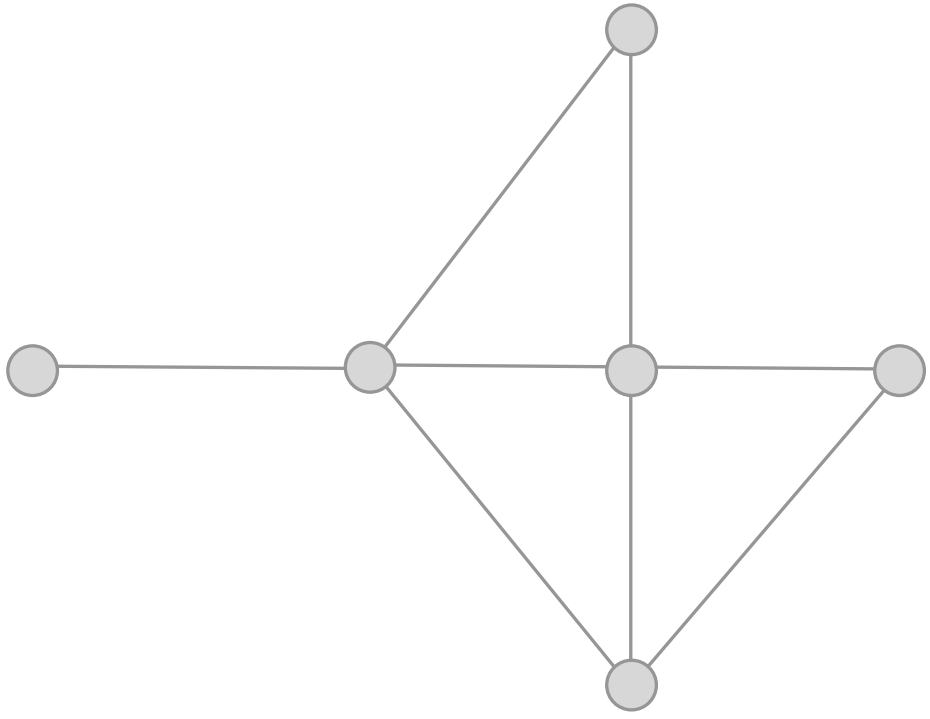
Parameterized Algorithms for the Max q-Colorable Induced Subgraph problem on Perfect Graphs

Neeldhara Misra, Fahad Panolan, Ashutosh Rai,
Venkatesh Raman, and Saket Saurabh

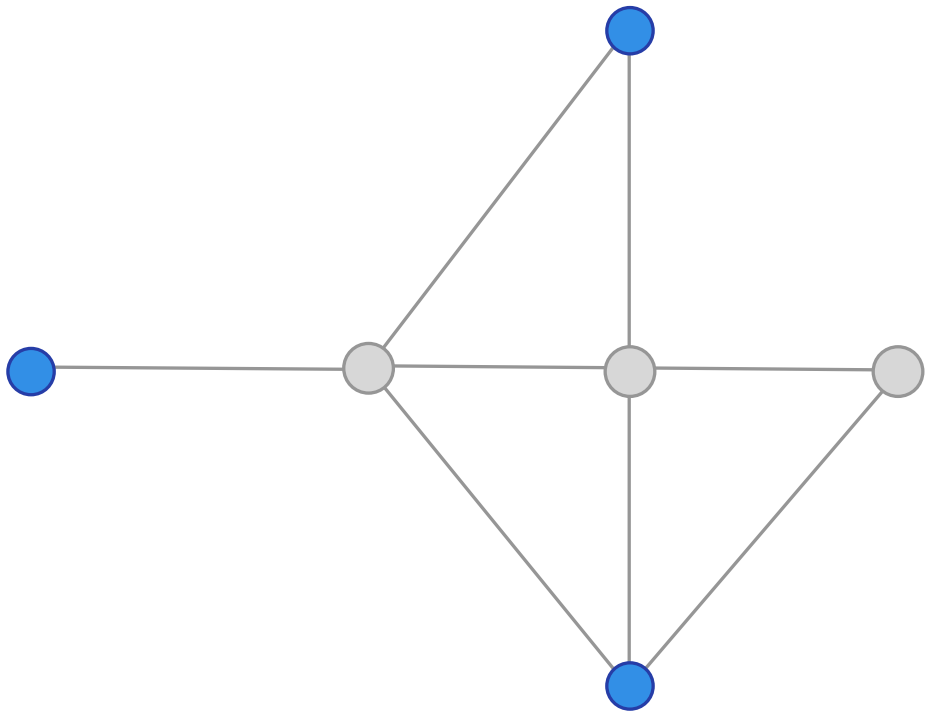
The Problem

Given a graph G, find the largest subgraph that is q-colorable.



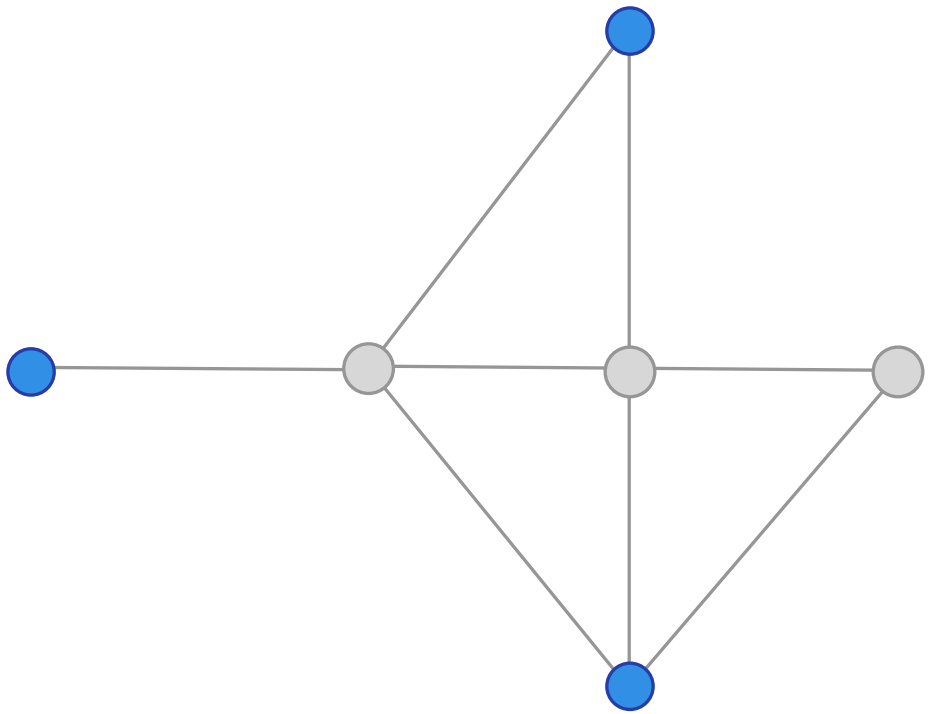
The Problem

Given a graph G, find the largest subgraph that is 1-colorable.



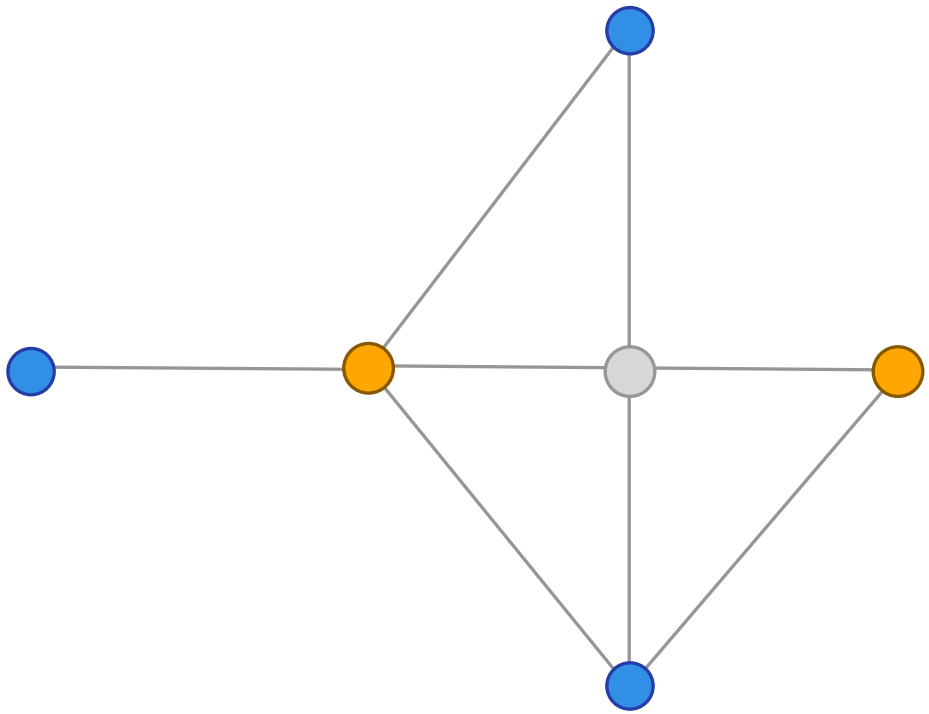
The Problem

Given a graph G, find the largest subgraph that is an independent set.



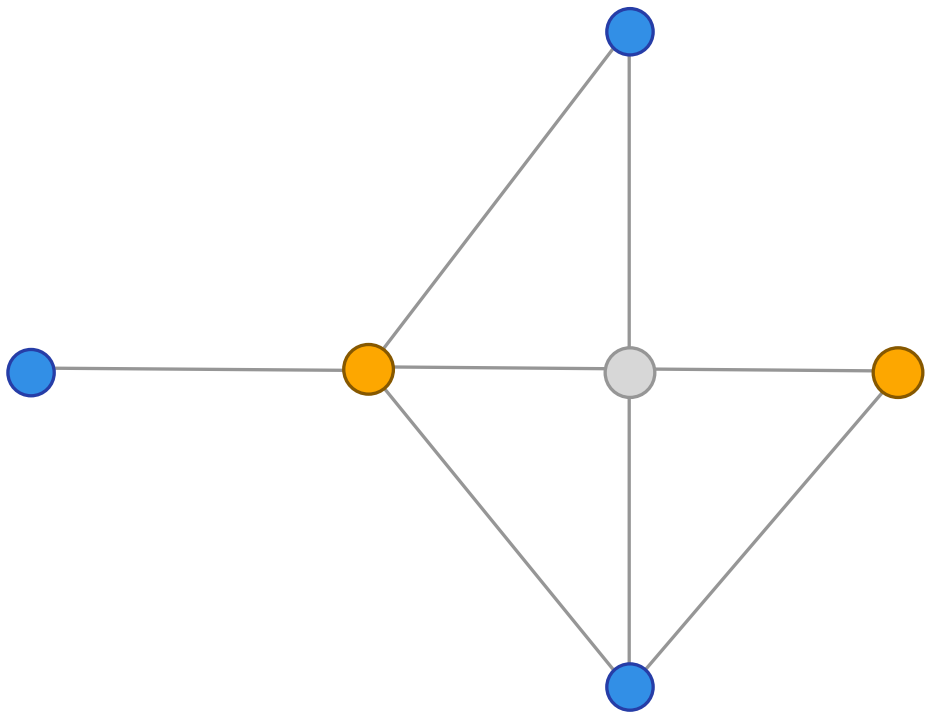
The Problem

Given a graph G, find the largest subgraph that is 2-colorable.



The Problem

Given a graph G, find the largest subgraph that is an induced bipartite subgraph.



Our Motivation

Yannakakis and Gavril [IPL 1987] showed that this problem is NP-complete even on split graphs if q is part of input, but gave an $n^{O(q)}$ algorithm on chordal graphs for fixed q.

Our Motivation

Yannakakis and Gavril [IPL 1987] showed that this problem is NP-complete even on split graphs if q is part of input, but gave an $n^{O(q)}$ algorithm on chordal graphs for fixed q.

An immediate natural question is: What is the status of this question in parametrized complexity?

Our Motivation

Yannakakis and Gavril [IPL 1987] showed that this problem is NP-complete even on split graphs if q is part of input, but gave an $n^{O(q)}$ algorithm on chordal graphs for fixed q.

An immediate natural question is: What is the status of this question in parametrized complexity?

In other words: Does this problem have an algorithm with running time $f(q) \cdot p(n)$ or $f(q, \ell) \cdot p(n, q)$? Here ℓ is the size of the subgraph we are looking for.

We first present a randomized algorithm with running time:

$f(\ell) \cdot$ [Time to enumerate maximal independent sets] $\cdot\, p(n, q)$,

to find an induced q-colorable subgraph of size at least ℓ.

We first present a randomized algorithm with running time:

$f(\ell) \cdot$ [Time to enumerate maximal independent sets] $\cdot\, p(n, q)$,

to find an induced q-colorable subgraph of size at least ℓ.

We first present a randomized algorithm with running time:

$$f(\ell) \cdot [\text{Time to enumerate maximal independent sets}] \cdot p(n, q),$$

to find an induced q-colorable subgraph of size at least ℓ.

Notice that we do not expect an algorithm with running time:

$$f(\ell) \cdot p(n, q),$$

since the problem is $W[1]$-hard on general graphs even when $q = 1$.

We first present a randomized algorithm with running time:

$f(\ell) \cdot$ [Time to enumerate maximal independent sets] $\cdot\, p(n, q)$,

to find an induced q-colorable subgraph of size at least ℓ.

This algorithm is reasonable for graph classes where maximal independent sets can be enumerated efficiently, for example, co-chordal and split graphs.

The problem remains NP-complete even when restricted to these graph classes.

We first present a randomized algorithm with running time:

$f(\ell) \cdot$ [Time to enumerate maximal independent sets] $\cdot\, p(n, q)$,

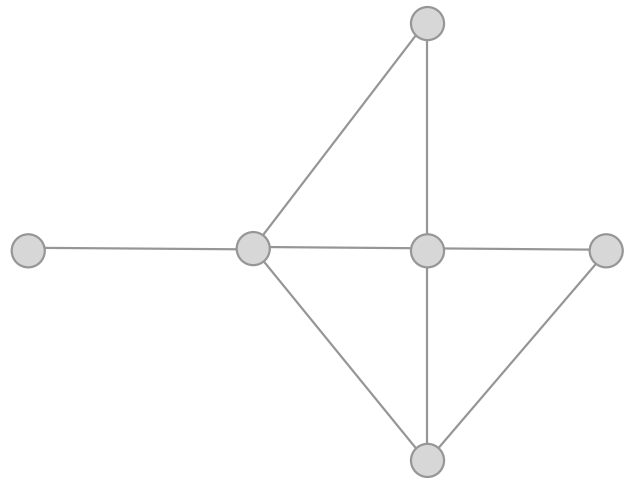
to find an induced q-colorable subgraph of size at least ℓ.

We also establish the non-existence of a polynomial kernels for the problem when parameterized by solution size alone.

(Under standard complexity-theoretic assumptions.)

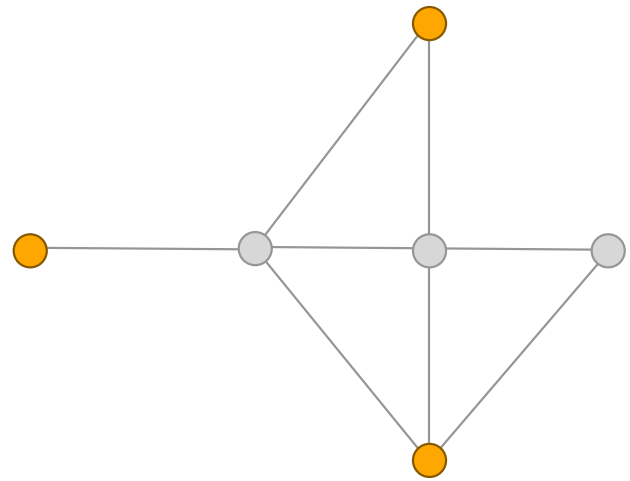
The Algorithm

Enumerate all maximal independent sets, and create an auxiliary graph that has one vertex m_i for every MIS.



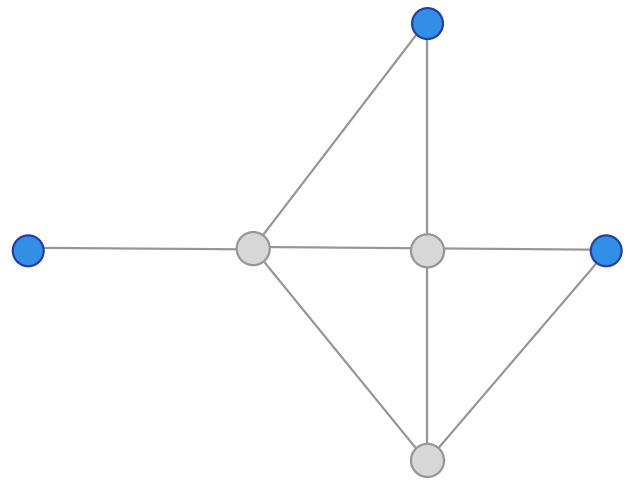
The Algorithm

Enumerate all maximal independent sets, and create an auxiliary graph that has one vertex m_i for every MIS.



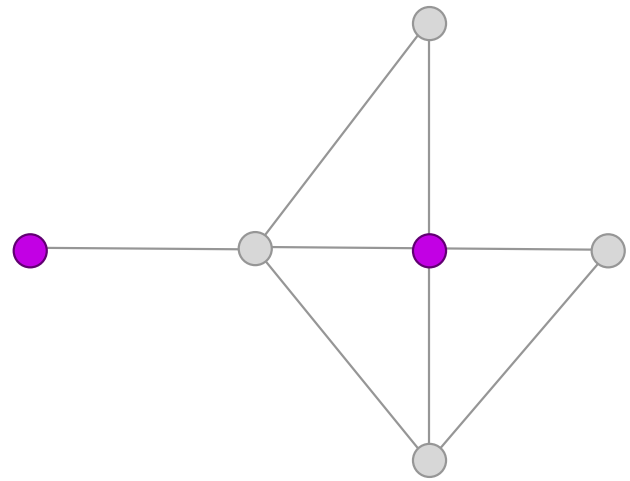
The Algorithm

Enumerate all maximal independent sets, and create an auxiliary graph that has one vertex m_i for every MIS.



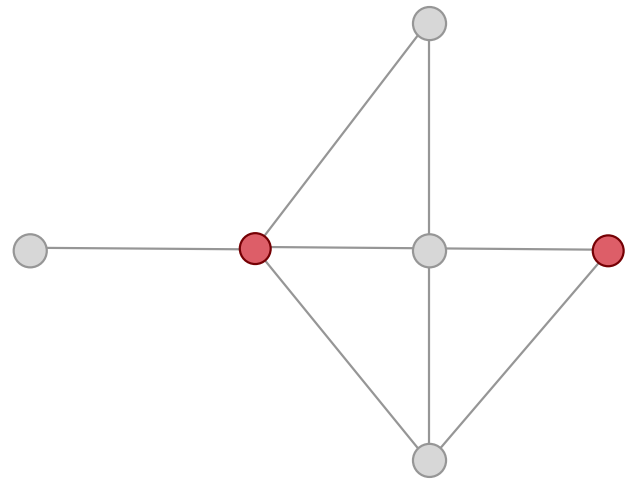
The Algorithm

Enumerate all maximal independent sets, and create an auxiliary graph that has one vertex m_i for every MIS.



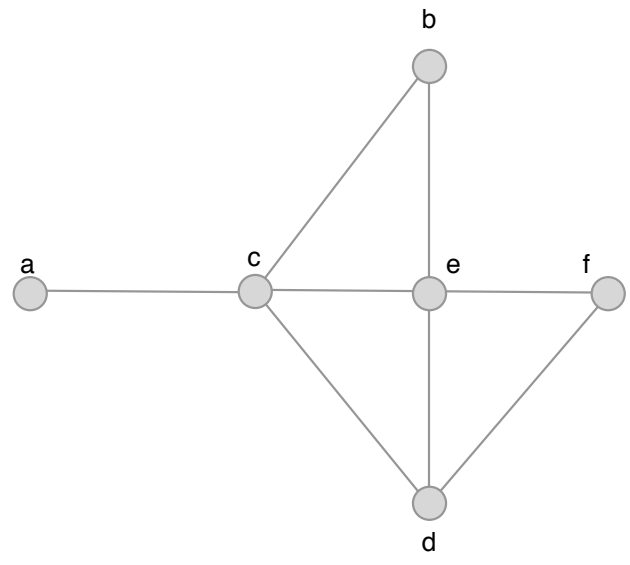
The Algorithm

Enumerate all maximal independent sets, and create an auxiliary graph that has one vertex m_i for every MIS.



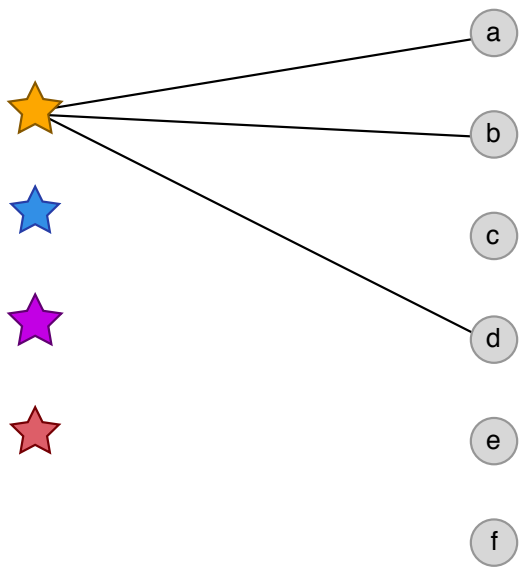
The Algorithm

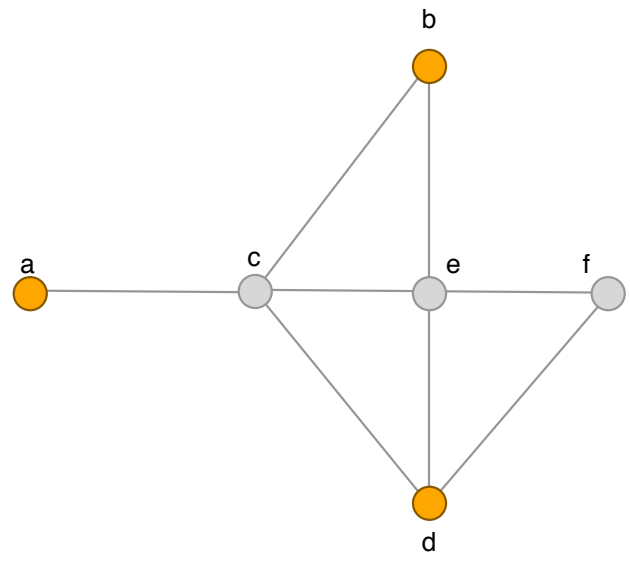
Introduce vertices corresponding to $V(G)$ and introduce the edge (v, m_i) if $v \in m_i$.



The Algorithm

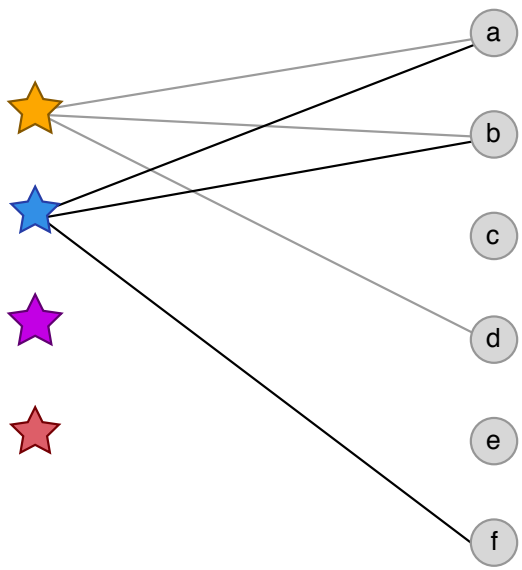
Introduce vertices corresponding to $V(G)$ and introduce the edge (v, m_i) if $v \in m_i$.

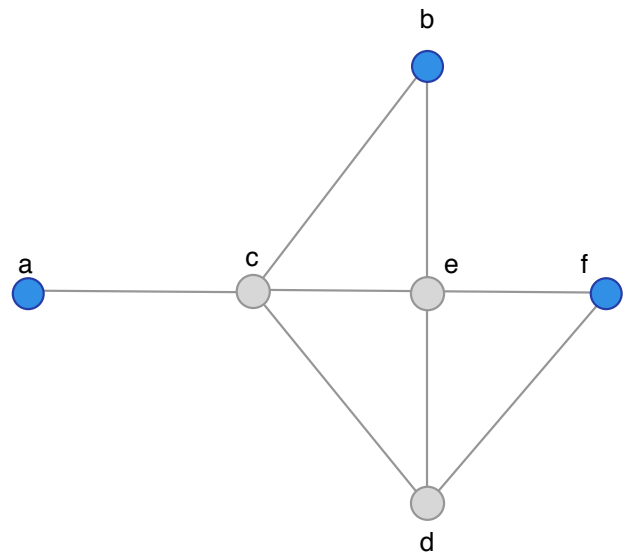




The Algorithm

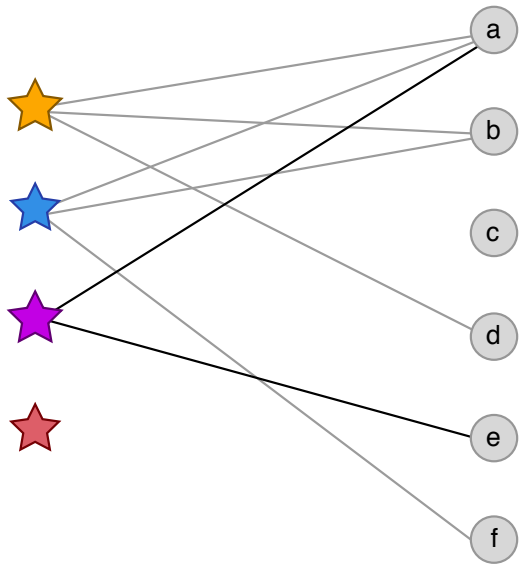
Introduce vertices corresponding to $V(G)$ and introduce the edge (v, m_i) if $v \in m_i$.

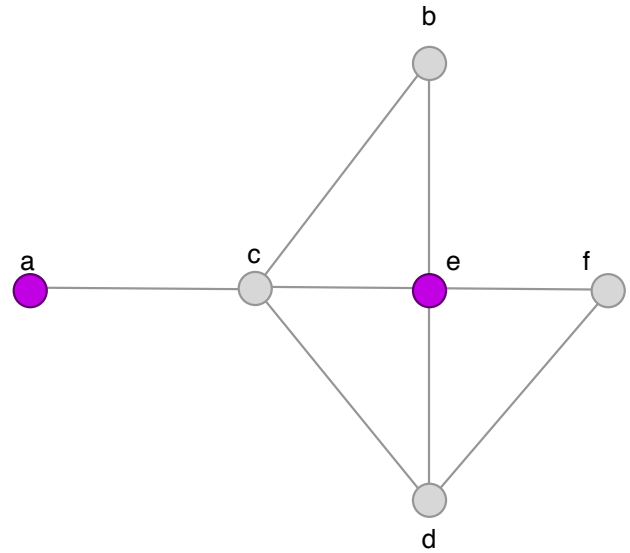




The Algorithm

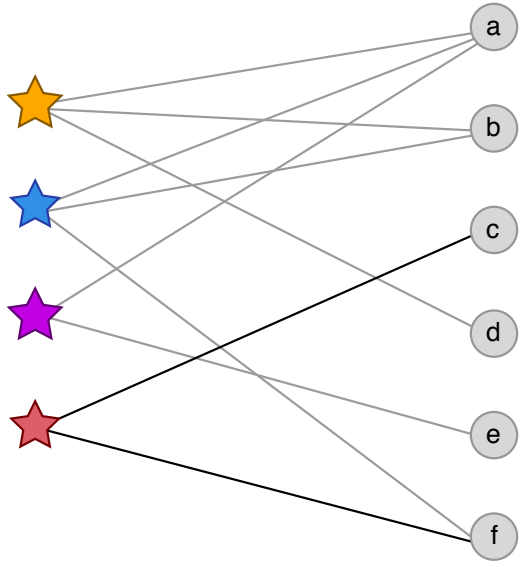
Introduce vertices corresponding to $V(G)$ and introduce the edge (v, m_i) if $v \in m_i$.

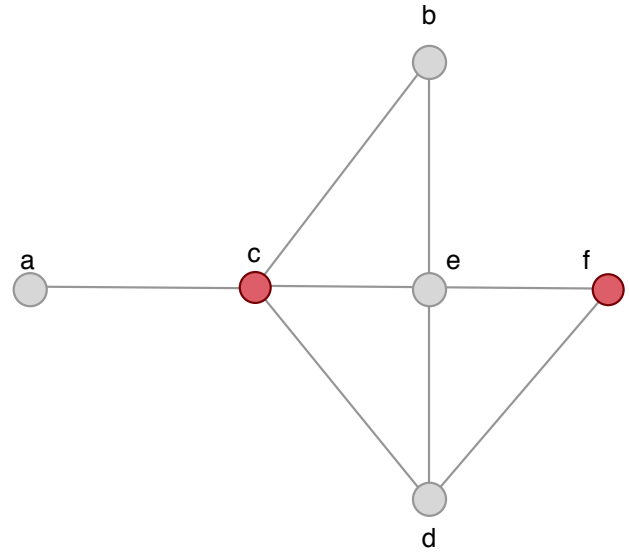




The Algorithm

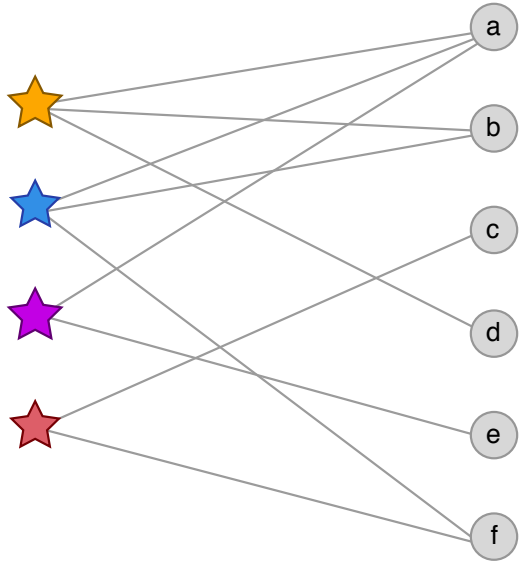
Introduce vertices corresponding to $V(G)$ and introduce the edge (v, m_i) if $v \in m_i$.

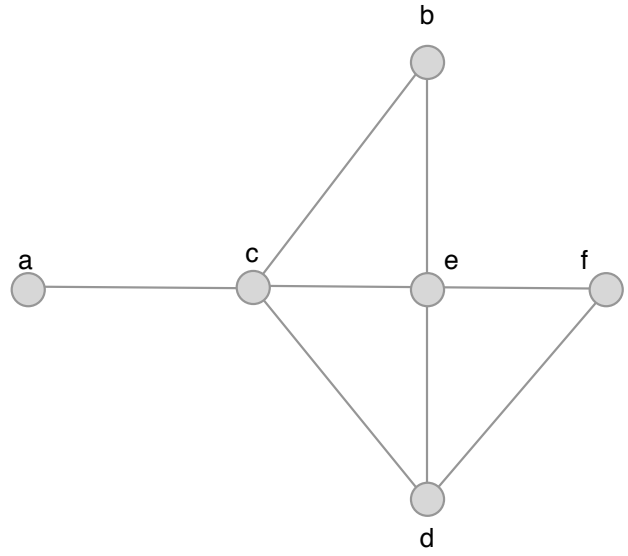




The Algorithm

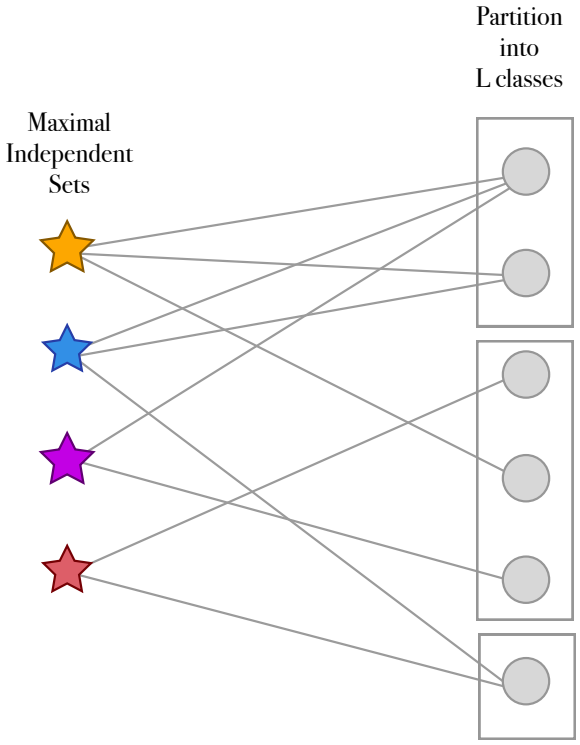
Introduce vertices corresponding to $V(G)$ and introduce the edge (v, m_i) if $v \in m_i$.

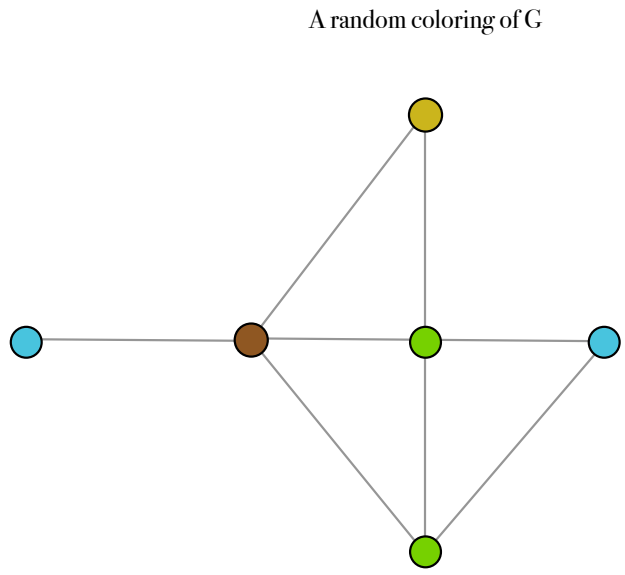




The Algorithm

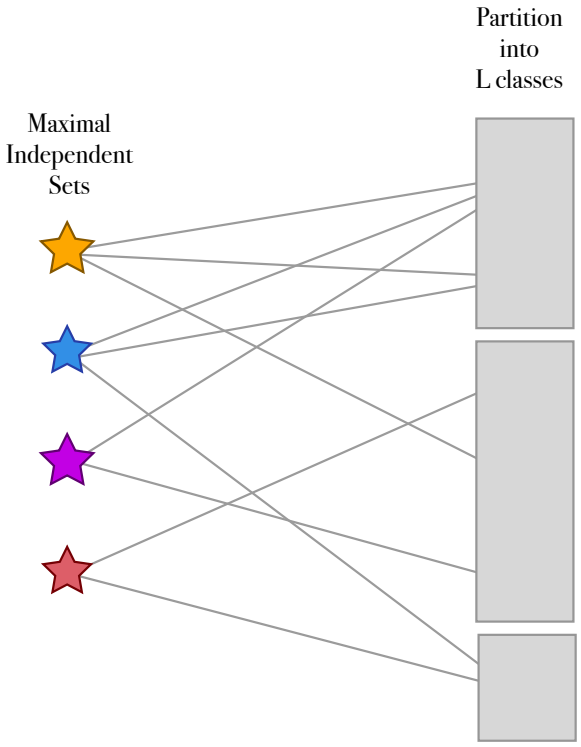
Color the vertices of the graph with ℓ colors, uniformly at random. Consider the color classes in the auxiliary graph.

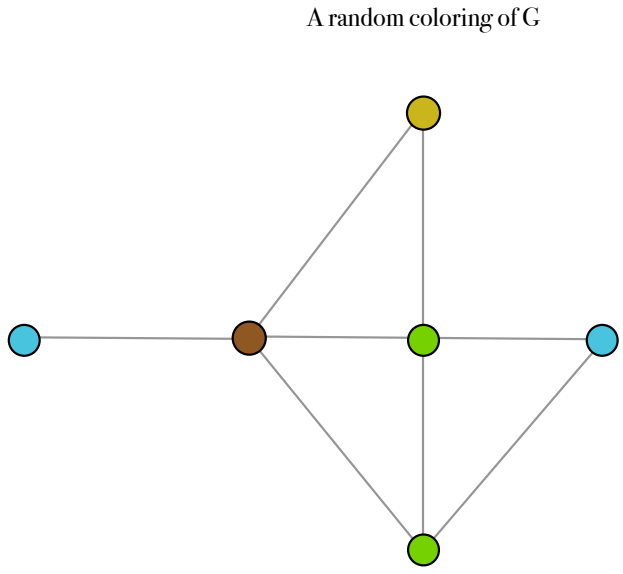




The Algorithm

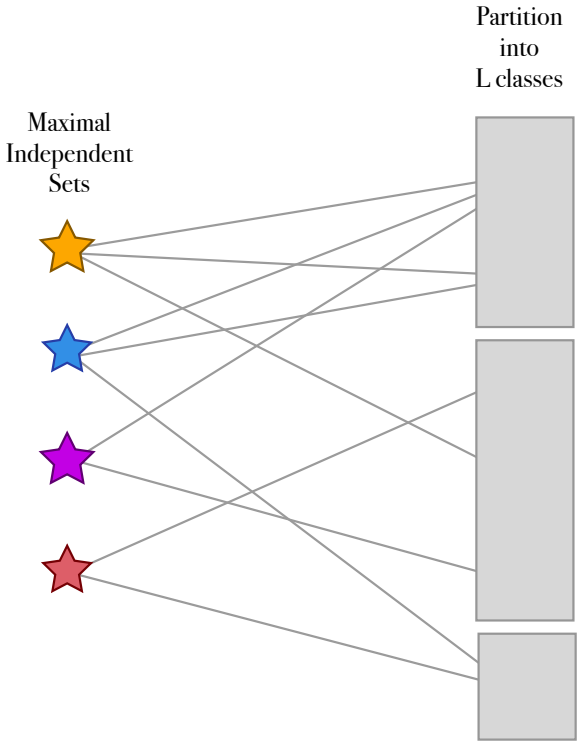
Now contract all the color classes into singletons, and find a dominating set of size at most q.

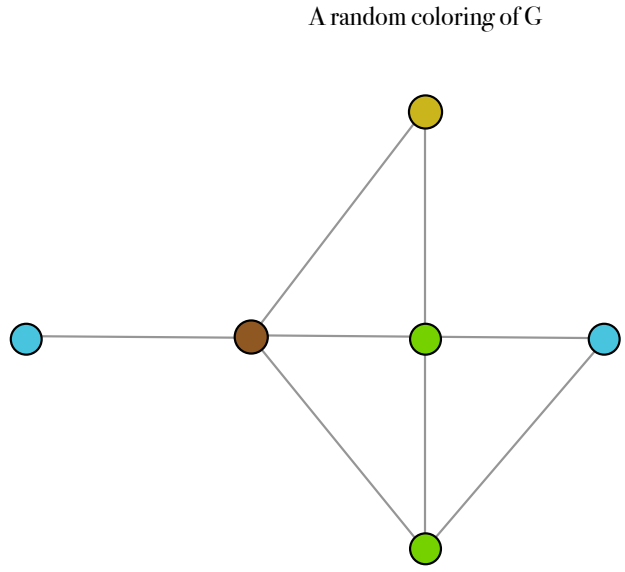




The Algorithm

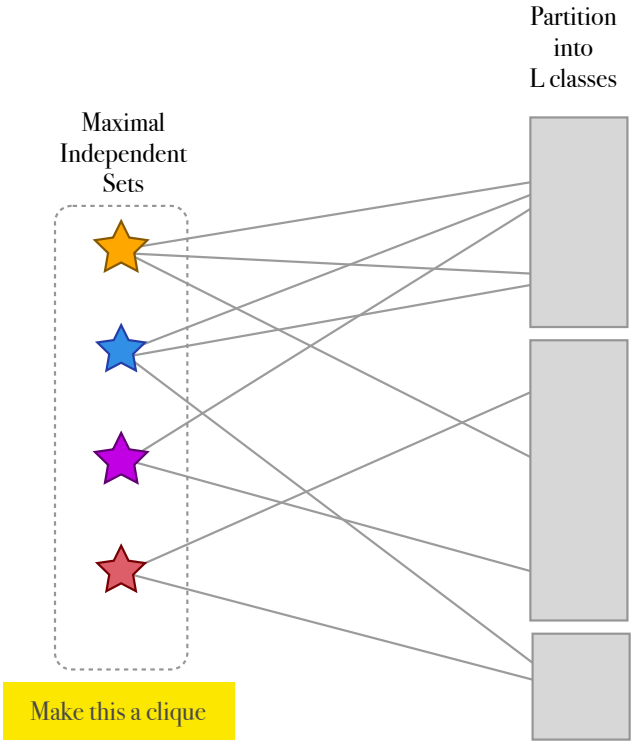
Clearly, if there is a dominating set, then we have found q-colorable subgraph of size at least ℓ.

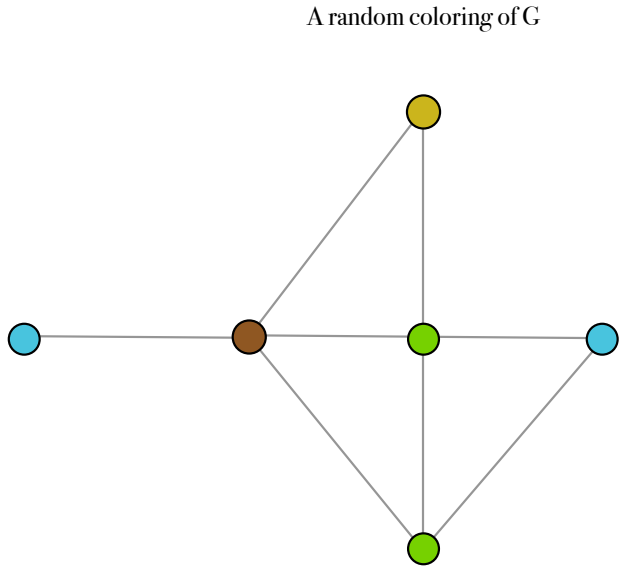




The Algorithm

To compute the dominating set, make the “MIS vertices” a clique and run Steiner Tree with the V as the terminals.





The Algorithm

When we randomly color the vertices, each vertex in a possible solution will get different colors with probability:

$$\frac{\ell!}{\ell^\ell} \geqslant e^{-\ell}$$

The Algorithm

When we randomly color the vertices, each vertex in a possible solution will get different colors with probability:

$$\frac{\ell!}{\ell^\ell} \geqslant e^{-\ell}$$

Once we have a nice coloring, clearly everything else works out fine.

The Algorithm

When we randomly color the vertices, each vertex in a possible solution will get different colors with probability:

$$\frac{\ell!}{\ell^\ell} \geqslant e^{-\ell}$$

Once we have a nice coloring, clearly everything else works out fine.

The Algorithm

We then present a randomized algorithm with running time:

$$f(\ell) \cdot [\text{Time to find a maximum sized independent sets}] \cdot p(n, q),$$

to find an induced q-colorable subgraph of size at least ℓ.

The Algorithm

We then present a randomized algorithm with running time:

$$f(\ell) \cdot [\text{Time to find a maximum sized independent sets}] \cdot p(n, q),$$

to find an induced q-colorable subgraph of size at least ℓ.

This algorithm is good for perfect graphs where maximum independent set can be found in polynomial time.

The Algorithm

A graph H is q colorable if and only if its vertex set can be partitioned into q independent sets.

The Algorithm

- Randomly color the vertices of G with $\{1, \ldots, q\}$.
- Let V_i be the set of vertices with color i.
- Find a maximum sized independent set $I_i \subseteq G[V_i]$.
- Return $I = \cup_{i=1}^{q} I_i$.

The Algorithm

Suppose we have a solution $W = \biguplus_{i=1}^{q} W_i$. Here W_i is an independent solution and $\sum_{1=1}^{q} W_i| = \ell$.

The Algorithm

Suppose we have a solution $W = \biguplus_{i=1}^{q} W_i$. Here W_i is an independent solution and $\sum_{1=1}^{q} W_i| = \ell$.

When we randomly color the vertices, a vertex in W_i gets color i

$$\frac{1}{q^{\ell}}$$

The Algorithm

Suppose we have a solution $W = \biguplus_{i=1}^{q} W_i$. Here W_i is an independent solution and $\sum_{1=1}^{q} W_i| = \ell$.

When we randomly color the vertices, a vertex in W_i gets color i

$$\frac{1}{q^{\ell}}$$

Once we have a nice coloring, clearly
everything else works out fine.

The Algorithm

Suppose we have a solution $W = \biguplus_{i=1}^{q} W_i$. Here W_i is an independent solution and $\sum_{1=1}^{q} W_i| = \ell$.

When we randomly color the vertices, a vertex in W_i gets color i

$$\frac{1}{q^{\ell}}$$

Once we have a nice coloring, clearly
everything else works out fine.

Kernelization Algorithm

A parameterized problem is said to admit a polynomial kernel if every instance (I, k) can be reduced in polynomial time to an equivalent instance (I', k') with both size and parameter value bounded by a polynomial in k.

No Kernel Results

Finally, we show that (under standard complexity-theoretic assumptions) the problem does not admit a polynomial kernel on split and perfect graphs in the following sense:

No Kernel Results

Finally, we show that (under standard complexity-theoretic assumptions) the problem does not admit a polynomial kernel on split and perfect graphs in the following sense:

(a) On split graphs, we do not expect a polynomial kernel if q is a part of the input.

(b) On perfect graphs, we do not expect a polynomial kernel even for fixed values of $q \geqslant 2$.

Composition

A OR-composition algorithm for a parameterized problem $\Pi \subseteq \Sigma^* \times \mathbb{N}$ is an algorithm

Composition

A OR-composition algorithm for a parameterized problem $\Pi \subseteq \Sigma^* \times \mathbb{N}$ is an algorithm that receives as input a sequence $((x_1, k), ..., (x_t, k))$, with

$$(x_i, k) \in \Sigma^* \times \mathbb{N} \text{ for each } 1 \leqslant i \leqslant t,$$

Composition

A OR-composition algorithm for a parameterized problem $\Pi \subseteq \Sigma^* \times \mathbb{N}$ is an algorithm that receives as input a sequence $((x_1, k), ..., (x_t, k))$, with

$$(x_i, k) \in \Sigma^* \times \mathbb{N} \text{ for each } 1 \leqslant i \leqslant t,$$

uses time polynomial in $\sum_{i=1}^{t} |x_i| + k$, and outputs $(y, k') \in \Sigma^* \times \mathbb{N}$

Composition

A OR-composition algorithm for a parameterized problem $\Pi \subseteq \Sigma^* \times \mathbb{N}$ is an algorithm that receives as input a sequence $((x_1, k), ..., (x_t, k))$, with

$(x_i, k) \in \Sigma^* \times \mathbb{N}$ for each $1 \leqslant i \leqslant t$,

uses time polynomial in $\sum_{i=1}^{t} |x_i| + k$, and outputs $(y, k') \in \Sigma^* \times \mathbb{N}$

with (a) $(y, k') \in \Pi \iff (x_i, k) \in \Pi$ for some $1 \leqslant i \leqslant t$ and (b) k' is polynomial in k.

Composition

A OR-composition algorithm for a parameterized problem $\Pi \subseteq \Sigma^* \times \mathbb{N}$ is an algorithm that receives as input a sequence $((x_1, k), ..., (x_t, k))$, with

$(x_i, k) \in \Sigma^* \times \mathbb{N}$ for each $1 \leqslant i \leqslant t$,

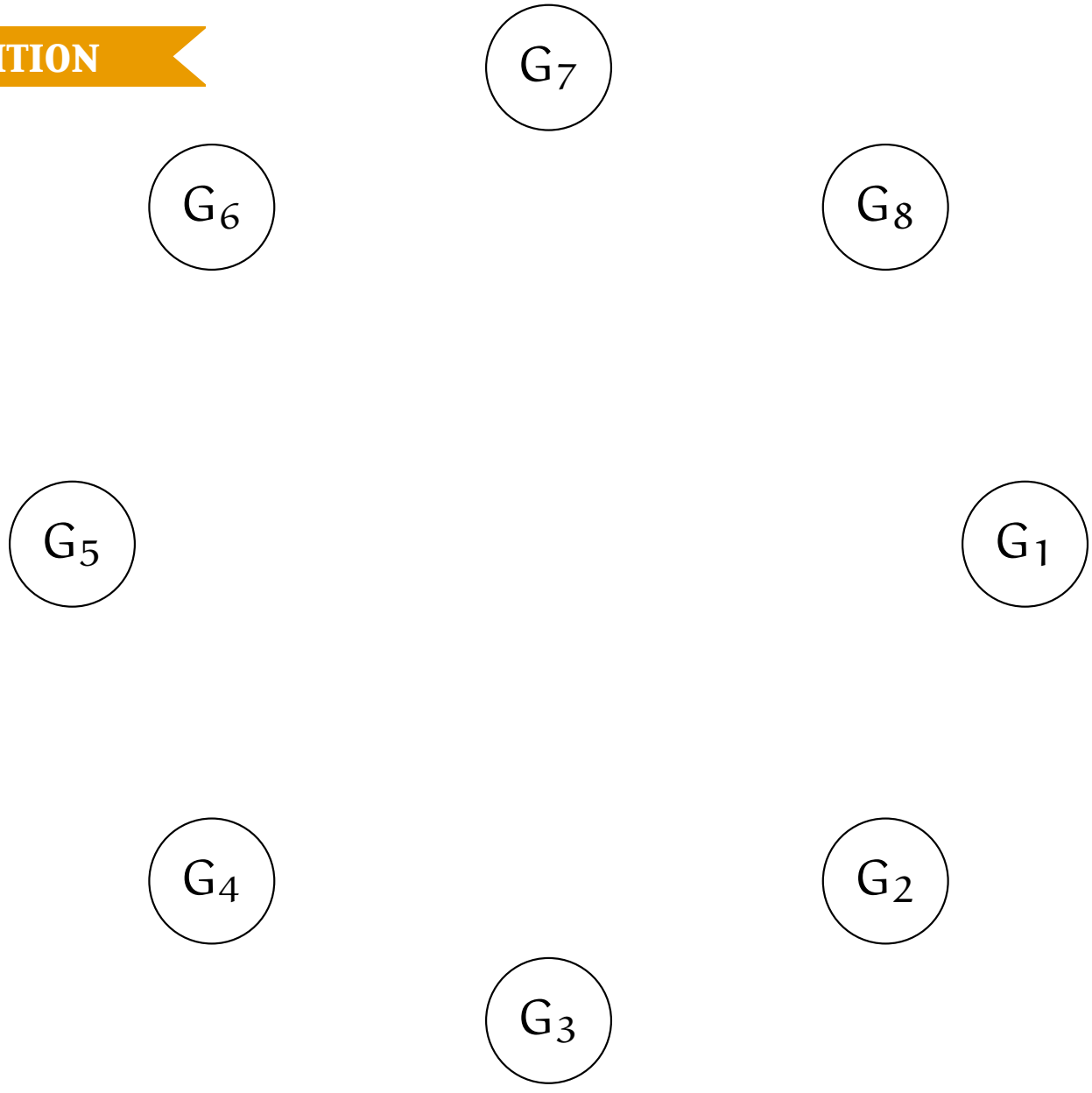
uses time polynomial in $\sum_{i=1}^{t} |x_i| + k$, and outputs $(y, k') \in \Sigma^* \times \mathbb{N}$

with (a) $(y, k') \in \Pi \iff (x_i, k) \in \Pi$ for some $1 \leqslant i \leqslant t$ and (b) k' is polynomial in k.

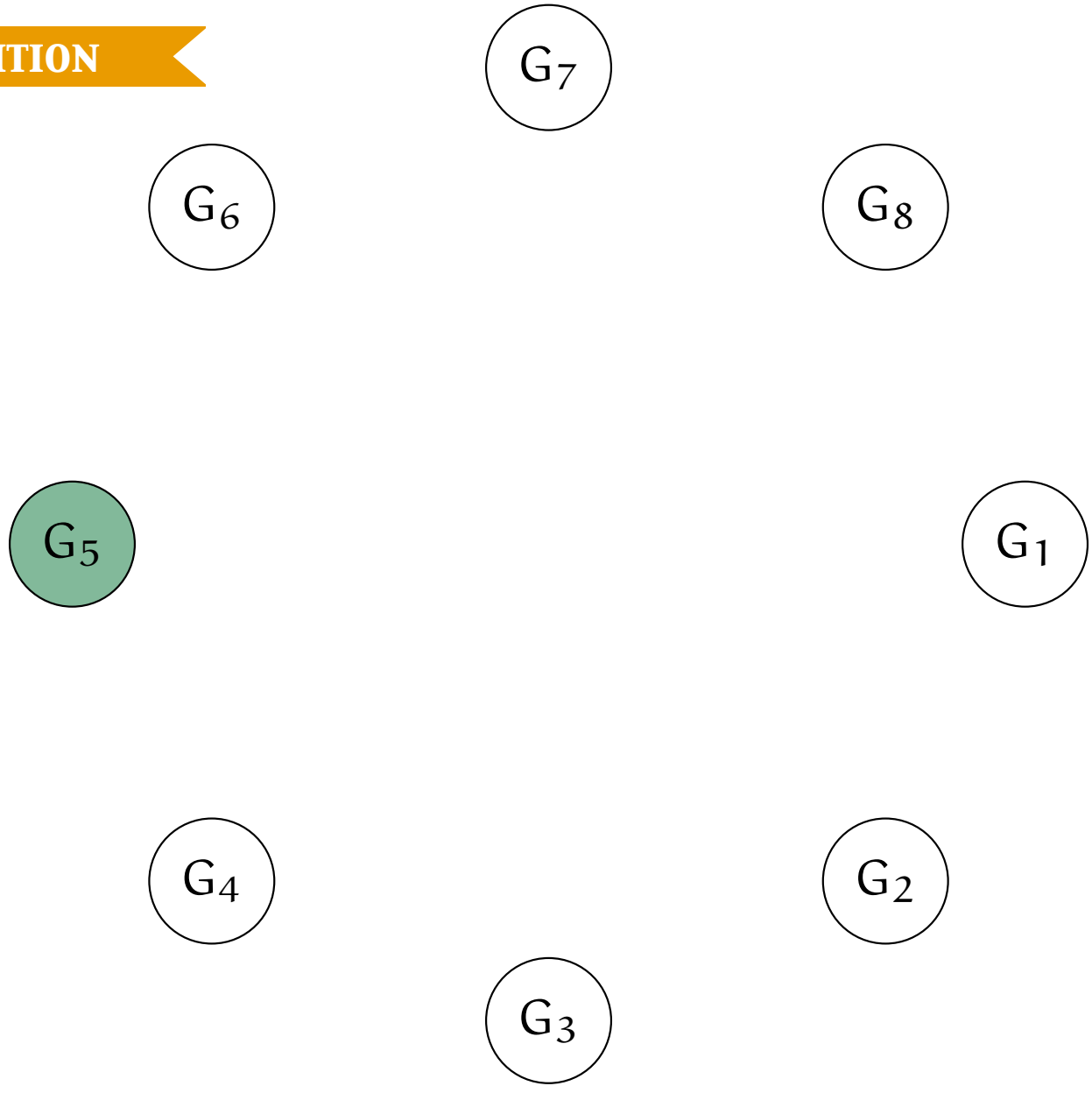
A parameterized problem is or OR-compositional if there is a composition algorithm for it.

The Composition Algorithm

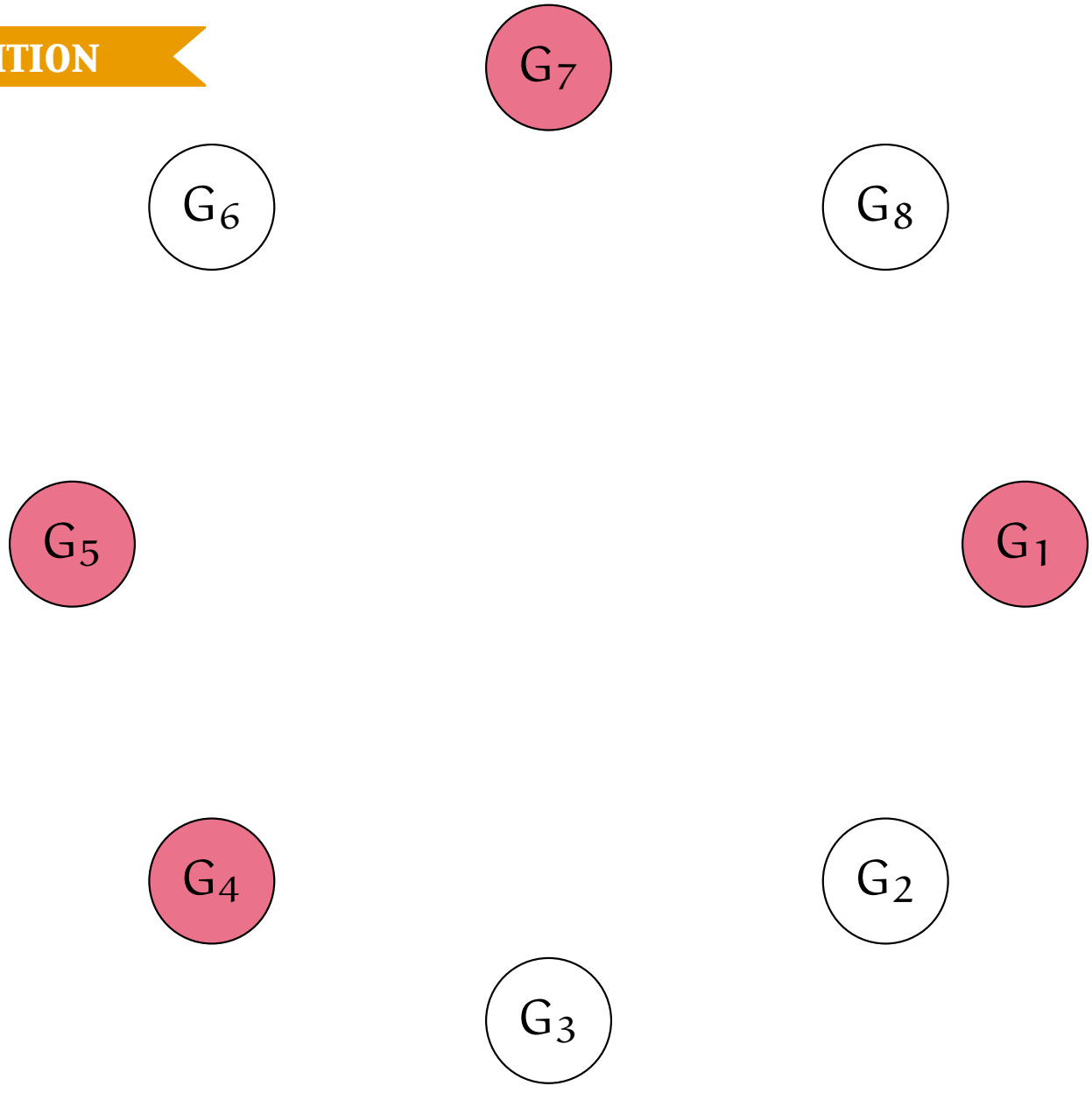
The Composition



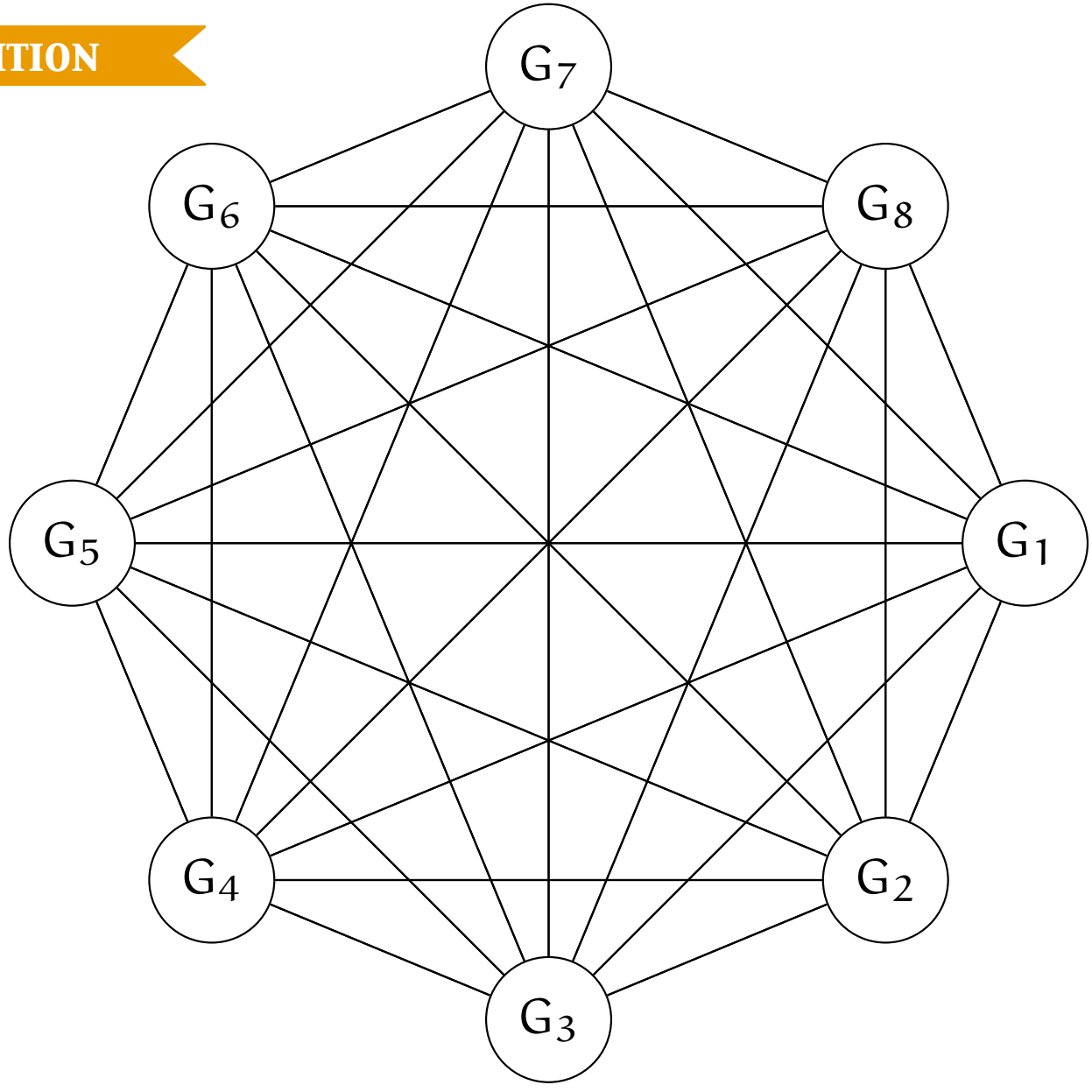
The Composition



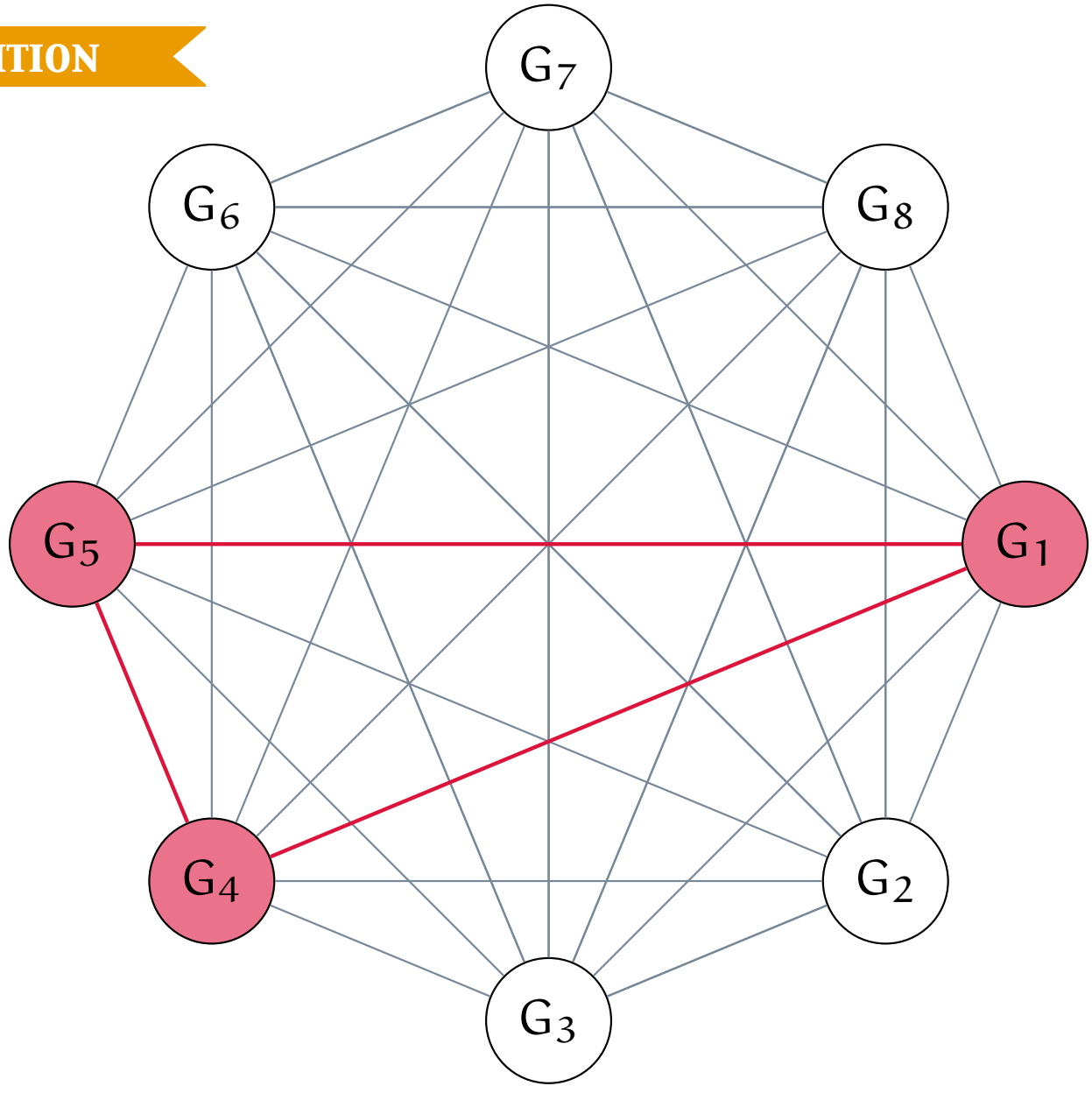
The Composition



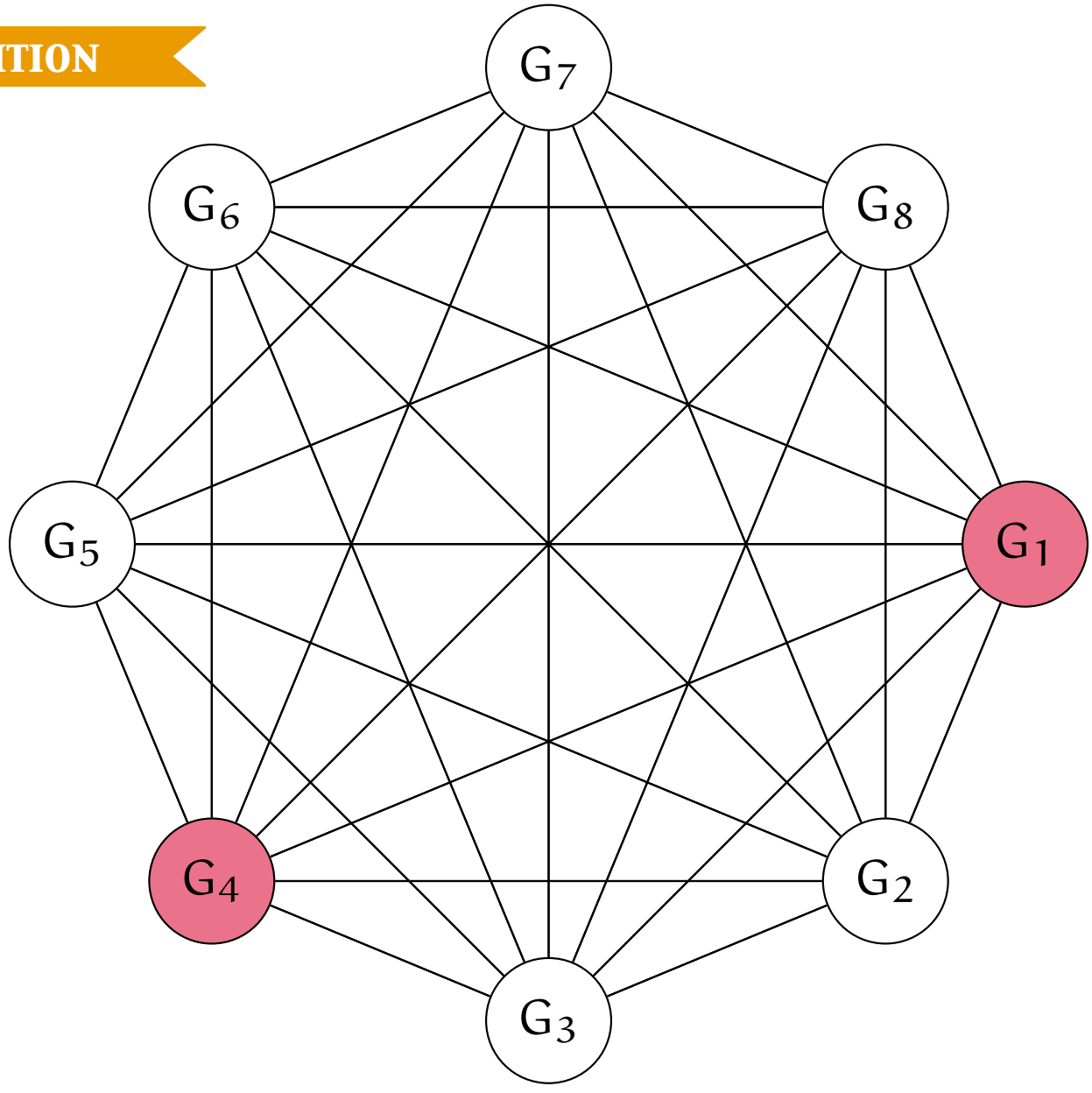
The Composition



The Composition

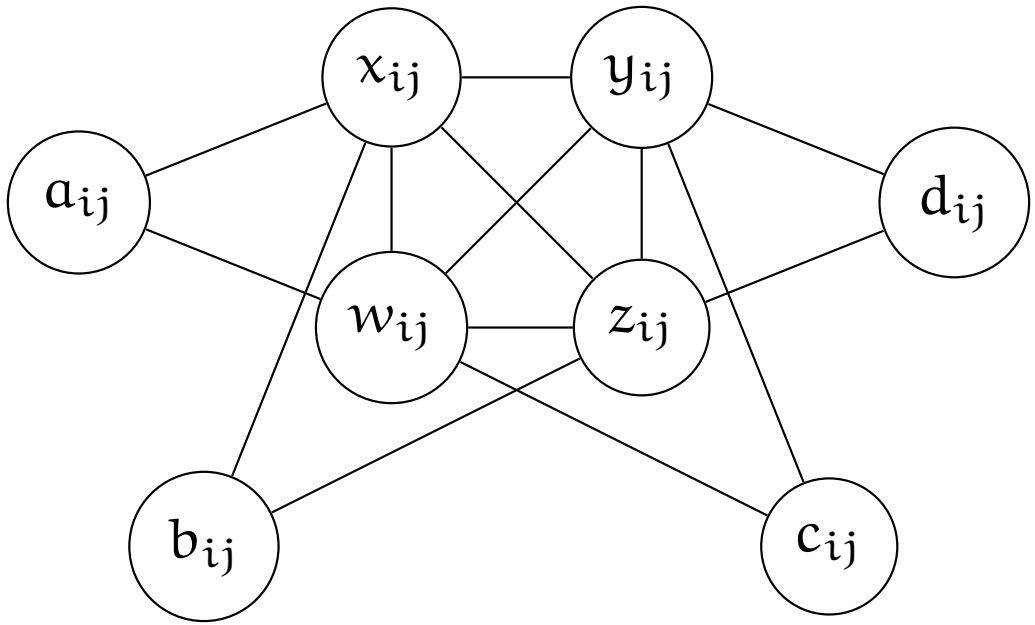


The Composition



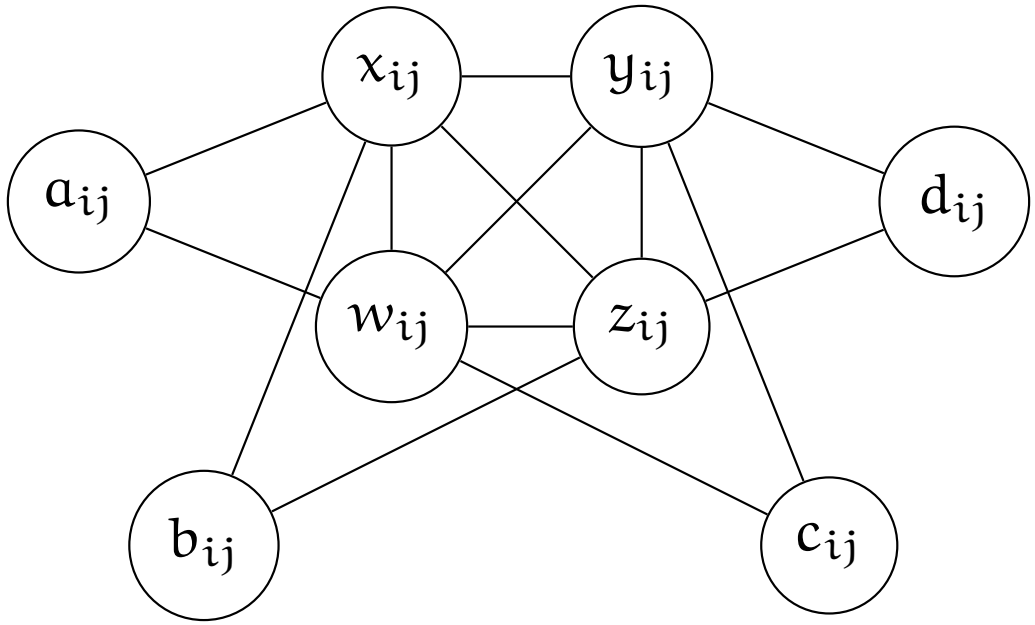
A spillover across two instances could still be a problem.

The Composition



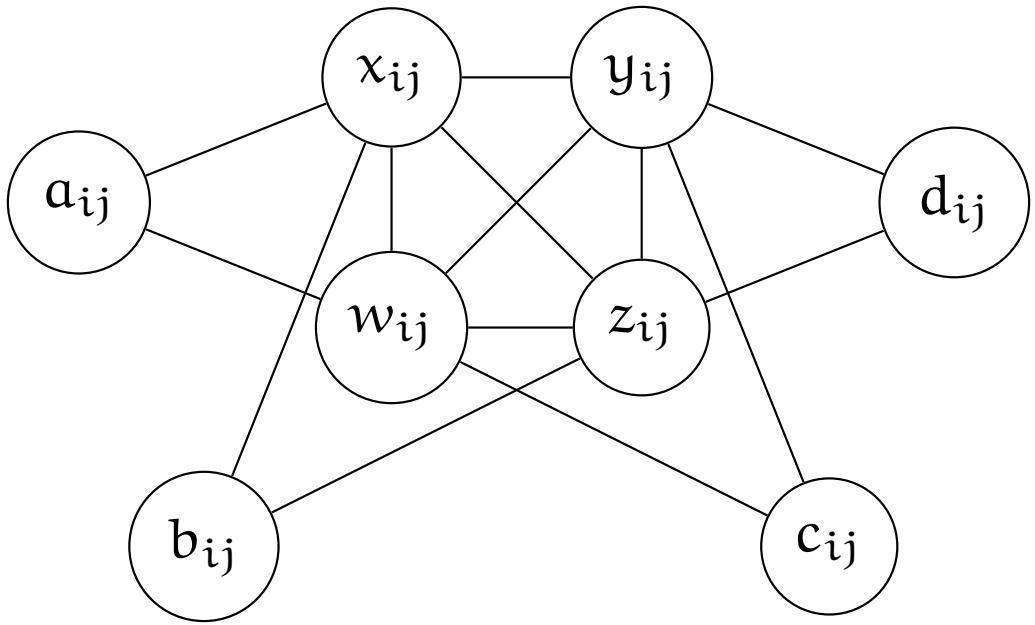
The Composition

What is this gadget trying to achieve?



The Composition

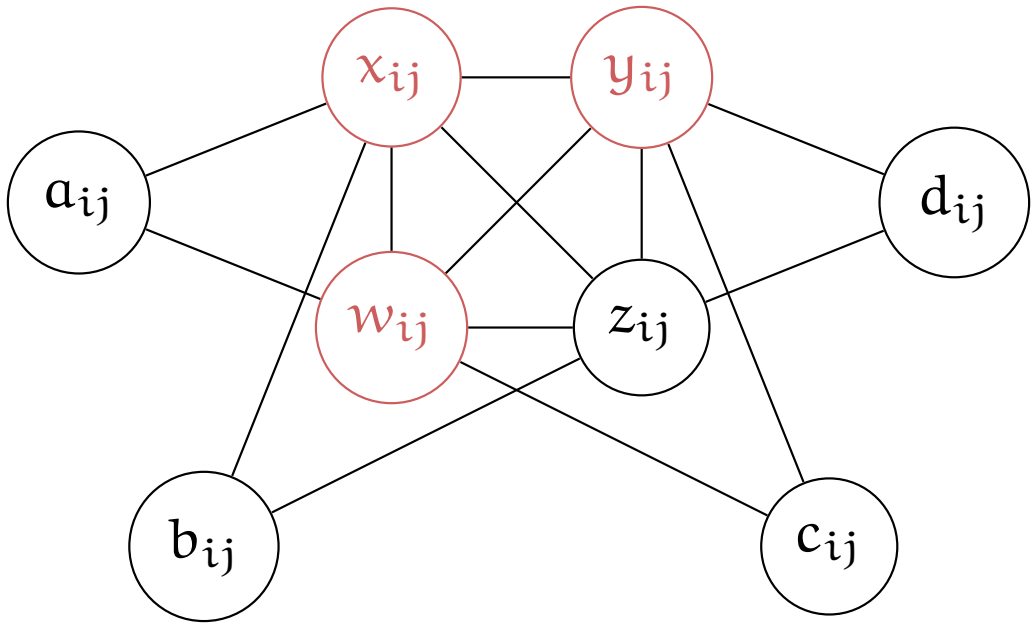
If the gadget contributes six vertices to any induced bipartite subgraph, then:



The Composition

If the gadget contributes six vertices to any induced bipartite subgraph, then:

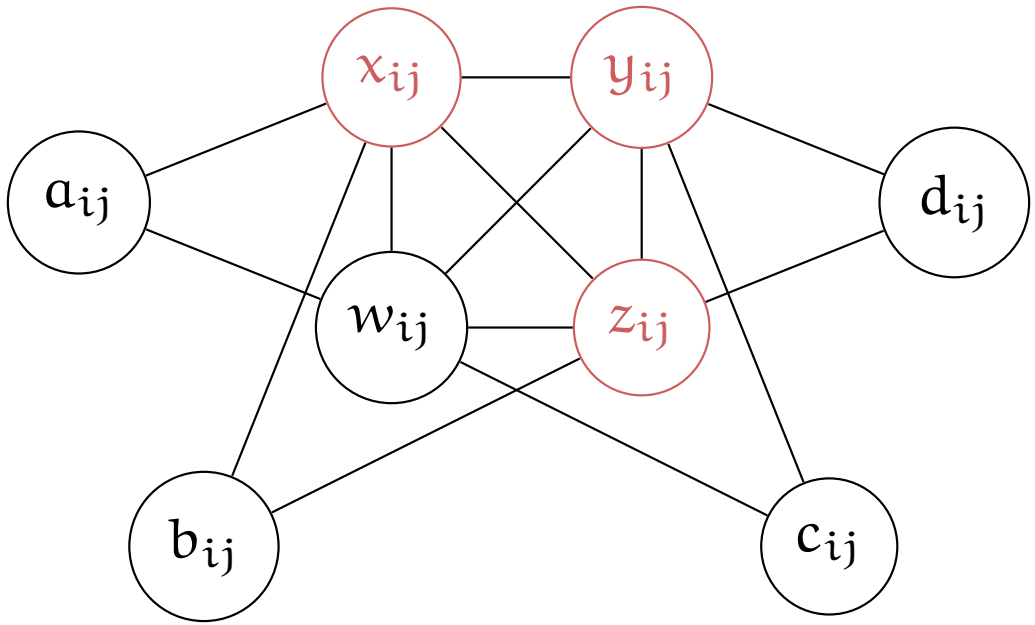
At most two inner vertices participate in the solution.



The Composition

If the gadget contributes six vertices to any induced bipartite subgraph, then:

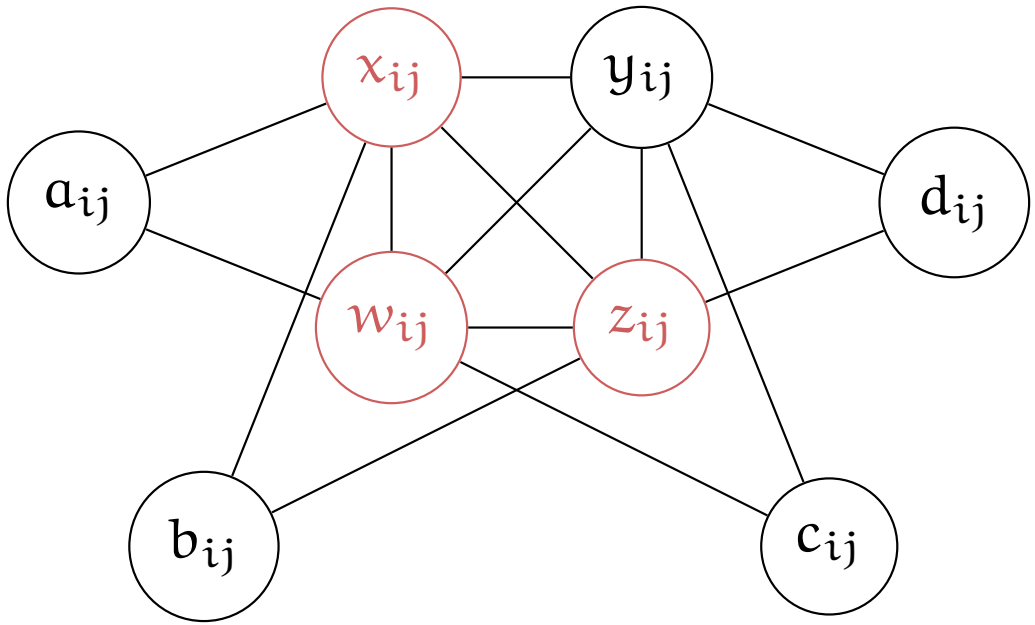
At most two inner vertices participate in the solution.



The Composition

If the gadget contributes six vertices to any induced bipartite subgraph, then:

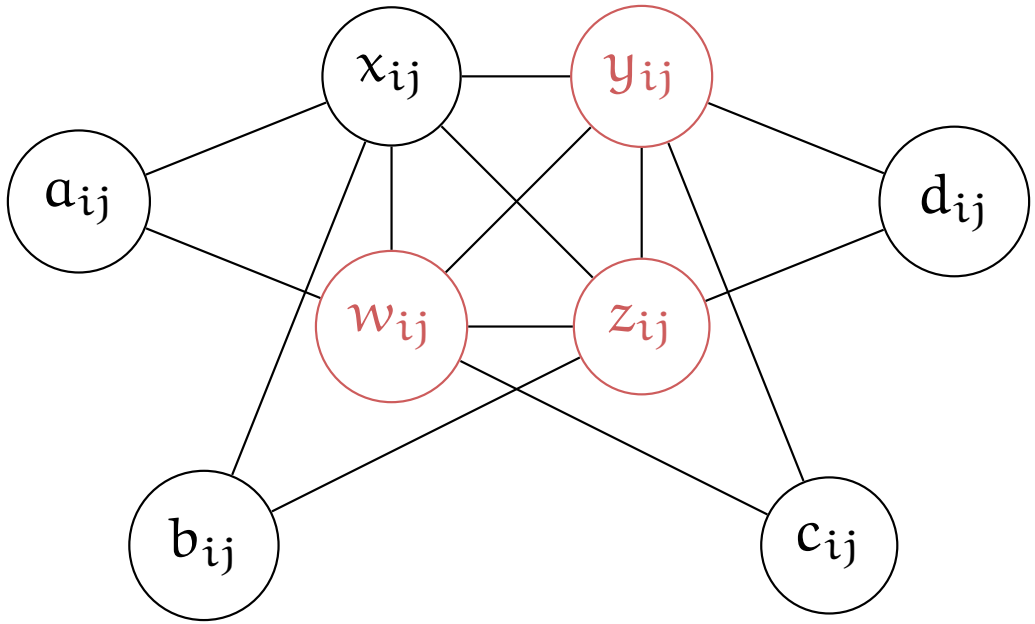
At most two inner vertices participate in the solution.



The Composition

If the gadget contributes six vertices to any induced bipartite subgraph, then:

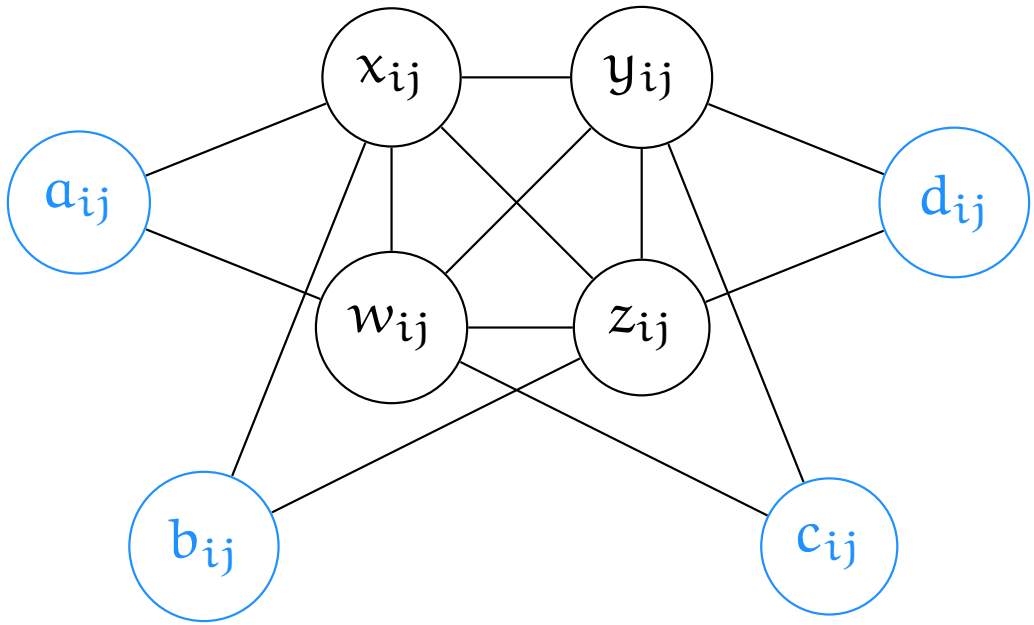
At most two inner vertices participate in the solution.



The Composition

If the gadget contributes six vertices to any induced bipartite subgraph, then:

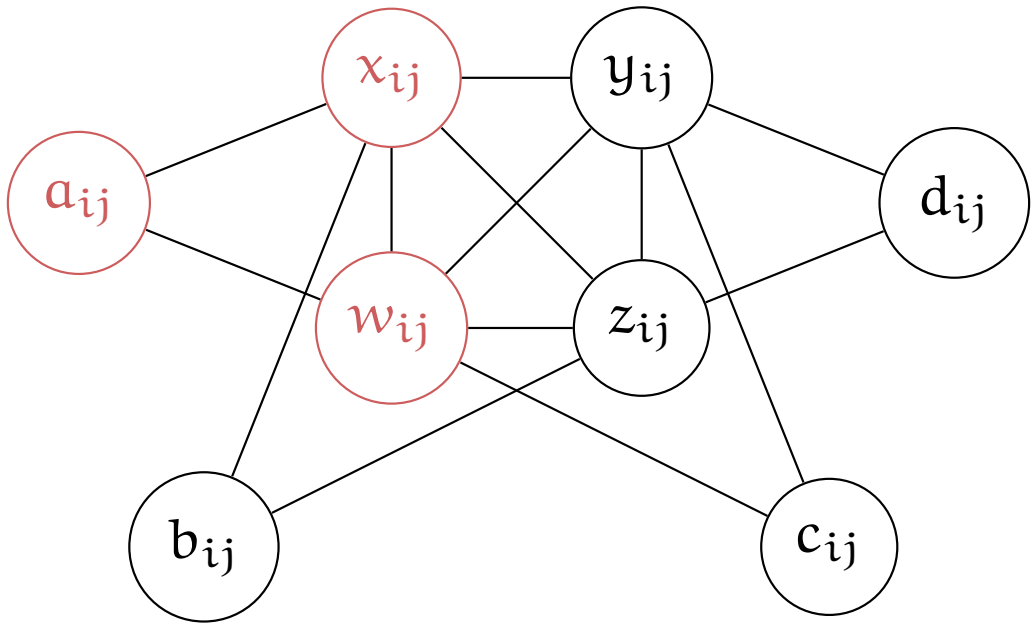
All four outer vertices participate in the solution.



The Composition

If the gadget contributes six vertices to any induced bipartite subgraph, then:

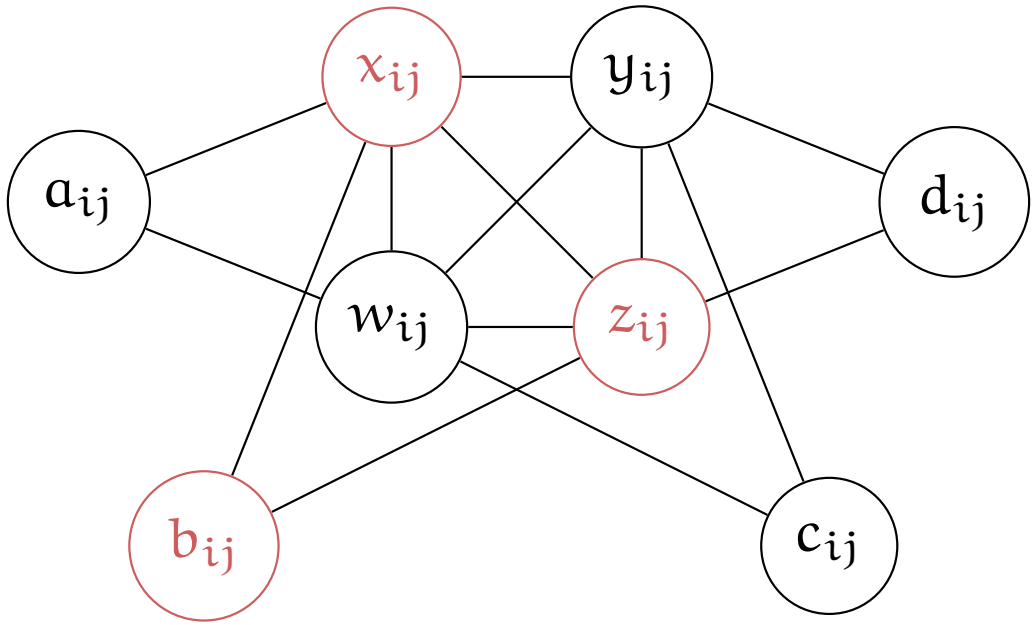
Inner vertices from two levels do not participate together.



The Composition

If the gadget contributes six vertices to any induced bipartite subgraph, then:

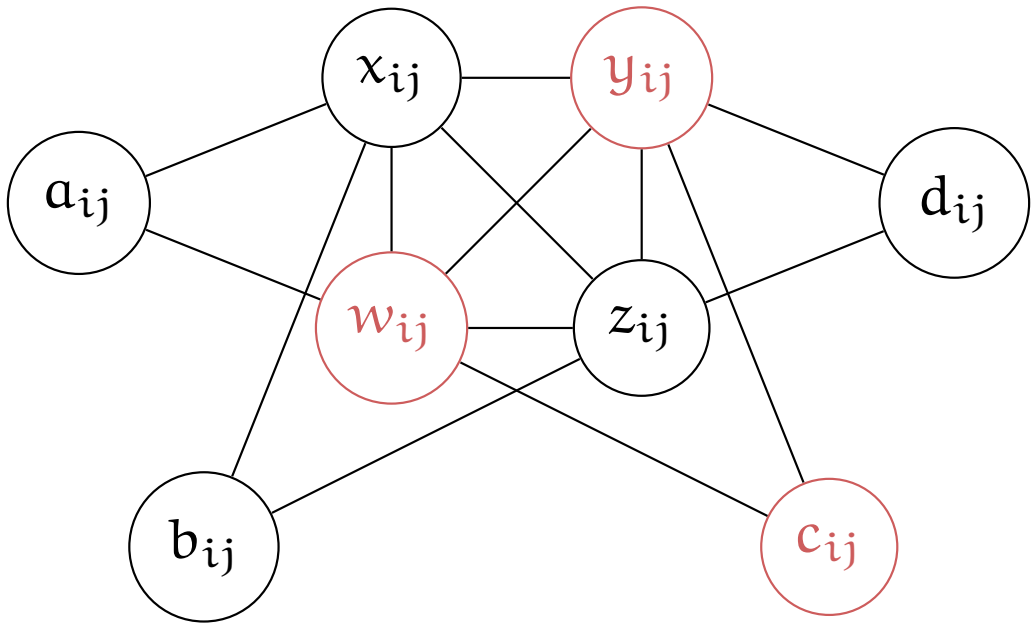
Inner vertices from two levels do not participate together.



The Composition

If the gadget contributes six vertices to any induced bipartite subgraph, then:

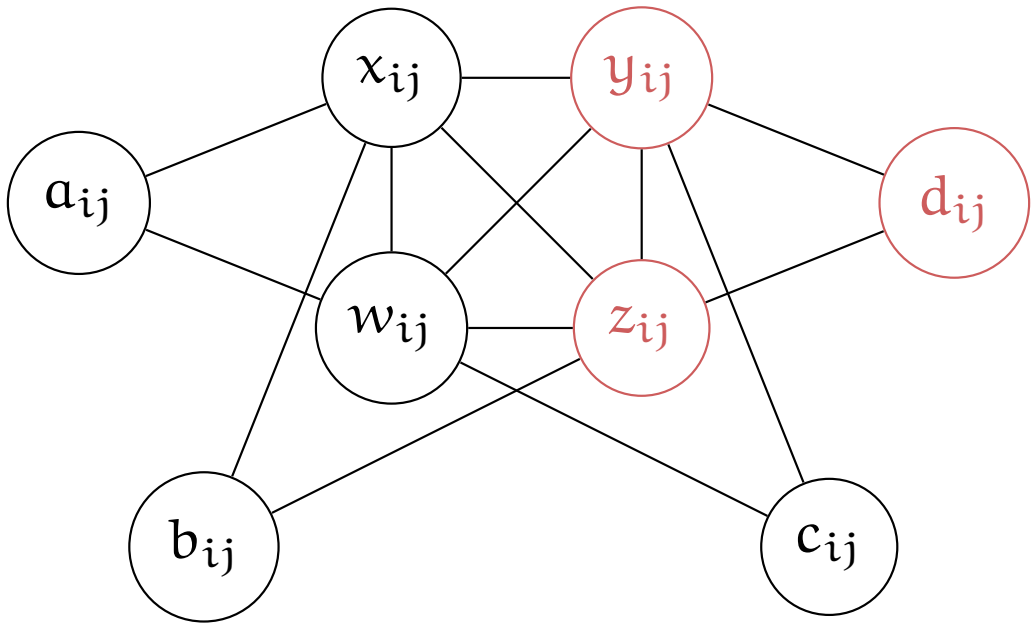
Inner vertices from two levels do not participate together.



The Composition

If the gadget contributes six vertices to any induced bipartite subgraph, then:

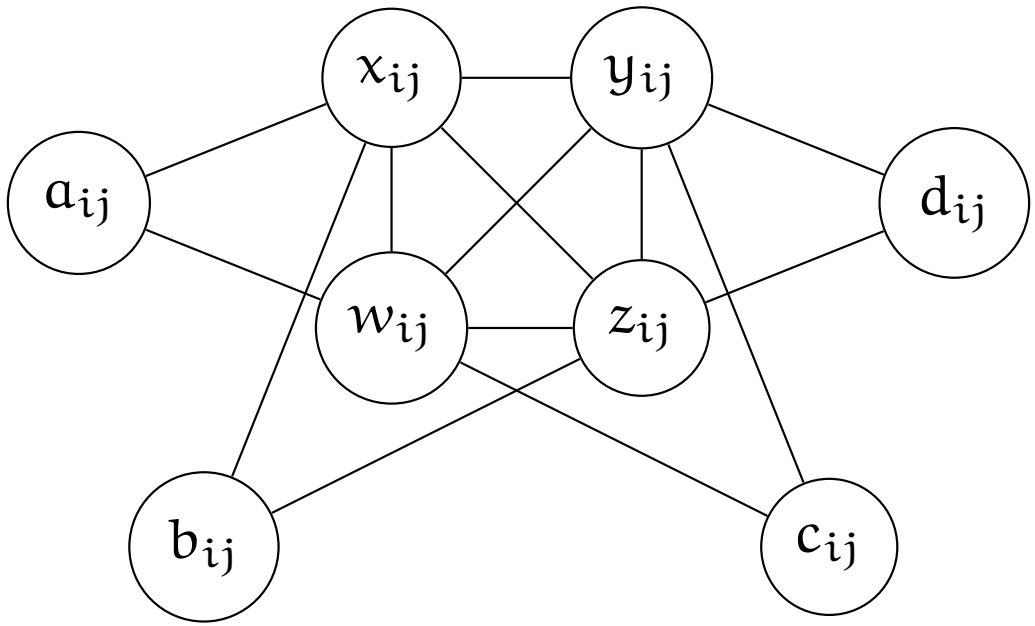
Inner vertices from two levels do not participate together.



The Composition

If the gadget contributes six vertices to any induced bipartite subgraph, then:

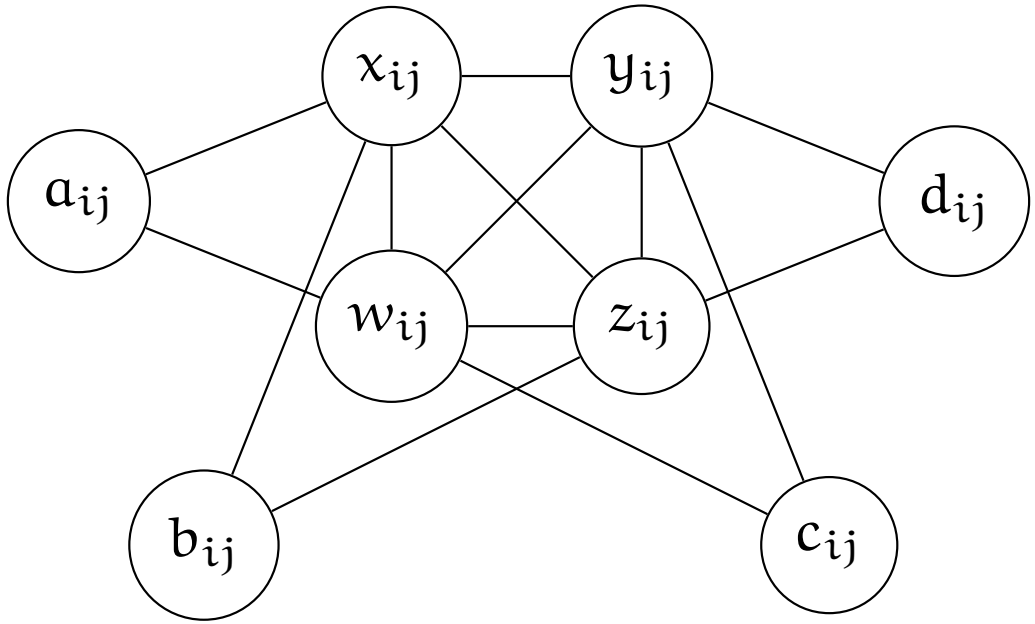
Inner vertices from two levels do not participate together.



The Composition

If the gadget contributes six vertices to any induced bipartite subgraph, then:

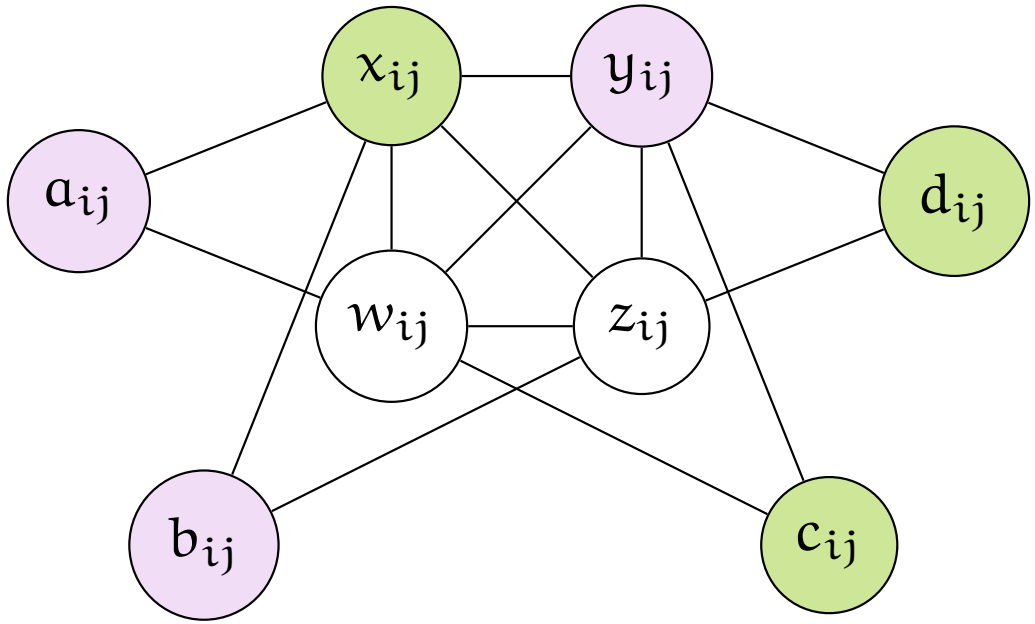
A valid contribution, therefore, is either …



The Composition

If the gadget contributes six vertices to any induced bipartite subgraph, then:

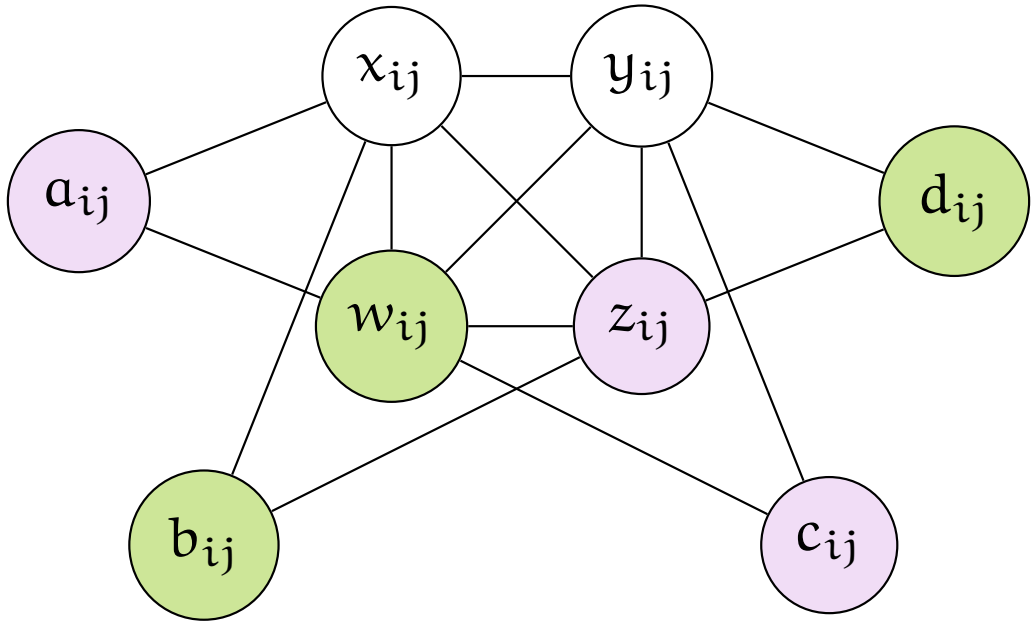
A valid contribution, therefore, is either this



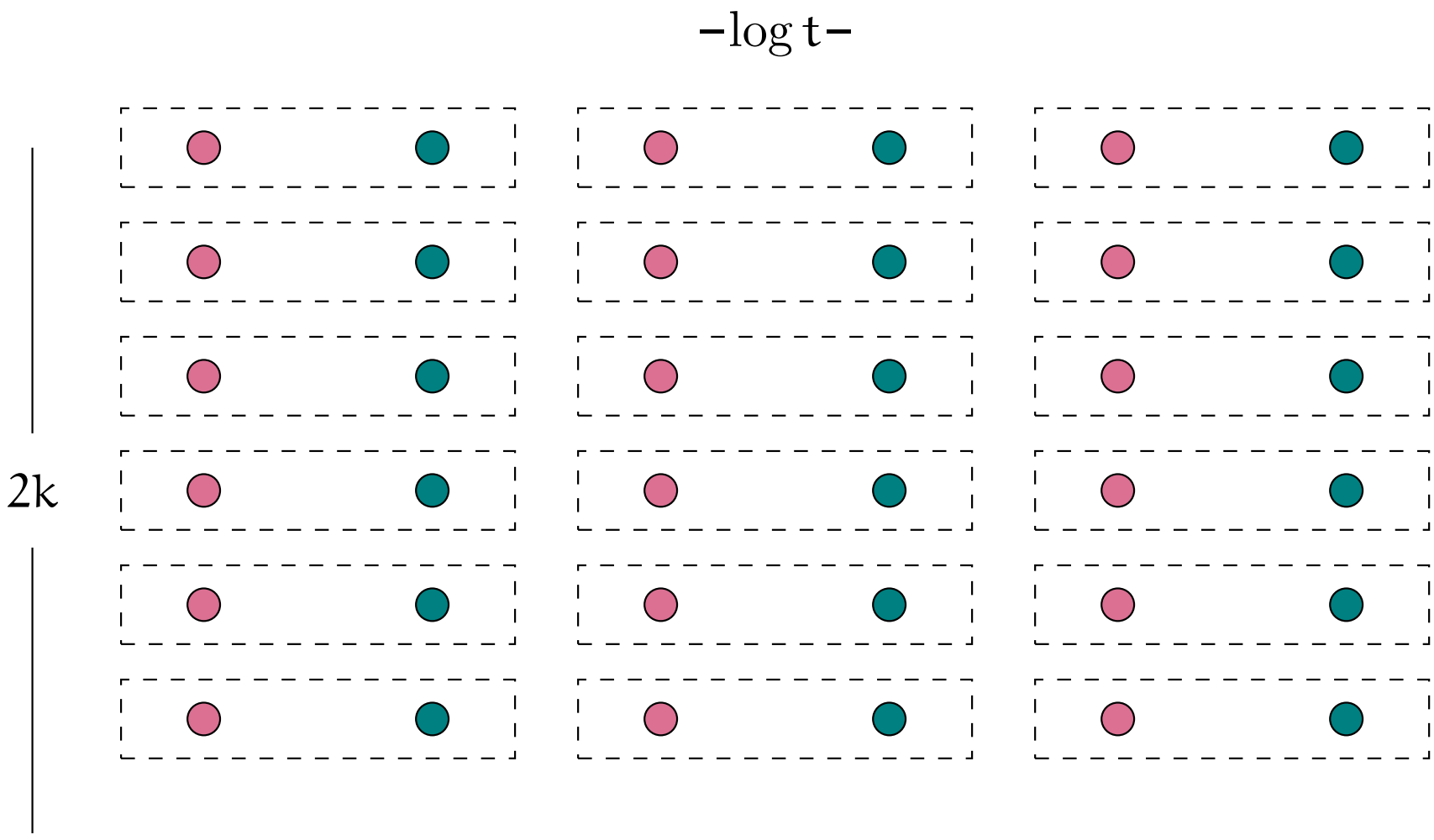
The Composition

If the gadget contributes six vertices to any induced bipartite subgraph, then:

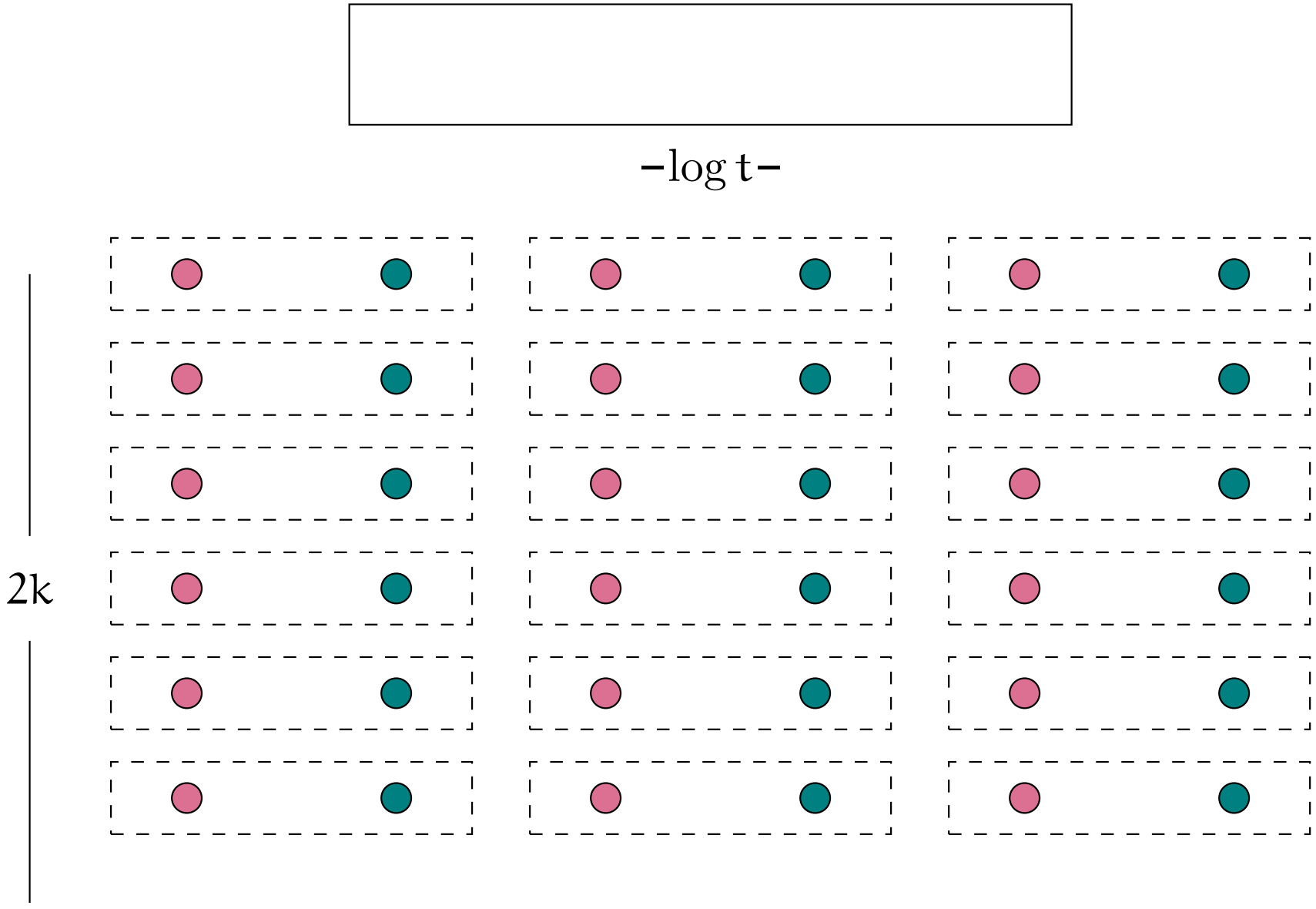
A valid contribution, therefore, is either or this:



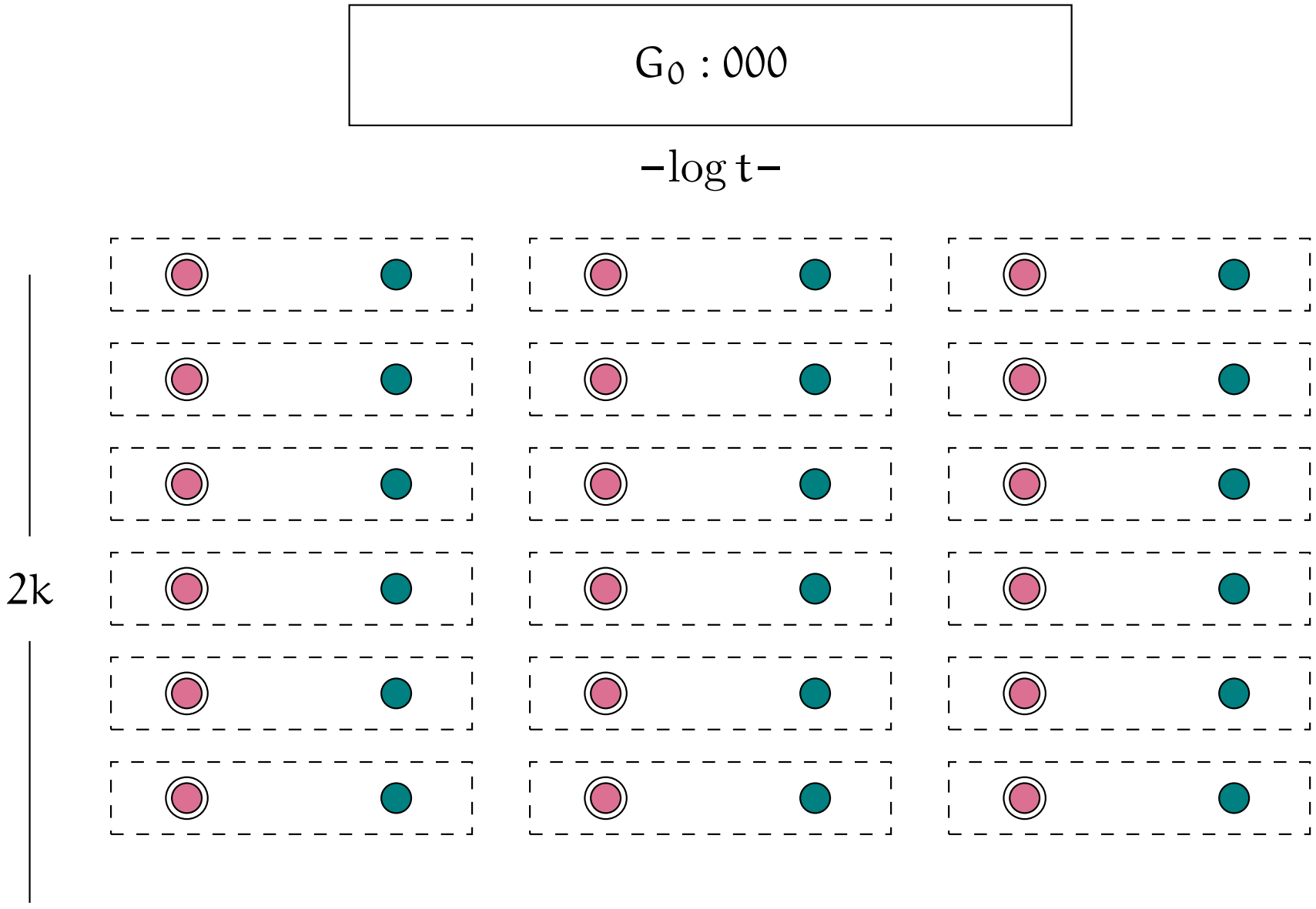
The Composition



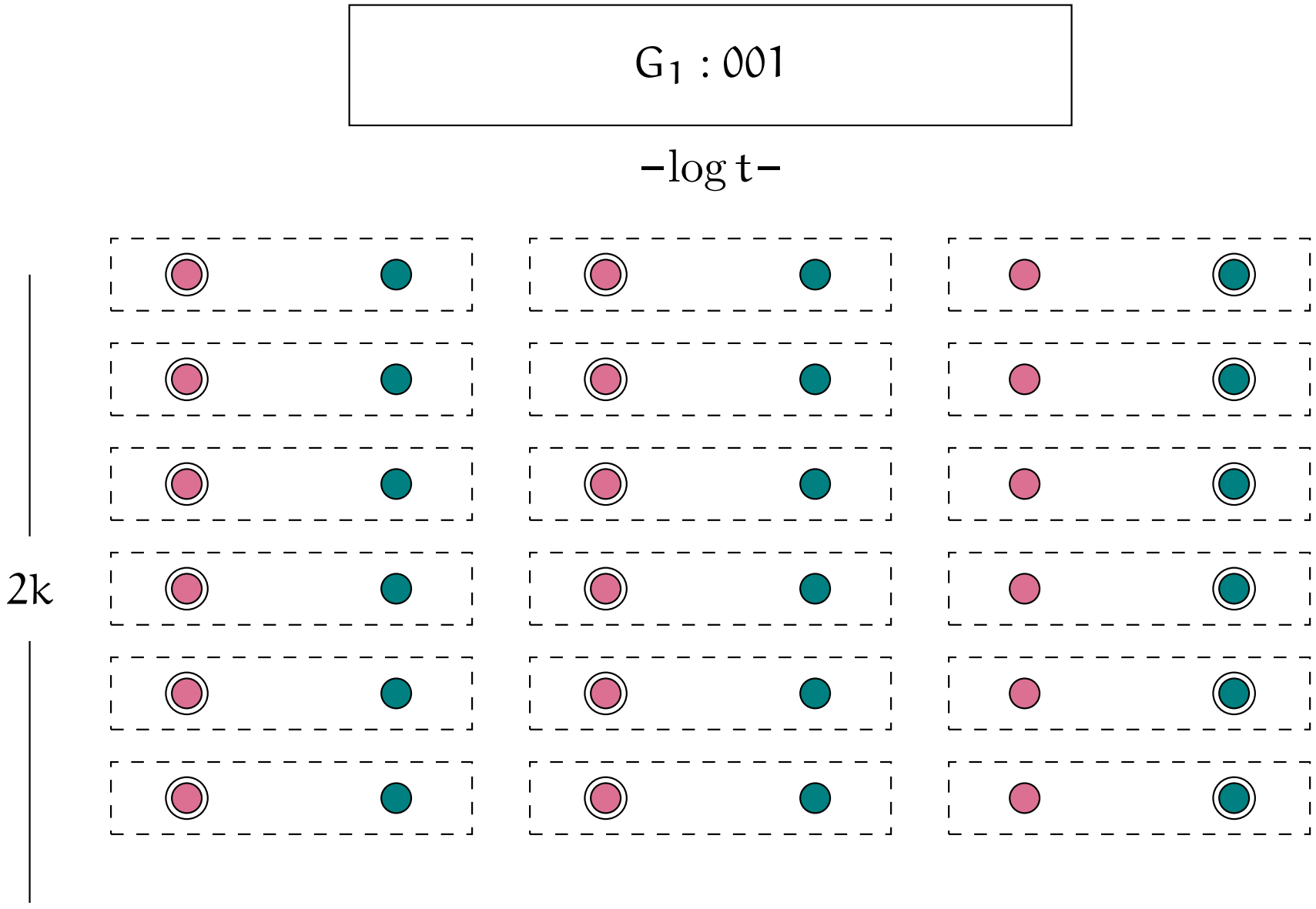
The Composition



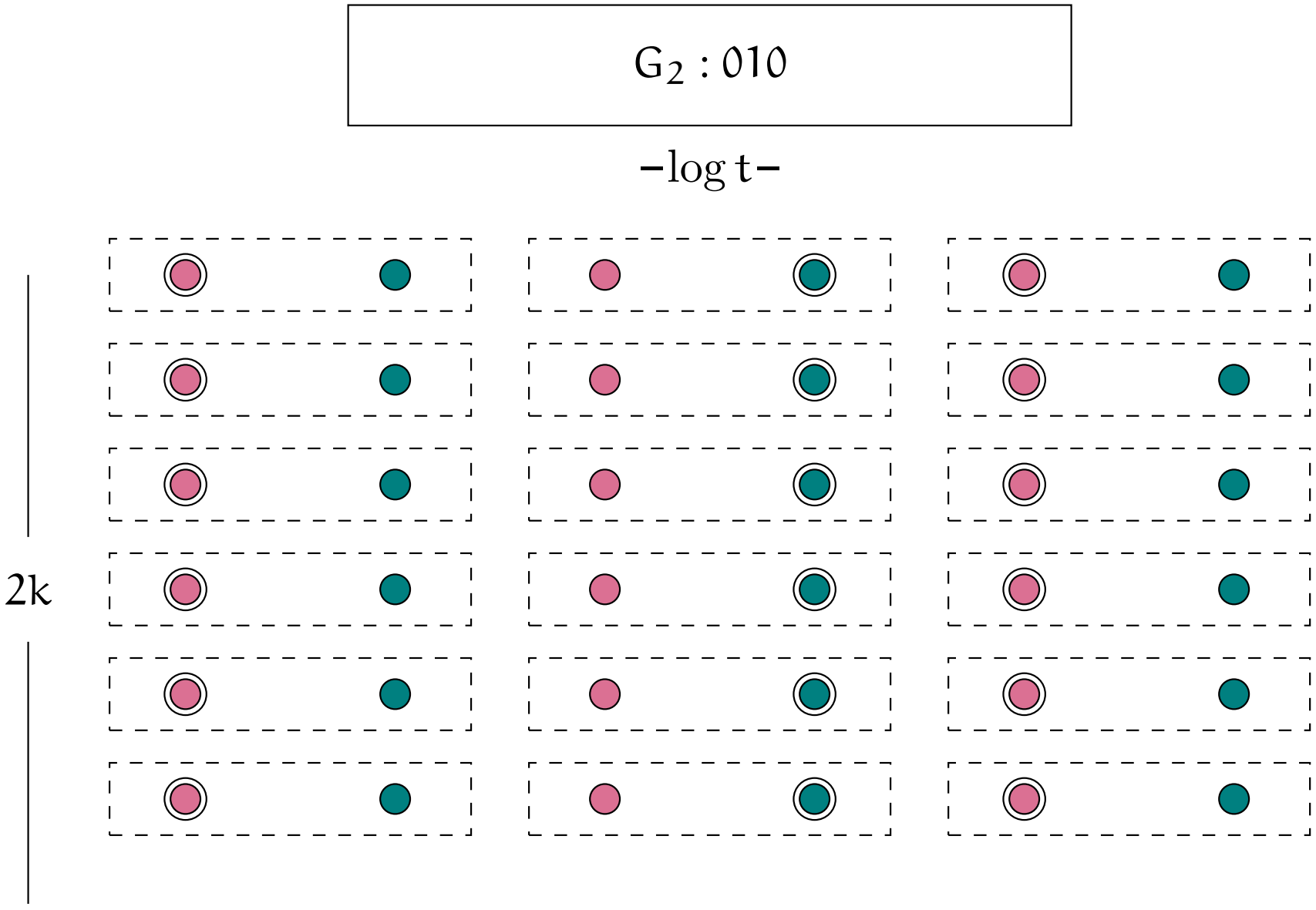
The Composition



The Composition



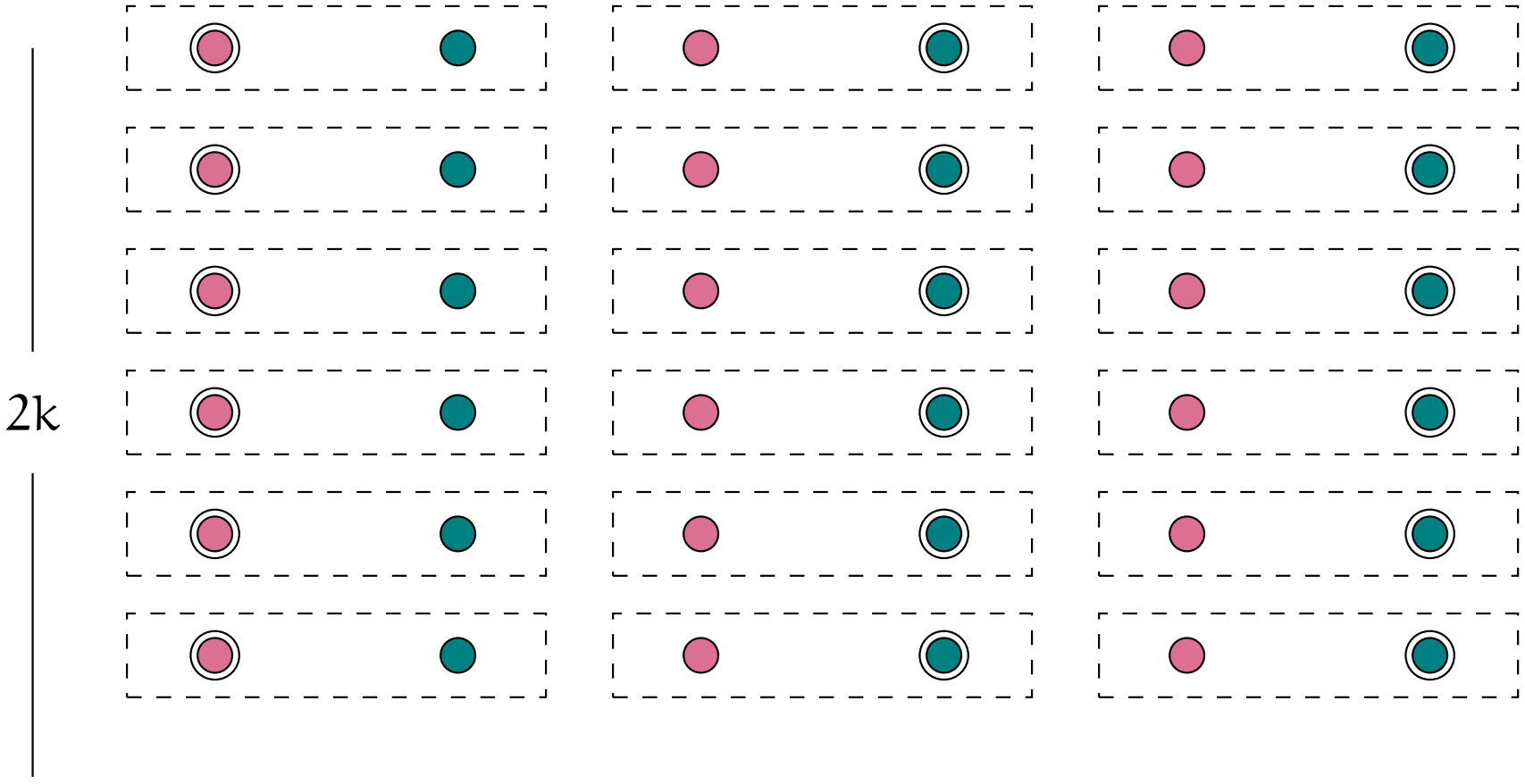
The Composition



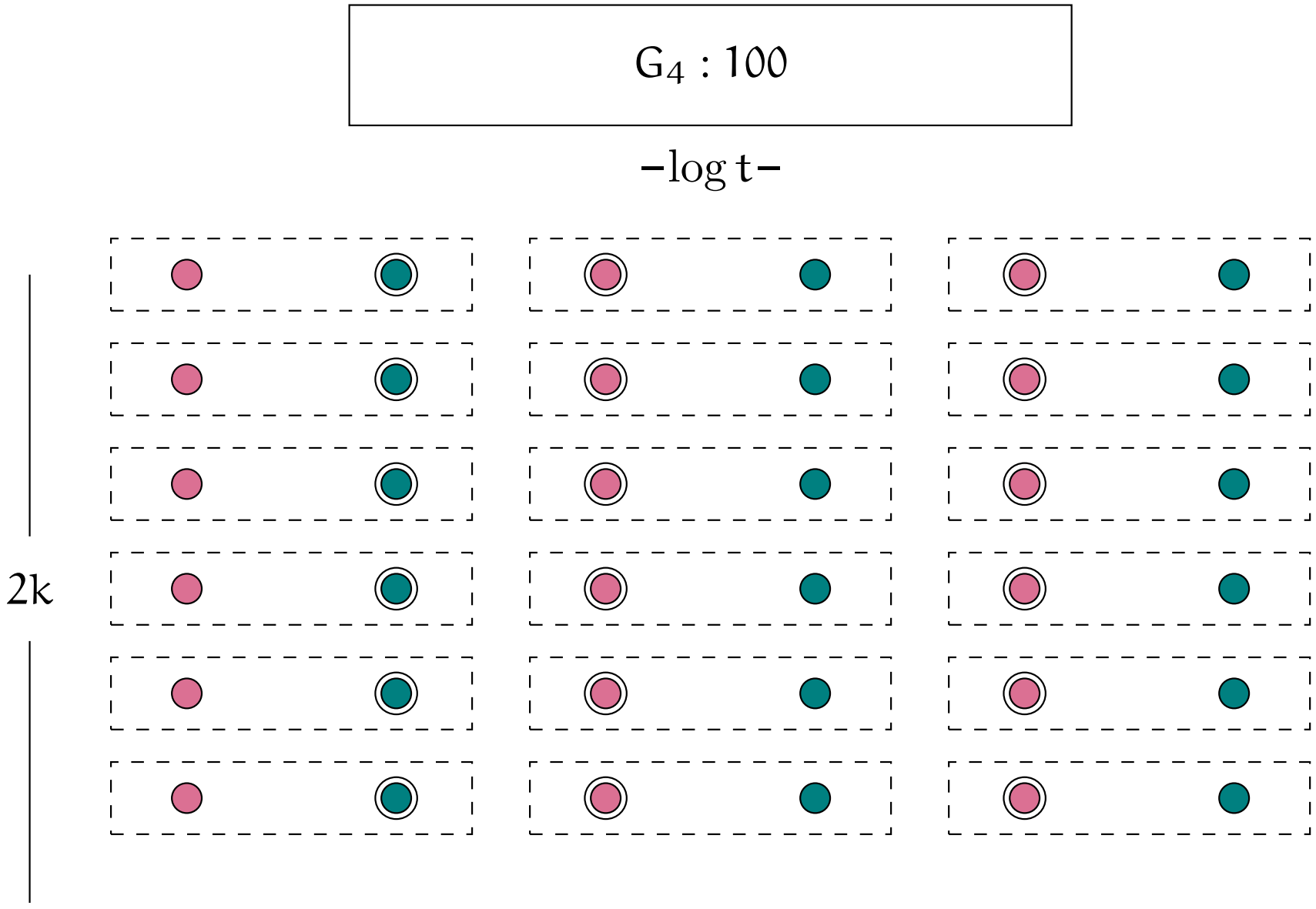
The Composition

$G_3 : 011$

$-\log t-$



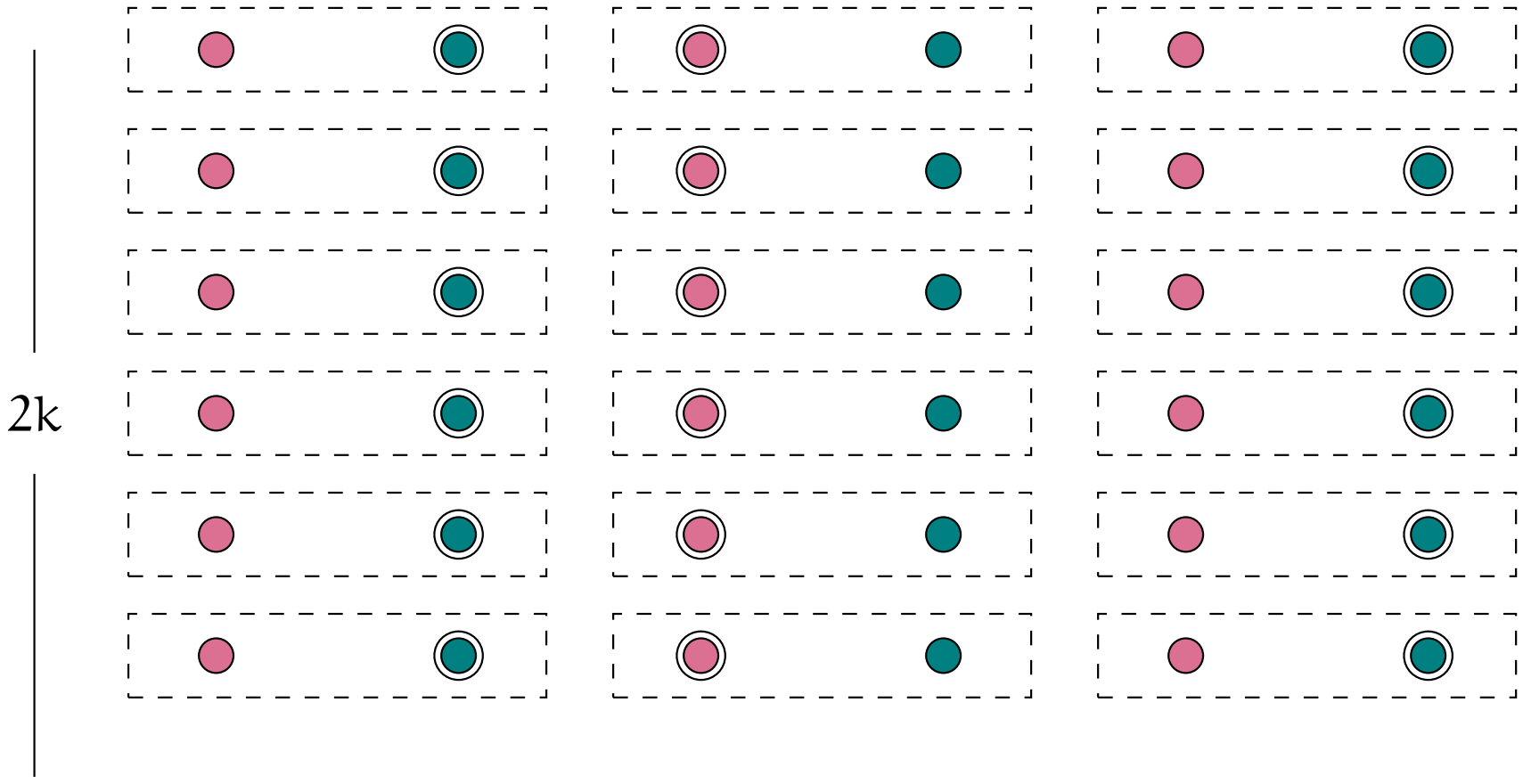
The Composition



The Composition

$G_5 : 101$

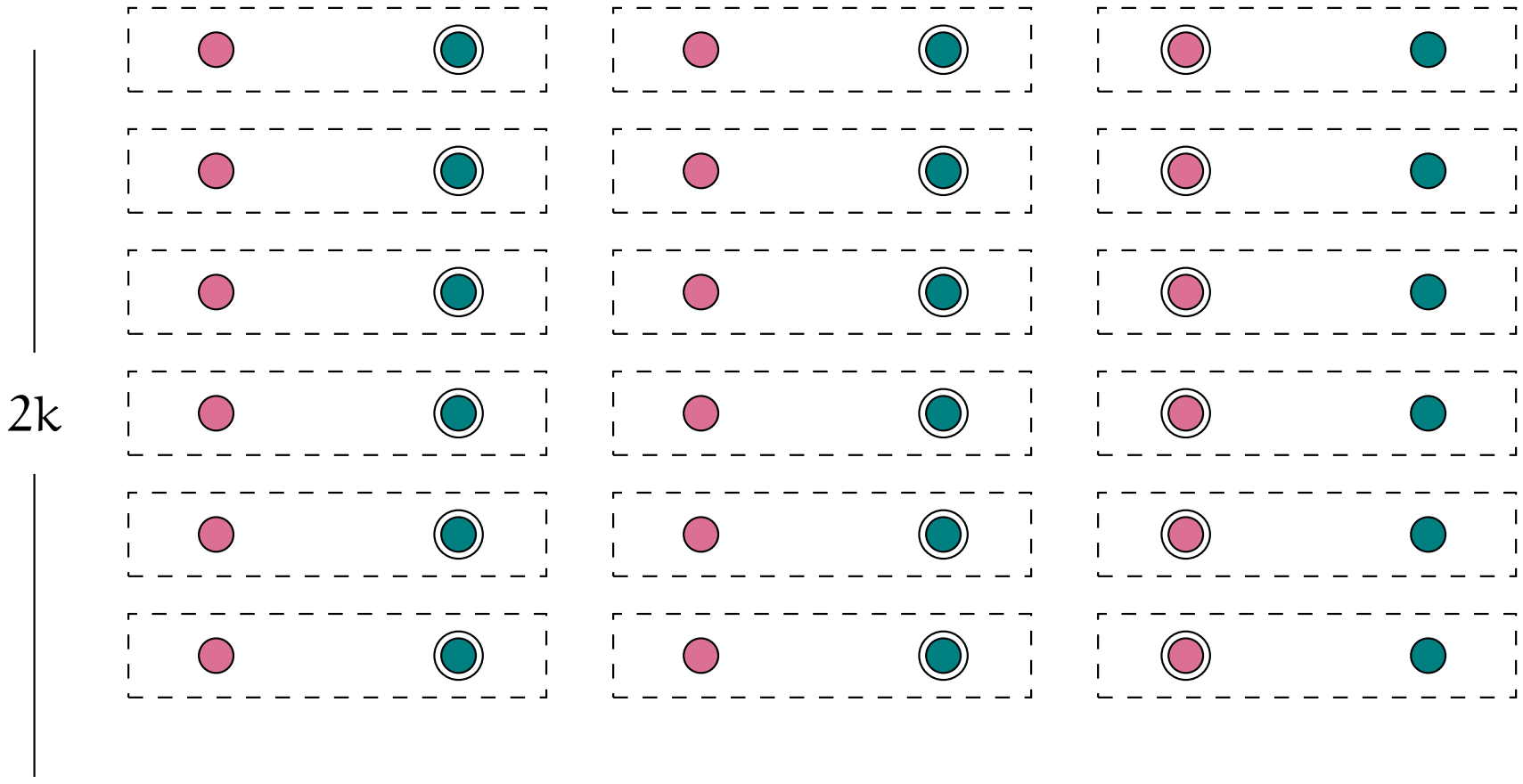
$-\log t-$



The Composition

$G_6 : 110$

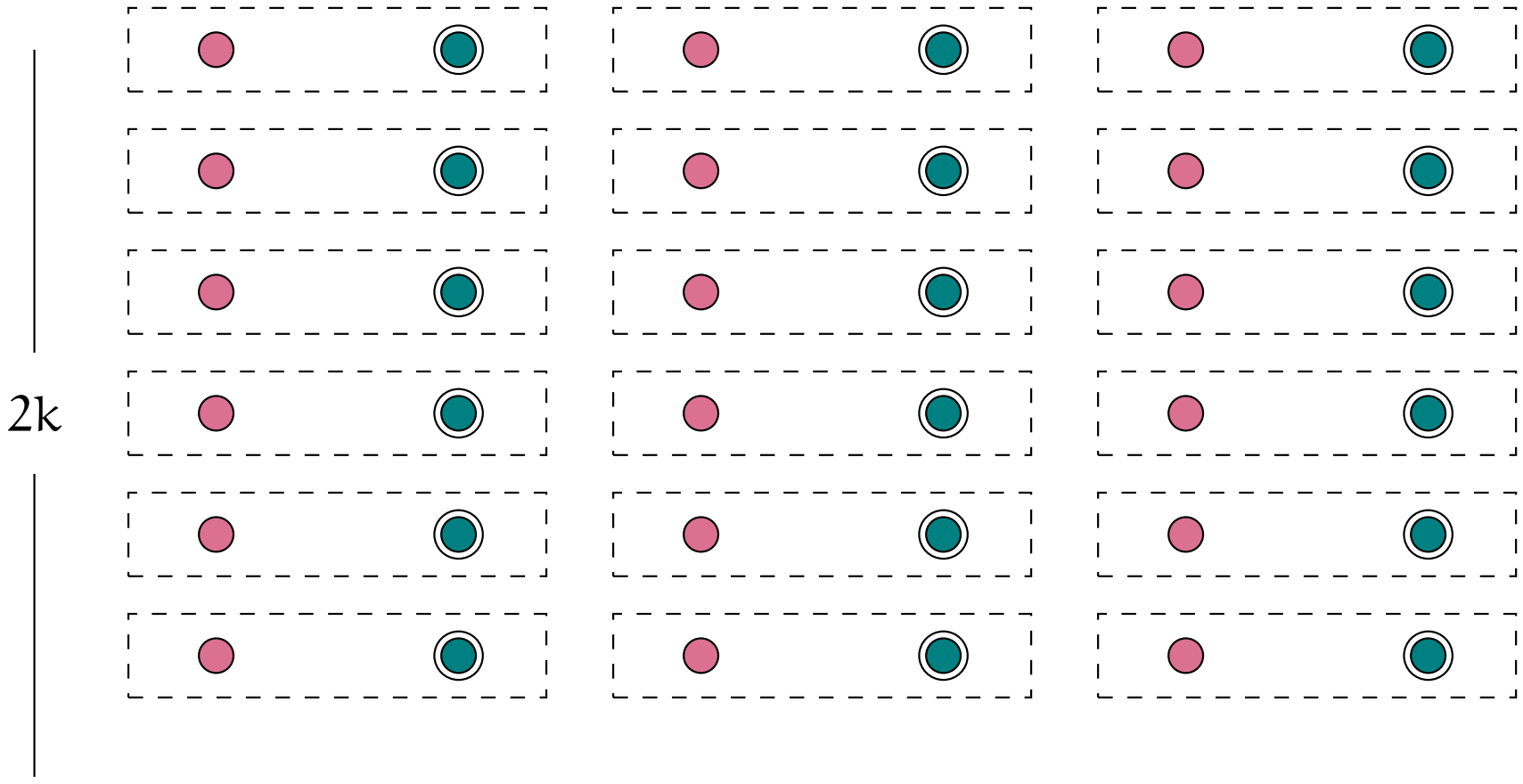
$-\log t-$



The Composition

$G_7 : 111$

$-\log t-$



$$k \longrightarrow k + 12k \cdot \log t$$

$$k \longrightarrow k + 12k \cdot \log t$$

...can be shown to be a polynomial in k without loss of generality.

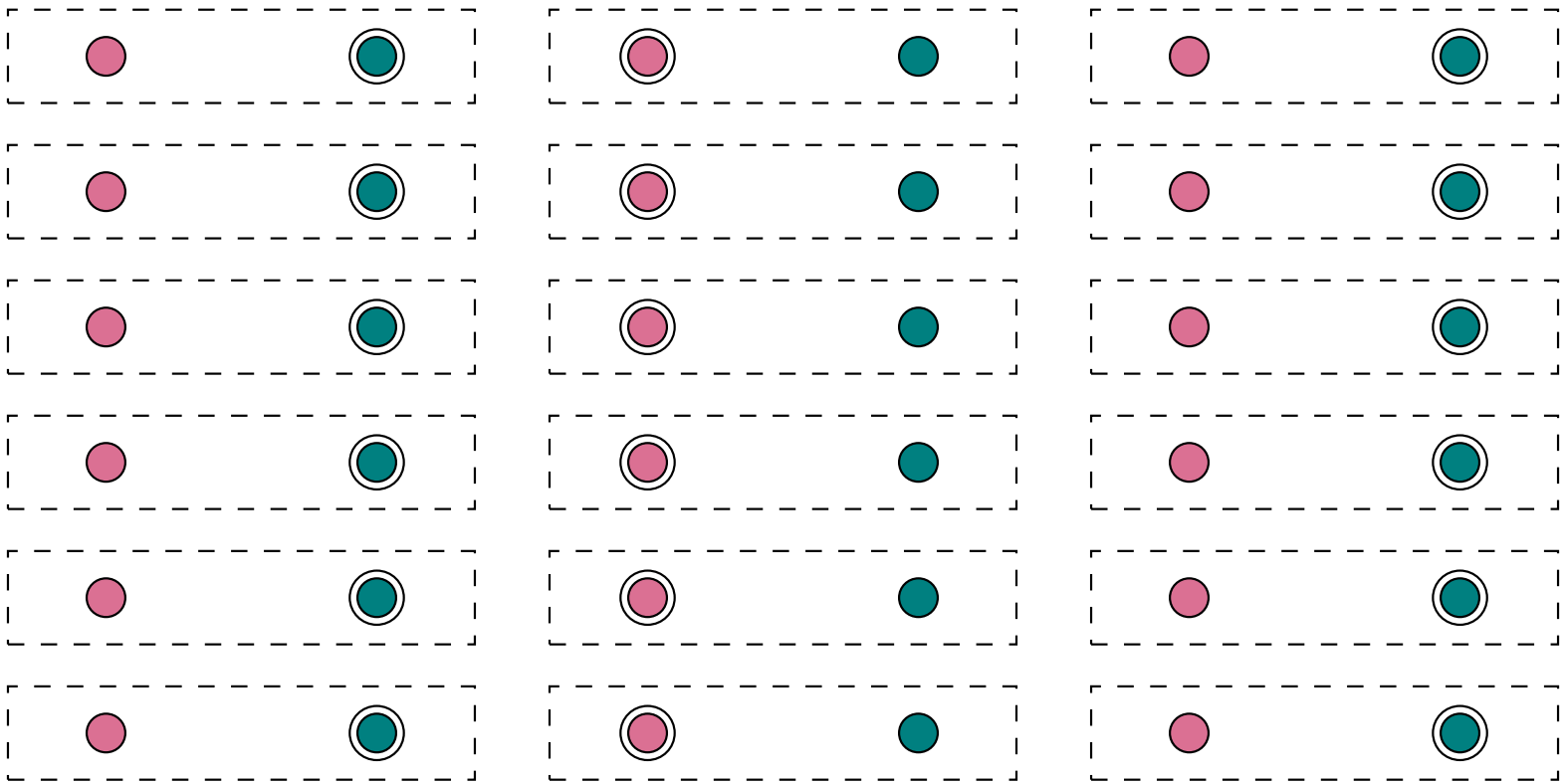
The Forward Direction

Suppose some G_i has a bipartite subgraph on k vertices.

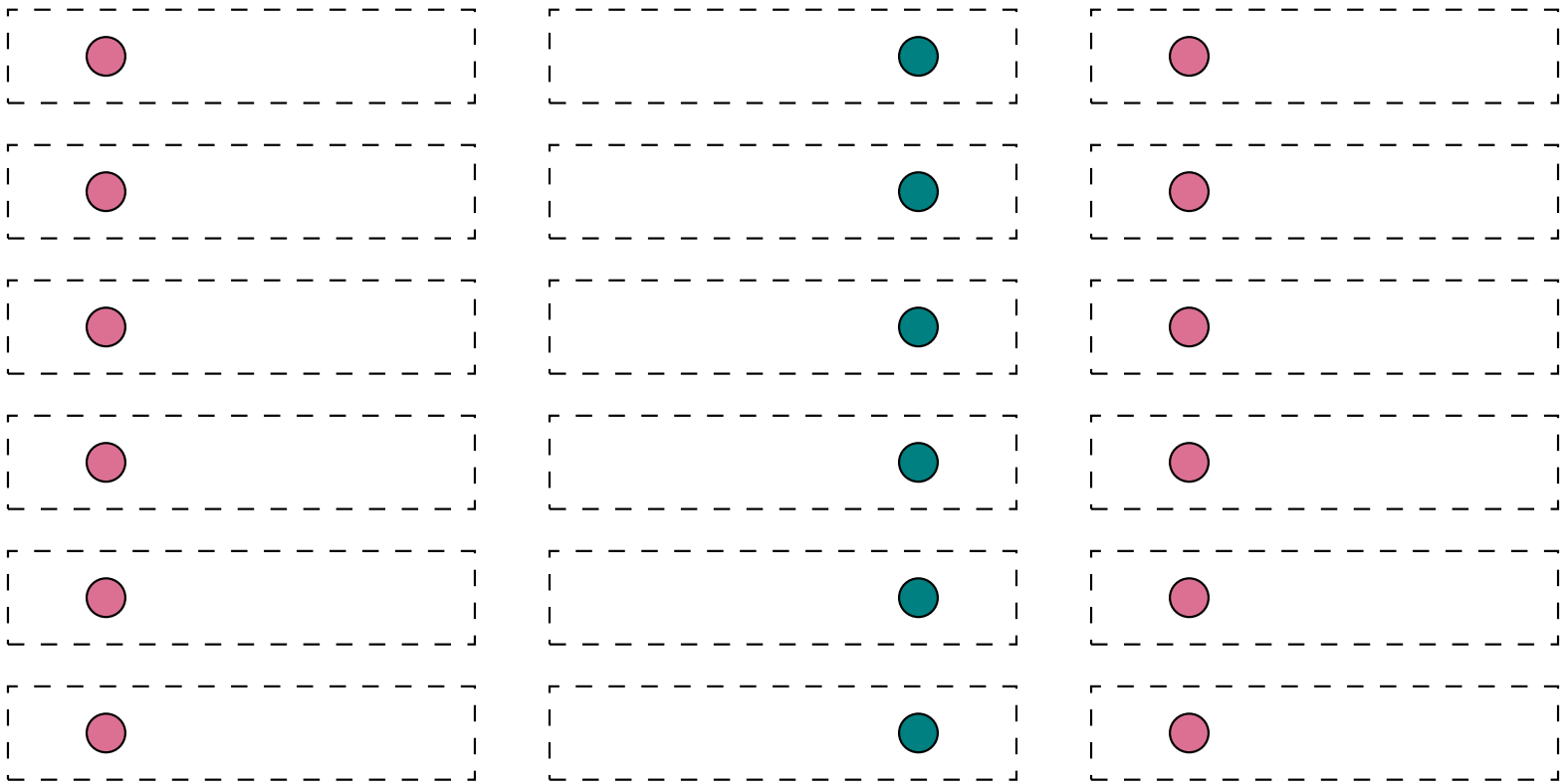
Suppose some G_i has a bipartite subgraph on k vertices.

Consider the binary representation of i. Pick the six vertices from the $2k \log t$ gadgets that G_i is non-adjacent to.

$G_5 : 101$



$G_5 : 101$



The Reverse Direction

Suppose the composed instance has an induced bipartite subgraph S on at least $k + 12k \cdot \log t$ vertices.

Suppose the composed instance has an induced bipartite subgraph S on at least $k + 12k \cdot \log t$ vertices.

Recall that each of the $(2k \log t)$ gadgets can contribute at most six vertices each.

Suppose the composed instance has an induced bipartite subgraph S on at least $k + 12k \cdot \log t$ vertices.

Recall that each of the $(2k \log t)$ gadgets can contribute at most six vertices each.

Consider:

$$(S \cap G_i) \text{ for } 1 \leqslant i \leqslant t.$$

Suppose the composed instance has an induced bipartite subgraph S on at least $k + 12k \cdot \log t$ vertices.

Recall that each of the $(2k \log t)$ gadgets can contribute at most six vertices each.

Consider:

$$(S \cap G_i) \text{ for } 1 \leqslant i \leqslant t.$$

If $S \cap G_i$ is non-empty for exactly one instance G_i, then we are done, because then we automatically have that $|S \cap G_i| \geqslant k$.

We also know that S cannot be shared by three of the instances, since that would induce a triangle.

We also know that S cannot be shared by three of the instances, since that would induce a triangle.

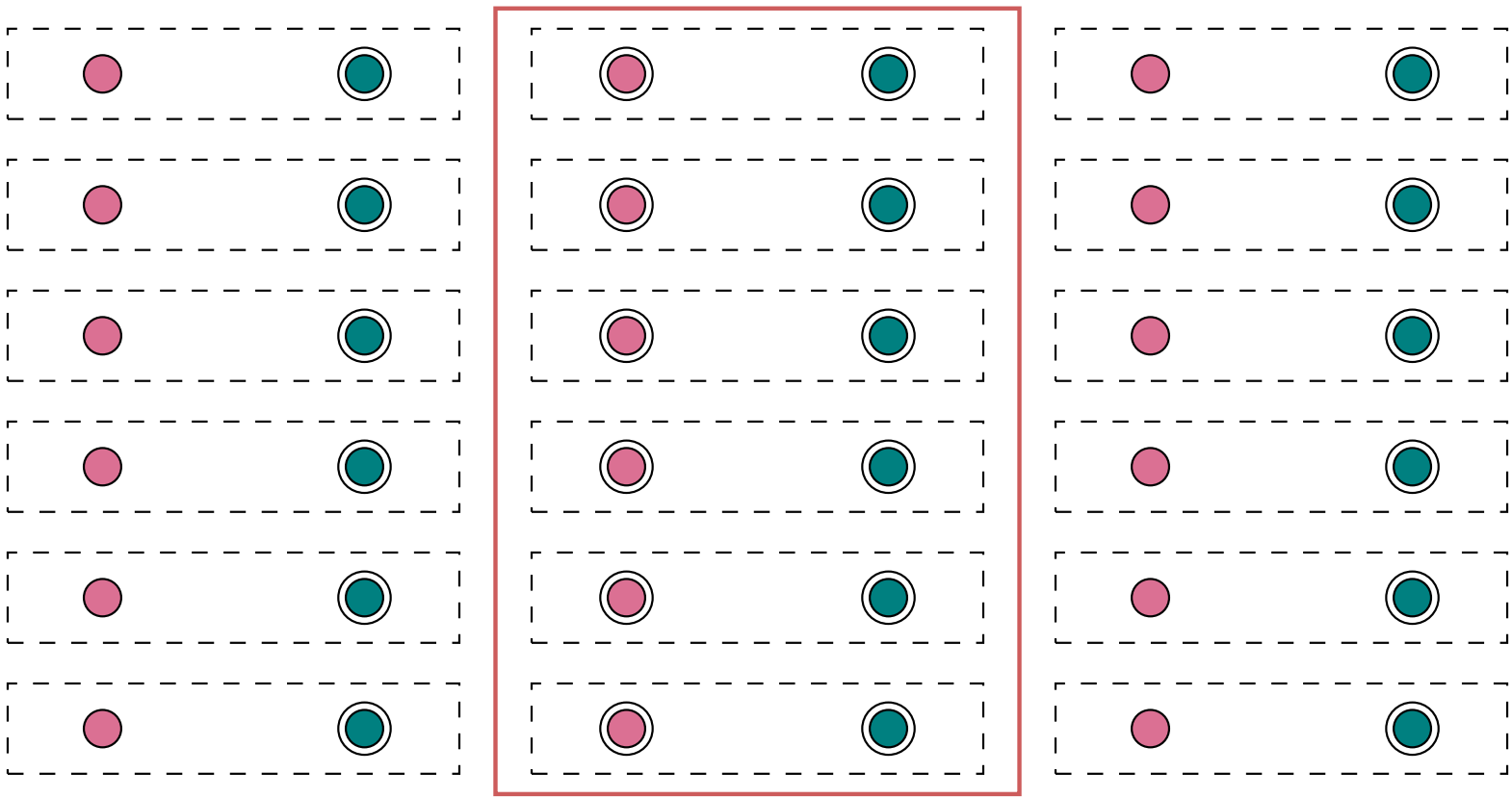
Let us now address the only remaining case: what if S intersects G_i and G_j non-trivially, for some $1 \leqslant i \neq j \leqslant t$?

If $i \neq j$, then the bit vectors corresponding to i and j are also different.

If $i \neq j$, then the bit vectors corresponding to i and j are also different.

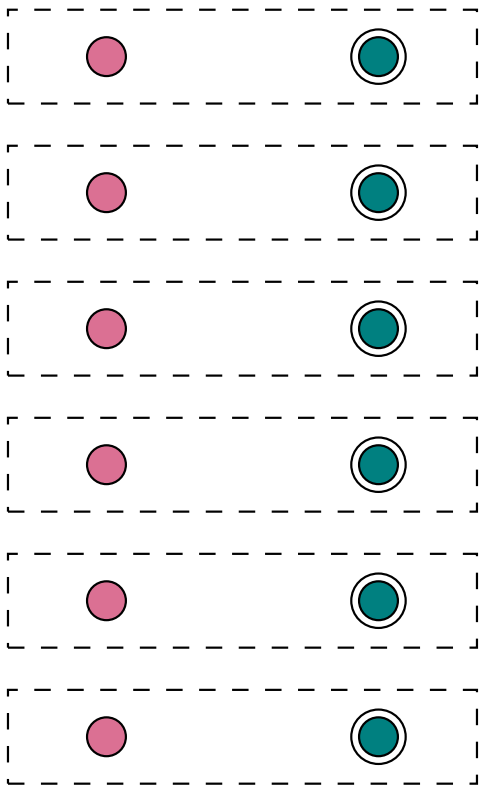
Let b be an index ($1 \leqslant b \leqslant \log t$) where i and j differ.

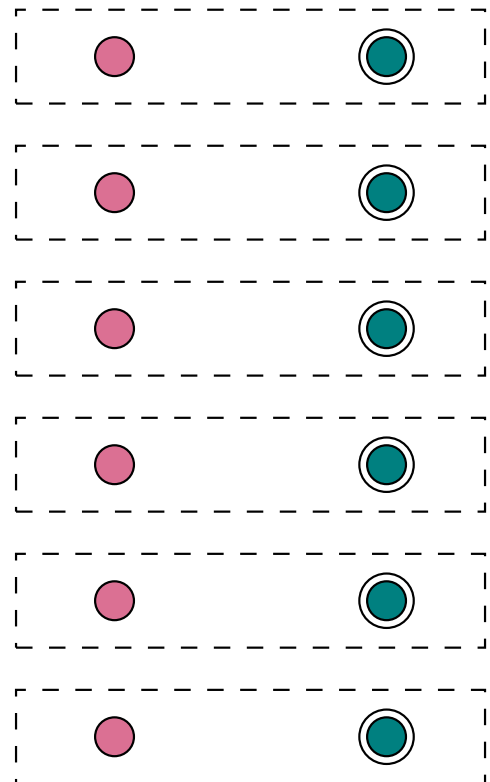
$G_5 : 101$ and $G_7 : 111$



These $2k$ gadgets, corresponding to the index b, cannot contribute to S at all, being global to vertices in G_i and G_j.

$G_5 : 101$ and $G_7 : 111$





The remaining gadgets can contribute at most:

$$[2k \cdot (\log t - 1)]6 + 4(2k) = 12k \log t - 4k.$$

The remaining gadgets can contribute at most:

$$[2k \cdot (\log t - 1)]6 + 4(2k) = 12k \log t - 4k.$$

Thus G_i and G_j together must account for at least $5k$ vertices between them.

The remaining gadgets can contribute at most:

$$[2k \cdot (\log t - 1)]6 + 4(2k) = 12k \log t - 4k.$$

Thus G_i and G_j together must account for at least $5k$ vertices between them.

This implies that at least one of them must be contributing at least k vertices, and this leads us to our conclusion.

Danke!