



Staged Self-Assembly

Assembly of Arbitrary Shapes with $O(1)$ Glues

Mashhood Ishaque

Research Talk

6th Dec, 2006

Advisor: Diane Souvaine

Committee: Judith Stafford, Lenore Cowen



Acknowledgement

Joint work with

- Erik D. Demaine (MIT CSAIL)
- Martin L. Demaine (MIT CSAIL)
- Sándor P. Fekete (Technische Universität Braunschweig)
- Eynat Rafalin (Google)
- Robert T. Schweller (Northwestern University)
- Diane L. Souvaine (Tufts University)

Useful discussions about tail fibers with

- Edward Goldberg (Sackler Biomedical School)
 - Timothy Harrah (Sackler Biomedical School)
-



Outline

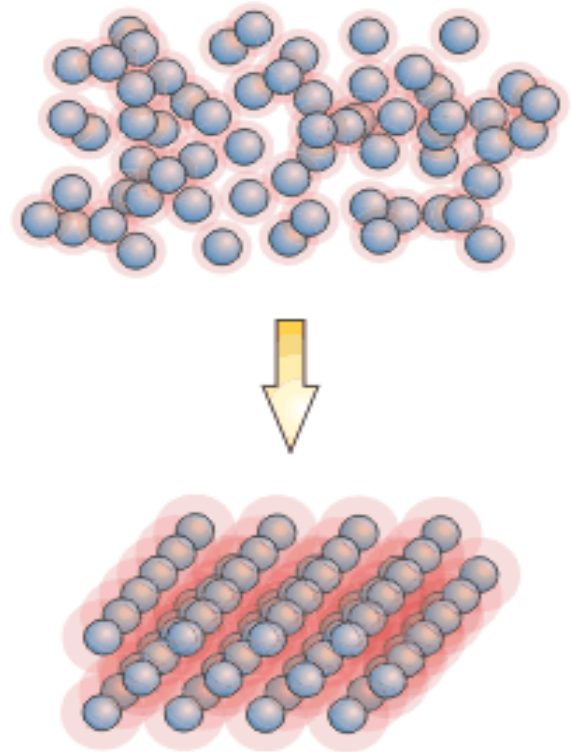
- Self-Assembly
 - Tile Model of Self-Assembly
 - Staged Self-Assembly
 - Assembly of $1 \times N$ Line
 - Assembly of $N \times N$ Square
 - Assembly of Monotones
 - Triangular Tile Model
 - Future Work
-



Self-Assembly

Self-assembly is the process in which simple parts **self-organize** into larger structures.

No central control

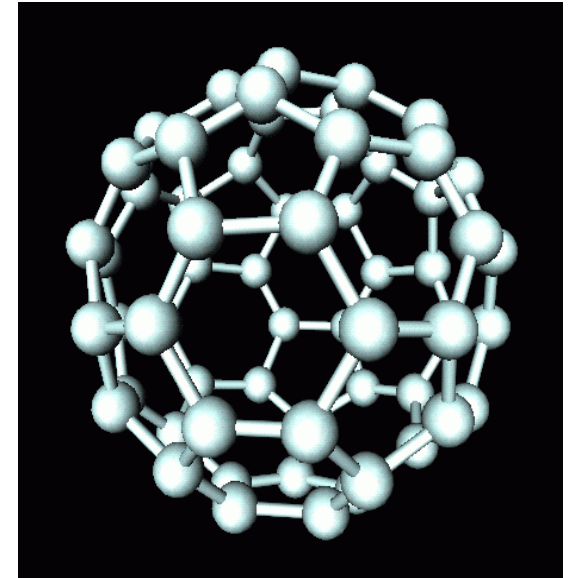




Why Self-Assembly?

Nature creates structures
by self-assembly:
crystals, DNA helices etc.

Physics at nanoscopic
level make centralized
control impractical.

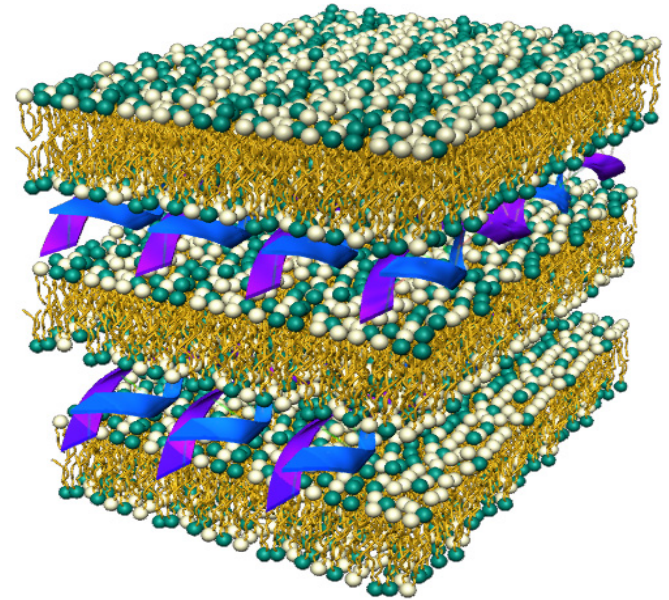


C_{60} , the Buckyball
Carbon nanotube



Applications of Self-Assembly

- Nano circuits.
- Sieve for removing viruses from serum.
- Tiny sensors that float in the blood stream (require biomaterial).
- Targeted drug delivery mechanism (minimize side-effects).

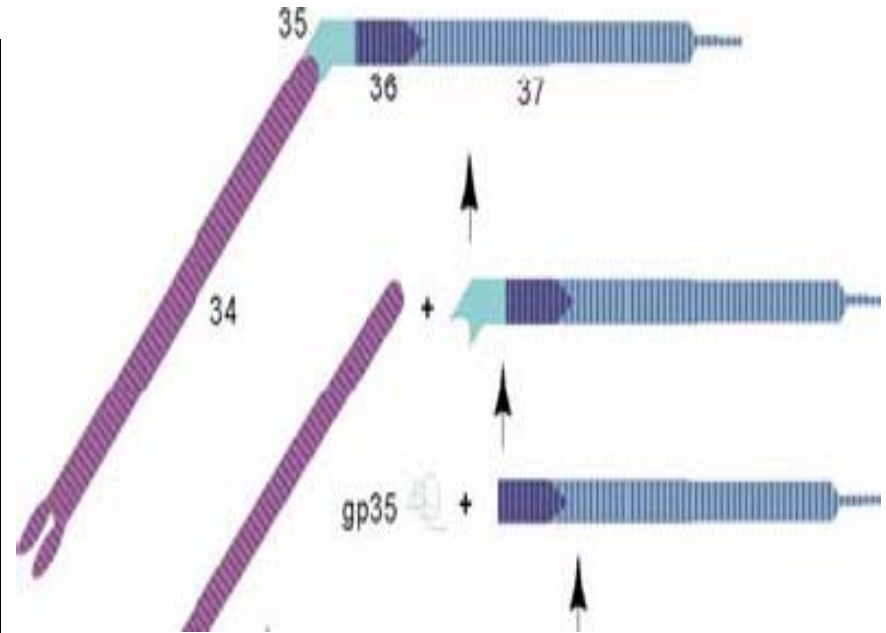
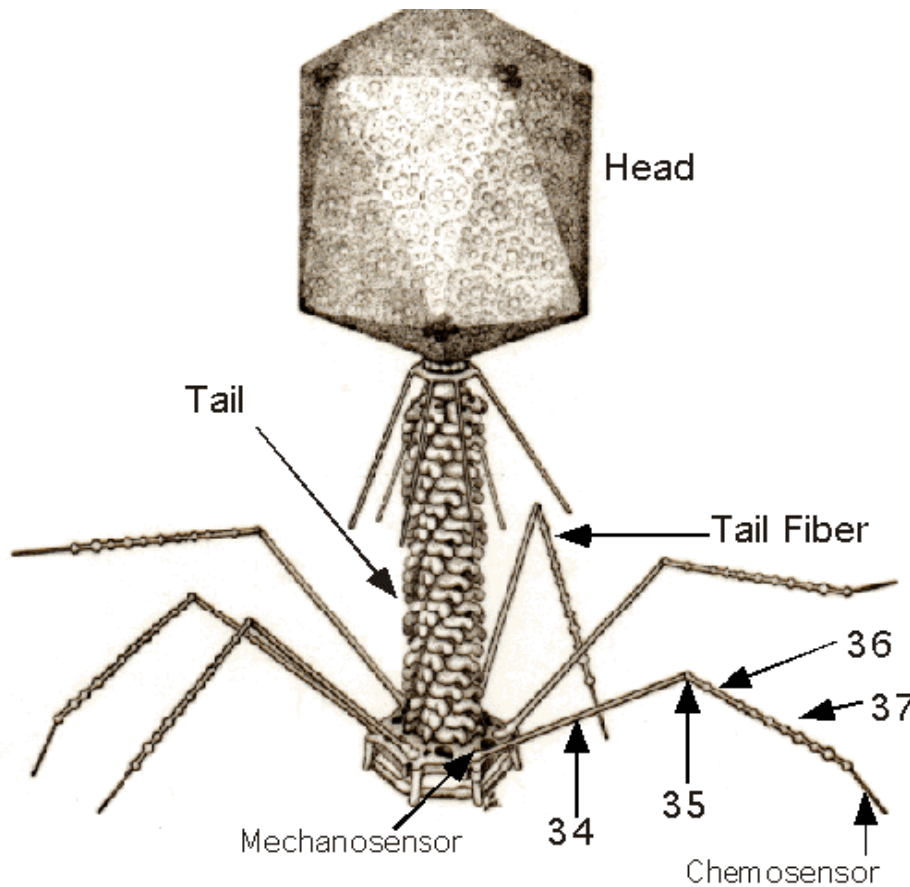


A DNA-membrane complex used as delivery vehicle in gene therapy.*

* Gerard C. L. Wong, Youli Li, Ilya Koltover, Cyrus R. Safinya, Zhonghou Cai, and Wenbing Yun in the October 5, 1998 issue of Applied Physics Letters.



Building Block for Self-Assembly



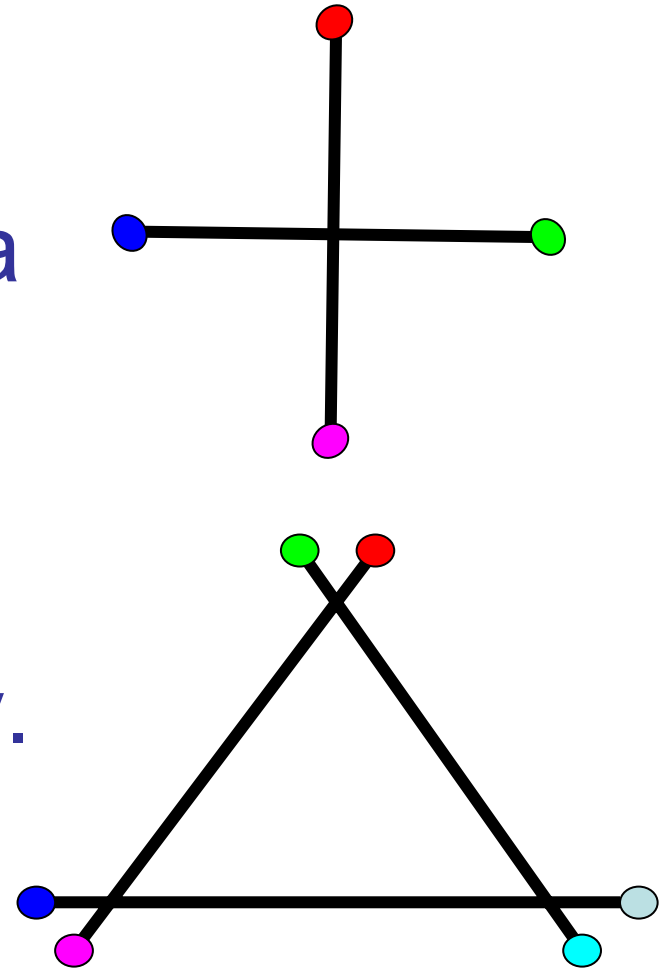
Bacteriophage T4 tail fiber can be (genetically) engineered into a rigid nano-rod*.

*Goldberg Laboratory, Sackler School of Graduate Biomedical Sciences.



Building Block for Self-Assembly

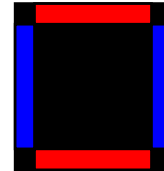
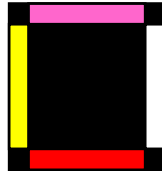
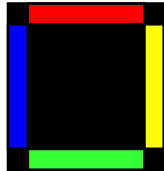
These rods can be cross-linked to create a **crossbar** or an **equilateral triangle**, to serve as a building block for self-assembly.





Crossbar as Wang Tile

The crossbar can be abstracted as a **Wang tile**, which is a four-sided tile with a specific color (glue) on each side.





Tile Model of Self-Assembly

Tile system consists of four pieces:

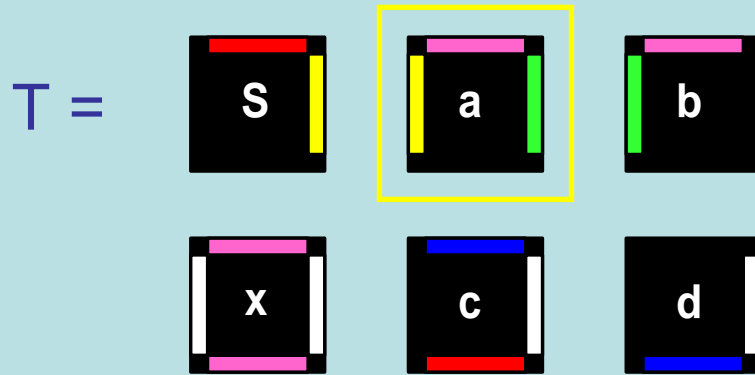
- tile set: $\left\{ \begin{array}{c} \text{red} \\ \text{black} \\ \text{yellow} \\ \text{black} \end{array} \begin{array}{c} \text{r} \\ \text{s} \\ \text{y} \\ \text{b} \end{array} \right\}, \left\{ \begin{array}{c} \text{pink} \\ \text{black} \\ \text{red} \\ \text{black} \end{array} \begin{array}{c} \text{p} \\ \text{w} \\ \text{r} \\ \text{b} \end{array} \right\}, \left\{ \begin{array}{c} \text{red} \\ \text{black} \\ \text{blue} \\ \text{black} \end{array} \begin{array}{c} \text{r} \\ \text{s} \\ \text{b} \\ \text{b} \end{array} \right\}, \dots \left. \right\}$

- seed tile:

	r	g	b	y	p	w
r	5	1	2	4	3	1
g	1	4	1	0	0	2
b	2	1	7	0	2	0
y	4	0	0	6	1	1
p	3	0	2	1	4	0
w	1	2	0	1	0	7

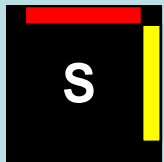
- temperature threshold: $t = 3$

How a tile system self-assembles



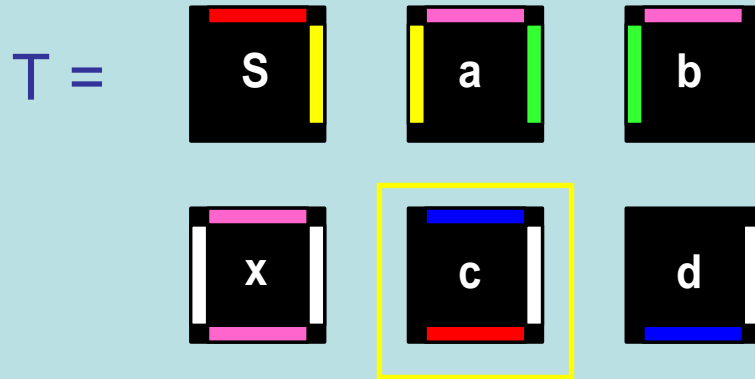
	r	g	b	y	p	w
r	2	0	0	0	0	0
g	0	2	0	0	0	0
b	0	0	2	0	0	0
y	0	0	0	2	0	0
p	0	0	0	0	1	0
w	0	0	0	0	0	1

$t = 2$



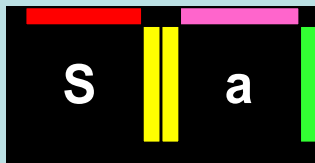
Tiles do not rotate or flip.

How a tile system self-assembles



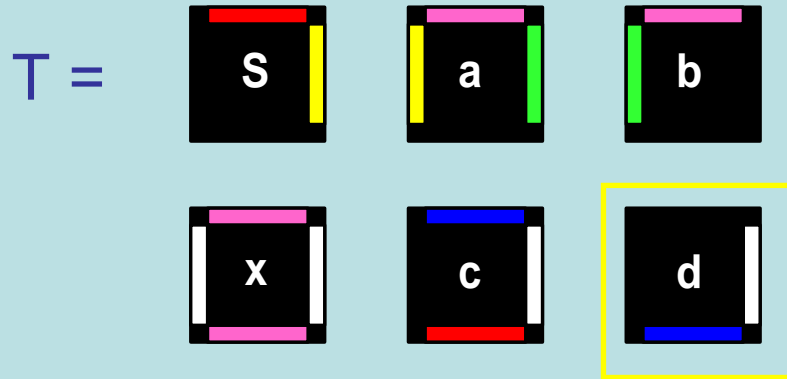
	r	g	b	y	p	w
r	2	0	0	0	0	0
g	0	2	0	0	0	0
b	0	0	2	0	0	0
y	0	0	0	2	0	0
p	0	0	0	0	1	0
w	0	0	0	0	0	1

t = 2



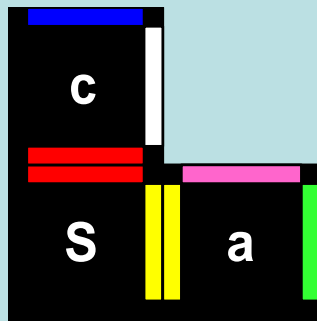
Tiles do not rotate or flip.

How a tile system self-assembles



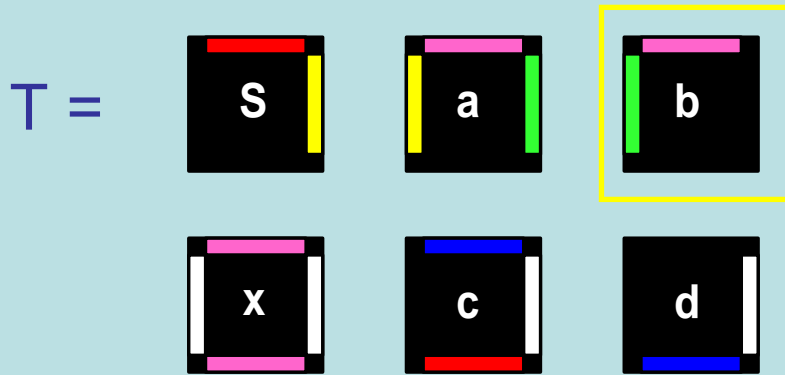
	r	g	b	y	p	w
r	2	0	0	0	0	0
g	0	2	0	0	0	0
b	0	0	2	0	0	0
y	0	0	0	2	0	0
p	0	0	0	0	1	0
w	0	0	0	0	0	1

$t = 2$



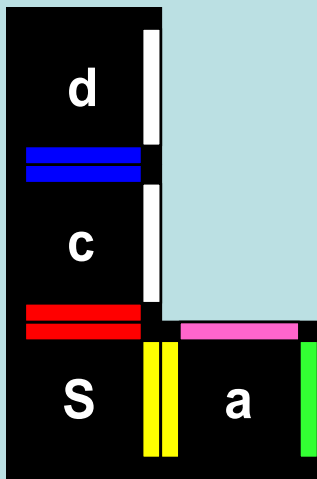
Tiles do not rotate or flip.

How a tile system self-assembles



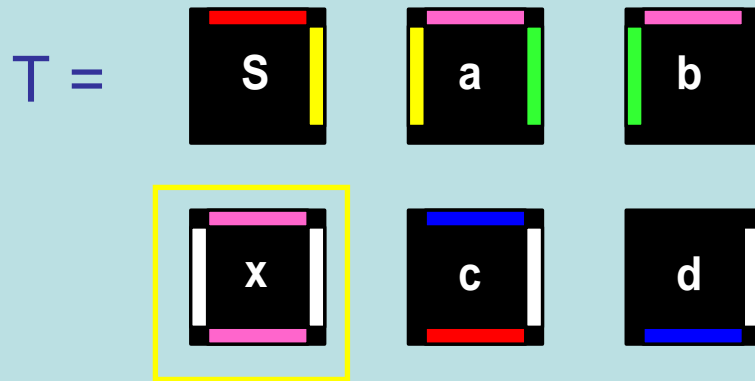
	r	g	b	y	p	w
r	2	0	0	0	0	0
g	0	2	0	0	0	0
b	0	0	2	0	0	0
y	0	0	0	2	0	0
p	0	0	0	0	1	0
w	0	0	0	0	0	1

t = 2



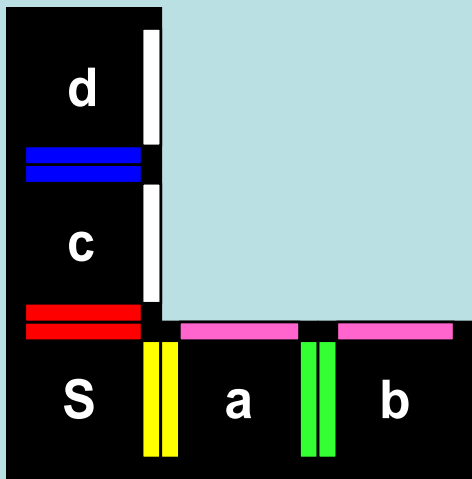
Tiles do not rotate or flip.

How a tile system self-assembles



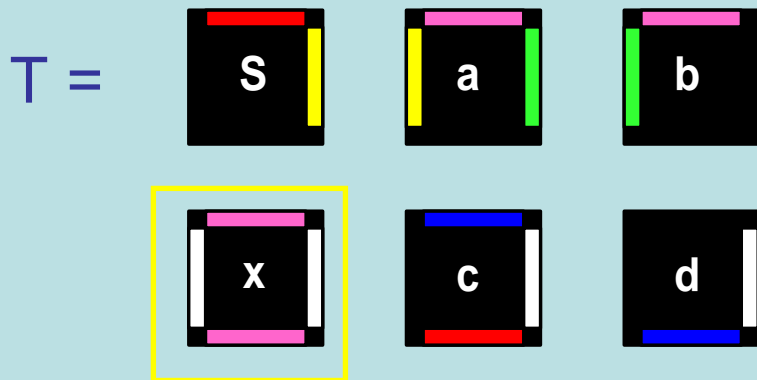
	r	g	b	y	p	w
r	2	0	0	0	0	0
g	0	2	0	0	0	0
b	0	0	2	0	0	0
y	0	0	0	2	0	0
p	0	0	0	0	1	0
w	0	0	0	0	0	1

$t = 2$



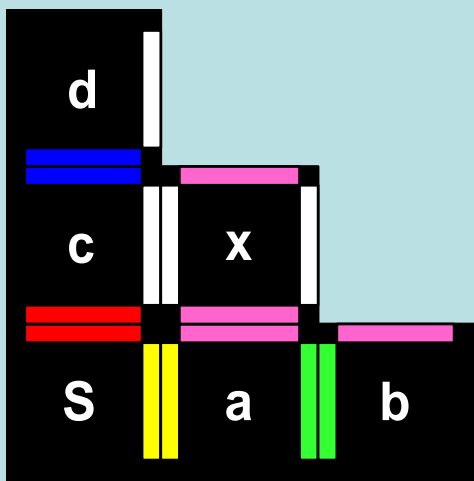
Tiles do not rotate or flip.

How a tile system self-assembles



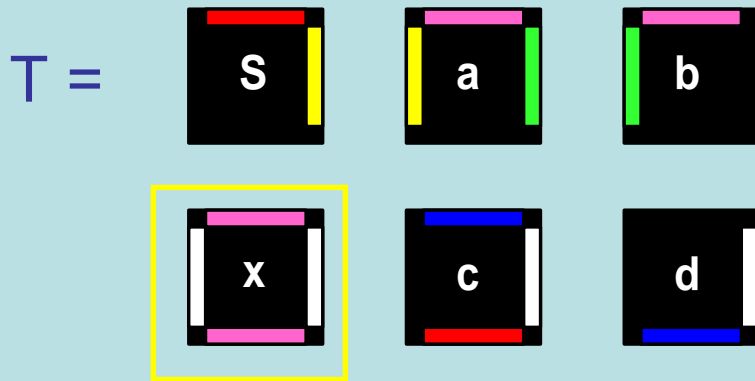
	r	g	b	y	p	w
r	2	0	0	0	0	0
g	0	2	0	0	0	0
b	0	0	2	0	0	0
y	0	0	0	2	0	0
p	0	0	0	0	1	0
w	0	0	0	0	0	1

$t = 2$



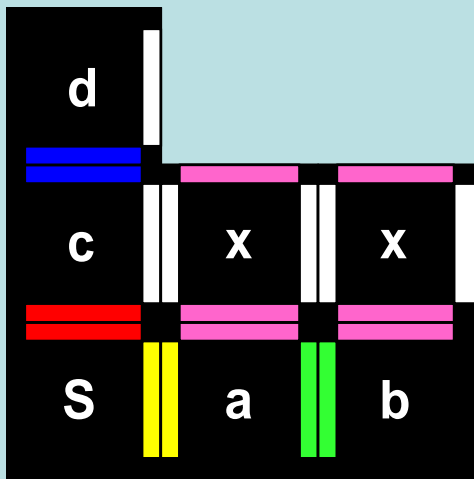
Tiles do not rotate or flip.

How a tile system self-assembles



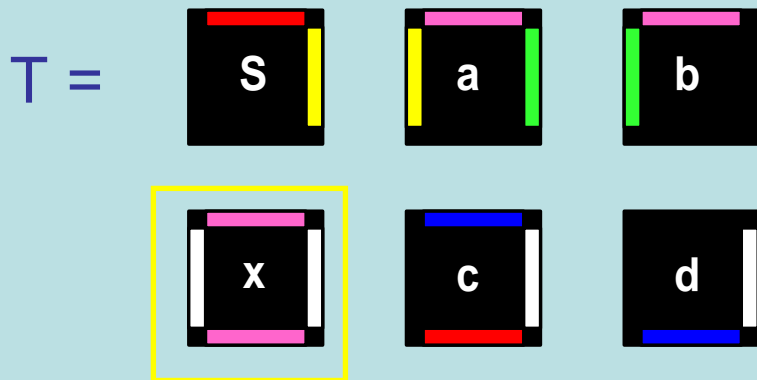
	r	g	b	y	p	w
r	2	0	0	0	0	0
g	0	2	0	0	0	0
b	0	0	2	0	0	0
y	0	0	0	2	0	0
p	0	0	0	0	1	0
w	0	0	0	0	0	1

$t = 2$



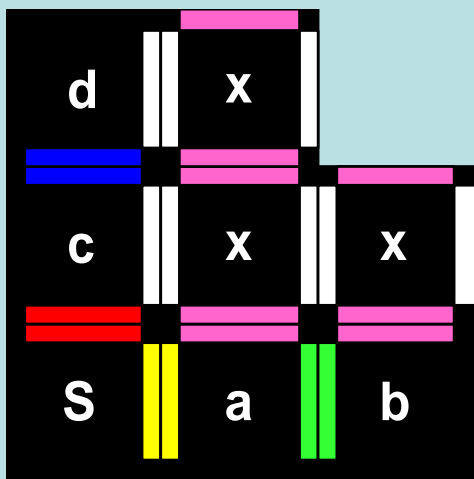
Tiles do not rotate or flip.

How a tile system self-assembles



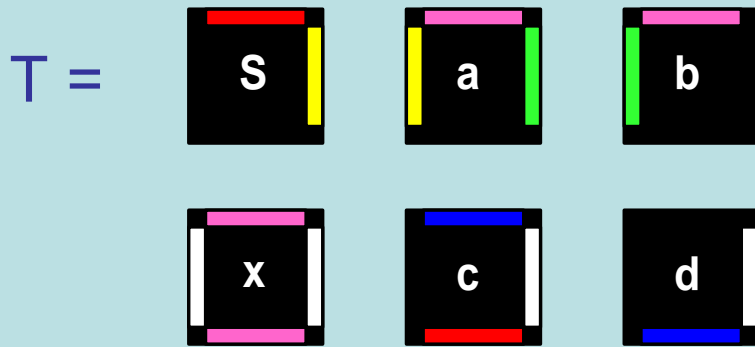
	r	g	b	y	p	w
r	2	0	0	0	0	0
g	0	2	0	0	0	0
b	0	0	2	0	0	0
y	0	0	0	2	0	0
p	0	0	0	0	1	0
w	0	0	0	0	0	1

$t = 2$



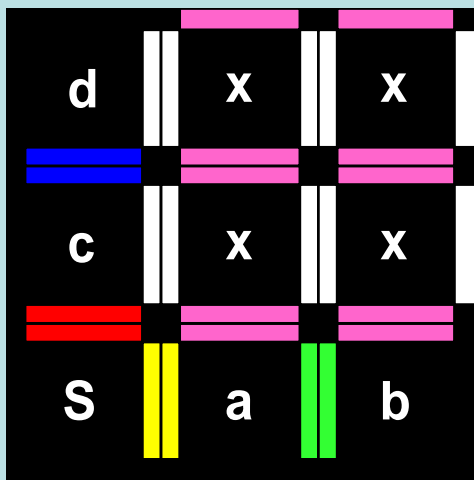
Tiles do not rotate or flip.

How a tile system self-assembles



	r	g	b	y	p	w
r	2	0	0	0	0	0
g	0	2	0	0	0	0
b	0	0	2	0	0	0
y	0	0	0	2	0	0
p	0	0	0	0	1	0
w	0	0	0	0	0	1

$t = 2$



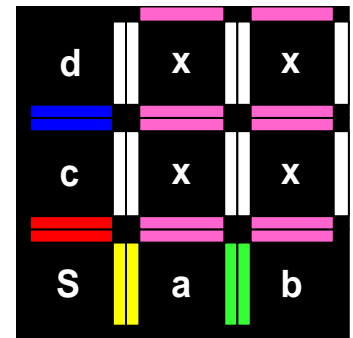
Tiles do not rotate or flip.



Program Size Complexity

Rothemund and Winfree defined the **program size complexity** of a shape S as the minimum number of distinct tiles required to self assemble S and no other shape, also known as the **tile complexity**.

In tile assembly model, the program size complexity of $N \times N$ square is $\Theta(\log N / \log \log N)$.

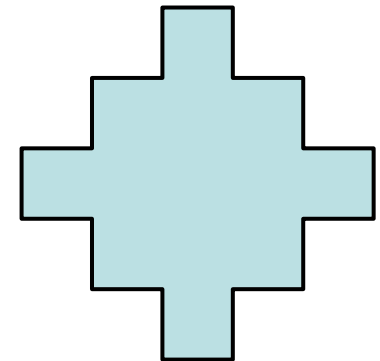




Program Size Complexity

Soloveichik et al. showed that in the tile assembly model the program size complexity of an arbitrary shape is $\Theta(K / \log K)$, where K is the **Kolmogorov complexity** of the shape (independent of scale).

Intuitively Kolmogorov complexity (aka *descriptive complexity*) of a shape is the size of the smallest program describing the shape.





Staged Self-Assembly

In **Staged Self-Assembly** model, tiles can be added dynamically in sequence and intermediate constructions can be stored for later mixing.

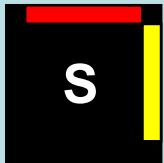
Staging allows us to break through the traditional lower bounds in tile assembly. The model is motivated by the practical assumption that only a constant number of tiles can be engineered.

Staged assembly of 5×5 square

$$T_1 = \begin{array}{|c|} \hline \text{red bar} \\ \hline \text{S} \\ \hline \text{yellow bar} \\ \hline \end{array}$$

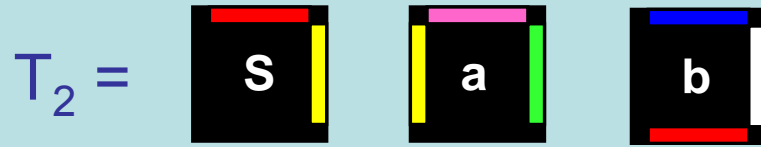
	r	g	b	y	p	w
r	2	0	0	0	0	0
g	0	2	0	0	0	0
b	0	0	2	0	0	0
y	0	0	0	2	0	0
p	0	0	0	0	1	0
w	0	0	0	0	0	1

$t = 2$



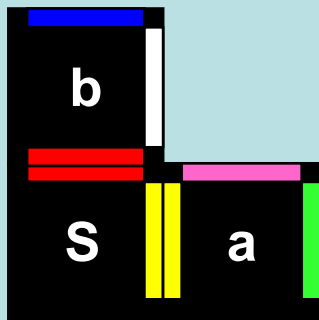
Tiles do not rotate or flip.

Staged assembly of 5×5 square



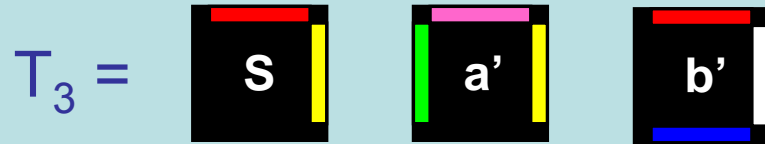
	r	g	b	y	p	w
r	2	0	0	0	0	0
g	0	2	0	0	0	0
b	0	0	2	0	0	0
y	0	0	0	2	0	0
p	0	0	0	0	1	0
w	0	0	0	0	0	1

$t = 2$



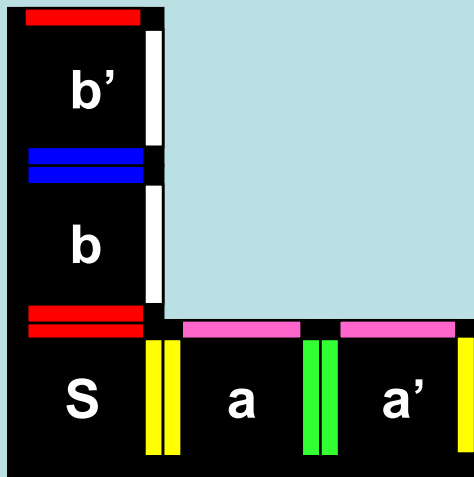
Tiles do not rotate or flip.

Staged assembly of 5×5 square



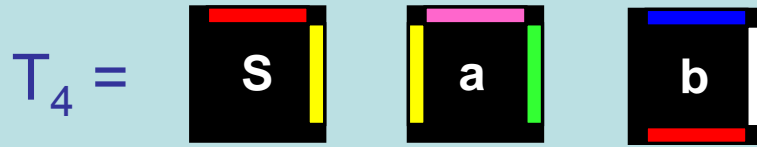
	r	g	b	y	p	w
r	2	0	0	0	0	0
g	0	2	0	0	0	0
b	0	0	2	0	0	0
y	0	0	0	2	0	0
p	0	0	0	0	1	0
w	0	0	0	0	0	1

$t = 2$



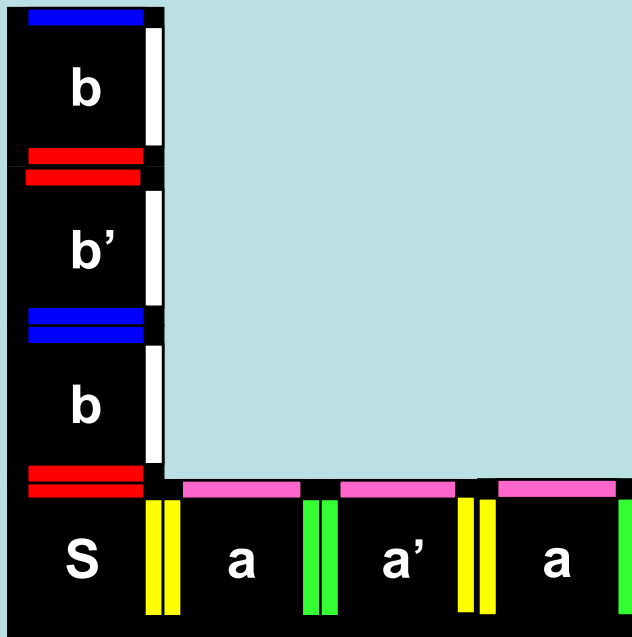
Tiles do not rotate or flip.

Staged assembly of 5×5 square



	r	g	b	y	p	w
r	2	0	0	0	0	0
g	0	2	0	0	0	0
b	0	0	2	0	0	0
y	0	0	0	2	0	0
p	0	0	0	0	1	0
w	0	0	0	0	0	1

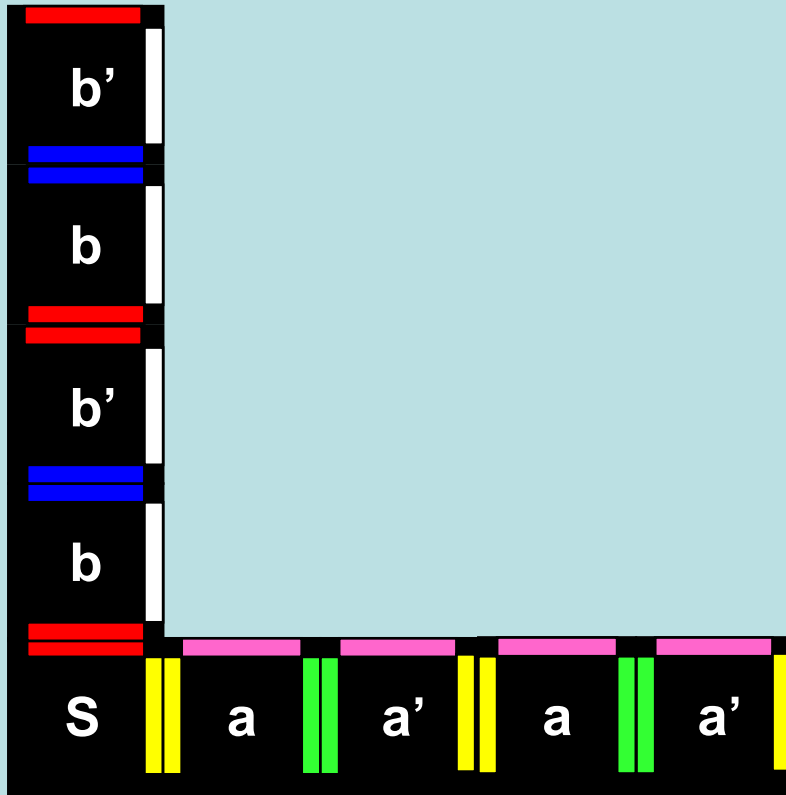
$t = 2$



Tiles do not rotate or flip.

Staged assembly of 5×5 square

$$T_5 = \begin{array}{|c|} \hline \text{S} \\ \hline \end{array} \quad \begin{array}{|c|} \hline \text{a}' \\ \hline \end{array} \quad \begin{array}{|c|} \hline \text{b}' \\ \hline \end{array}$$

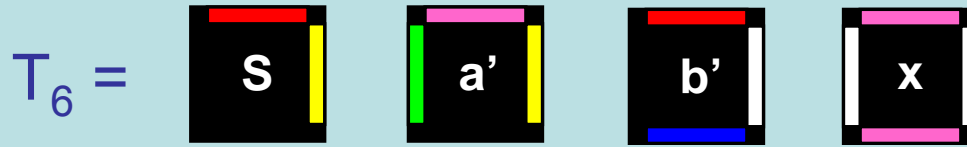


	r	g	b	y	p	w
r	2	0	0	0	0	0
g	0	2	0	0	0	0
b	0	0	2	0	0	0
y	0	0	0	2	0	0
p	0	0	0	0	1	0
w	0	0	0	0	0	1

$t = 2$

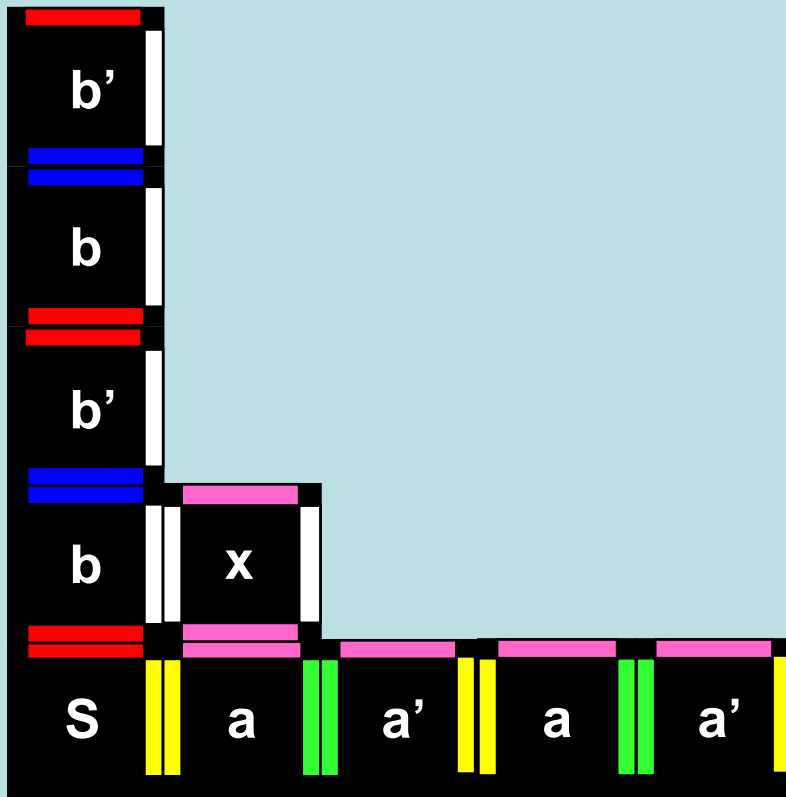
Tiles do not rotate or flip.

Staged assembly of 5×5 square



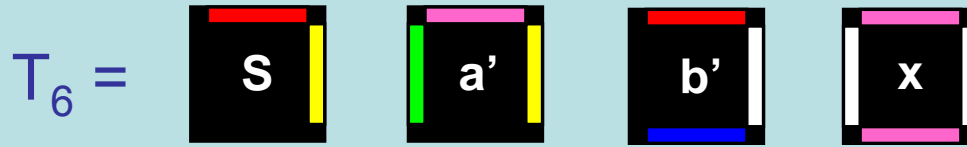
	r	g	b	y	p	w
r	2	0	0	0	0	0
g	0	2	0	0	0	0
b	0	0	2	0	0	0
y	0	0	0	2	0	0
p	0	0	0	0	1	0
w	0	0	0	0	0	1

$t = 2$



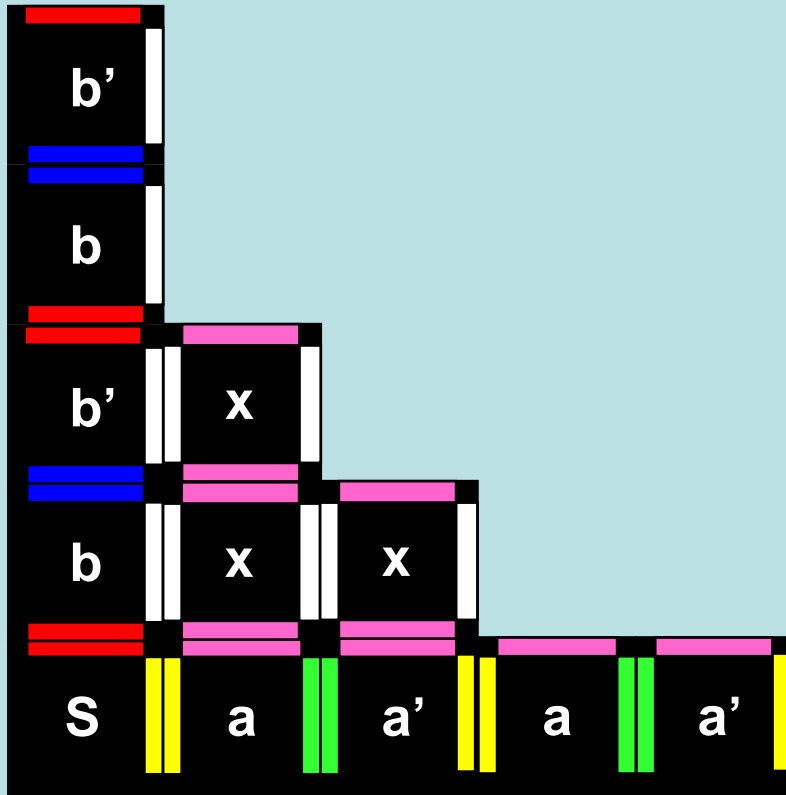
Tiles do not rotate or flip.

Staged assembly of 5×5 square



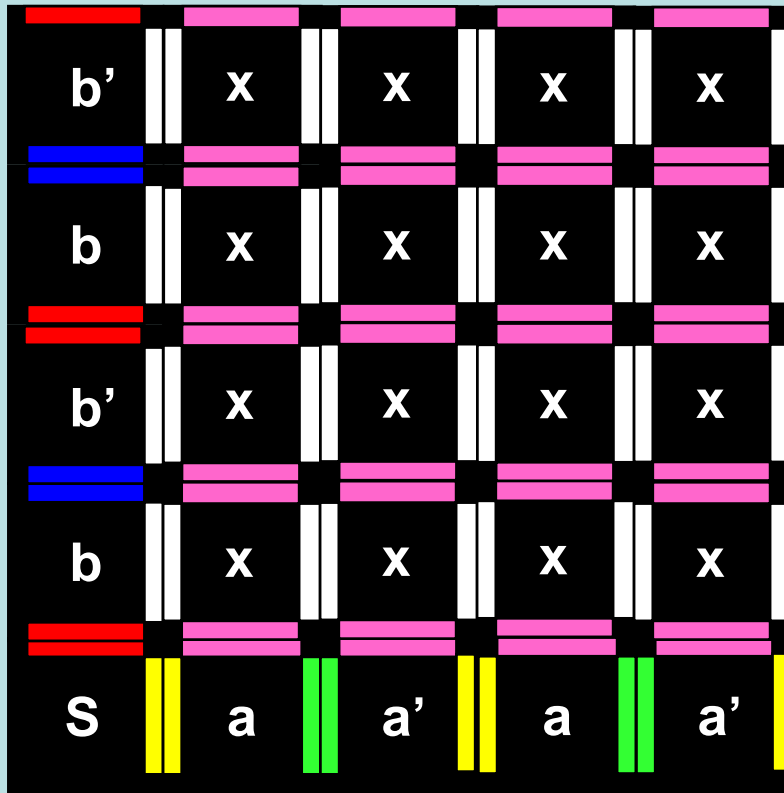
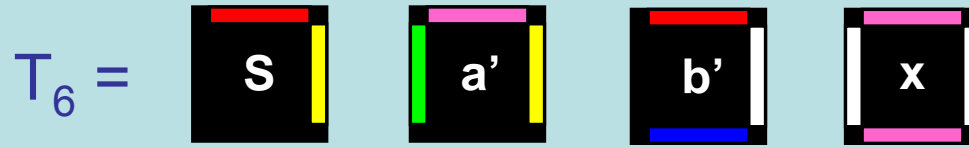
	r	g	b	y	p	w
r	2	0	0	0	0	0
g	0	2	0	0	0	0
b	0	0	2	0	0	0
y	0	0	0	2	0	0
p	0	0	0	0	1	0
w	0	0	0	0	0	1

$t = 2$



Tiles do not rotate or flip.

Staged assembly of 5×5 square



	r	g	b	y	p	w
r	2	0	0	0	0	0
g	0	2	0	0	0	0
b	0	0	2	0	0	0
y	0	0	0	2	0	0
p	0	0	0	0	1	0
w	0	0	0	0	0	1

$t = 2$

Tiles do not rotate or flip.



Metrics

- **Tile complexity**

Number of distinct tile types
(related to number of distinct glues).

- **Bin complexity**

Number of distinct containers used to contain
intermediate results.

- **Stage complexity**

Number of stages (operator time).

- **Temperature sensitivity**

The value of the temperature threshold



Metrics

- **Planarity**

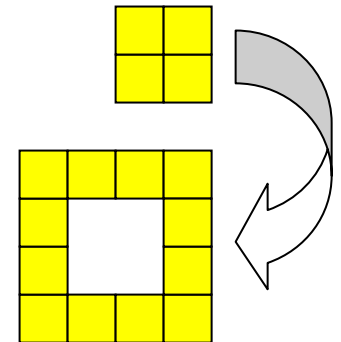
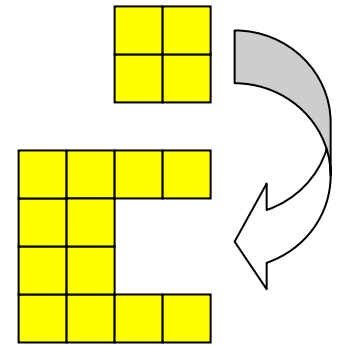
Tiles can be moved into position without intersecting each other. This avoids supertiles with holes to fill.

- **Full-Connectivity**

Every pair of adjacent tiles is connected with strength > 0 .

- **Scale factor**

The created shape is a scaled version of the desired shape.



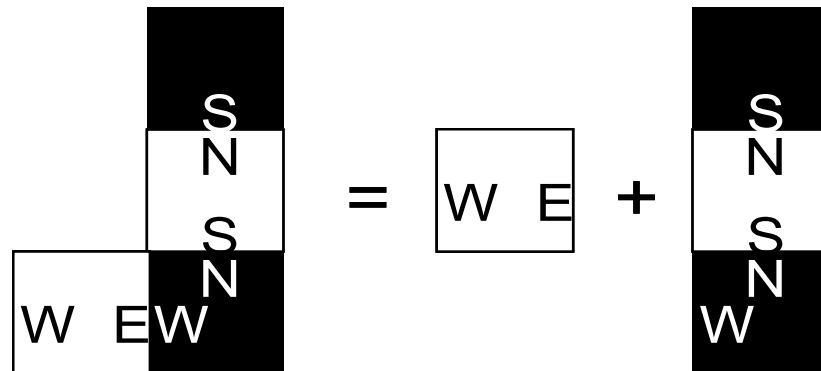
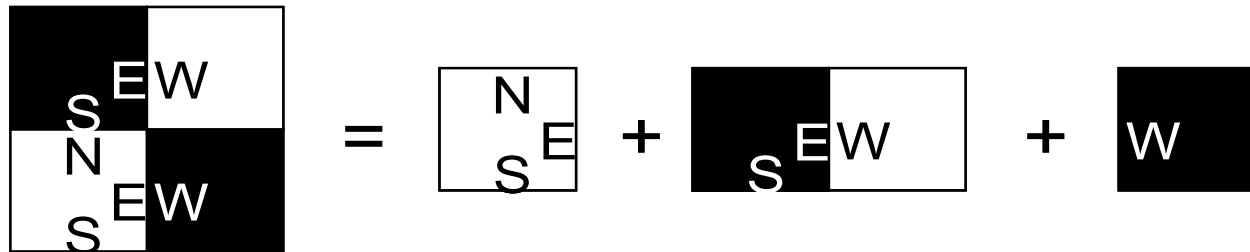
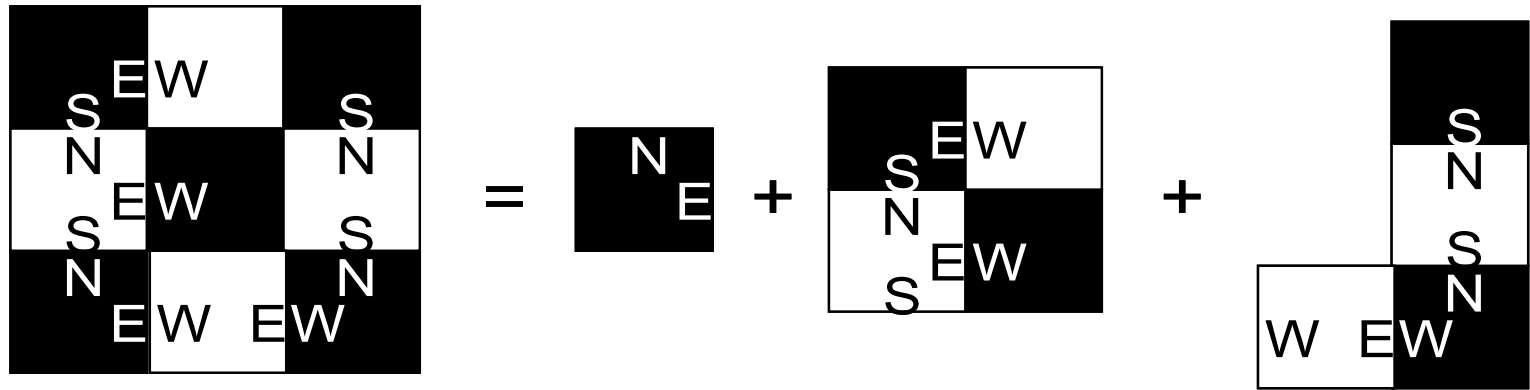


Summary of Results

$n \times n$ square	Tiles	Bins	Stages	τ	Scale	Conn.	Planar
Previous work	$\Theta(\frac{\log n}{\log \log n})$	1	1	2	1	full	yes
Jigsaw approach	$O(1)$	$O(1)$	$O(\log n)$	1	1	full	yes
General shapes	Tiles	Bins	Stages	τ	Scale	Conn.	Planar
Previous work	$\Theta(K/\log K)$	1	1	2	unbounded	partial	no
Spanning Tree Method	$O(1)$	$O(1)$	$O(n)$	1	1	partial	no
Jigsaw approach Monotones	$O(1)$	$O(n)$	$O(\log n)$	1	1	full	yes



Spanning Tree Method





Spanning Tree Method

Spanning tree method can create any shape S .

- **Tile complexity:** $O(1)$
 - **Stage complexity:** $O(\text{depth of spanning tree})$
 - **Bin complexity:** $O(\text{number of distinct tiles})$
 - **Temperature sensitivity:** 1
 - **Planar:** No
Achieving planarity is an **Open** problem .
 - **Fully connected:** No
-



Assembly of 1xN Line

Consider 1xN line: any spanning tree will have depth $O(N)$ and thus the stage complexity using spanning tree method would be $O(N)$.

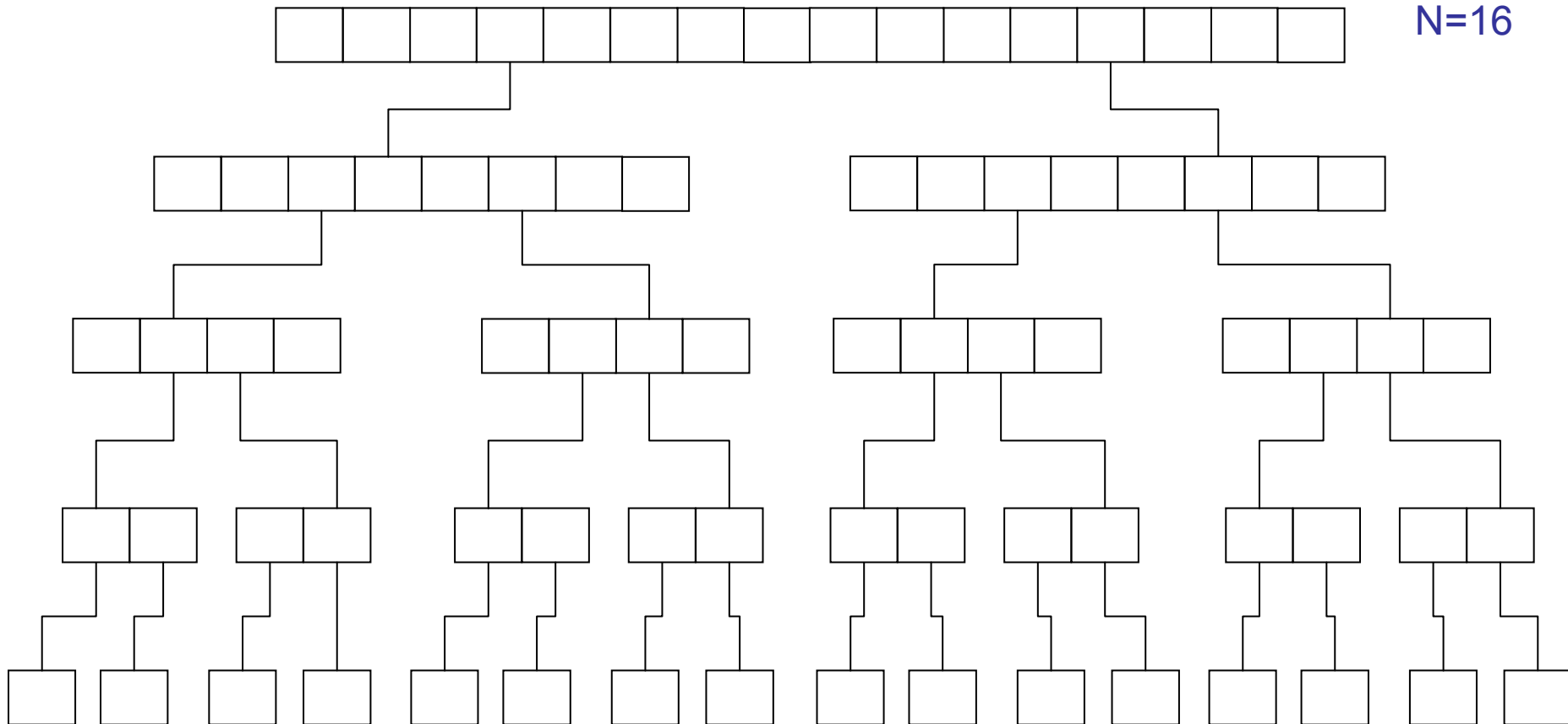


1xN Line

We can use a divide and conquer approach to reduce the stage complexity. The idea is to recursively split the shape into pieces such that the pieces can be combined in a unique way.



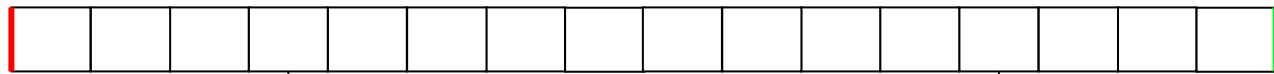
Assembly of $1 \times N$ Line



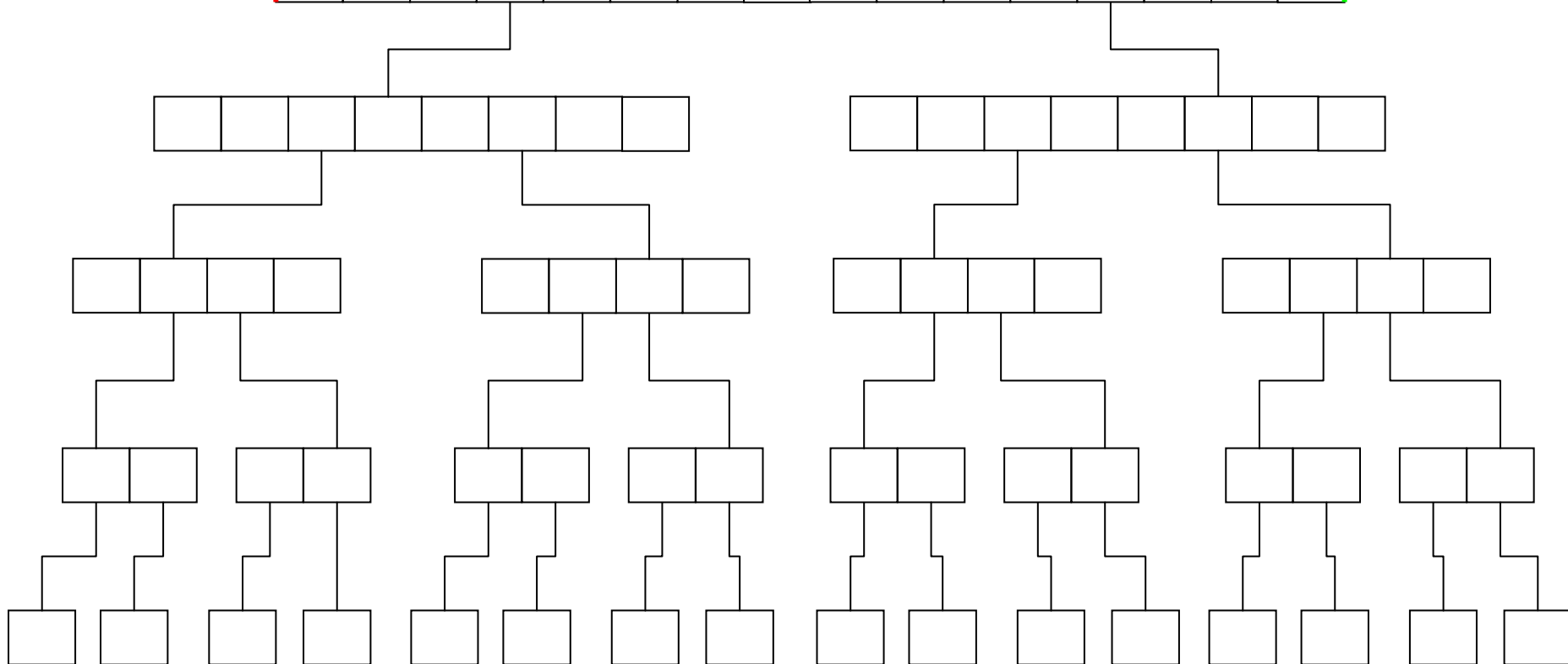
Decomposition Tree



Assembly of $1 \times N$ Line

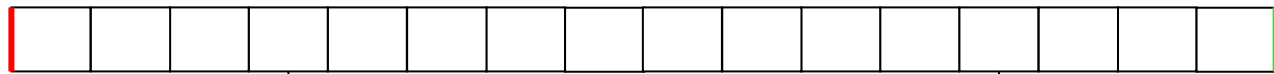


N=16

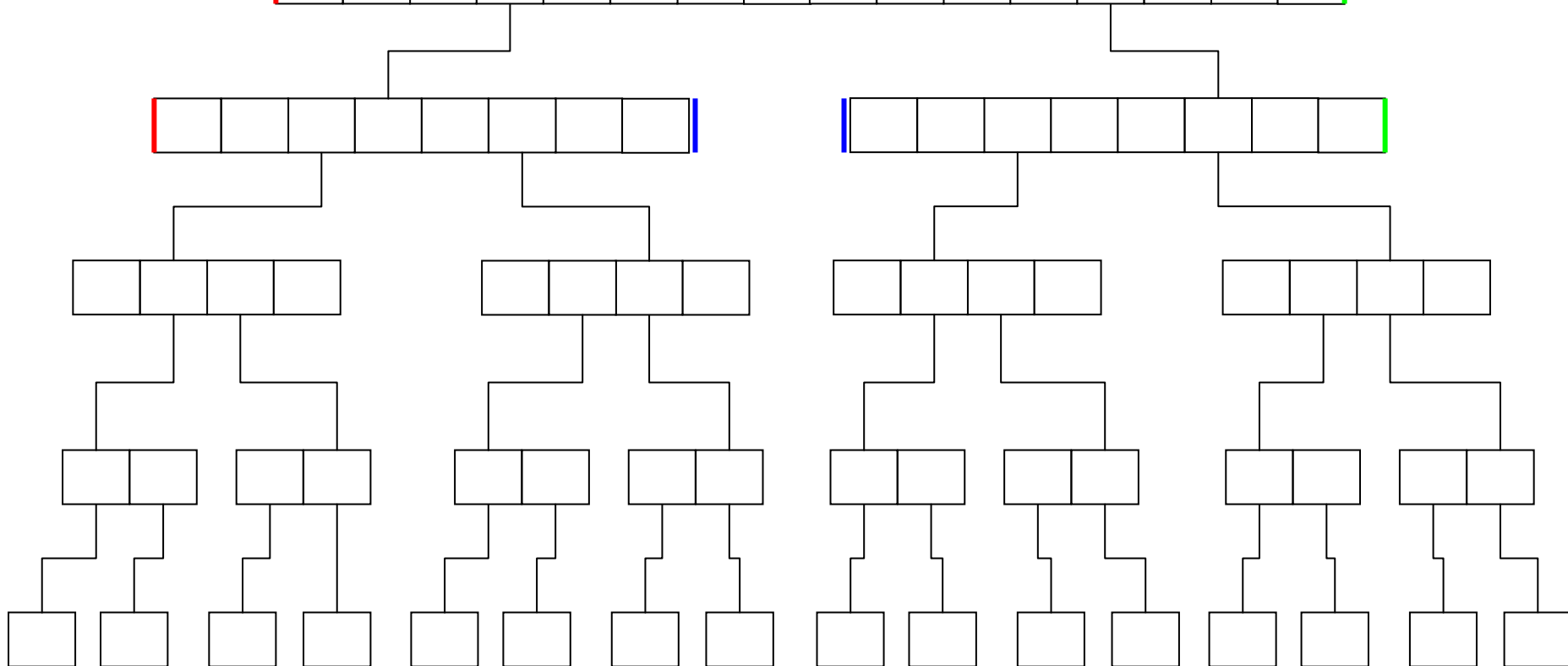




Assembly of $1 \times N$ Line

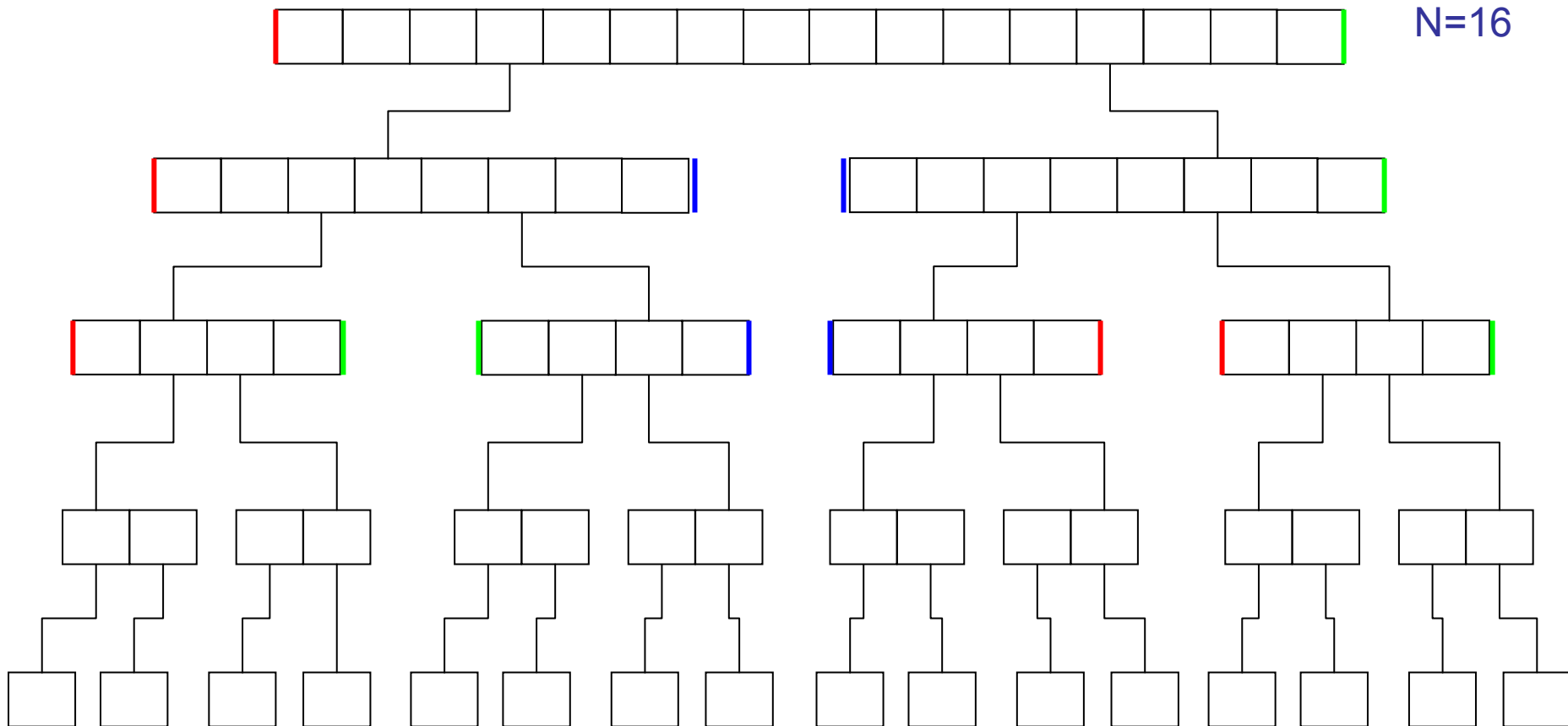


N=16



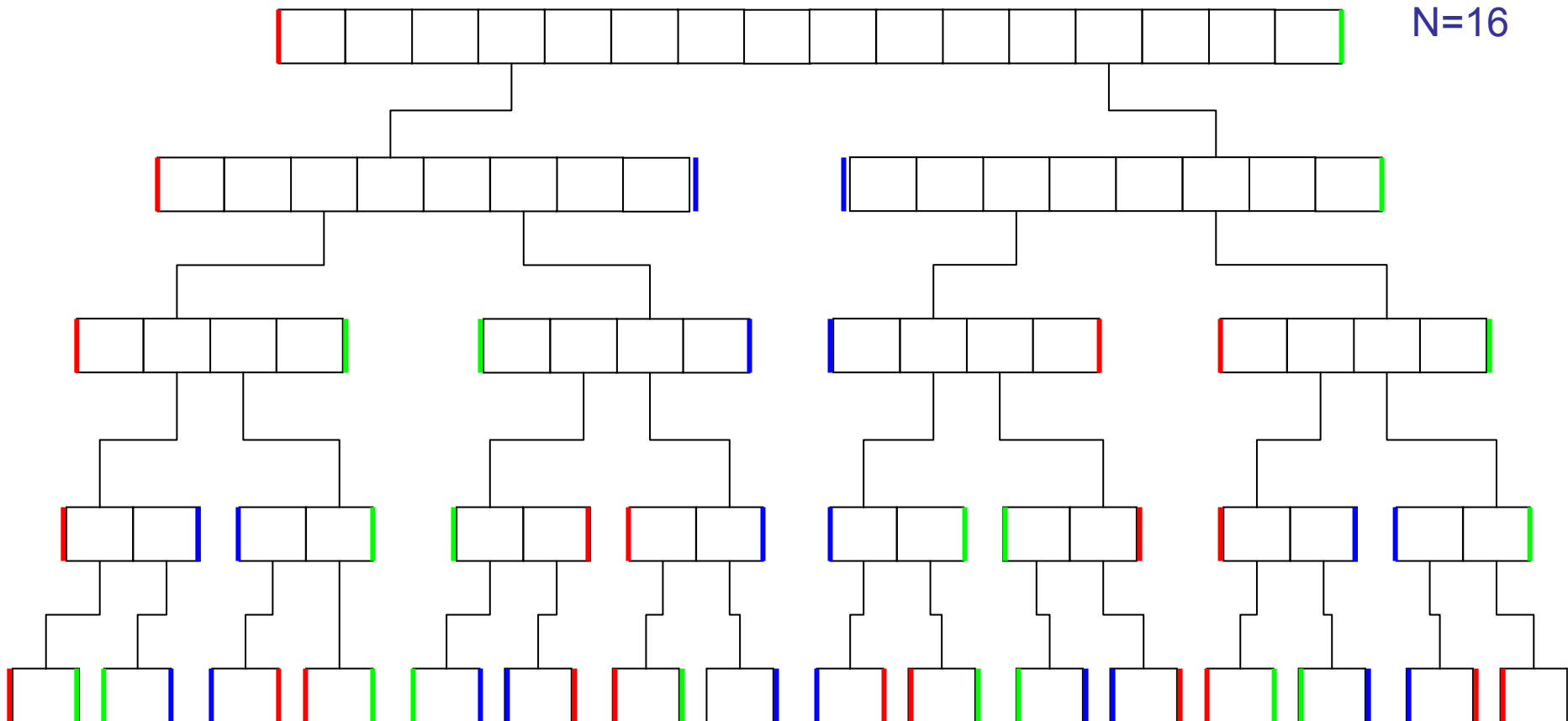


Assembly of $1 \times N$ Line





Assembly of $1 \times N$ Line



Decomposition Tree

Height of Tree = # of Stages, Distinct nodes at a level = # of bins



Assembly of $1 \times N$ Line

Lemma:

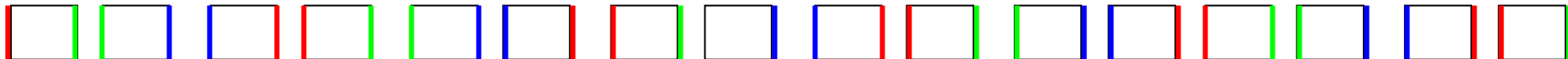
Every supertile in the decomposition tree has different colors on its left and right side. (proof by induction)

Theorem: We can assemble a fully connected $1 \times N$ line, with planarity at temperature 1, using only $O(1)$ tiles, $O(1)$ bins, and $O(\log N)$ stages.



Assembly of $1 \times N$ Line

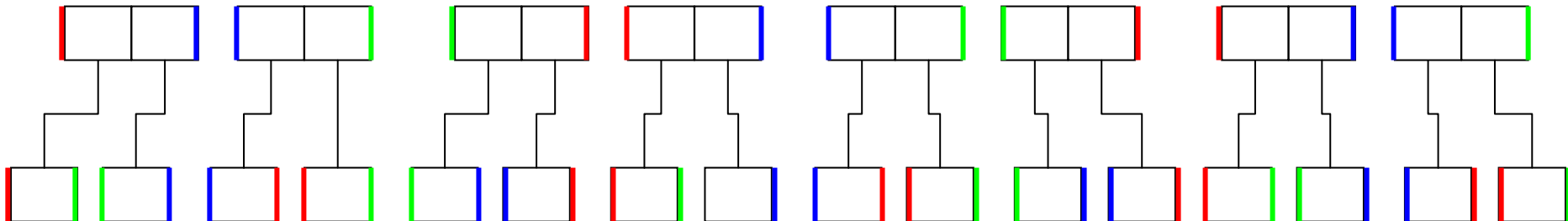
N=16





Assembly of $1 \times N$ Line

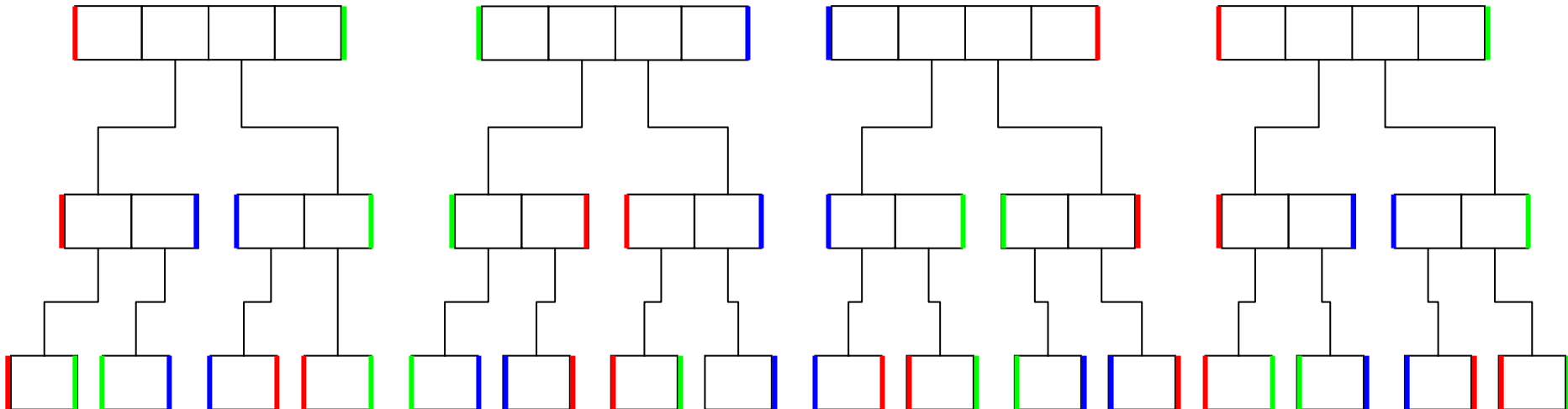
N=16





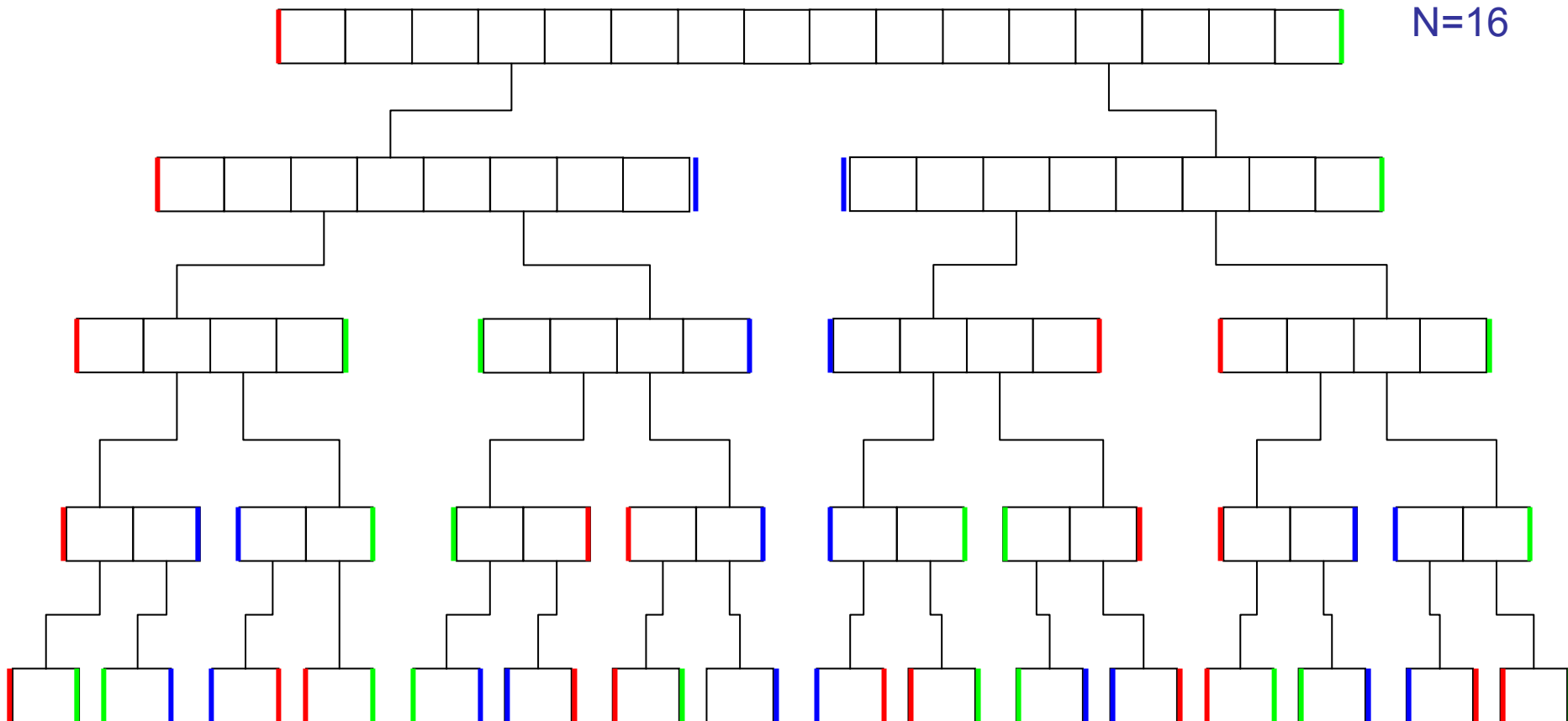
Assembly of $1 \times N$ Line

N=16





Assembly of $1 \times N$ Line



Height of Tree = # of Stages, Distinct nodes at a level = # of bins



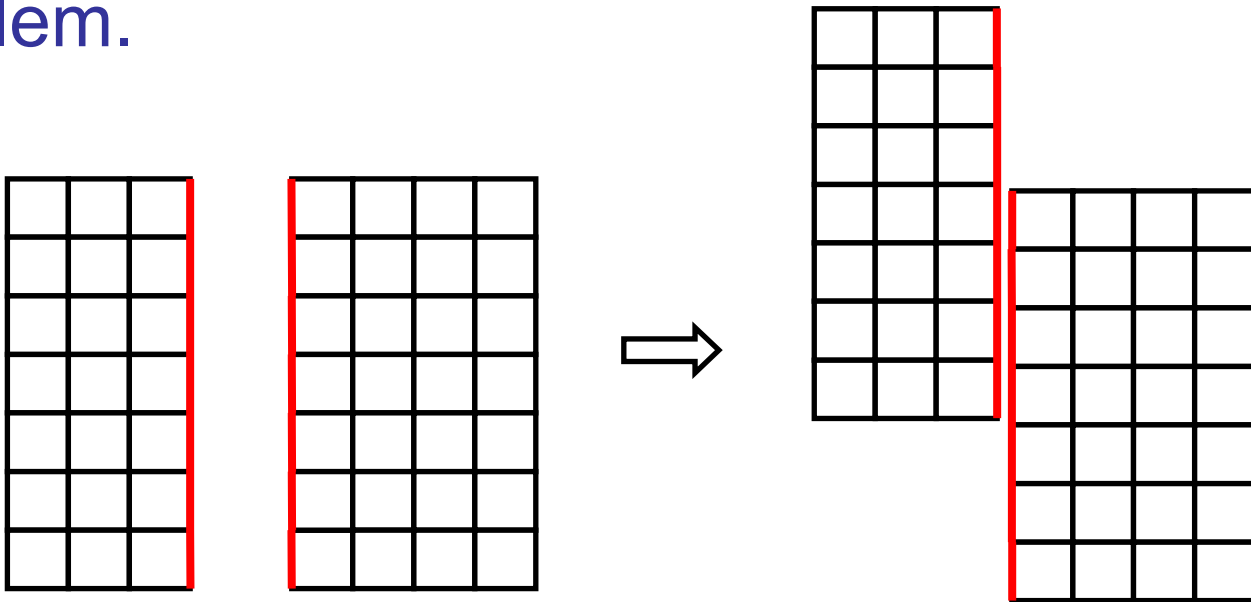
Assembly of $N \times N$ Square?

We would like to create $N \times N$ square using the same technique but we encounter shifting problem.



Assembly of $N \times N$ Square?

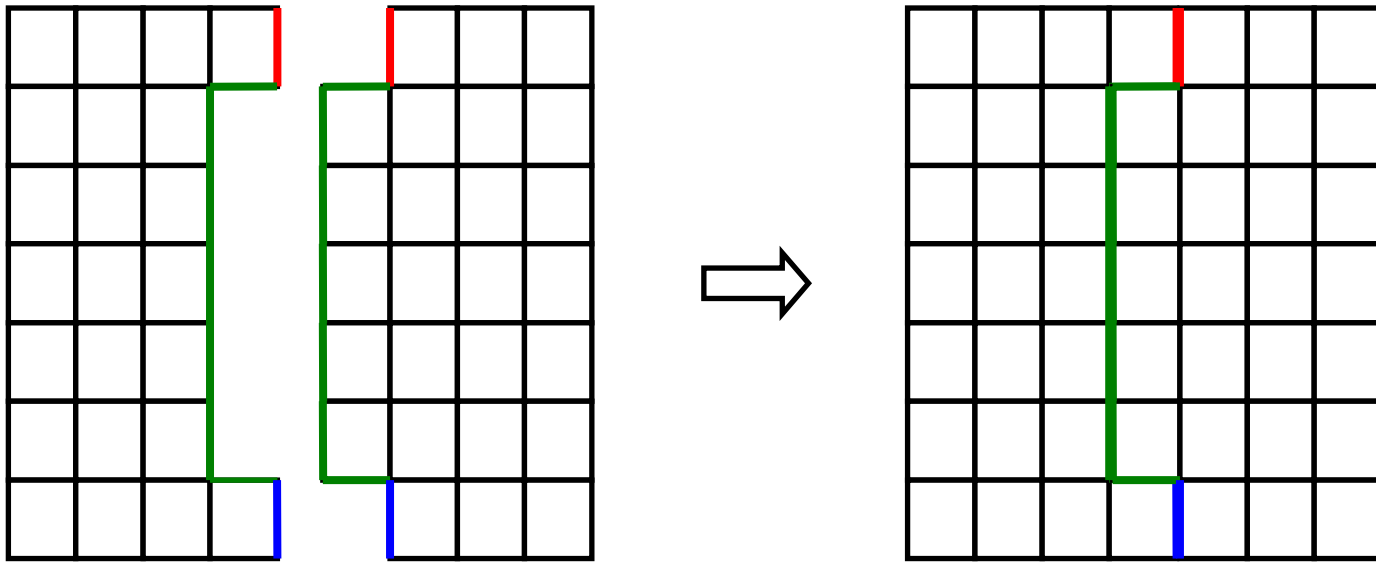
We would like to create $N \times N$ square using the same technique but we encounter shifting problem.



Shifting Problem



Jigsaw Technique



Jigsaw technique can get around the shifting problem.
And then we can use the construction similar to that for
1xN line.

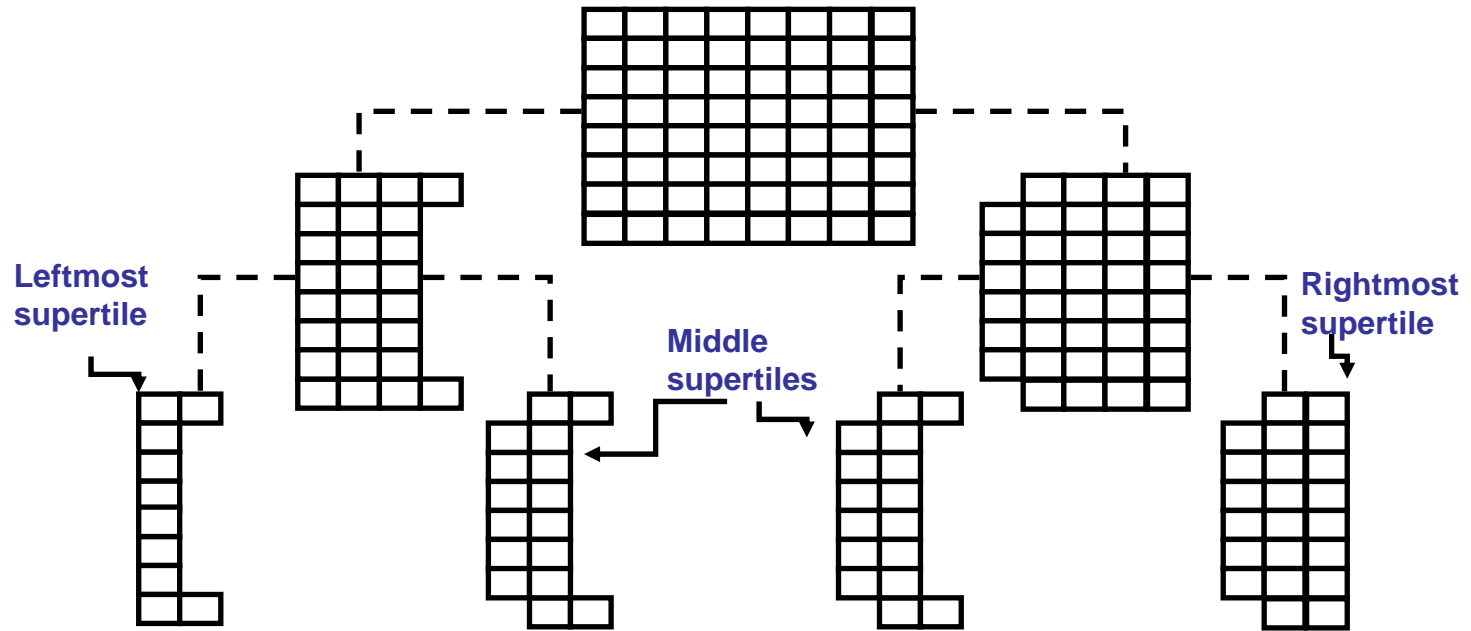


Assembly of $N \times N$ Square

Theorem: Jigsaw technique can assemble a fully connected $N \times N$ square, with planarity at temperature 1, using only $O(1)$ tiles, $O(1)$ bins, and $O(\log N)$ stages.

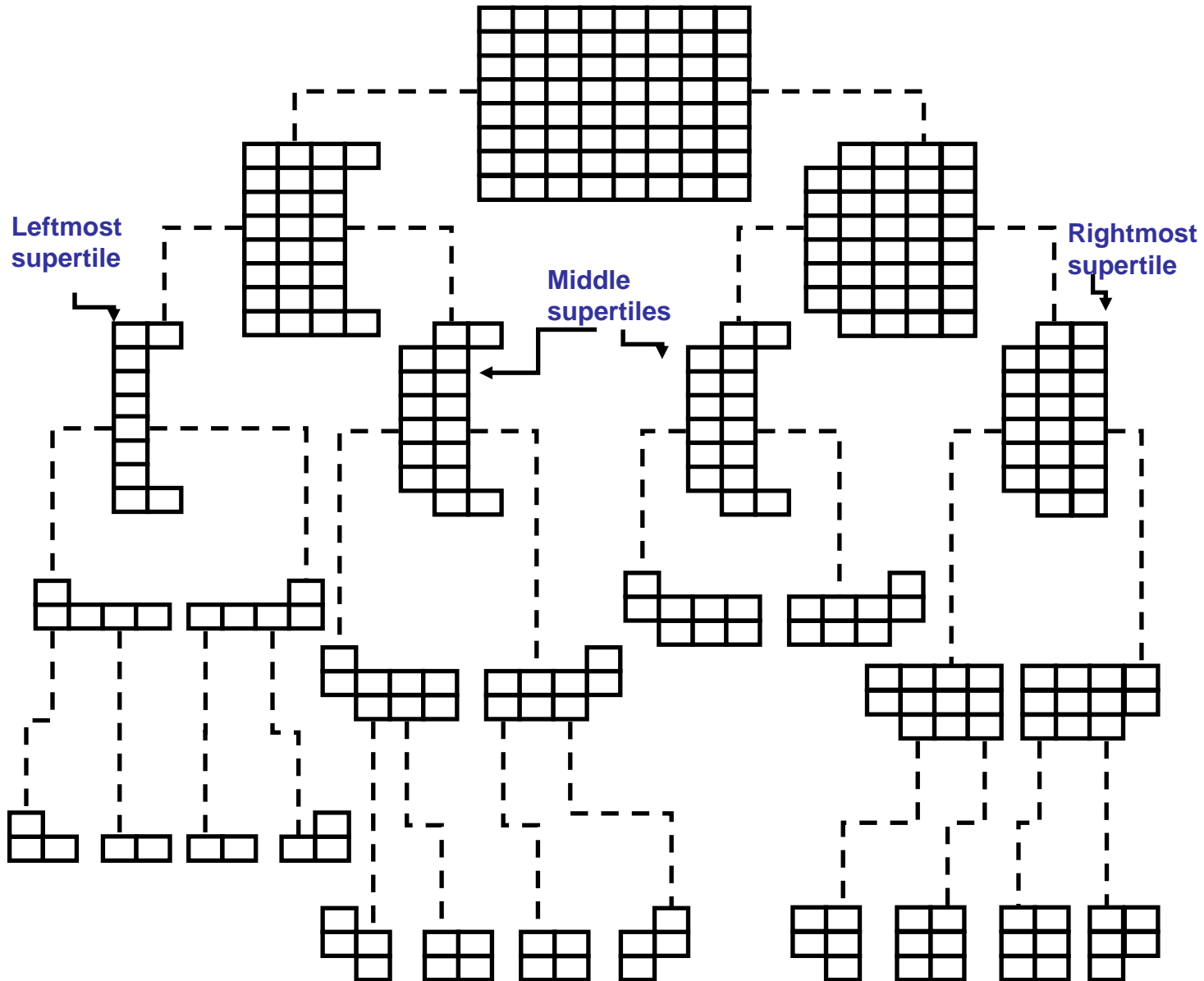
Notice that jigsaw technique overcomes the two drawbacks of spanning tree technique. But creating arbitrary shapes using this method is still **Open**.

Decomposition Tree for NxN Square



Vertical Decomposition

Decomposition Tree for NxN Square

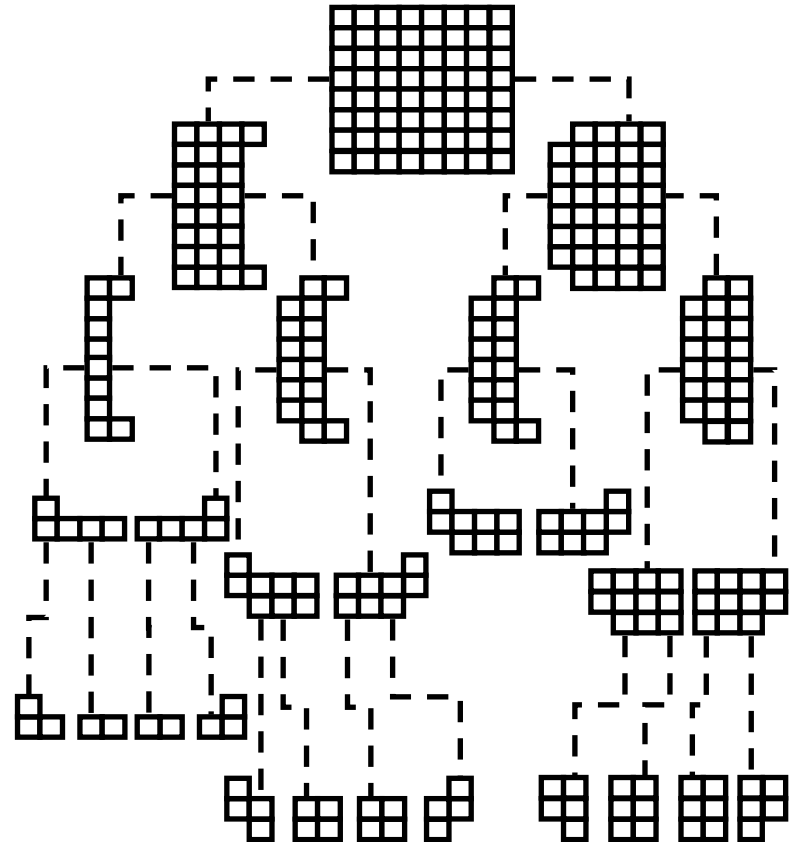


Vertical Decomposition is followed by Horizontal Decomposition



Stage Complexity is $O(\log N)$

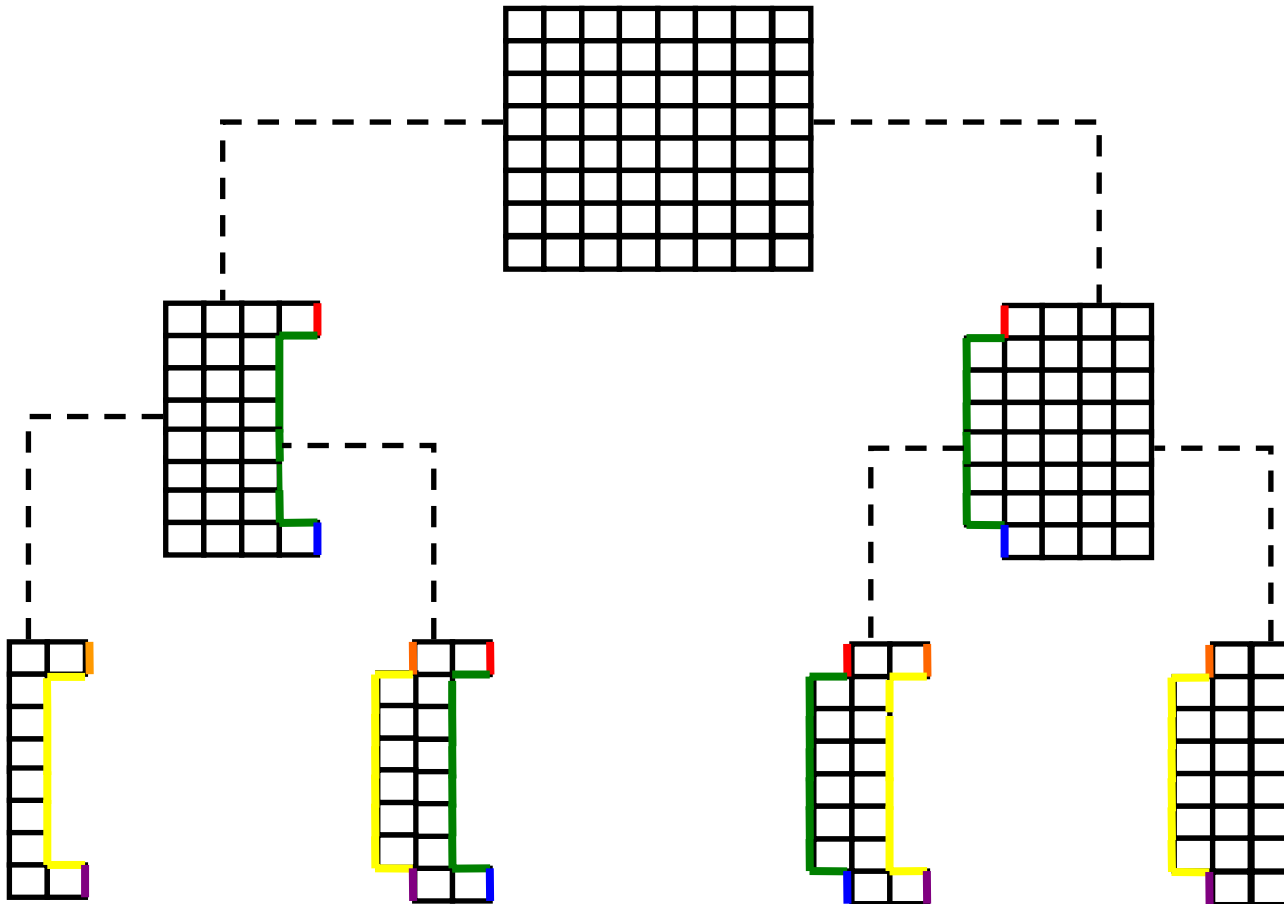
The height of the decomposition tree, which is a balanced binary tree, corresponds to stage complexity.





Tile Complexity is $O(1)$

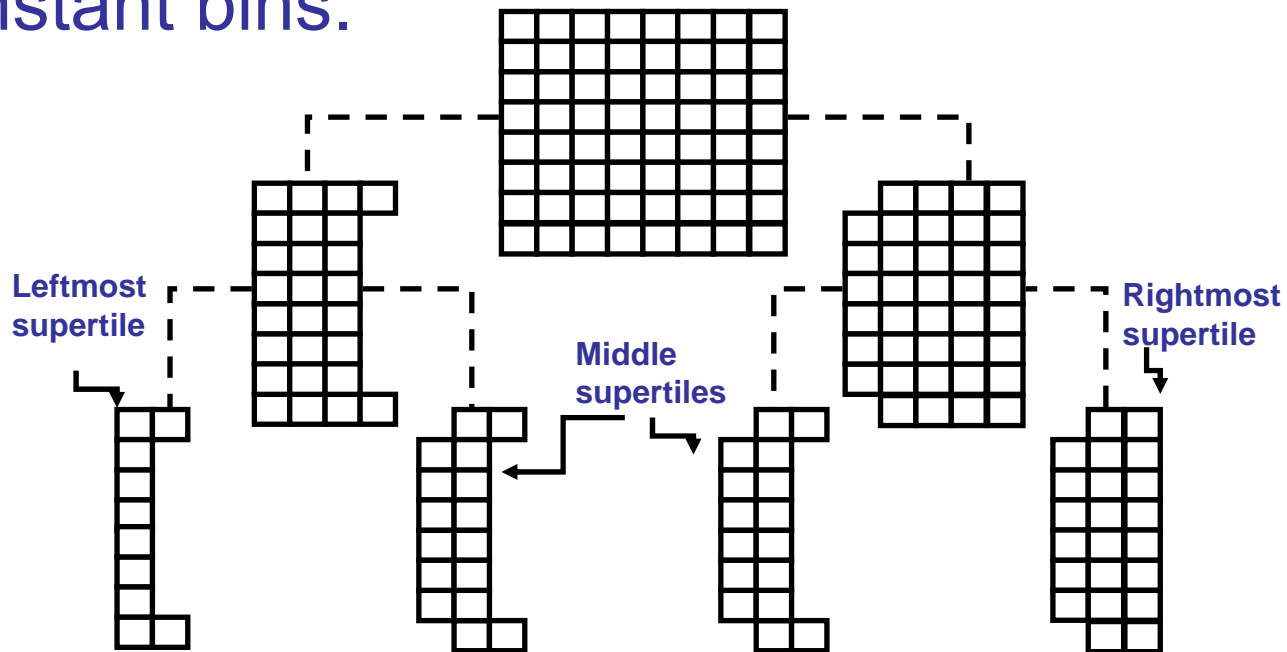
We use at most 12 colors which implies constant number of tiles.





Bin Complexity is $O(1)$

- Separate bins for leftmost, rightmost and middle supertiles.
- Middle supertiles always have the same shape (almost).
- Constant color choices for tile walls imply constant bins.





Assembly for NxN Square

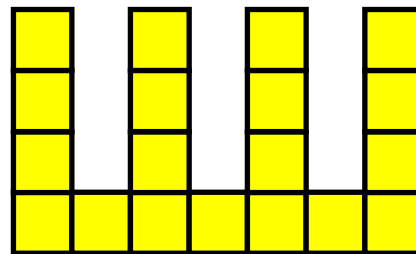
We can create an NxN square.

- **Tile complexity:** $O(1)$
 - **Stage complexity:** $O(\log N)$
 - **Bin complexity:** $O(1)$
 - **Temperature sensitivity:** 1
 - **Planar**
 - **Fully connected**
-



Assembly of Monotones

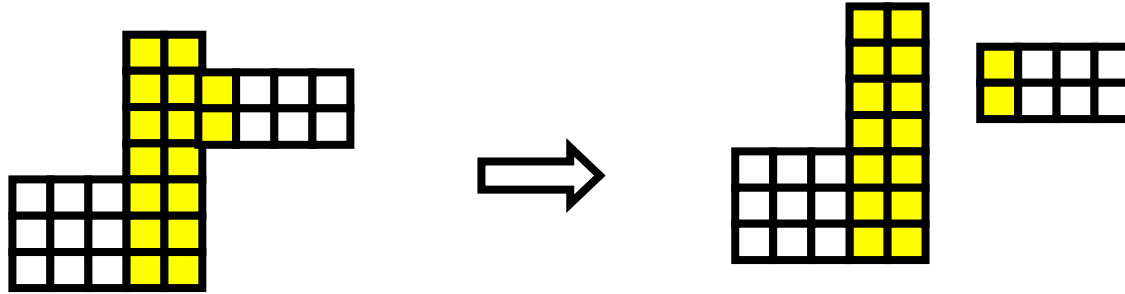
Theorem: Jigsaw technique can assemble a monotone shape with planarity at temperature 1, using only $O(1)$ tiles, $O(N)$ bins, and $O(\log N)$ stages, where N is the side length of smallest square bounding the shape.



x-Monotone

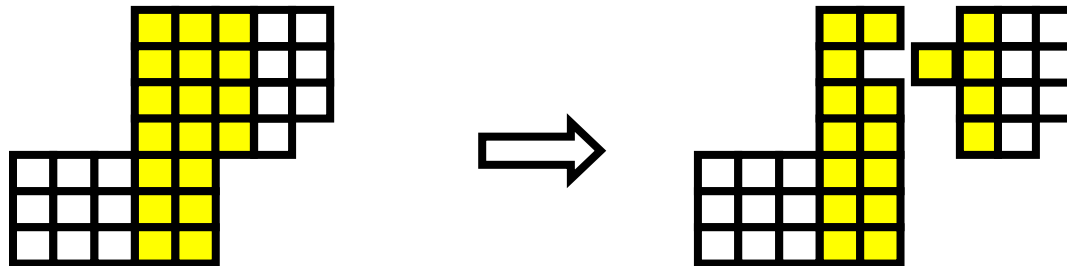
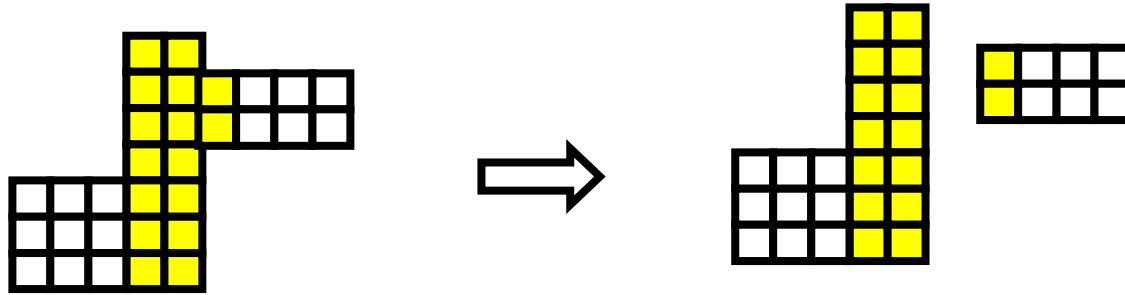


Assembly of Monotones



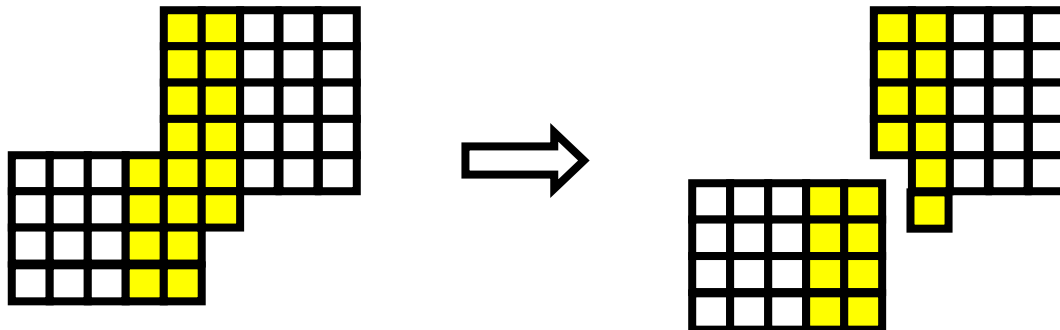
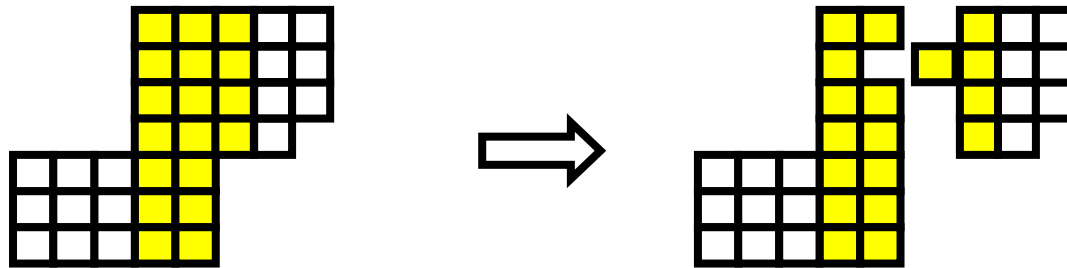
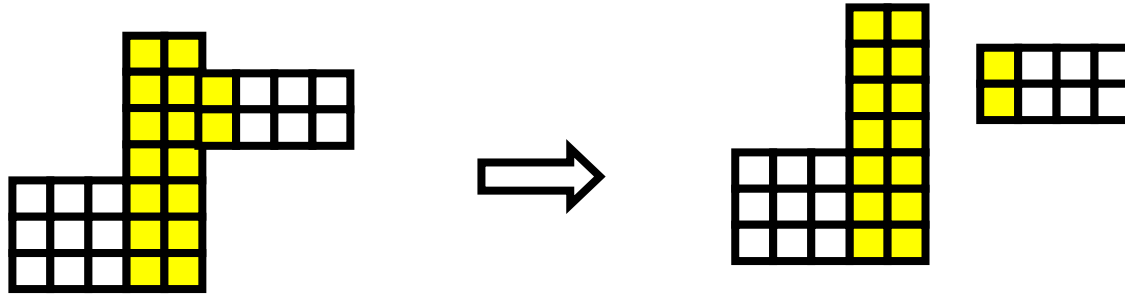


Assembly of Monotones





Assembly of Monotones

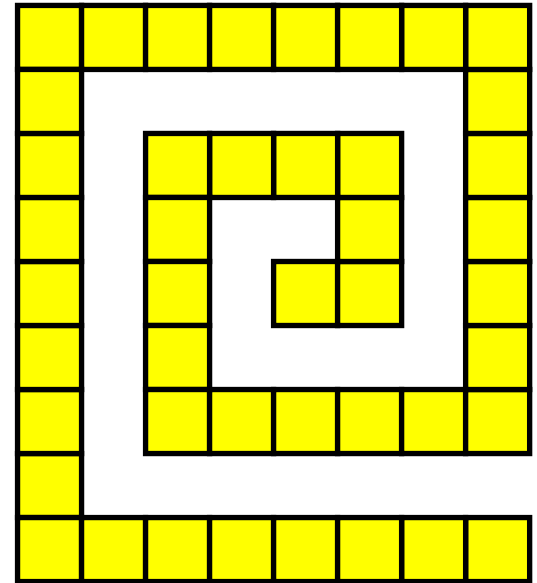




Assembly of Spirals

To create spirals: decompose at the turns and then build inside out.

Note a N -tile spiral (max. turns) has a $\Omega(\sqrt{N})$ lower bound on stage complexity for any planar method. Hence it cannot be constructed by any constant-stage tile system.

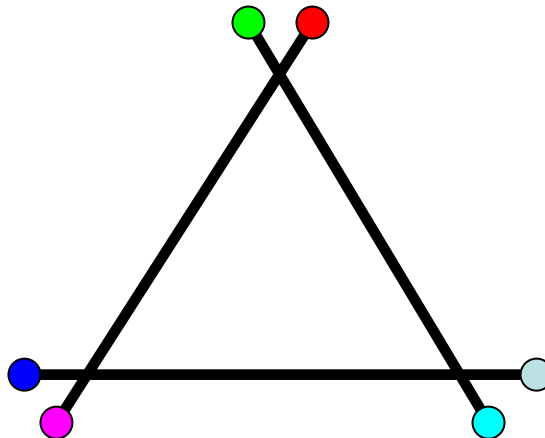




Triangular Tile Model

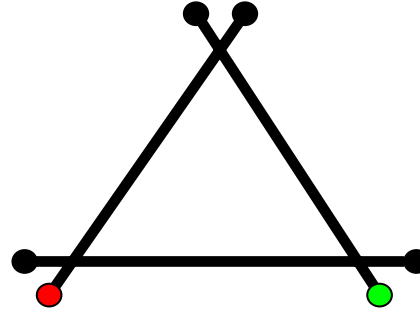
In triangular tile model:

- Equilateral triangle as the building block.
- Six binding sites—two near each vertex.
- Two sites stick to each other only when their corresponding rods are aligned.



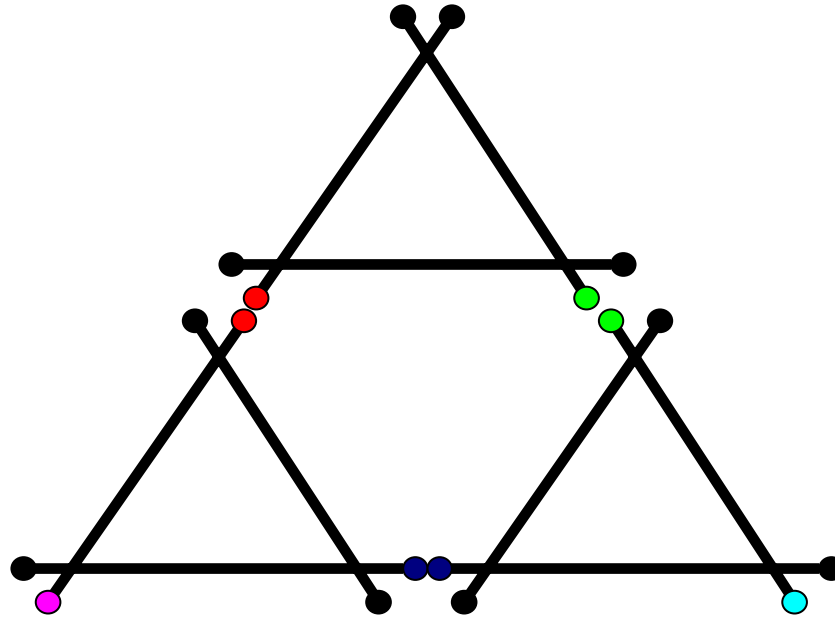


Assembling Sierpinski Triangle



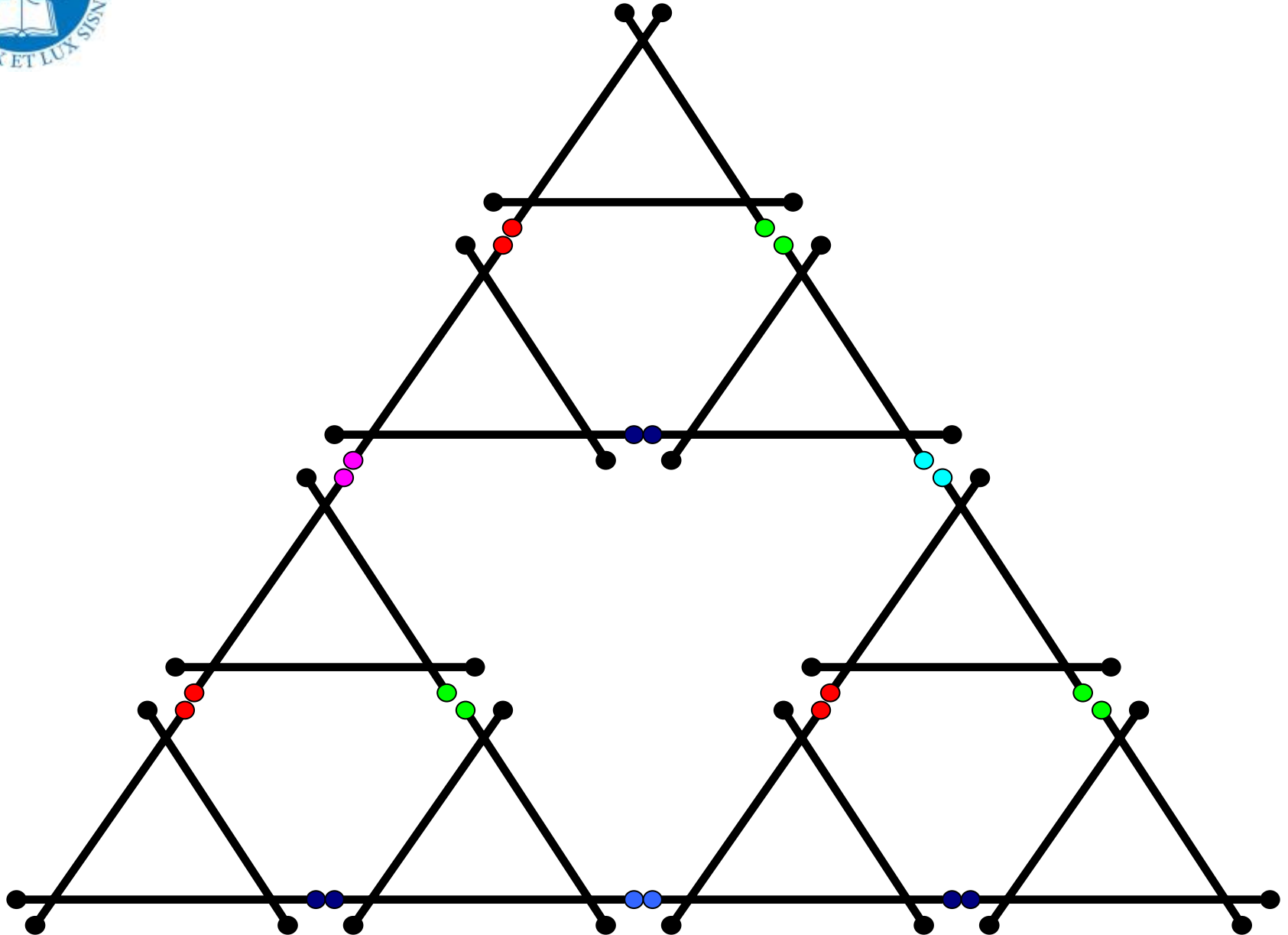


Assembling Sierpinski Triangle





Assembling Sierpinski Triangle

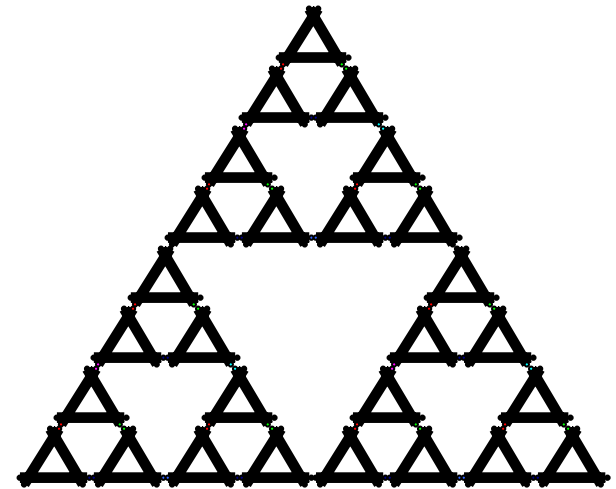




Assembling Sierpinski Triangle

Sierpinski triangle of side length N can be created using:

- **Tile complexity:** $O(1)$
- **Stage complexity:** $O(\log N)$
- **Bin complexity:** $O(1)$
- **Temperature sensitivity:** 1
- **Planar**
- **Fully connected**

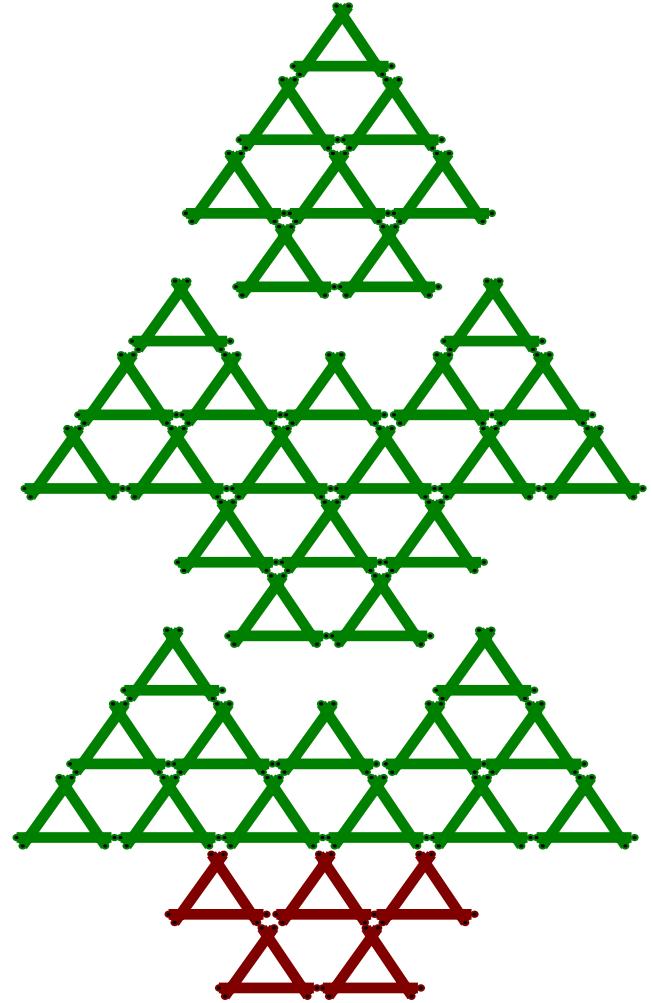




More Shapes in Triangular Model

We also know how to create other interesting shapes in this model such as equilateral triangles, $N \times N$ square, Star of David etc.

Both spanning tree method and the jigsaw technique are applicable to triangular model as well.





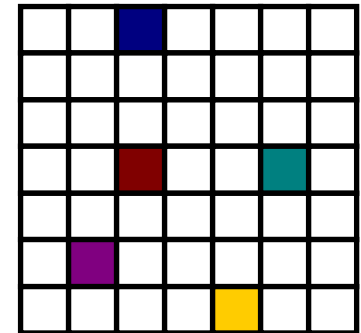
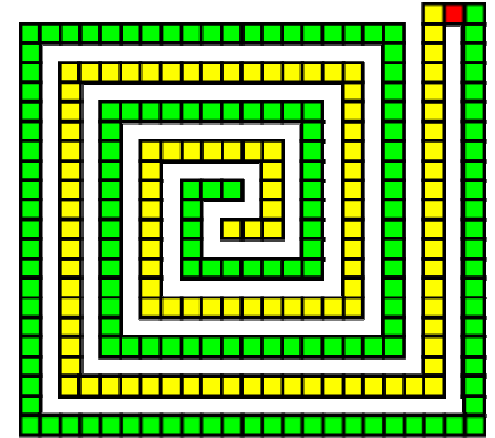
Summary

- Self-Assembly
 - Tile Model of Self-Assembly
 - Staged Self-Assembly
 - Assembly of $1 \times N$ Line
 - Assembly of $N \times N$ Square
 - Assembly of Monotones
 - Triangular Tile Model
-



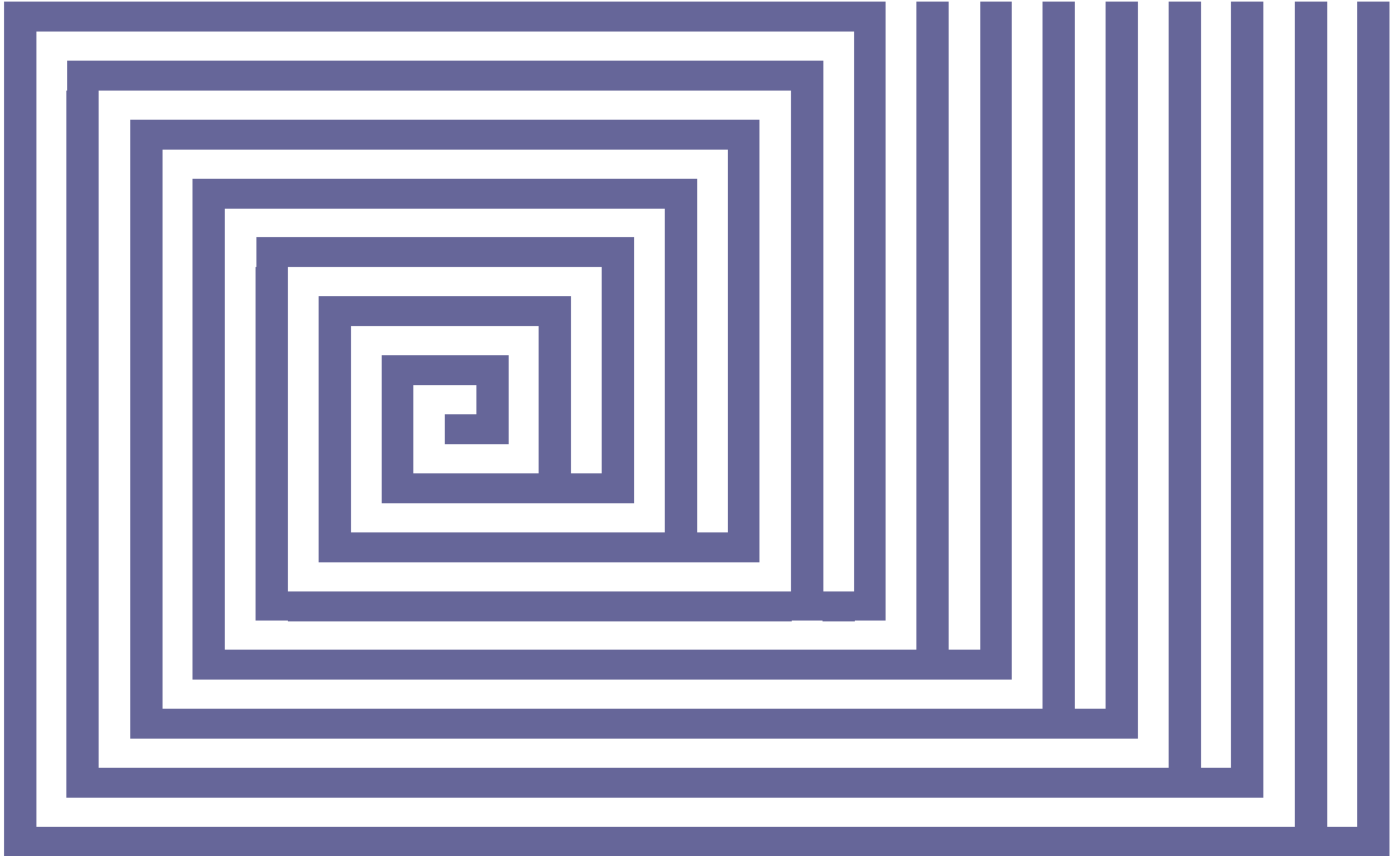
Future Work

- Fully connected (planarity?)
 - Shapes without holes
 - Arbitrary shapes
- Lower bound on stage complexity of a shape.
- Assembly of 3-D structures.
- Self-assembly of *functionalized* 2-D and 3-D structures.





Questions?





References

- L. M. Adleman. Towards a mathematical theory of self-assembly (extended abstract). Technical report, University of Southern California, 1999.
 - Len Adleman, Qi Cheng, Ashish Goel, Ming-Deh Huang, David Kempe, Pablo Moisset de Espan'es, and Paul Wilhelm Karl Rothemund. Combinatorial optimization problems in self-assembly. In Proceedings of the Thirty-Fourth Annual ACM Symposium on Theory of Computing, pages 23–32 (electronic), New York, 2002. ACM.
 - Leonard Adleman, Qi Cheng, Ashish Goel, and Ming-Deh Huang. Running time and program size for self-assembled squares. In Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing, pages 740–748 (electronic), New York, 2001. ACM.
 - Gagan Aggarwal, Qi Cheng, Michael H. Goldwasser, Ming-Yang Kao, Pablo Moisset de Es-panes, and Robert T. Schweller. Complexities for generalized models of self-assembly. *SIAM J. Comput.*, 34(6):1493–1515 (electronic), 2005.
-



References

- Ming-Yang Kao and Robert Schweller. Reducing tile complexity for self-assembly through temperature programming. In Proceedings of the 17th Annual ACM SIAM Symposium on Discrete Algorithms (SODA 2006), pages 571–580., Jan 2006.
 - Paul W. K. Rothmund and Erik Winfree. The program-size complexity of self- assembled squares (extended abstract). In Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, pages 459–468 (electronic), New York, 2000. ACM.
 - David Soloveichik and Erik Winfree. Complexity of self-assembled shapes, 2004. ACM Computing Research Repository, cs.CG/0412096.
-