal{C}$:

Proving a Lower Bound (real proof)

Theorem: For the hex grid, $\tau \geq \frac{2}{5}$.

Proof. Each vertex in $\mathcal{C}$ gives $\frac{2}{5k}$ cake to each neighbor not in $\mathcal{C}$ that has k neighbors in $\mathcal{C}$.

We must show that each vertex v gets at least $\frac{2}{5}$ of a cake.
We consider cases, based on what size cluster contains v.

- $v \notin \mathcal{C}$: $k(\frac{2}{5k}) = \frac{2}{5}$

Proving a Lower Bound (real proof)

Theorem: For the hex grid, $\tau \geq \frac{2}{5}$.

Proof. Each vertex in $\mathcal{C}$ gives $\frac{2}{5k}$ cake to each neighbor not in $\mathcal{C}$ that has k neighbors in $\mathcal{C}$.

We must show that each vertex v gets at least $\frac{2}{5}$ of a cake. We consider cases, based on what size cluster contains v.

- $v \notin \mathcal{C}$: $k(\frac{2}{5k}) = \frac{2}{5}$
- v in a 1-cluster:

Proving a Lower Bound (real proof)

Theorem: For the hex grid, $\tau \geq \frac{2}{5}$.

Proof. Each vertex in $\mathcal{C}$ gives $\frac{2}{5k}$ cake to each neighbor not in $\mathcal{C}$ that has k neighbors in $\mathcal{C}$.

We must show that each vertex v gets at least $\frac{2}{5}$ of a cake. We consider cases, based on what size cluster contains v.

- $v \notin \mathcal{C}$: $k(\frac{2}{5k}) = \frac{2}{5}$
- v in a 1-cluster: $1 - 3(\frac{2}{5(2)}) = \frac{2}{5}$.

Proving a Lower Bound (real proof)

Theorem: For the hex grid, $\tau \geq \frac{2}{5}$.

Proof. Each vertex in $\mathcal{C}$ gives $\frac{2}{5k}$ cake to each neighbor not in $\mathcal{C}$ that has k neighbors in $\mathcal{C}$.

We must show that each vertex v gets at least $\frac{2}{5}$ of a cake.
We consider cases, based on what size cluster contains v.

- $v \notin \mathcal{C}$: $k(\frac{2}{5k}) = \frac{2}{5}$
- v in a 1-cluster: $1 - 3(\frac{2}{5(2)}) = \frac{2}{5}$.
- v in a 3^+-cluster:

Proving a Lower Bound (real proof)

Theorem: For the hex grid, $\tau \geq \frac{2}{5}$.

Proof. Each vertex in $\mathcal{C}$ gives $\frac{2}{5k}$ cake to each neighbor not in $\mathcal{C}$ that has k neighbors in $\mathcal{C}$.

We must show that each vertex v gets at least $\frac{2}{5}$ of a cake.
We consider cases, based on what size cluster contains v.

- $v \notin \mathcal{C}$: $k(\frac{2}{5k}) = \frac{2}{5}$
- v in a 1-cluster: $1 - 3(\frac{2}{5(2)}) = \frac{2}{5}$.
- v in a 3^+-cluster:
  - v has 3 neighbors in cluster:

Proving a Lower Bound (real proof)

Theorem: For the hex grid, $\tau \geq \frac{2}{5}$.

Proof. Each vertex in $\mathcal{C}$ gives $\frac{2}{5k}$ cake to each neighbor not in $\mathcal{C}$ that has k neighbors in $\mathcal{C}$.

We must show that each vertex v gets at least $\frac{2}{5}$ of a cake.
We consider cases, based on what size cluster contains v.

- $v \notin \mathcal{C}$: $k(\frac{2}{5k}) = \frac{2}{5}$
- v in a 1-cluster: $1 - 3(\frac{2}{5(2)}) = \frac{2}{5}$.
- v in a 3^+-cluster:
  - v has 3 neighbors in cluster: $1 - 0 = 1$

Proving a Lower Bound (real proof)

Theorem: For the hex grid, $\tau \geq \frac{2}{5}$.

Proof. Each vertex in $\mathcal{C}$ gives $\frac{2}{5k}$ cake to each neighbor not in $\mathcal{C}$ that has k neighbors in $\mathcal{C}$.

We must show that each vertex v gets at least $\frac{2}{5}$ of a cake. We consider cases, based on what size cluster contains v.

- $v \notin \mathcal{C}$: $k(\frac{2}{5k}) = \frac{2}{5}$
- v in a 1-cluster: $1 - 3(\frac{2}{5(2)}) = \frac{2}{5}$.
- v in a 3^+-cluster:
  - v has 3 neighbors in cluster: $1 - 0 = 1$
  - v has 2 neighbors in cluster:

Proving a Lower Bound (real proof)

Theorem: For the hex grid, $\tau \geq \frac{2}{5}$.

Proof. Each vertex in $\mathcal{C}$ gives $\frac{2}{5k}$ cake to each neighbor not in $\mathcal{C}$ that has k neighbors in $\mathcal{C}$.

We must show that each vertex v gets at least $\frac{2}{5}$ of a cake. We consider cases, based on what size cluster contains v.

- $v \notin \mathcal{C}$: $k(\frac{2}{5k}) = \frac{2}{5}$
- v in a 1-cluster: $1 - 3(\frac{2}{5(2)}) = \frac{2}{5}$.
- v in a 3^+-cluster:
  - v has 3 neighbors in cluster: $1 - 0 = 1$
  - v has 2 neighbors in cluster: $1 - 1(\frac{2}{5}) = \frac{3}{5}$

Proving a Lower Bound (real proof)

Theorem: For the hex grid, $\tau \geq \frac{2}{5}$.

Proof. Each vertex in $\mathcal{C}$ gives $\frac{2}{5k}$ cake to each neighbor not in $\mathcal{C}$ that has k neighbors in $\mathcal{C}$.

We must show that each vertex v gets at least $\frac{2}{5}$ of a cake. We consider cases, based on what size cluster contains v.

- $v \notin \mathcal{C}$: $k(\frac{2}{5k}) = \frac{2}{5}$
- v in a 1-cluster: $1 - 3(\frac{2}{5(2)}) = \frac{2}{5}$.
- v in a 3^+-cluster:
  - v has 3 neighbors in cluster: $1 - 0 = 1$
  - v has 2 neighbors in cluster: $1 - 1(\frac{2}{5}) = \frac{3}{5}$
  - v has 1 neighbor in cluster:

Proving a Lower Bound (real proof)

Theorem: For the hex grid, $\tau \geq \frac{2}{5}$.

Proof. Each vertex in $\mathcal{C}$ gives $\frac{2}{5k}$ cake to each neighbor not in $\mathcal{C}$ that has k neighbors in $\mathcal{C}$.

We must show that each vertex v gets at least $\frac{2}{5}$ of a cake. We consider cases, based on what size cluster contains v.

- $v \notin \mathcal{C}$: $k(\frac{2}{5k}) = \frac{2}{5}$
- v in a 1-cluster: $1 - 3(\frac{2}{5(2)}) = \frac{2}{5}$.
- v in a 3^+-cluster:
  - v has 3 neighbors in cluster: $1 - 0 = 1$
  - v has 2 neighbors in cluster: $1 - 1(\frac{2}{5}) = \frac{3}{5}$
  - v has 1 neighbor in cluster: $1 - 2(\frac{2}{5}) = \frac{1}{5}$

Proving a Lower Bound (real proof)

Theorem: For the hex grid, $\tau \geq \frac{2}{5}$.

Proof. Each vertex in $\mathcal{C}$ gives $\frac{2}{5k}$ cake to each neighbor not in $\mathcal{C}$ that has k neighbors in $\mathcal{C}$.

We must show that each vertex v gets at least $\frac{2}{5}$ of a cake.
We consider cases, based on what size cluster contains v.

- $v \notin \mathcal{C}$: $k(\frac{2}{5k}) = \frac{2}{5}$
- v in a 1-cluster: $1 - 3(\frac{2}{5(2)}) = \frac{2}{5}$.
- v in a 3^+-cluster:
  - v has 3 neighbors in cluster: $1 - 0 = 1$
  - v has 2 neighbors in cluster: $1 - 1(\frac{2}{5}) = \frac{3}{5}$
  - v has 1 neighbor in cluster: ~~$1 - 2(\frac{2}{5}) = \frac{1}{5}$~~
    $1 - \frac{2}{5} - \frac{1}{5} = \frac{2}{5}$ ■

Proving a Lower Bound (real proof)

Theorem: For the hex grid, $\tau \geq \frac{2}{5}$.

Proof. Each vertex in $\mathcal{C}$ gives $\frac{2}{5k}$ cake to each neighbor not in $\mathcal{C}$ that has k neighbors in $\mathcal{C}$.

We must show that each vertex v gets at least $\frac{2}{5}$ of a cake. We consider cases, based on what size cluster contains v.

- $v \notin \mathcal{C}$: $k(\frac{2}{5k}) = \frac{2}{5}$
- v in a 1-cluster: $1 - 3(\frac{2}{5(2)}) = \frac{2}{5}$.
- v in a 3^+-cluster:
  - v has 3 neighbors in cluster: $1 - 0 = 1$
  - v has 2 neighbors in cluster: $1 - 1(\frac{2}{5}) = \frac{3}{5}$
  - v has 1 neighbor in cluster: ~~$1 - 2(\frac{2}{5}) = \frac{1}{5}$~~
    $1 - \frac{2}{5} - \frac{1}{5} = \frac{2}{5}$ ■

Question: Can we do better?

Proving a Lower Bound (real proof)

Theorem: For the hex grid, $\tau \geq \frac{2}{5}$.

Proof. Each vertex in $\mathcal{C}$ gives $\frac{2}{5k}$ cake to each neighbor not in $\mathcal{C}$ that has k neighbors in $\mathcal{C}$.

We must show that each vertex v gets at least $\frac{2}{5}$ of a cake. We consider cases, based on what size cluster contains v.

- $v \notin \mathcal{C}$: $k(\frac{2}{5k}) = \frac{2}{5}$
- v in a 1-cluster: $1 - 3(\frac{2}{5(2)}) = \frac{2}{5}$.
- v in a 3^+-cluster:
  - v has 3 neighbors in cluster: $1 - 0 = 1$
  - v has 2 neighbors in cluster: $1 - 1(\frac{2}{5}) = \frac{3}{5}$
  - v has 1 neighbor in cluster: ~~$1 - 2(\frac{2}{5}) = \frac{1}{5}$~~ $1 - \frac{2}{5} - \frac{1}{5} = \frac{2}{5}$ ∎

Question: Can we do better?

Theorem: For the hex grid, $\tau \geq \frac{12}{29}$.

Proving a Lower Bound (real proof)

Theorem: For the hex grid, $\tau \geq \frac{2}{5}$.

Proof. Each vertex in $\mathcal{C}$ gives $\frac{2}{5k}$ cake to each neighbor not in $\mathcal{C}$ that has k neighbors in $\mathcal{C}$.

We must show that each vertex v gets at least $\frac{2}{5}$ of a cake. We consider cases, based on what size cluster contains v.

- $v \notin \mathcal{C}$: $k(\frac{2}{5k}) = \frac{2}{5}$
- v in a 1-cluster: $1 - 3(\frac{2}{5(2)}) = \frac{2}{5}$.
- v in a 3^+-cluster:
  - v has 3 neighbors in cluster: $1 - 0 = 1$
  - v has 2 neighbors in cluster: $1 - 1(\frac{2}{5}) = \frac{3}{5}$
  - v has 1 neighbor in cluster: ~~$1 - 2(\frac{2}{5}) = \frac{1}{5}$~~
    $1 - \frac{2}{5} - \frac{1}{5} = \frac{2}{5}$ ∎

Question: Can we do better?

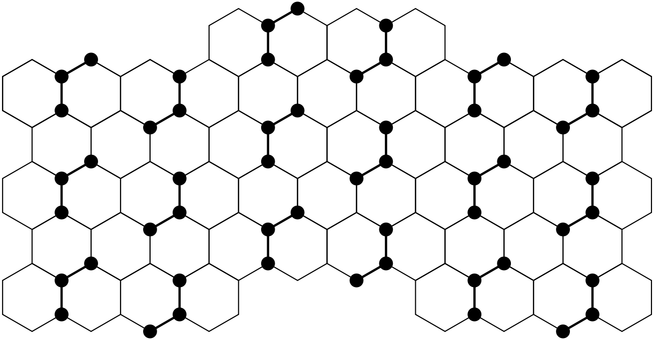
Theorem: For the hex grid, $\tau \geq \frac{12}{29}$. (vs. $\frac{2}{5} = \frac{12}{30}$ and $\frac{3}{7} = \frac{12}{28}$)

$\tau(G)$ for the Hex Grid

How to check if $\mathcal{C}$ is a code for the Hex Grid:

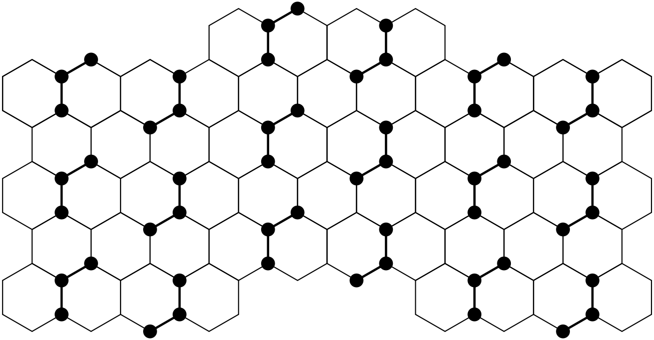
$\tau(G)$ for the Hex Grid

How to check if $\mathcal{C}$ is a code for the Hex Grid:



$\tau(G)$ for the Hex Grid

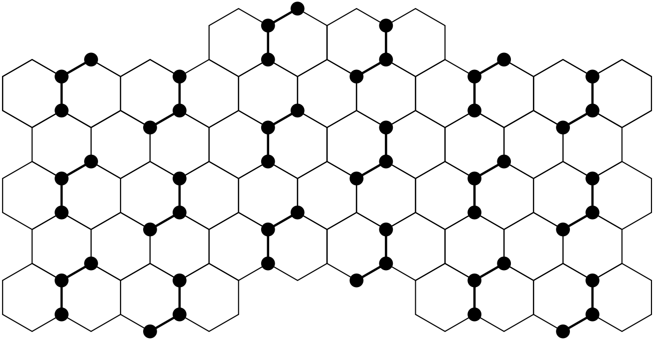
How to check if $\mathcal{C}$ is a code for the Hex Grid:



- For all $v \in V(G)$, we need $N[v] \cap \mathcal{C} \neq \emptyset$.

$\tau(G)$ for the Hex Grid

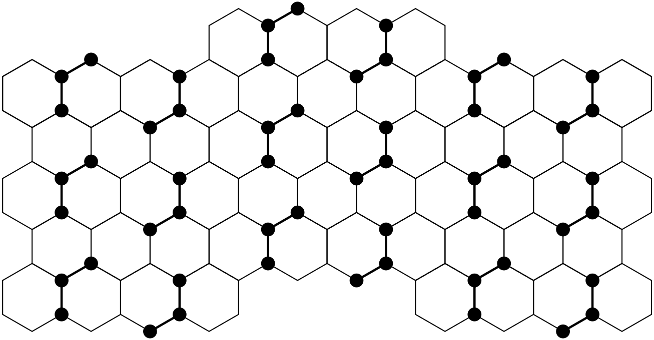
How to check if $\mathcal{C}$ is a code for the Hex Grid:



- For all $v \in V(G)$, we need $N[v] \cap \mathcal{C} \neq \emptyset$.
- For all $v \in \mathcal{C}$, there is at most one u s.t. $N[u] \cap \mathcal{C} = \{v\}$.

$\tau(G)$ for the Hex Grid

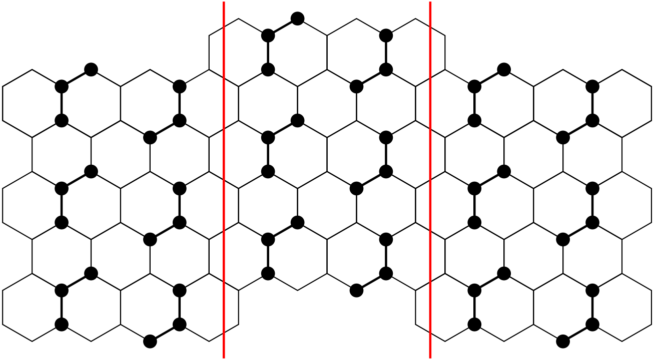
How to check if $\mathcal{C}$ is a code for the Hex Grid:



- For all $v \in V(G)$, we need $N[v] \cap \mathcal{C} \neq \emptyset$.
- For all $v \in \mathcal{C}$, there is at most one u s.t. $N[u] \cap \mathcal{C} = \{v\}$.
- $\mathcal{C}$ has no 2-clusters.

$\tau(G)$ for the Hex Grid

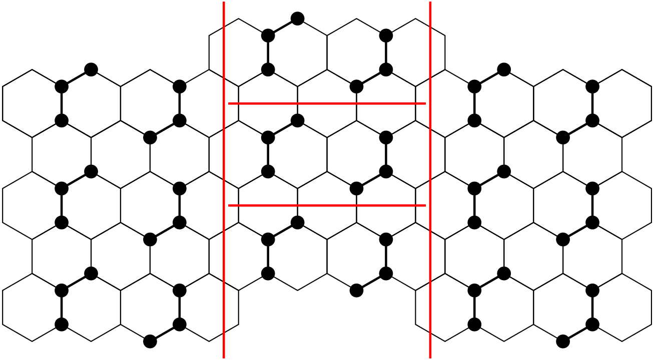
How to check if $\mathcal{C}$ is a code for the Hex Grid:



- For all $v \in V(G)$, we need $N[v] \cap \mathcal{C} \neq \emptyset$.
- For all $v \in \mathcal{C}$, there is at most one u s.t. $N[u] \cap \mathcal{C} = \{v\}$.
- $\mathcal{C}$ has no 2-clusters.

$\tau(G)$ for the Hex Grid

How to check if $\mathcal{C}$ is a code for the Hex Grid:



- For all $v \in V(G)$, we need $N[v] \cap \mathcal{C} \neq \emptyset$.
- For all $v \in \mathcal{C}$, there is at most one u s.t. $N[u] \cap \mathcal{C} = \{v\}$.
- $\mathcal{C}$ has no 2-clusters.