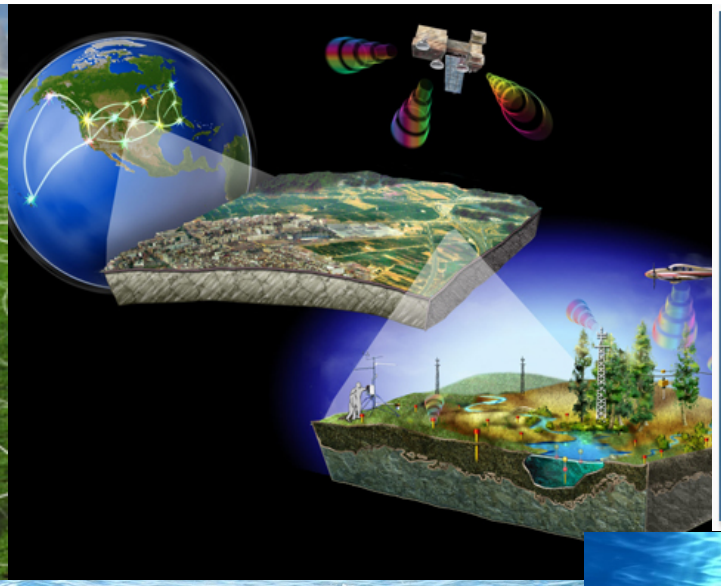
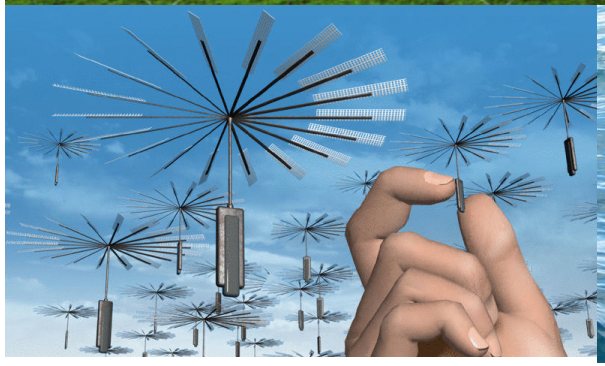
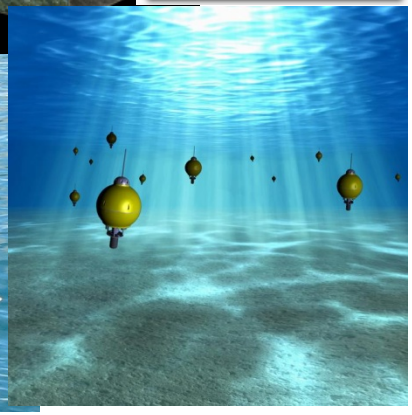
Self-Assembling Sensor Networks

Christian Scheideler
Dept. of Computer Science
University of Paderborn

Sensor Networks

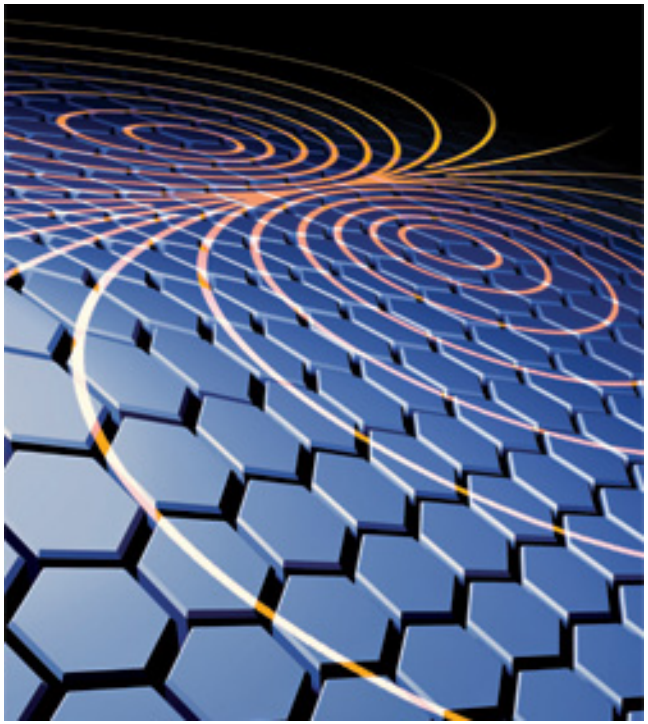






Sensor Networks

Our focus: self-assembling nano-sensors

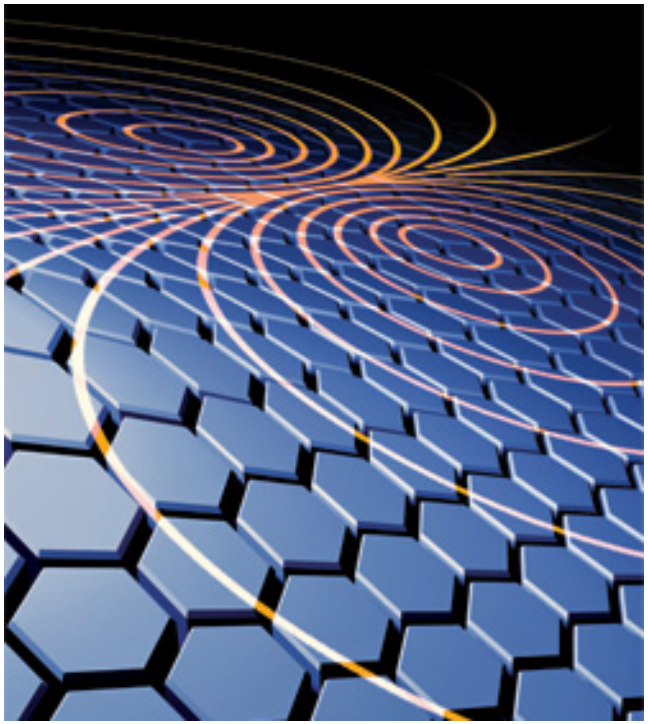


Why self-assembly?

- More flexible
- Easier to solve covering problems

Sensor Networks

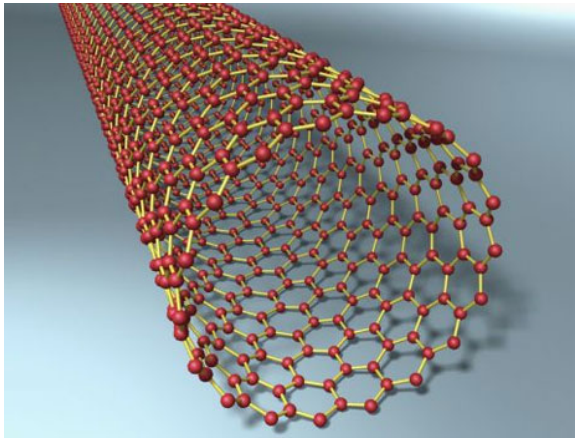
Our focus: self-assembling nano-sensors

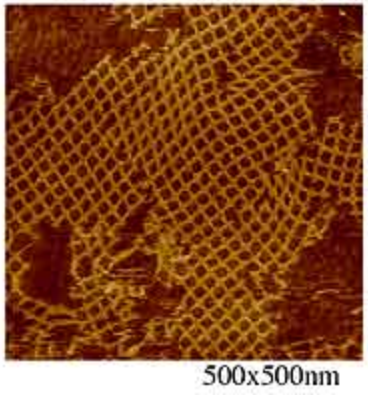


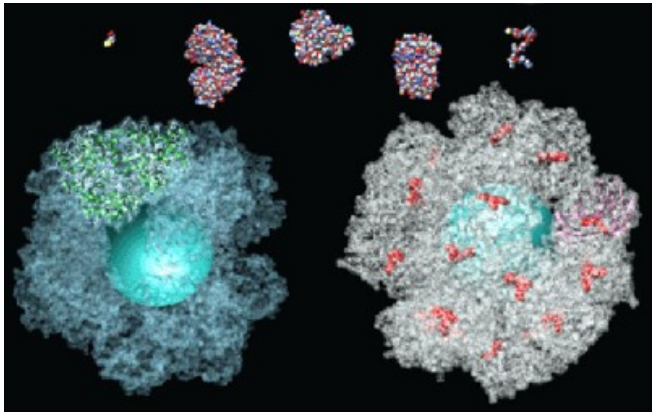
Self-organizing nano-structures occur in many contexts

Self-organizing Nano Structures

Nanotubes, crystals, DNA tiles,…

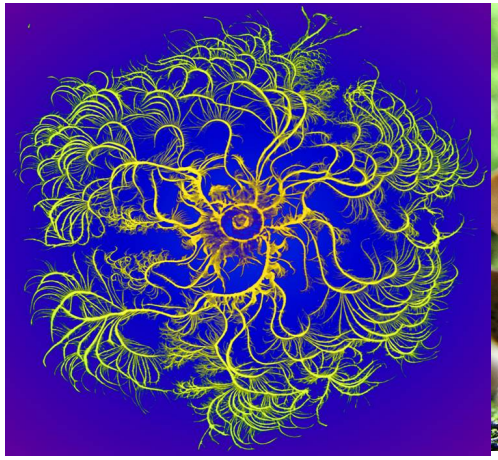






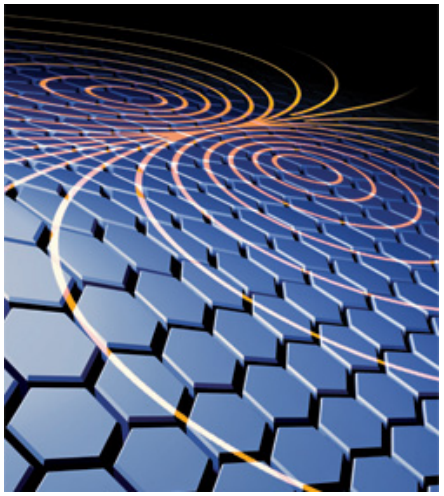
Self-organizing Nano Structures

Bacterial colonies, mushrooms,corals,…



Self-assembling Nano-Sensors

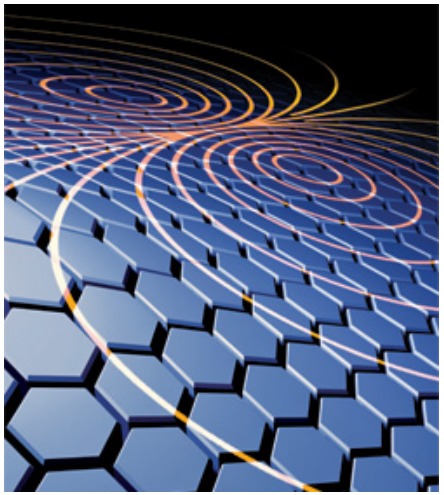
Applications:



- Measure environmental conditions

Self-assembling Nano-Sensors

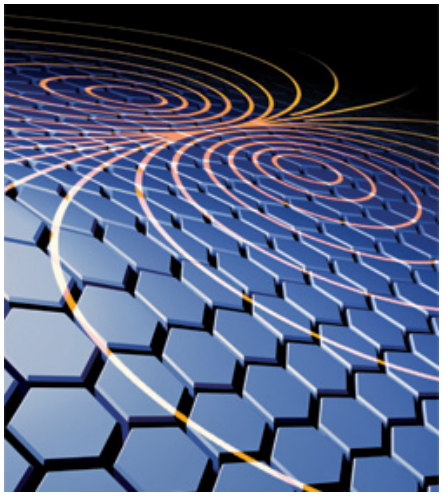
Applications:



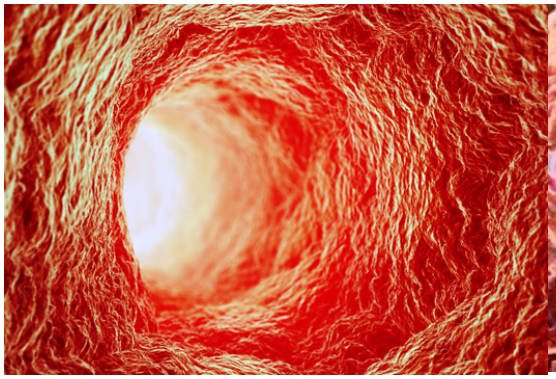
- Measure structural conditions

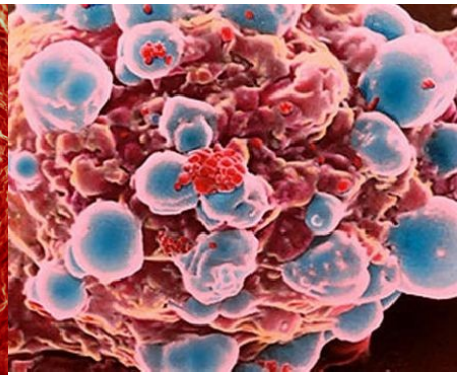
Self-assembling Nano-Sensors

Applications:



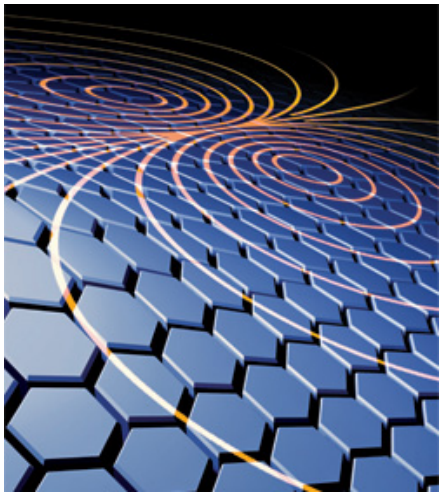
- Monitor health and assist medical surgery



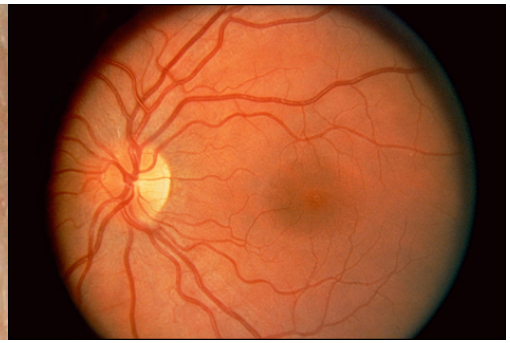


Self-assembling Nano-Sensors

Applications:

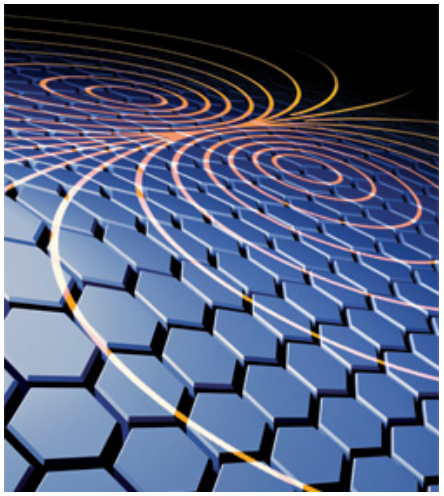


- Artificial skin or retina



Self-assembling Nano-Sensors

Basic approach: nano-sensors can move and bond

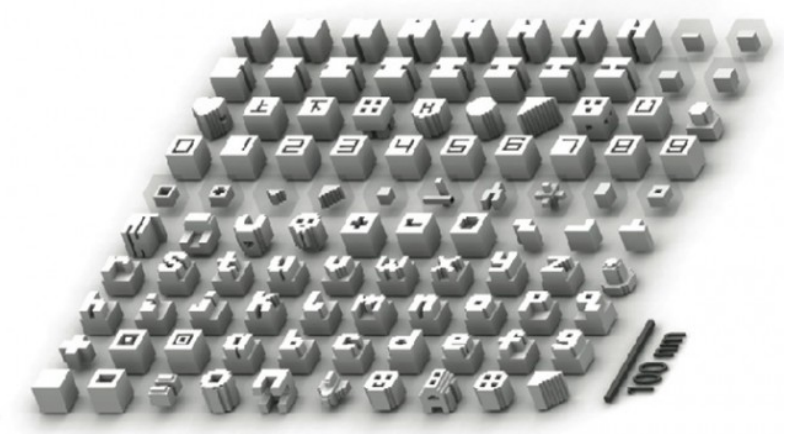


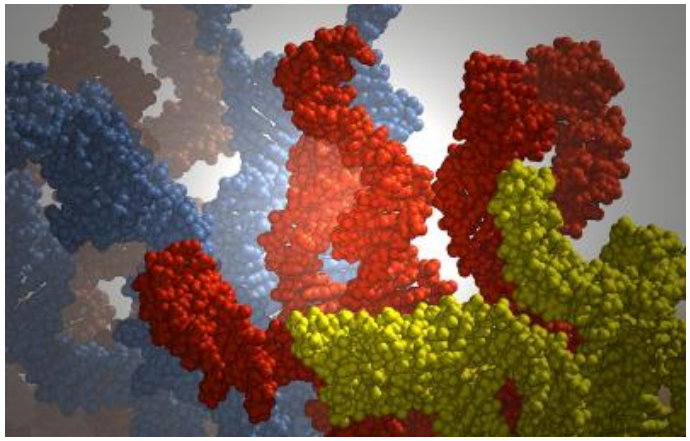
Cases:

- sensors are passive
- sensors are intelligent (finite automata)

Self-assembling DNA Structures

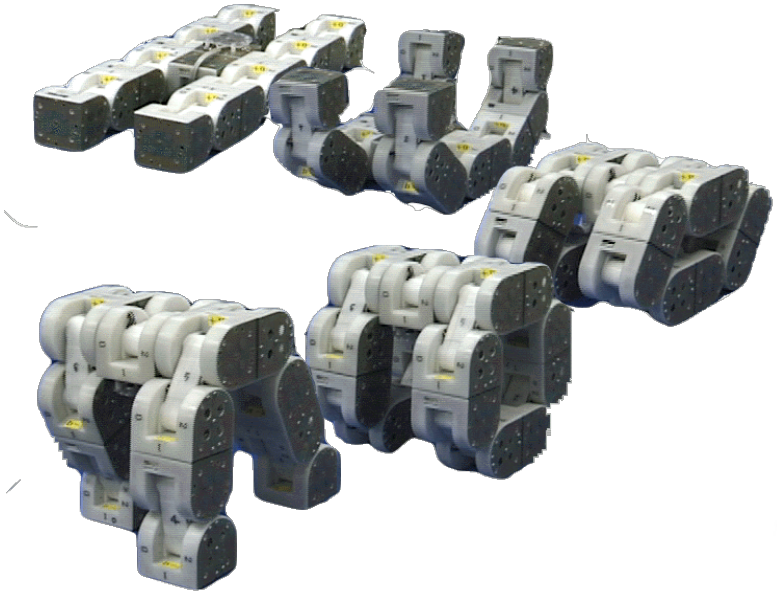
Self-organizing, passive elements have been studied before in the context of self-assembling DNA structures.





Modular Robotics

Self-organizing, intelligent elements have been studied before in modular robotics.



Self-Assembling Nano-Sensors

Basic problems:

- Sensors self-organize to form a predefined shape

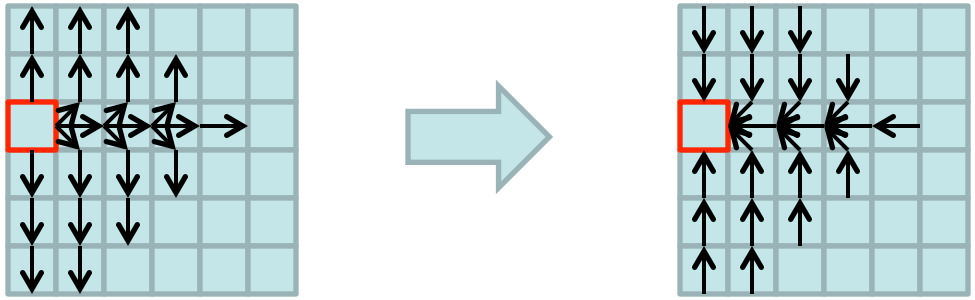
- Sensors self-organize in order to coat a given surface

Self-Assembling Nano-Sensors

Aggregation of information:

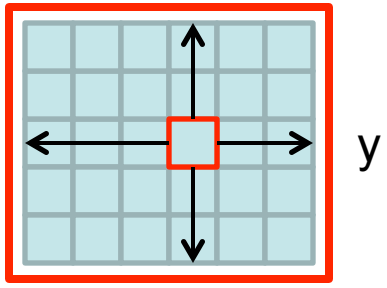
Via directed diffusion

observer
(human, computer)



Localization of events:

border is observed



Overview

Passive nano-sensors:

- DNA self-assembly

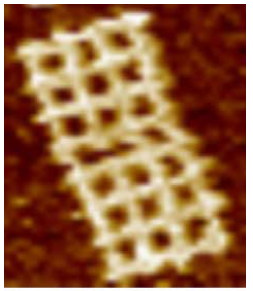
Intelligent nano-sensors:

- Finite automata and cellular automata
- Models for self-assembling nano-sensors
- Some problems and initial results

DNA self-assembly

DNA self-assembly useful in two respects:

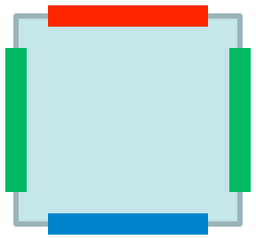
- Self-assembling nano-circuitry to establish a nano-sensor



- Self-assembling structures of nano-sensors

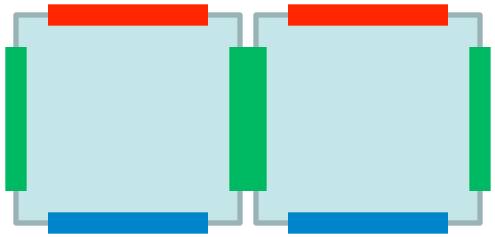
DNA self-assembly

Basic building block: Wang tile.



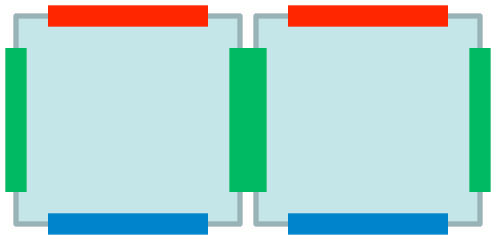
- Every side has a color out of Σ.
- So a Wang tile is a tuple $T \in \Sigma^4$.

Two Wang tiles bond if their sides have the same color.



DNA self-assembly

Two Wang tiles bond if their sides have the same color.

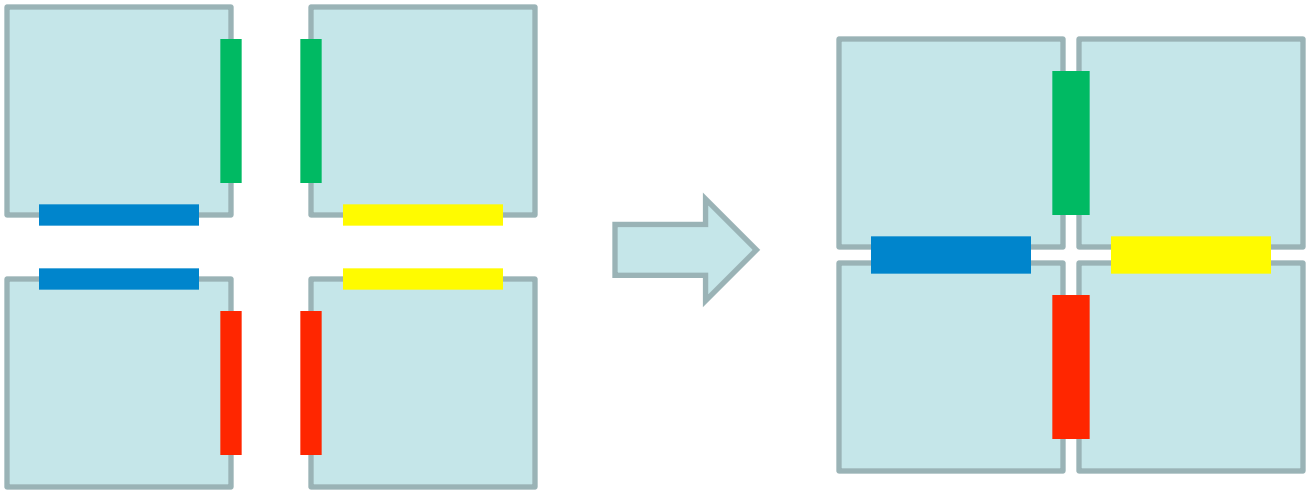


- Each bond color c has a strength $s(c) \in \mathbb{N}$.
- A tile is added to an assembly:
  summed strength of bonds $\geq T$.

DNA self-assembly

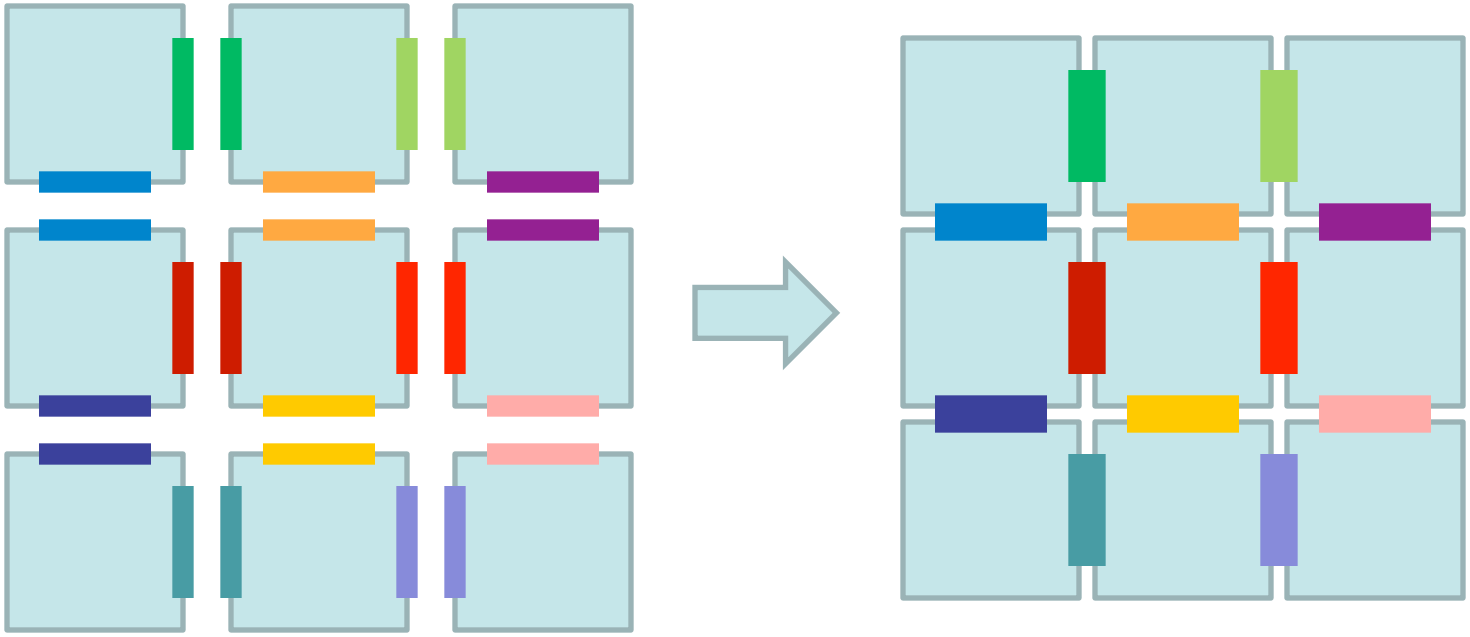
Problem: What is the minimum number of tile types needed to form a n×n-square for some temperature T?

T=1:



DNA self-assembly

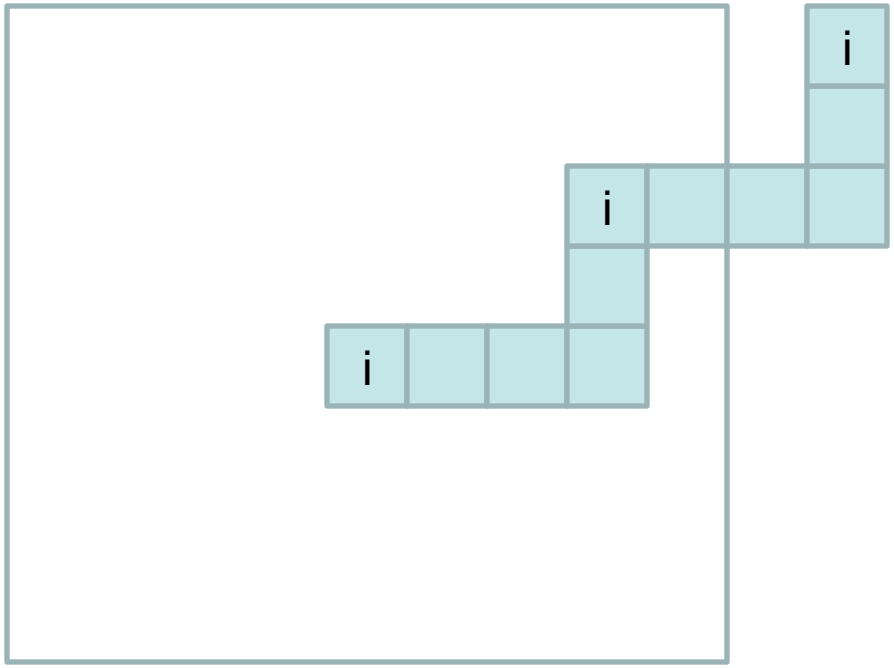
3x3-square:



Question: Is it possible to use less than n^2 tile types for a $n \times n$-square?

DNA self-assembly

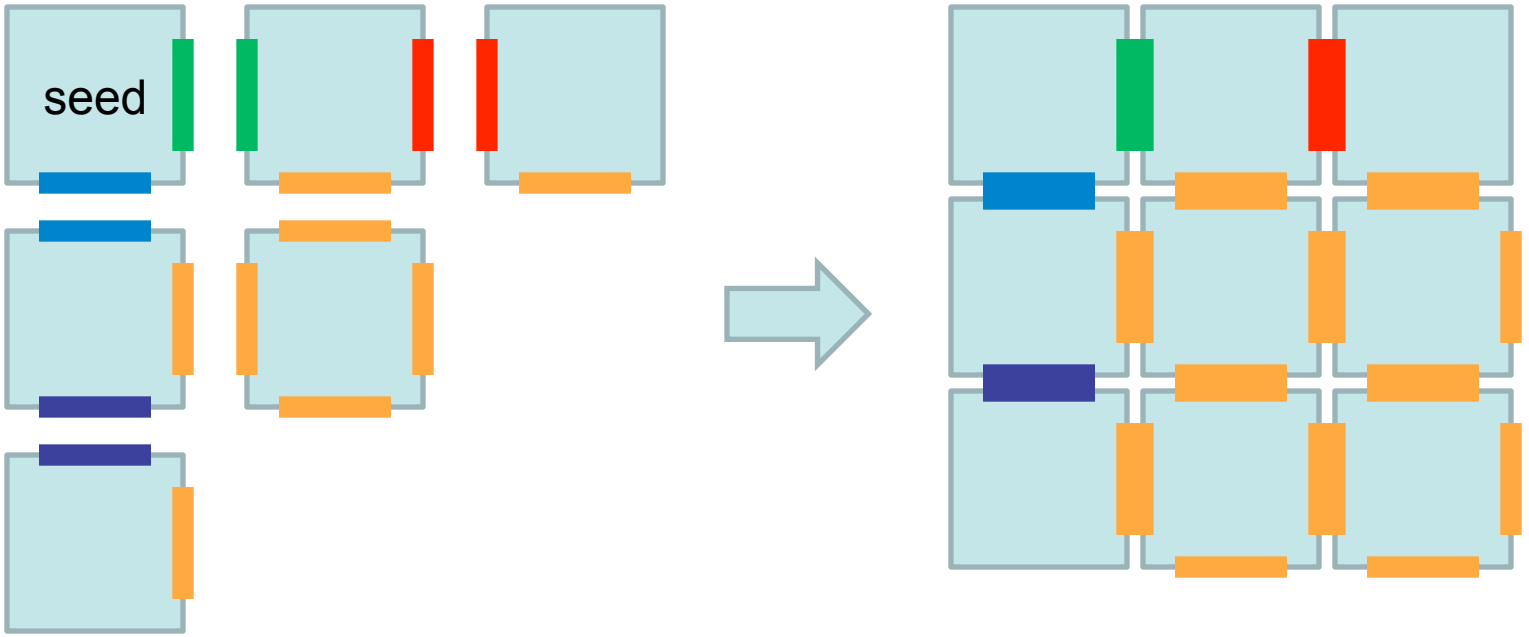
No!



DNA self-assembly

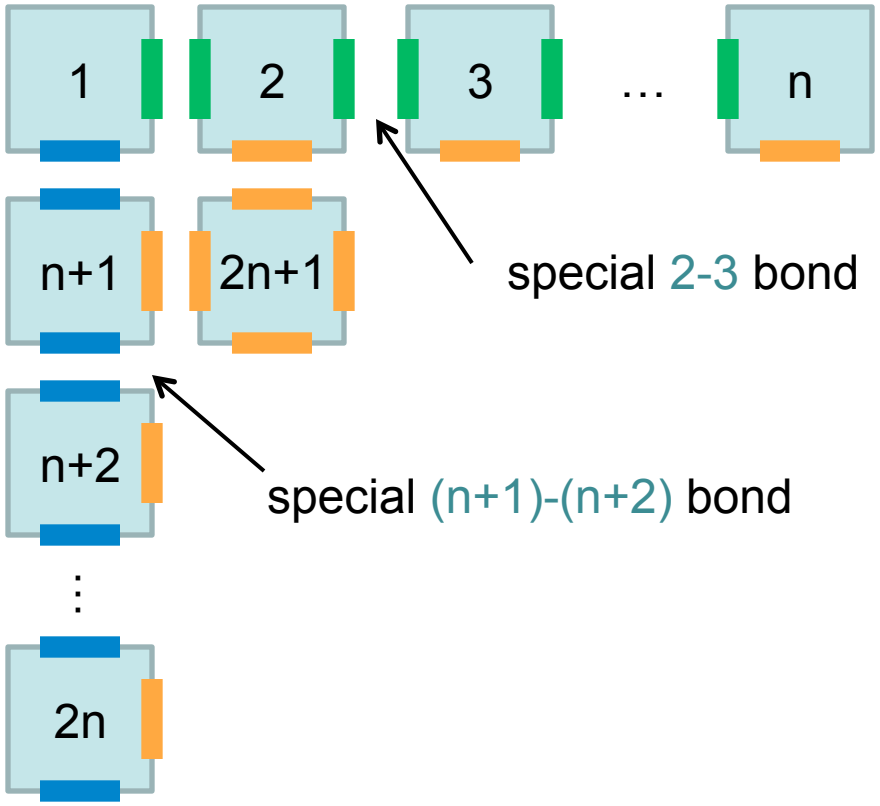
T=2:

- : strength 2
- : strength 1



DNA self-assembly

T=2, general construction:



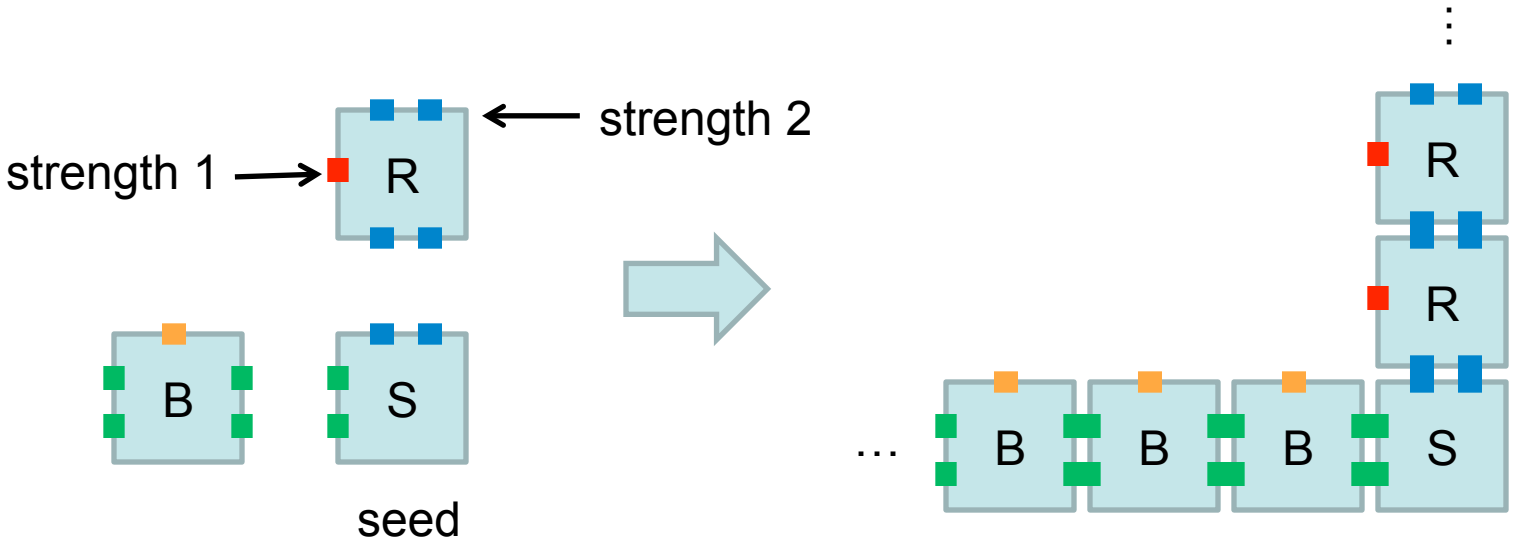
2n+1 tile types are sufficient

DNA self-assembly

Can we do better for T=2?

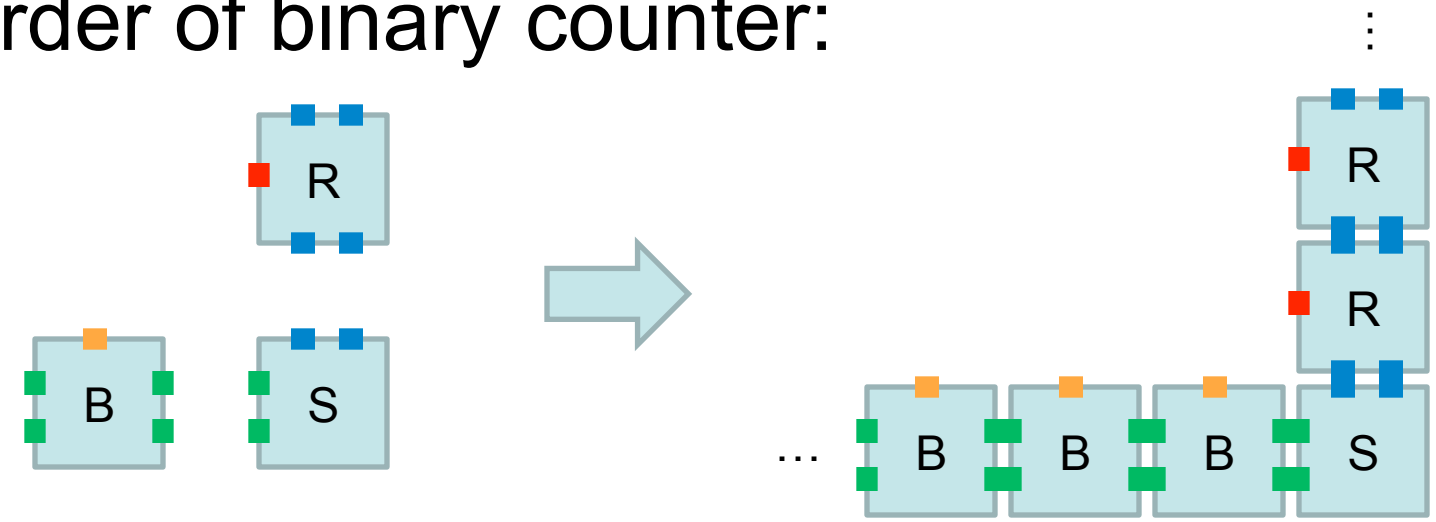
Yes, we can with a binary counter approach.

Border squares of binary counter:

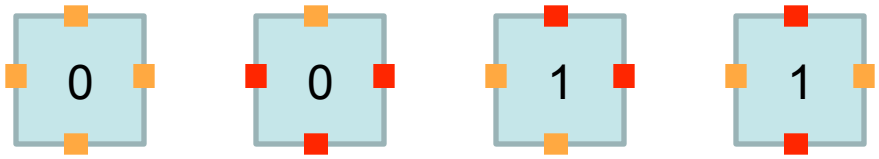


DNA self-assembly

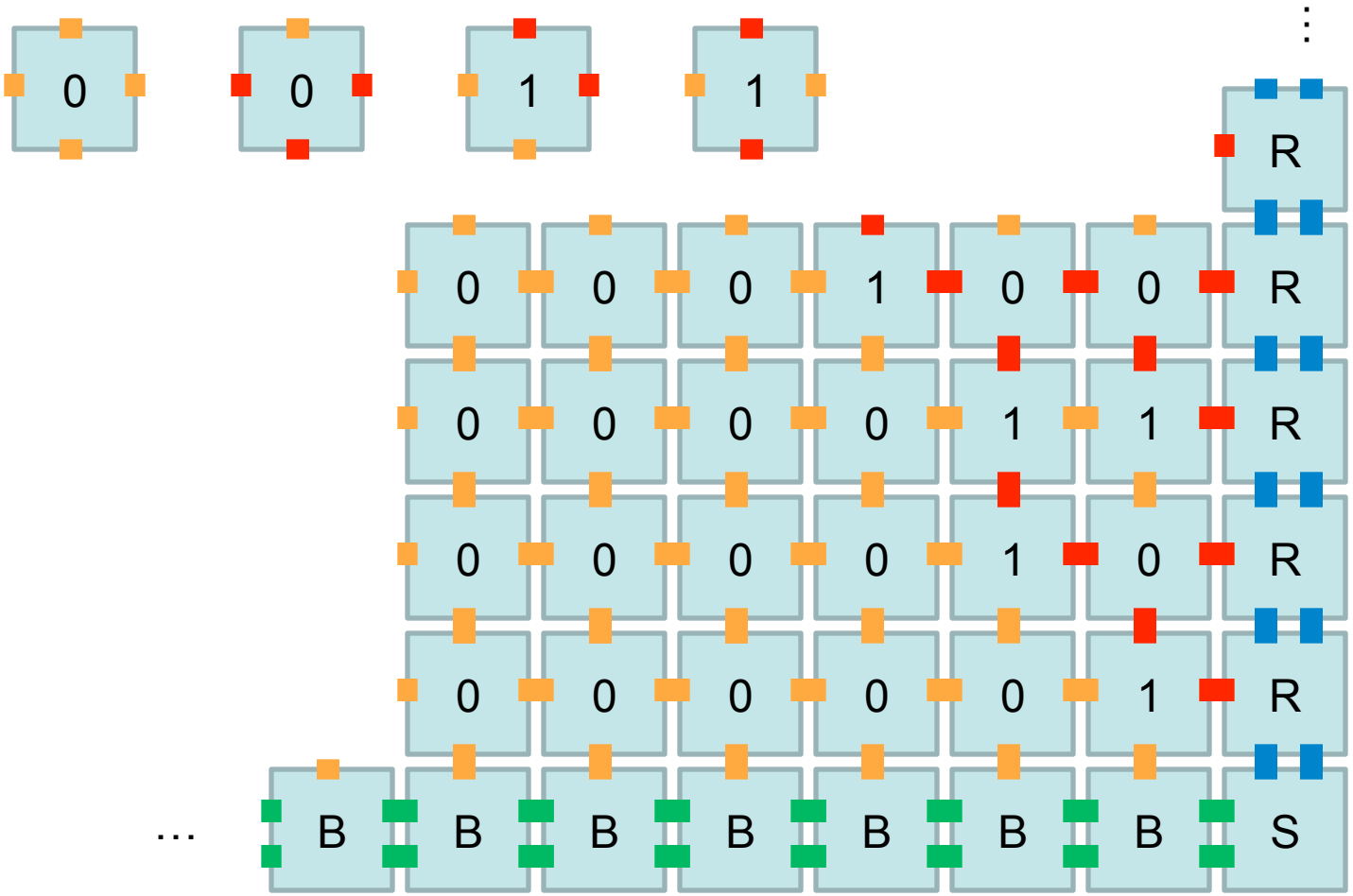
Border of binary counter:



Tiles for binary counter:

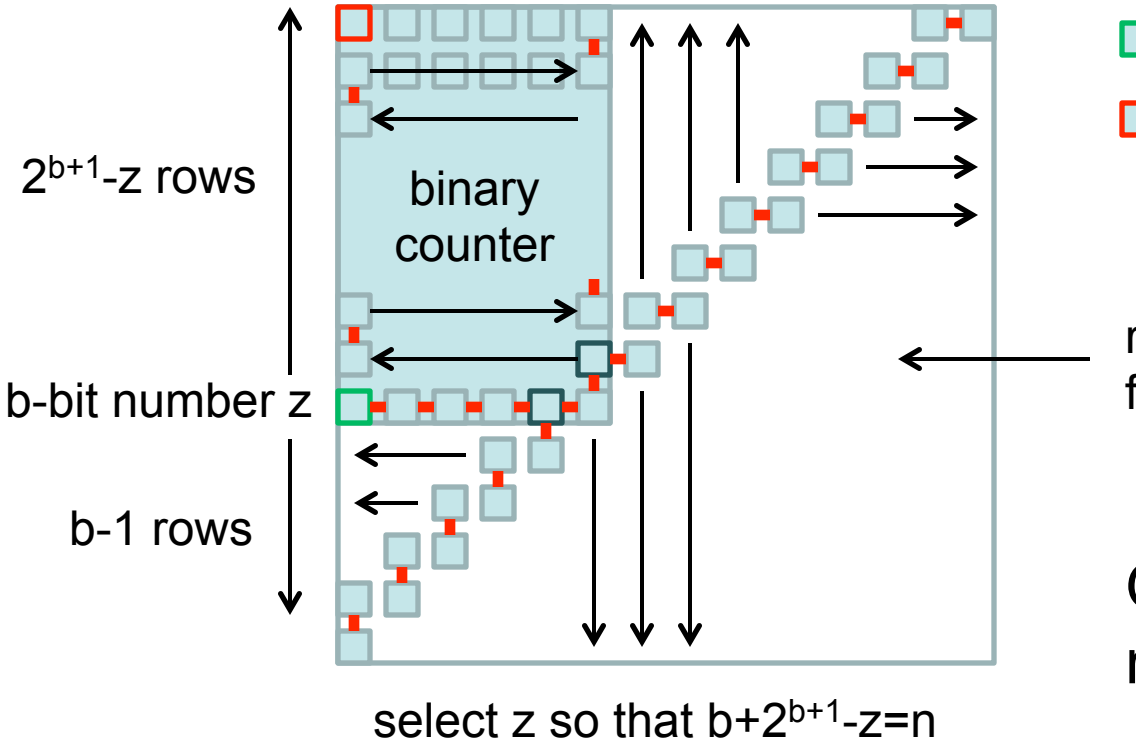


DNA self-assembly



DNA self-assembly

Strategy to construct n×n-square:



: seed tile of counter

: final tile of counter

: bond of strength 2

rest filled with generic filling tile

Only O(log n) tiles needed!

DNA self-assembly

One can do even better:

Theorem: For an infinite set of n-values, only f(n) tiles are needed to construct a n×n-square, where f(n) can be an arbitrarily slow growing function.

Theorem: There is an infinite set of n-values that require Ω(log n / loglog n) tiles to construct a n×n-square.

Proof:
after a few slides

For more details see:
P. Rothemund and E. Winfree
The program-size complexity of self-assembling squares
In ACM STOC 2000.

DNA self-assembly

Given a shape S let

- T(S) be the minimum number of tiles needed to construct S and
- K(S) be the size of the minimum program ouputting S as a list of locations (also known as the Kolmogorov complexity of S)

Theorem: For any shape S,

$$K(S) = \Theta(T(S) \log T(S))$$

Proof:

D. Soloveichik and E. Winfree
Complexity of self-assembled shapes
SIAM Journal on Computing 36(6), 2007.

DNA self-assembly

Kolmogorov complexity:

- Easier to understand for natural numbers.

Theorem: There are numbers $n \in \mathbb{N}$ with Kolmogorov complexity $\Omega(\log n)$.

Proof:

- Consider any $k \in \mathbb{N}$.
- There are 2^k numbers that have a k-bit binary representation.
- Each of these numbers must have a unique smallest program to construct it.
- Therefore, each of these programs must have a unique binary representation.
- Hence, one of these numbers must have a program of size $\Omega(\log n)$.

Note: most of prime numbers have Kolmogorov complexity $o(\log n)$.

DNA self-assembly

What is the Kolmogorov complexity of a n×n-square?

- From the previous theorem, there must be a n×n-square with Kolmogorov complexity $\Omega(\log n)$.
- Hence, it follows from $K(S) = \Theta(T(S) \log T(S))$ that there are n×n-squares that require $\Omega(\log n / \log\log n)$ tiles, which confirms our previous lower bound.

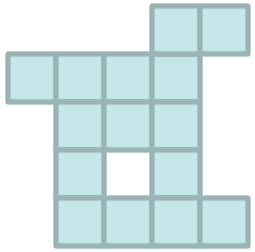
DNA self-assembly

How to design tiles for arbitrary shapes?

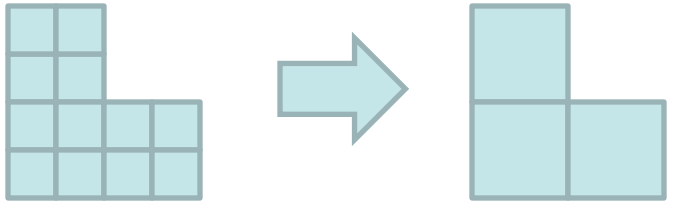
Definition: An irreducible shape is a shape in which we cannot shrink $k \times k$-slots to 1×1-slots without changing the shape.

Examples:

Irreducible shape:



Reducible shape:



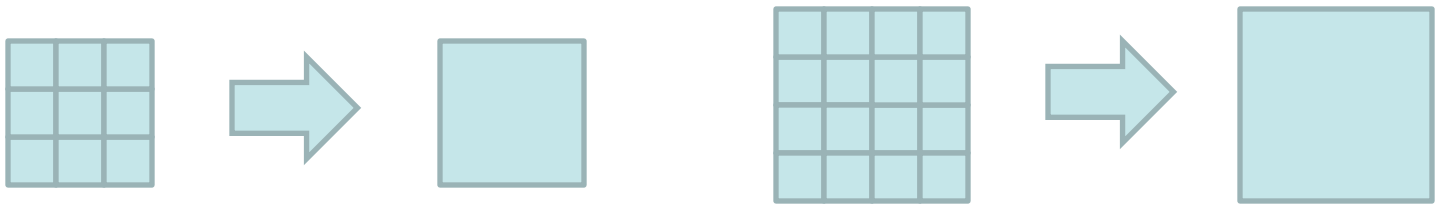
DNA self-assembly

How to design tiles for arbitrary shapes?

Definition: An irreducible shape is a shape in which we cannot shrink $k \times k$-slots to 1×1-slots without changing the shape.

Examples:

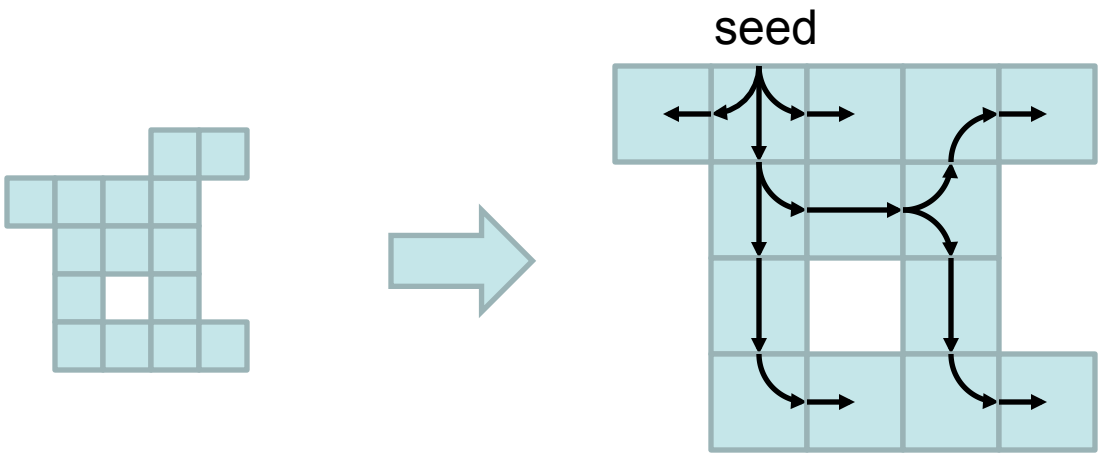
Reducible shapes:



DNA self-assembly

Irreducible shape:

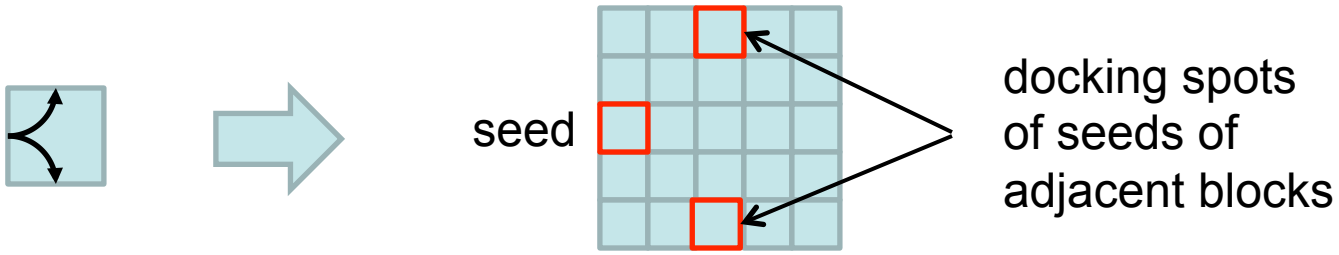
- Each slot gets its own tile type.
- The tiles form a spanning tree of the shape.



DNA self-assembly

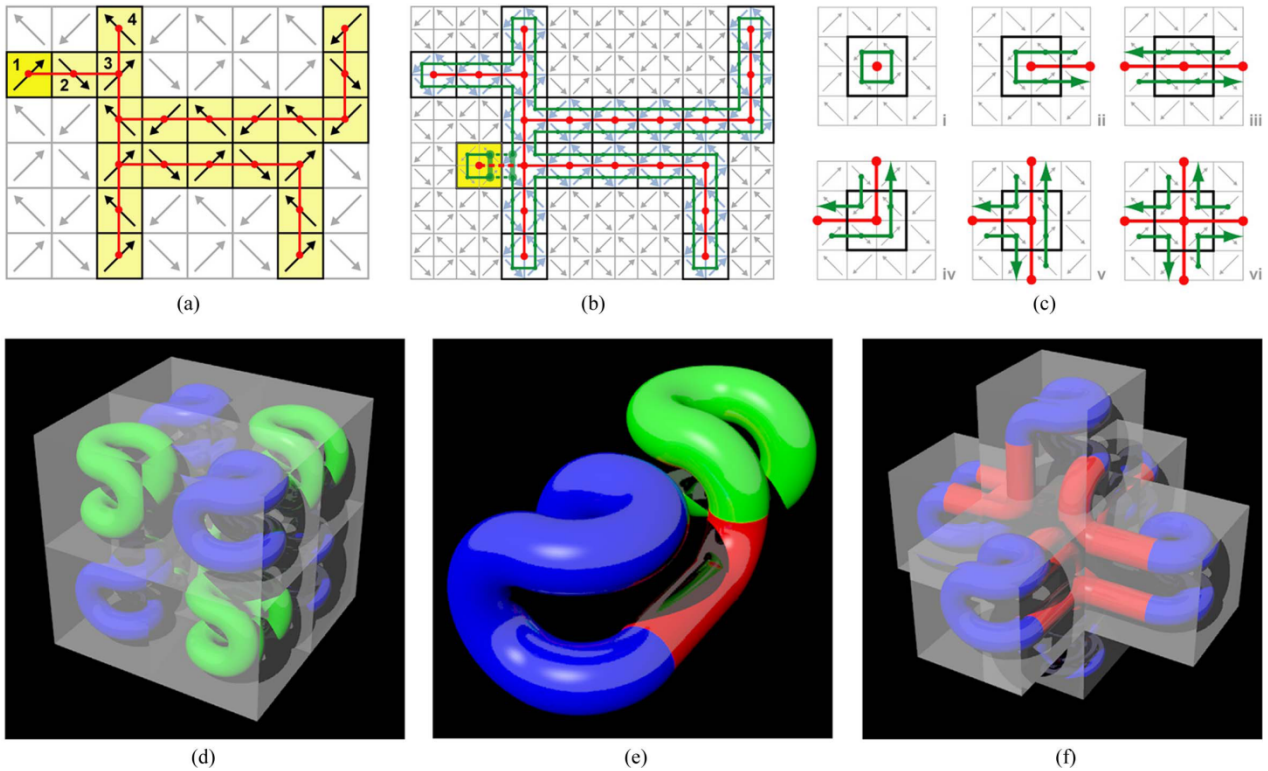
Reducible shape S:

- Determine the unique k that shrinks S to an irreducible shape S´.
- Construct a tile system T´ for S´.
- Construct a tile system T for S so that for each tile $T'_i \in T'$, a k×k-square B_i is built.



DNA self-assembly

Alternative approach: foldable strings



DNA self-assembly

Basic approach:

- Consider any shape S.
- Determine the unique k that shrinks S to an irreducible shape S´.
- Construct a (foldable) string L´ for S´.
- Extend every unit in L´ by a foldable substring folding into a k×k-square.

DNA self-assembly

Wang tiles:

- For an introduction see
  Matthew J. Patitz
  An Introduction to Tile-Based Self-Assembly
- For a simulator see
  http://www.self-assembly.net

Foldable strings: see
K. Cheung, E. Demaine, J. Bachrach, and S. Griffith
Programmable assembly with universally foldable strings (moteins)
In IEEE Transactions on Robotics 4(27), 2011.

Overview

Passive nano-sensors:

- DNA self-assembly

Intelligent nano-sensors:

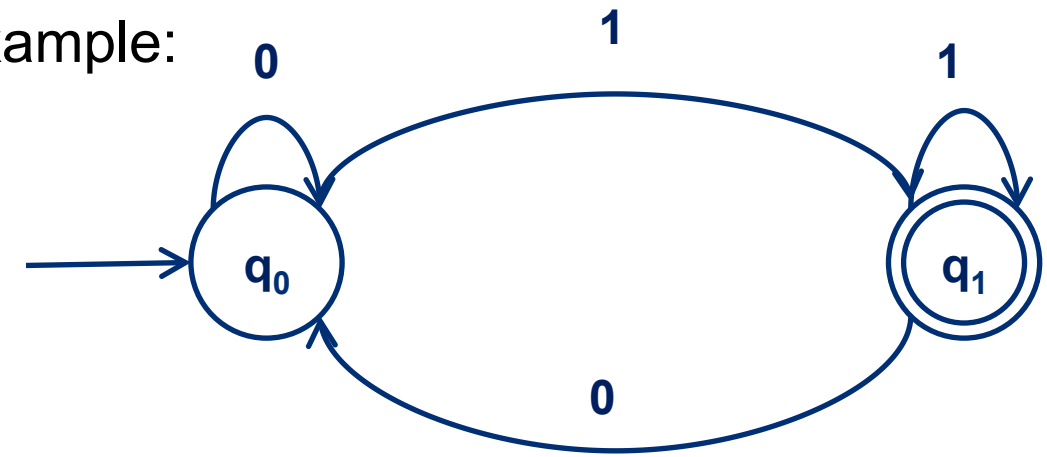
- Finite automata and cellular automata
- Models for self-assembling nano-sensors
- Some problems and initial results

Finite Automaton

A finite automaton M is a tuple $M=(Q, \Sigma, \delta, q_0, F)$ where

- Q is a finite set of states
- Σ is the input alphabet
- $\delta: Q \times \Sigma \rightarrow Q$ is the transition function
- $q_0 \in Q$ is the initial state
- $F \subseteq Q$ is the set of final states

Example:



- $Q=\{q_0,q_1\}$
- $\Sigma=\{0,1\}$
- $F=\{q_1\}$

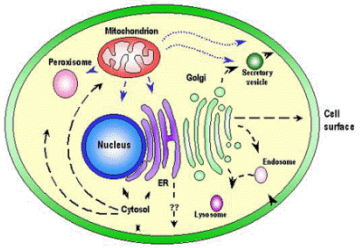
Finite Automaton

A finite automaton M is a tuple $M=(Q, \Sigma, \delta, q_0, F)$ where

- Q is a finite set of states
- Σ is the input alphabet
- $\delta: Q \times \Sigma \rightarrow Q$ is the transition function
- $q_0 \in Q$ is the initial state
- $F \subseteq Q$ is the set of final states

Advantage:

- easy to build



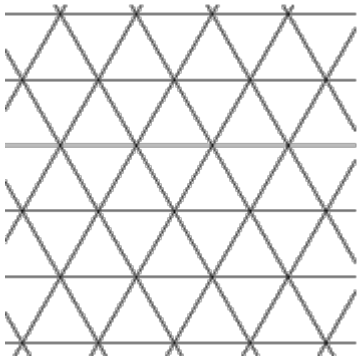
Disadvantage:

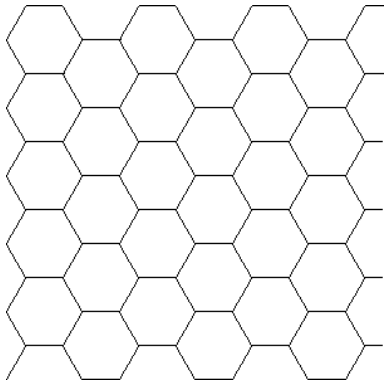
- very limited computational capabilities (regular languages)

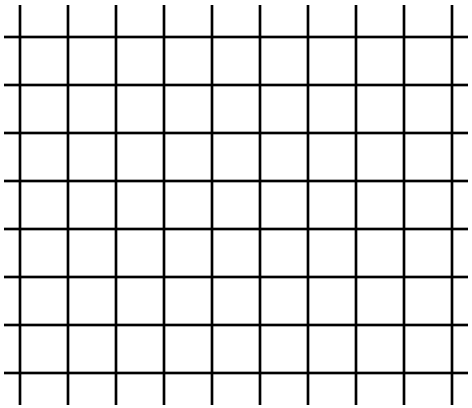
Cellular Automaton

A cellular automaton $C=(Q,\delta)$ is a finite automaton that determines the behavior of any cell in a grid.

Examples of grids:

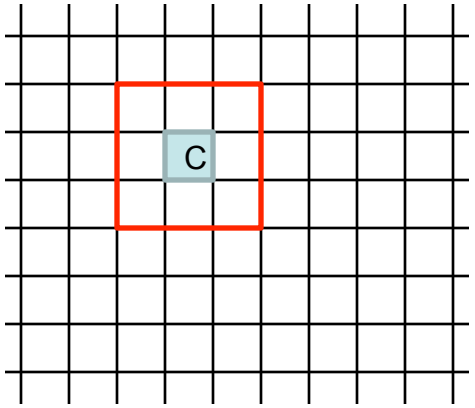






Cellular Automaton

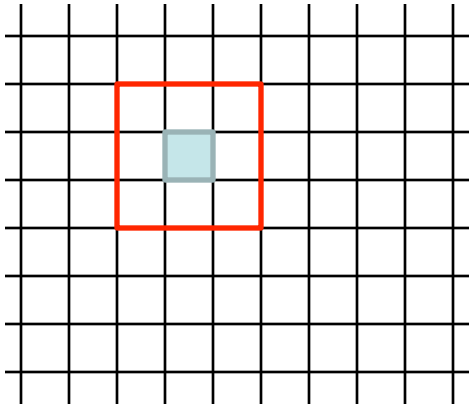
A cellular automaton $C=(Q,\delta)$ is a finite automaton that determines the behavior of any cell in a grid.



C takes its own state and all 8 neighboring states into account: $\delta:Q^9 \rightarrow Q$

Cellular Automaton

A cellular automaton $C=(Q,\delta)$ is a finite automaton that determines the behavior of any cell in a grid.



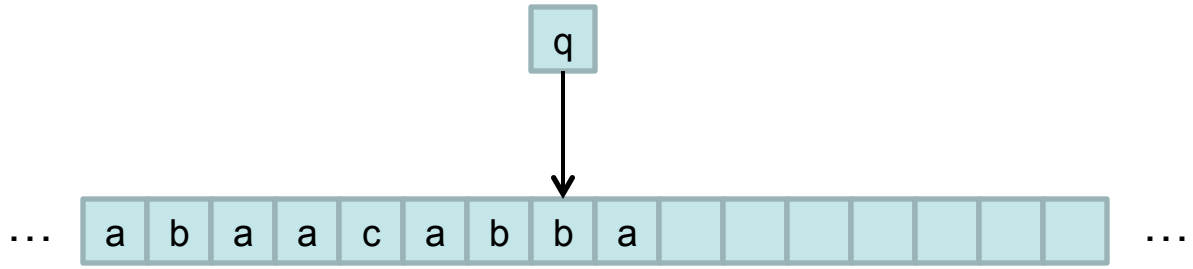
Standard assumption:
Cells act in a perfectly synchronized way, i.e., the system operates in discrete time steps t

Cellular Automaton

Theorem: On an infinite grid, a cellular automaton can emulate any Turing machine algorithm.

Turing machine:

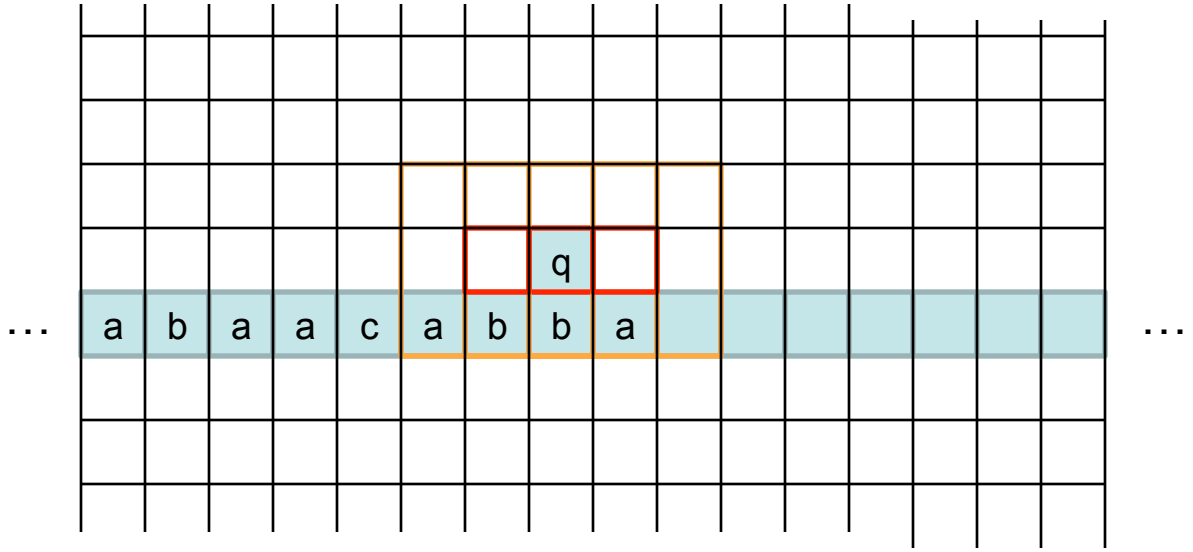
$\delta: Q \times \Sigma \rightarrow Q \times \{L,N,R\}$, i.e., a TM can move one cell to left or right in each step



Cellular Automaton

Theorem: On an infinite grid, a cellular automaton can emulate any Turing machine algorithm.

Cellular automaton:

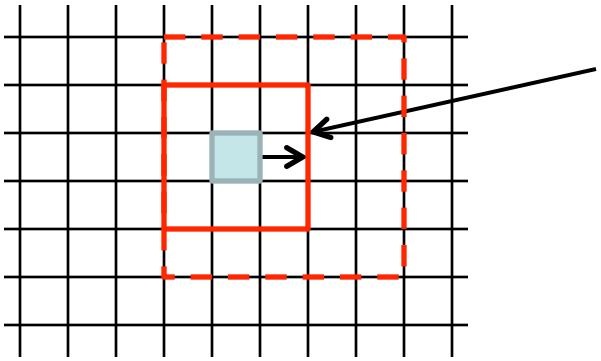


Cellular Automaton

Are cellular automata an appropriate model for self-assembling nano-sensors?

Rule:

- Each nano-sensor (particle) can observe its direct neighborhood and decide based on that where to move next

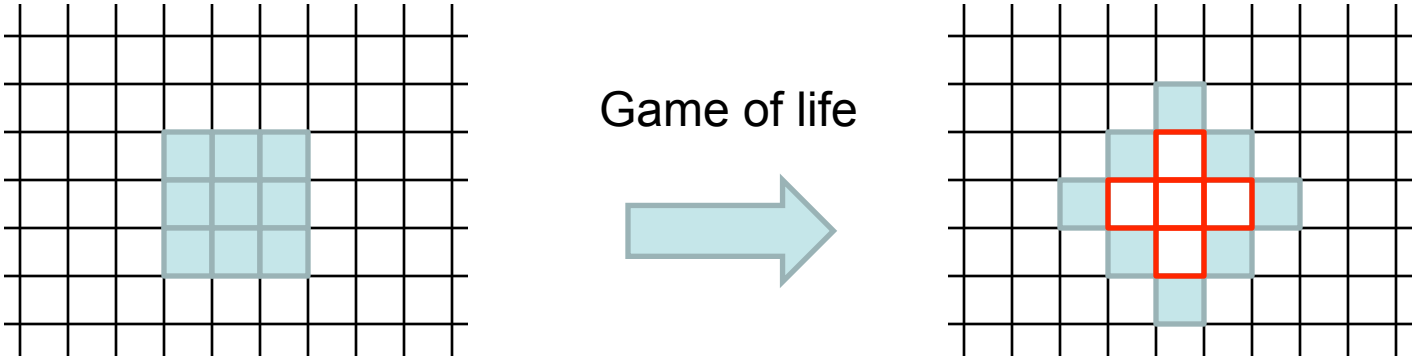


New position needs to know 2-neighborhood to decide that!

Cellular Automaton

Are cellular automata an appropriate model for self-assembling nano-sensors?

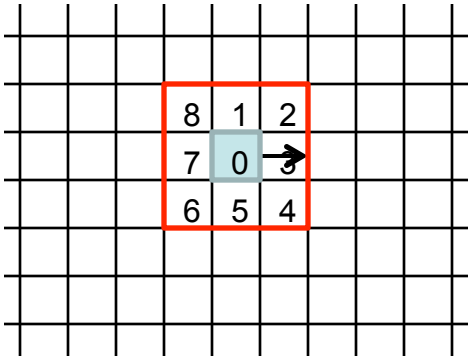
- On the other hand, a cellular automaton can do many things a particle is not allowed to do



Particle Model

Better: separate particle model

- A particle is a finite automaton of the form $P=(Q,\delta)$ where $\delta:Q^9 \rightarrow Q \times \{0,\ldots,8\}$

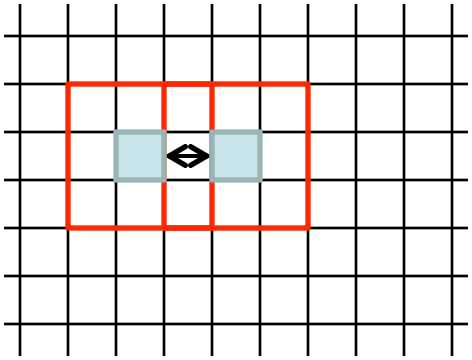


- A particle can only move to an empty spot (otherwise difficult to backoff as previous position might get occupied by another particle).

Particle Model

Better: separate particle model

- A particle is a finite automaton of the form $P=(Q,\delta)$ where $\delta:Q^9 \rightarrow Q \times \{0,\ldots,8\}$

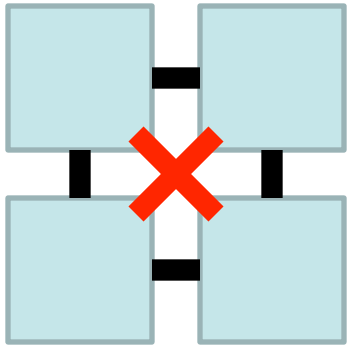


- If two particles collide, they backoff to their previous spot.

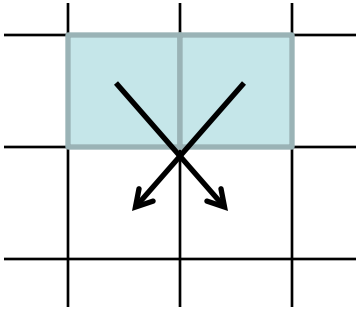
Particle Model

Technical problems:

- How can a particle interact with all 8 neighboring particles?



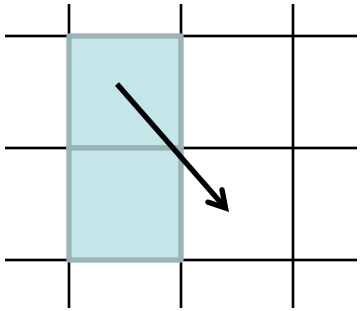
- How can two diagonally moving particles pass each other?



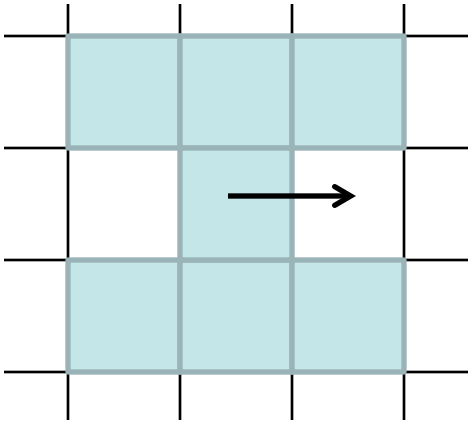
Particle Model

Technical problems:

- How does a particle move around the corner?



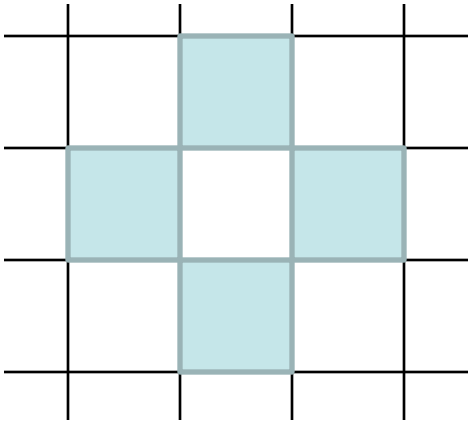
- How can particles move while preserving connec-tivity?



Particle Model

Technical problems:

- Are diagonal connections sufficient for connectivity?

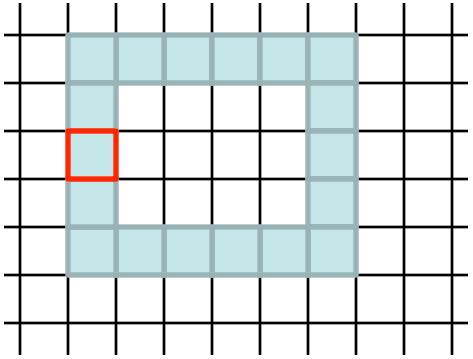


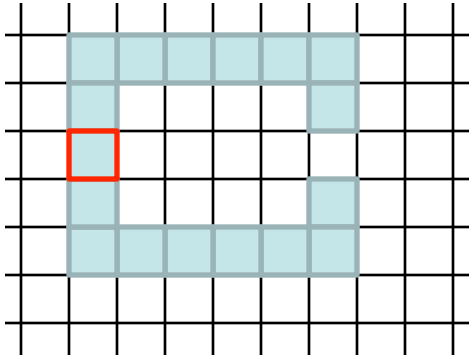
- We will see that deterministic particles can run into problems.
  How to realize randomization?

Particle Model

Algorithmic problems:

- How can a particle find out whether its movement would disconnect the structure?

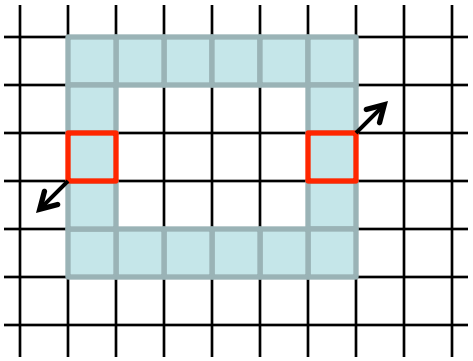


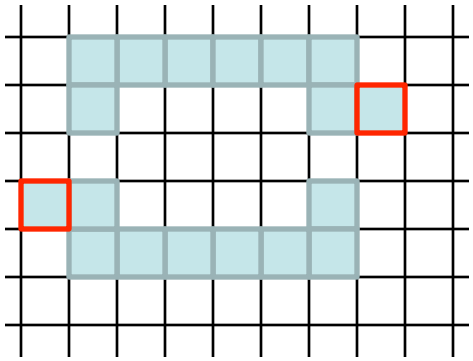


Particle Model

Algorithmic problems:

- Even if the particles had connectivity information, how can joint movements be coordinated?

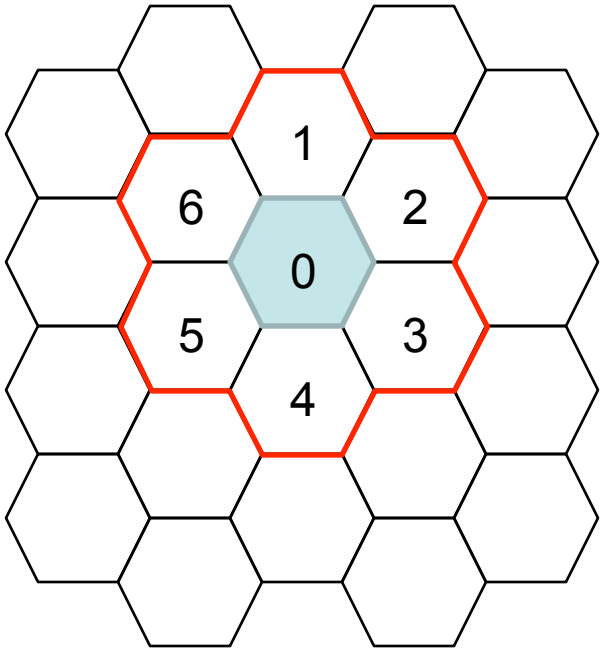




→ also randomization needed to break symmetry!

Particle Model

How about a different grid:

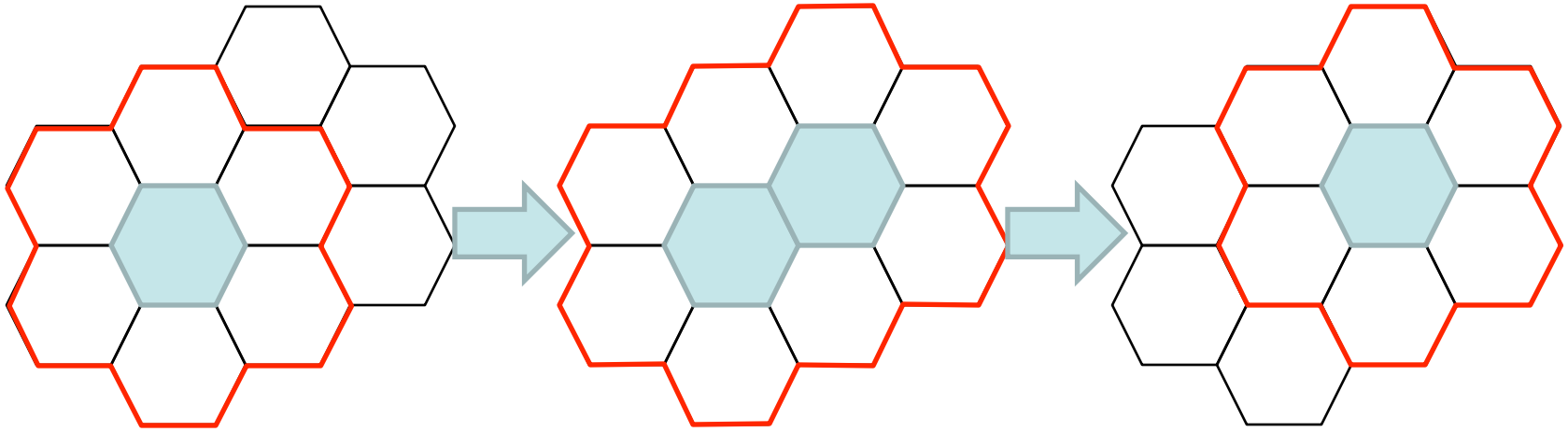


A particle is a finite automaton of the form $P=(Q,\delta)$ with

$\delta:Q^7 \rightarrow Q \times \{0,\ldots,6\}$

Particle Model

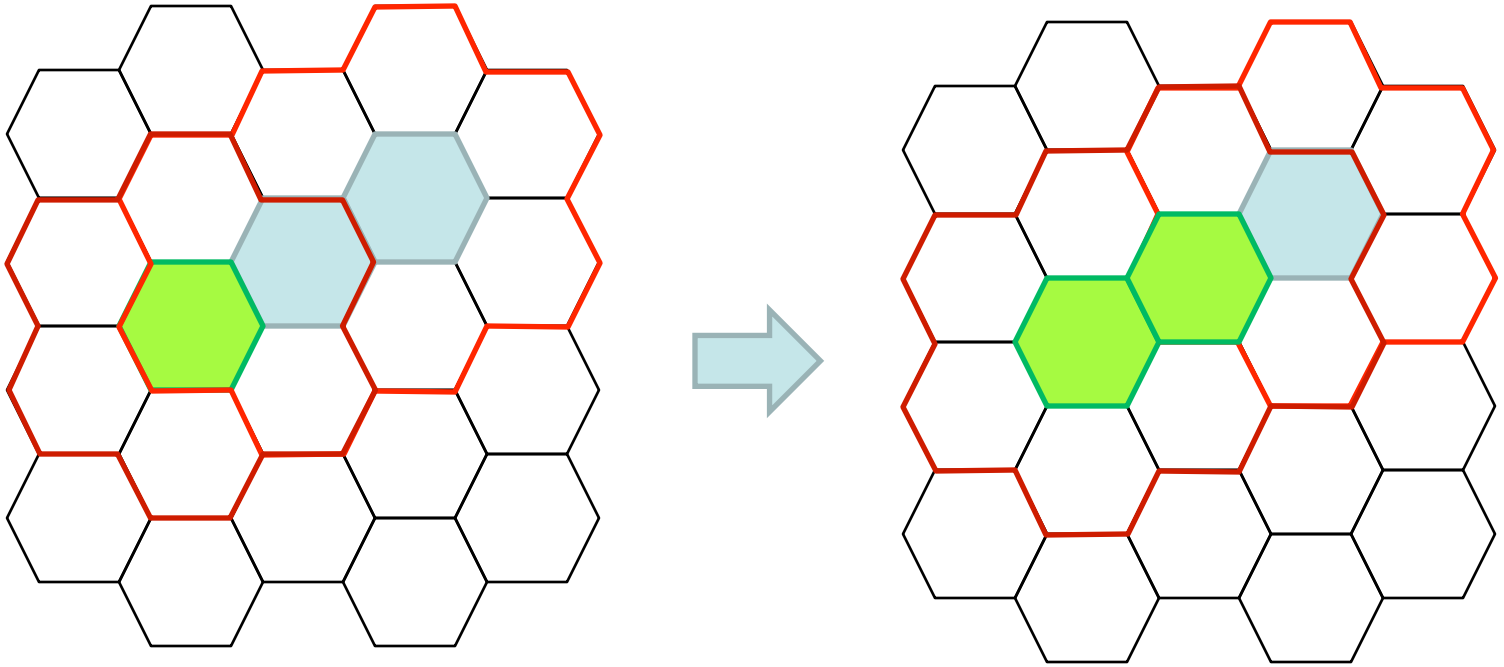
Movement of a particle:



A particle expands and contracts.

Particle Model

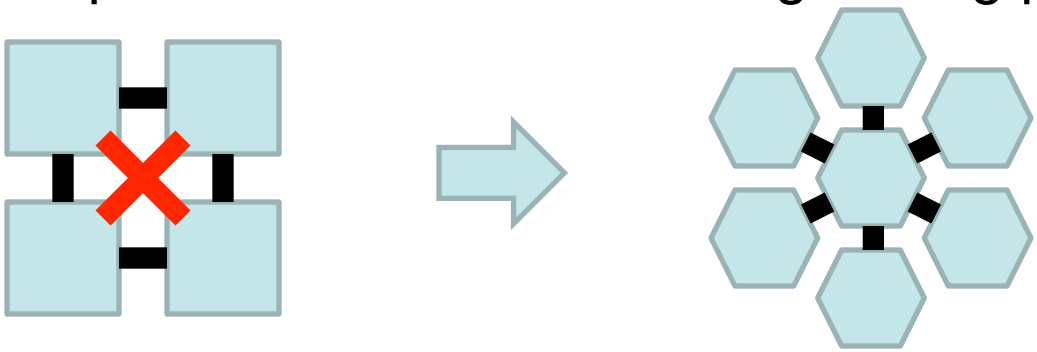
This also allows handover of slot without running into backoff problems.



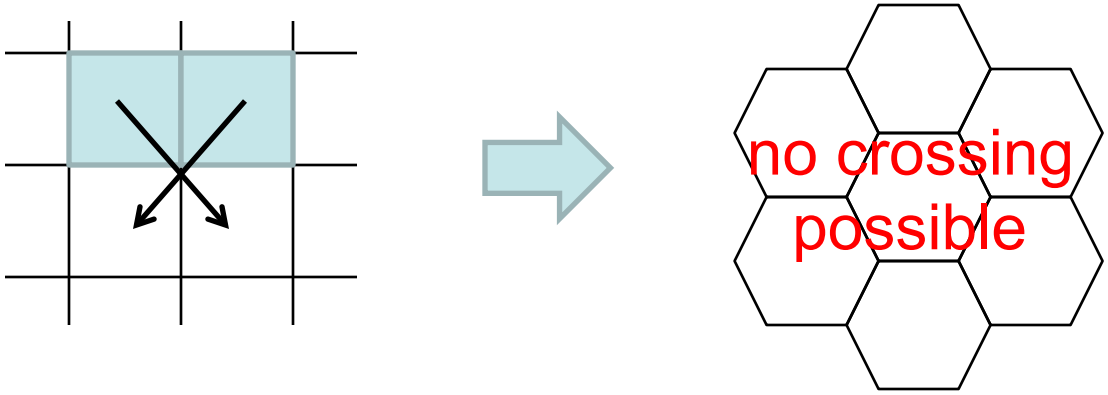
Particle Model

Technical problems:

- How can a particle interact with all neighboring particles?



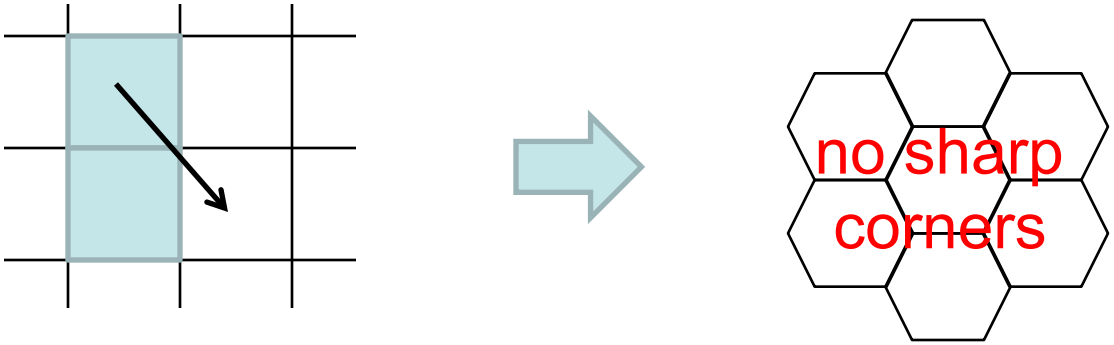
- How can two diagonally moving particles pass each other?



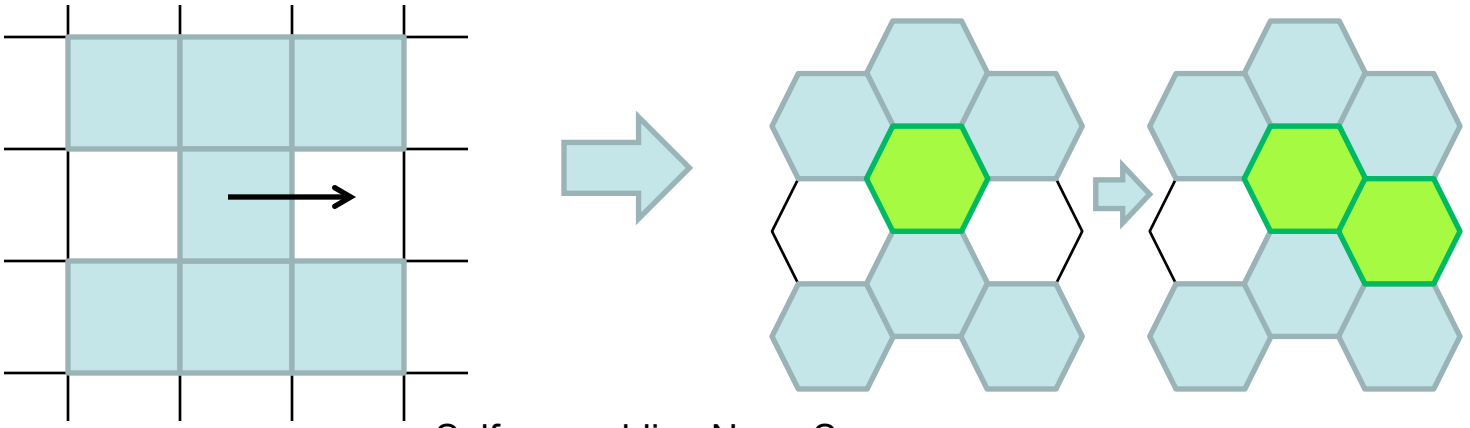
Particle Model

Technical problems:

- How does a particle move around the corner?



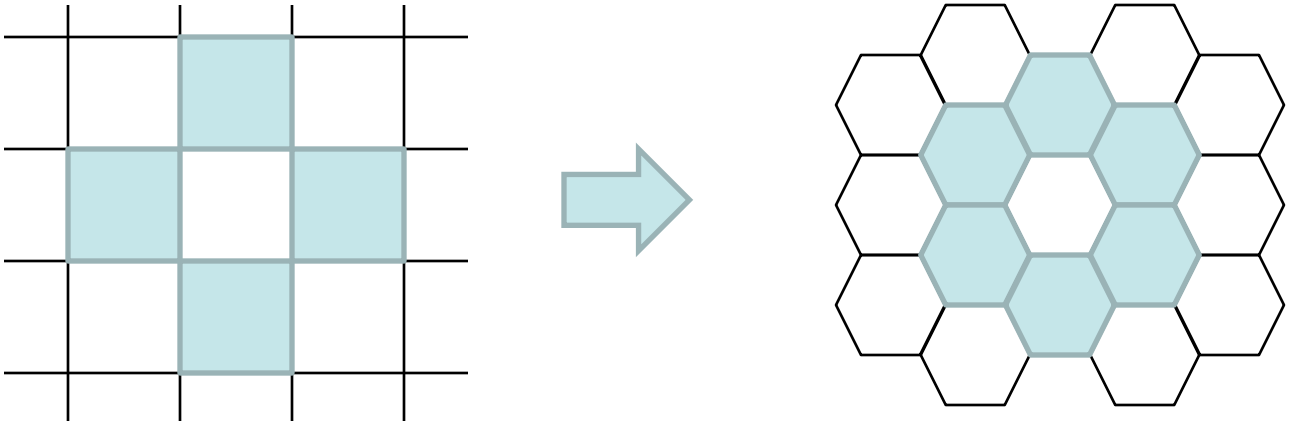
- How can particles move while preserving connectivity?



Particle Model

Technical problems:

- Are diagonal connections sufficient for connectivity?

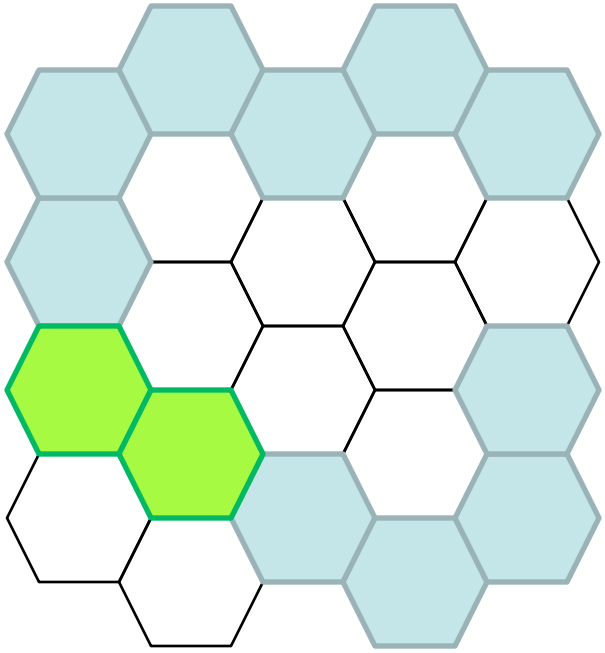


- Deterministic particles may still run into problems.
  How to realize randomization?

Particle Model

Algorithmic problems:

- How can a particle find out whether its movement would disconnect the structure?



Still a hard problem…

Self-Assembling Nano-Sensors

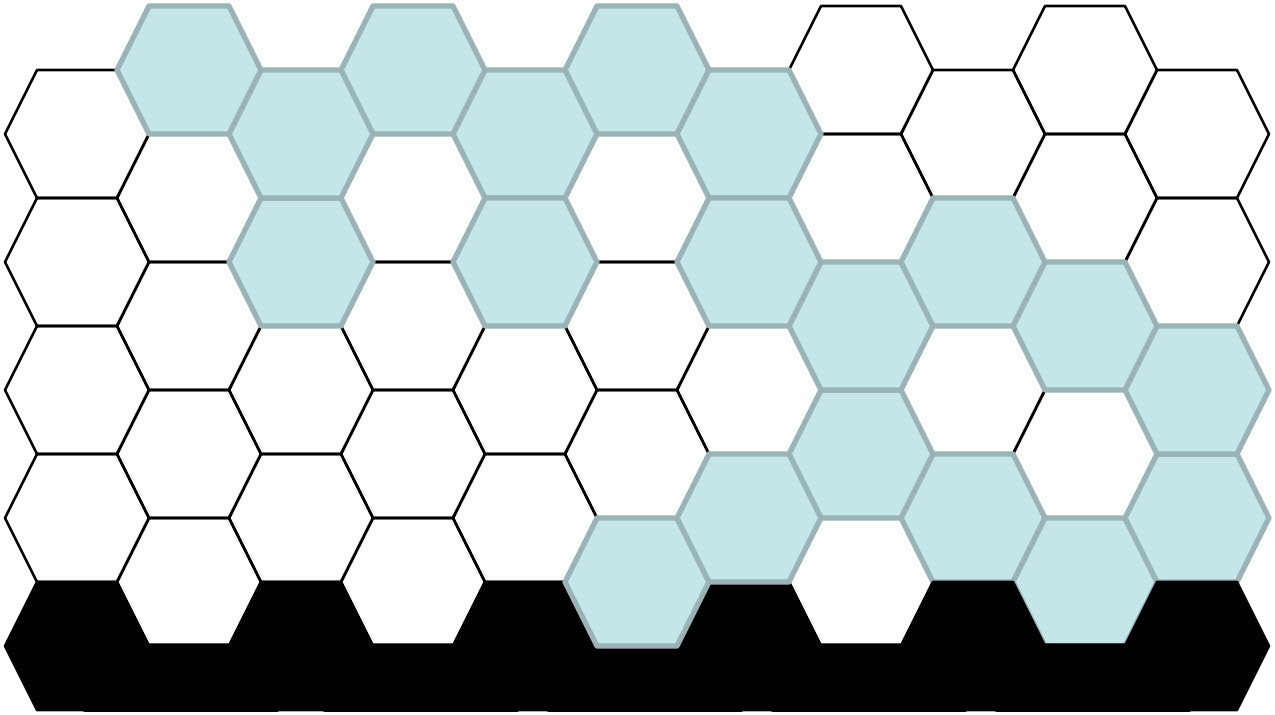
Basic problems:

- Sensors self-organize to form a predefined shape

- Sensors self-organize in order to coat a given surface

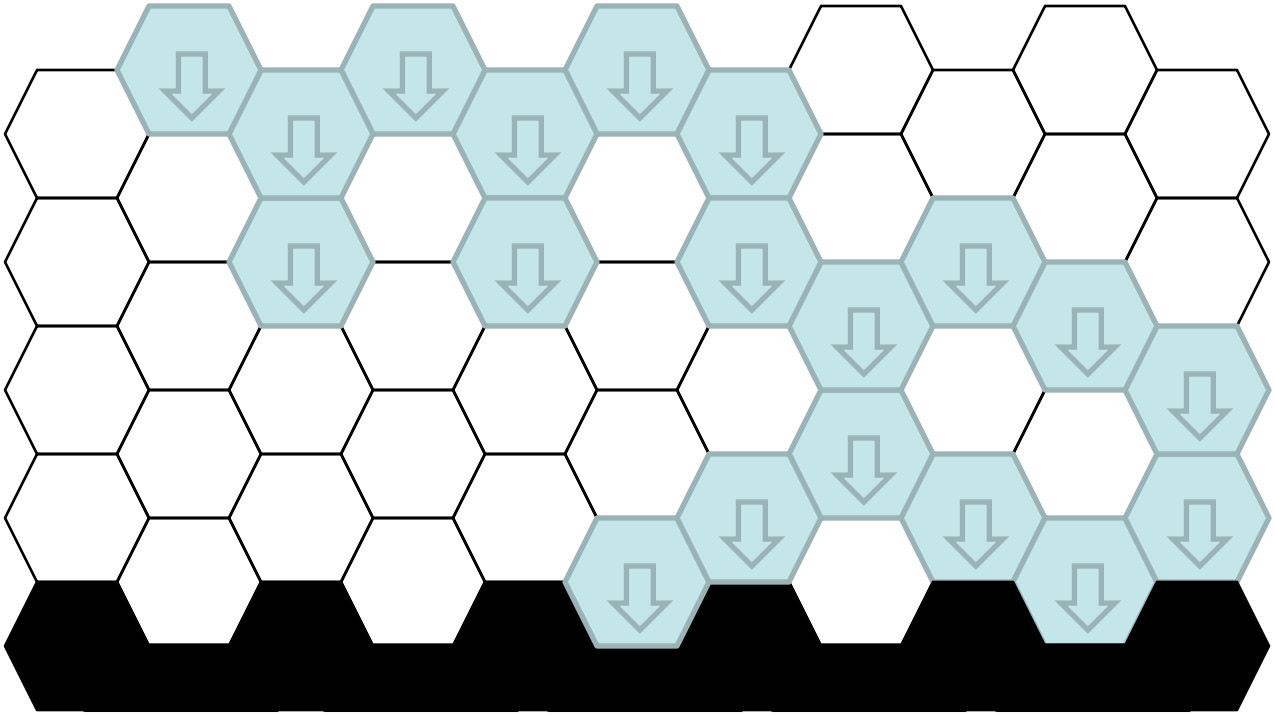
Covering Problem

Consider the problem of covering a surface.



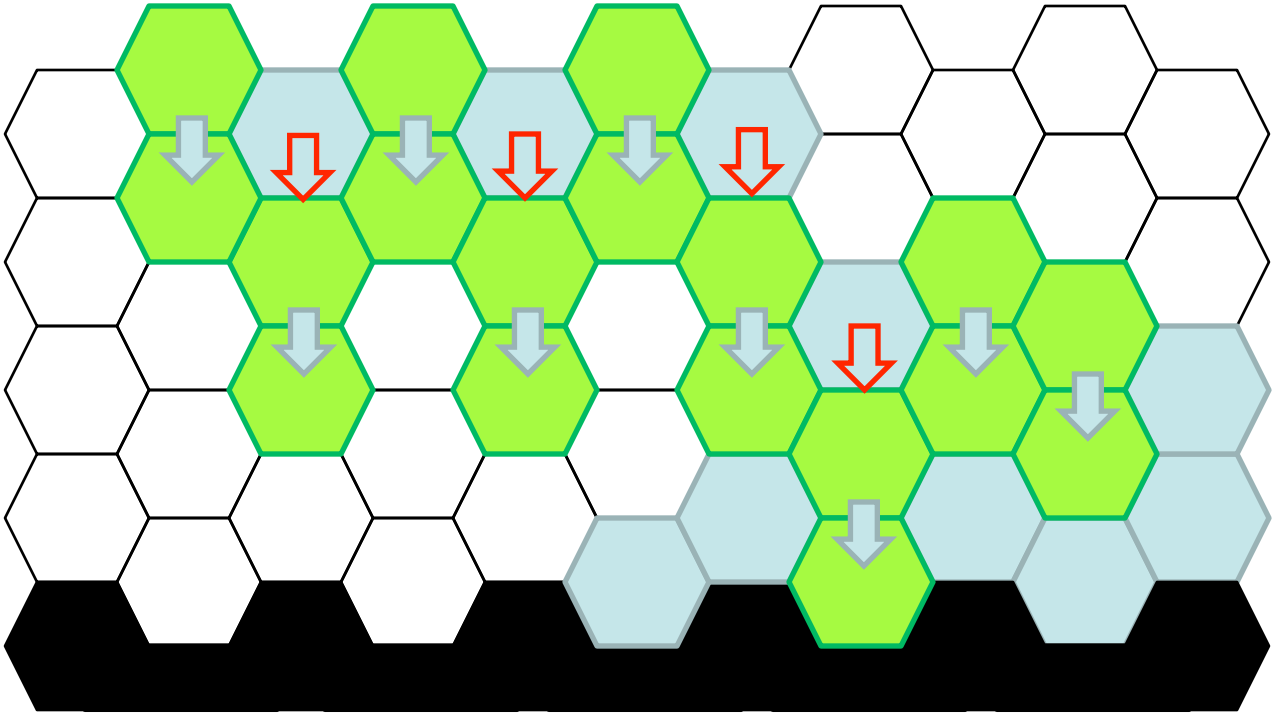
Covering Problem

Consider the problem of covering a surface.



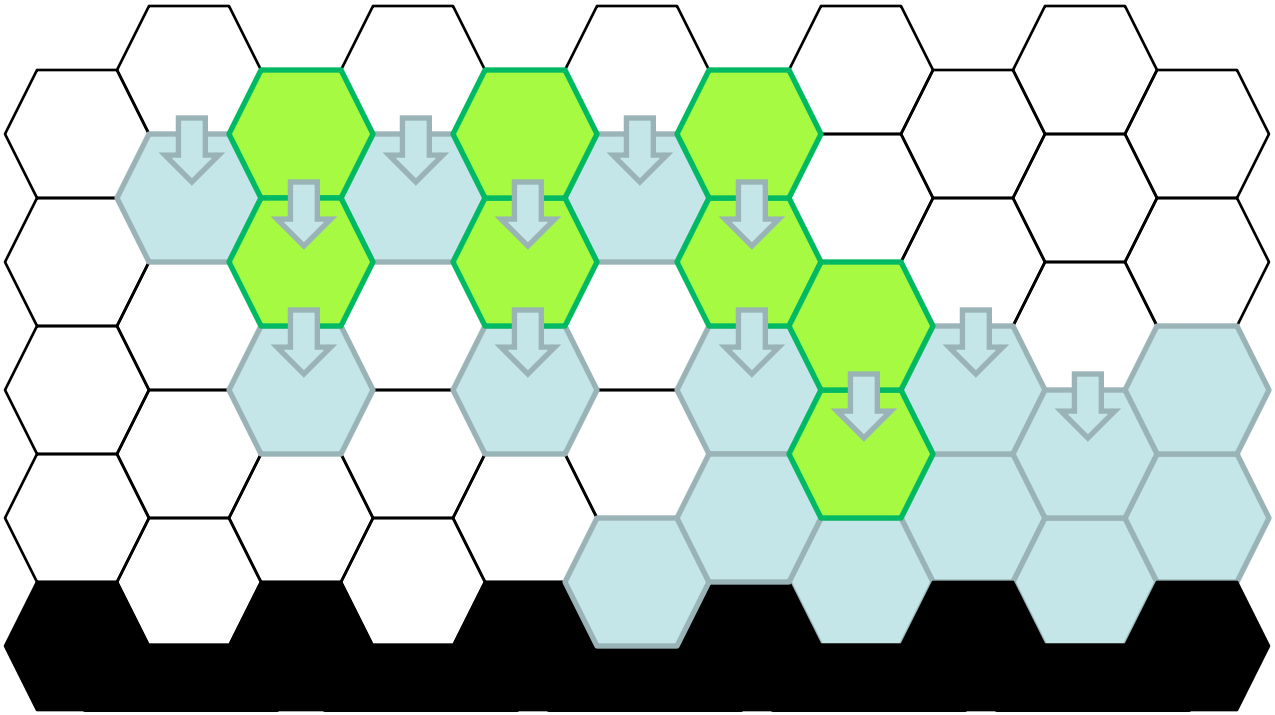
Covering Problem

Consider the problem of covering a surface.



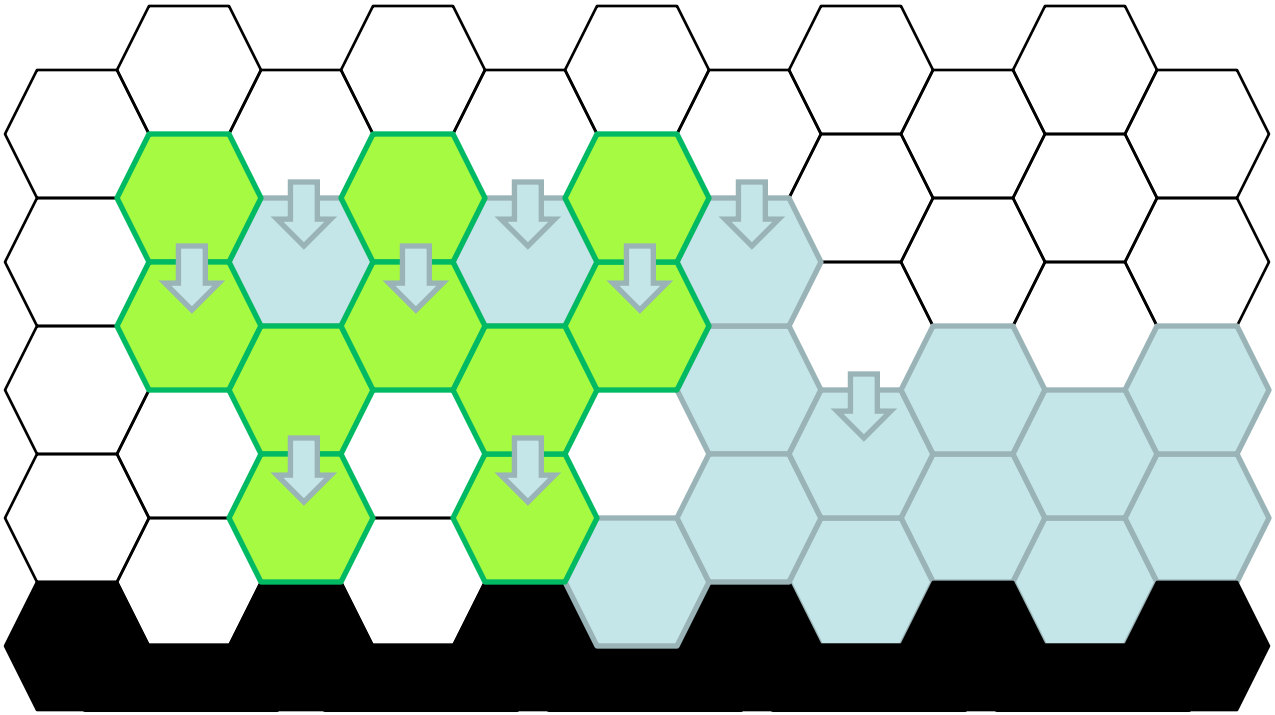
Covering Problem

Consider the problem of covering a surface.



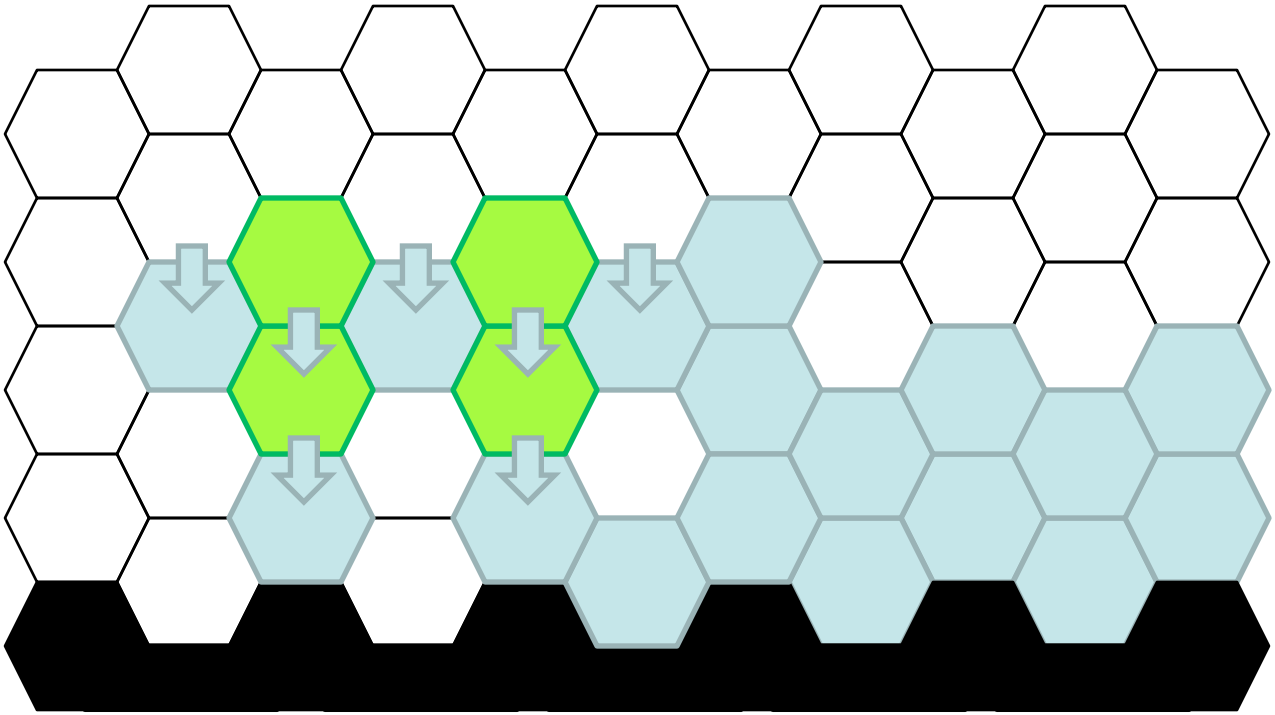
Covering Problem

Consider the problem of covering a surface.



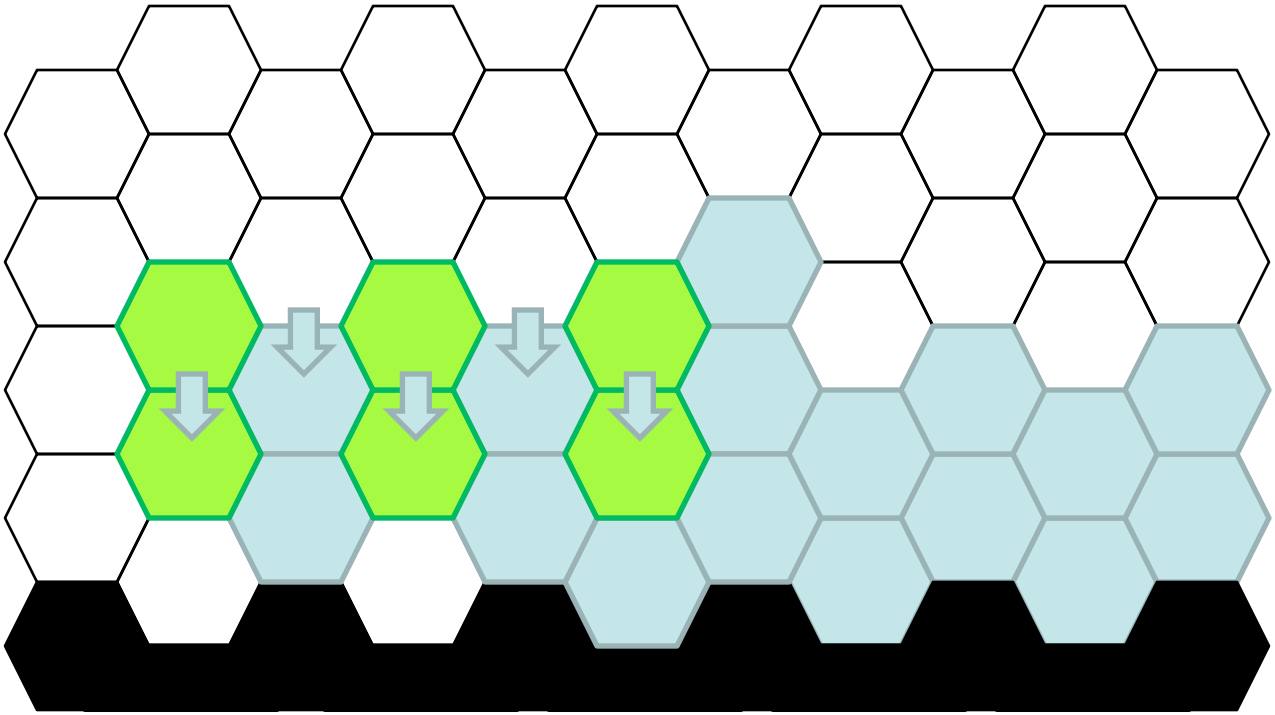
Covering Problem

Consider the problem of covering a surface.



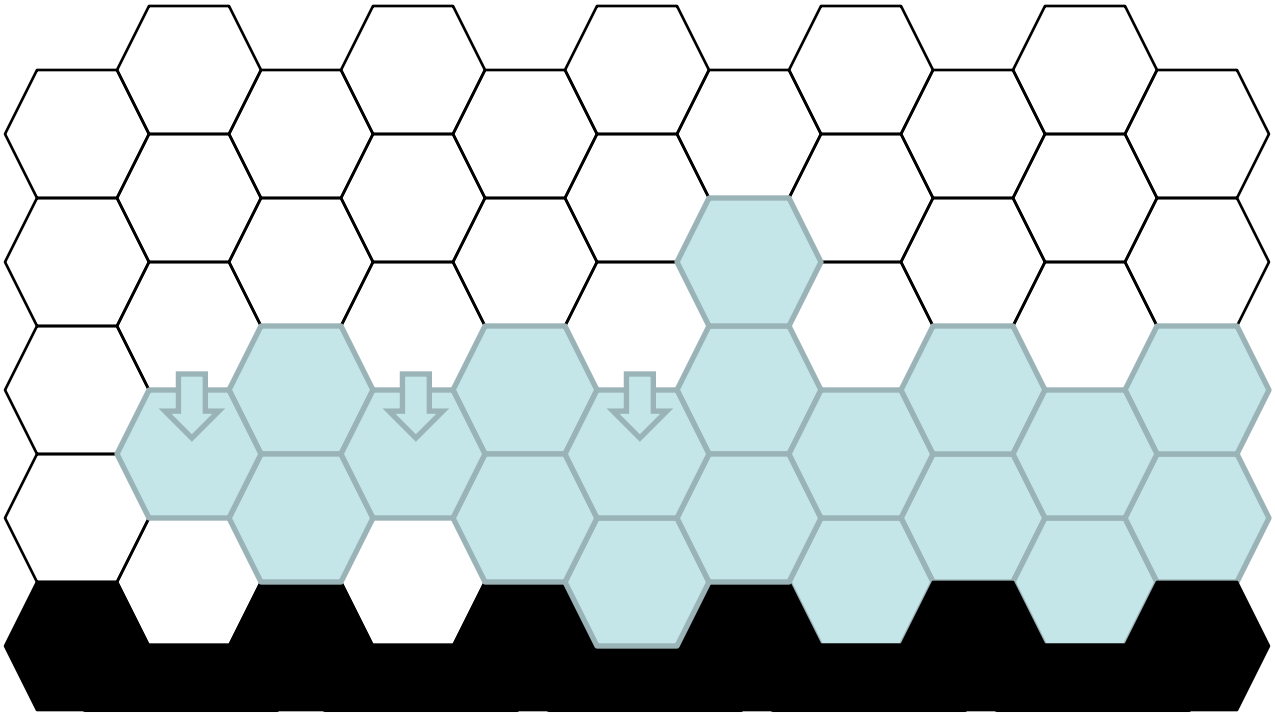
Covering Problem

Consider the problem of covering a surface.



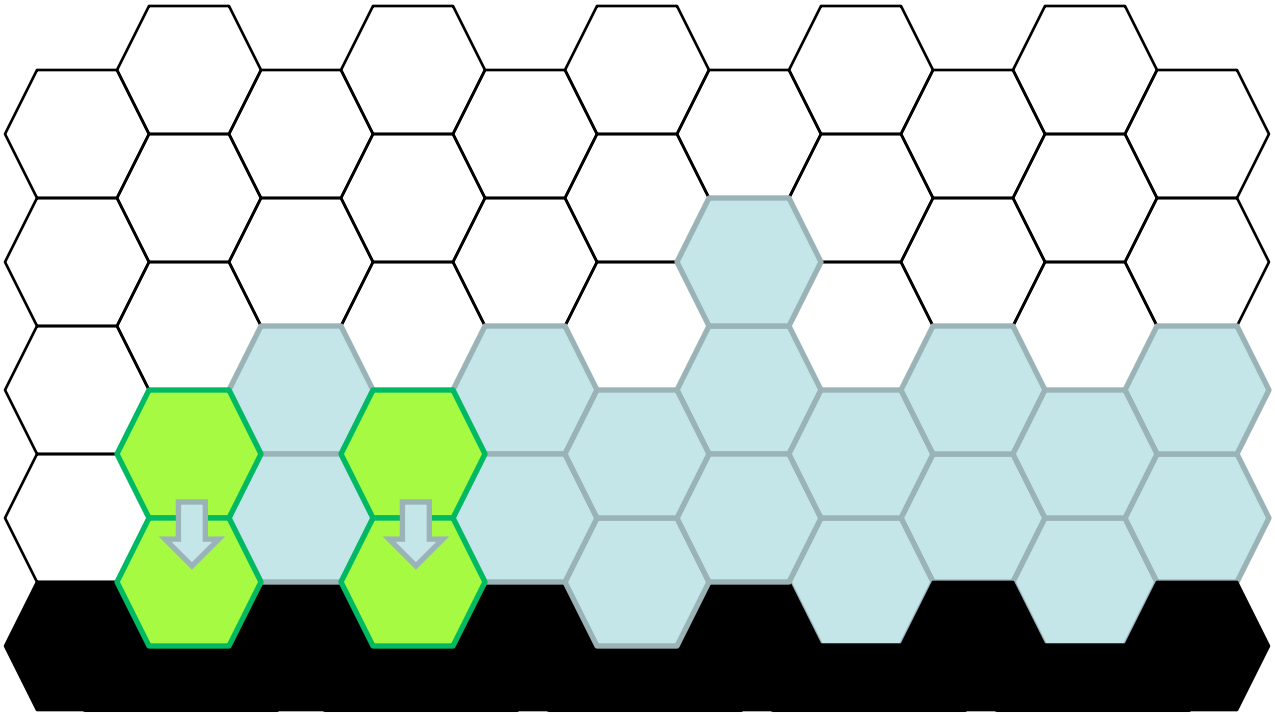
Covering Problem

Consider the problem of covering a surface.



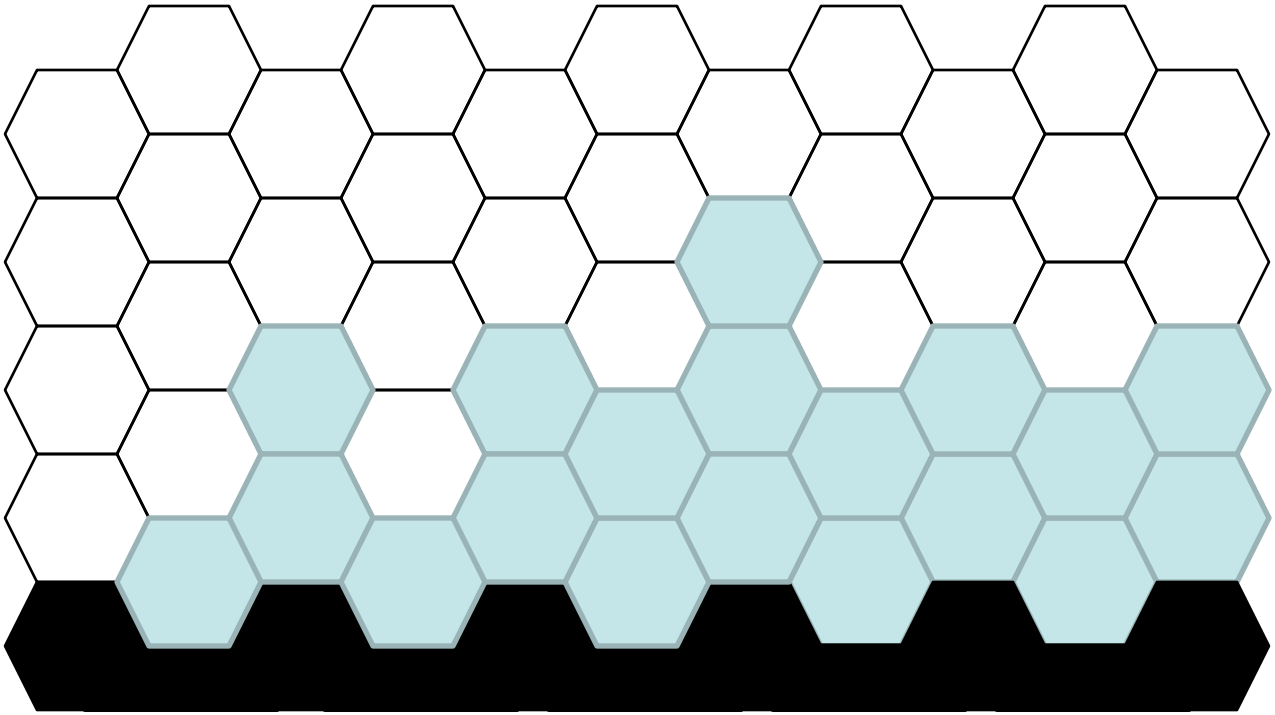
Covering Problem

Consider the problem of covering a surface.



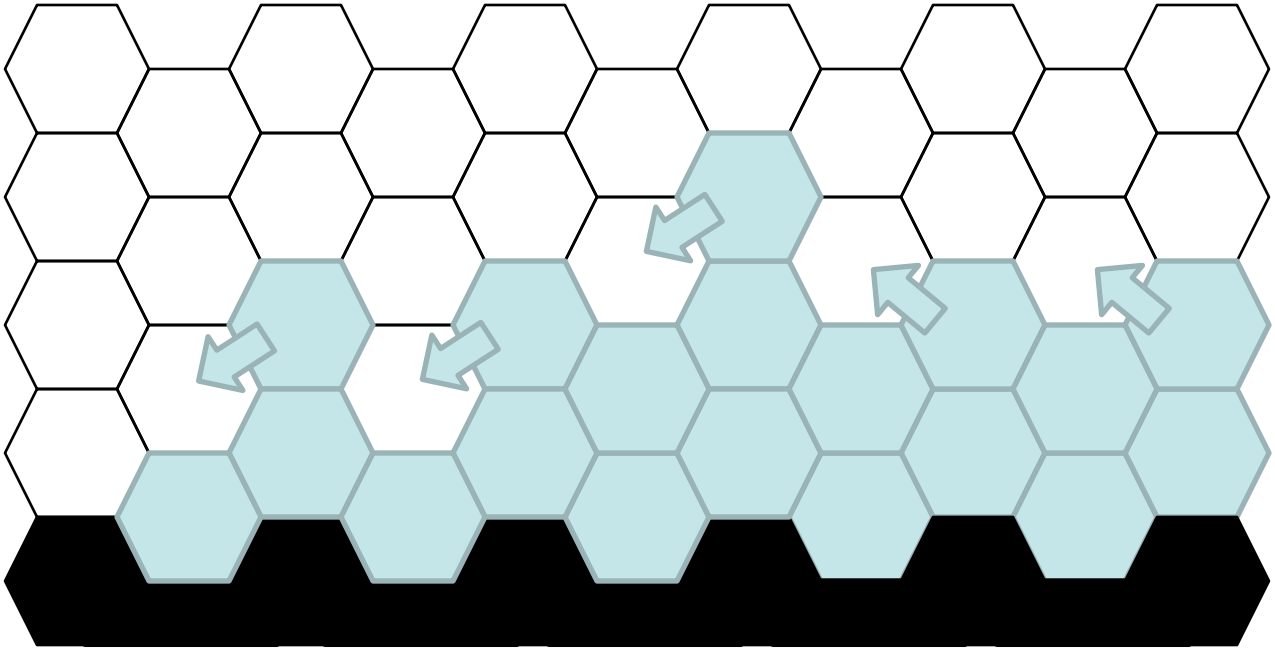
Covering Problem

Consider the problem of covering a surface.



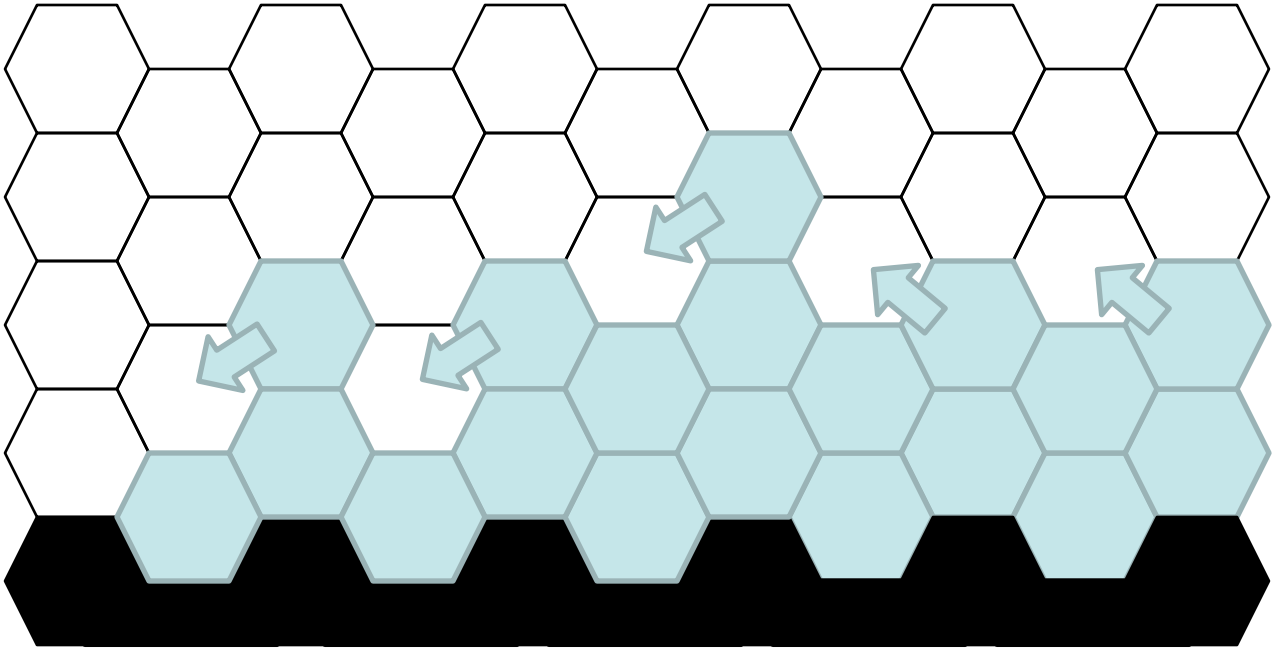
Covering Problem

Additional rule: every particle that is not in contact with surface moves to left.



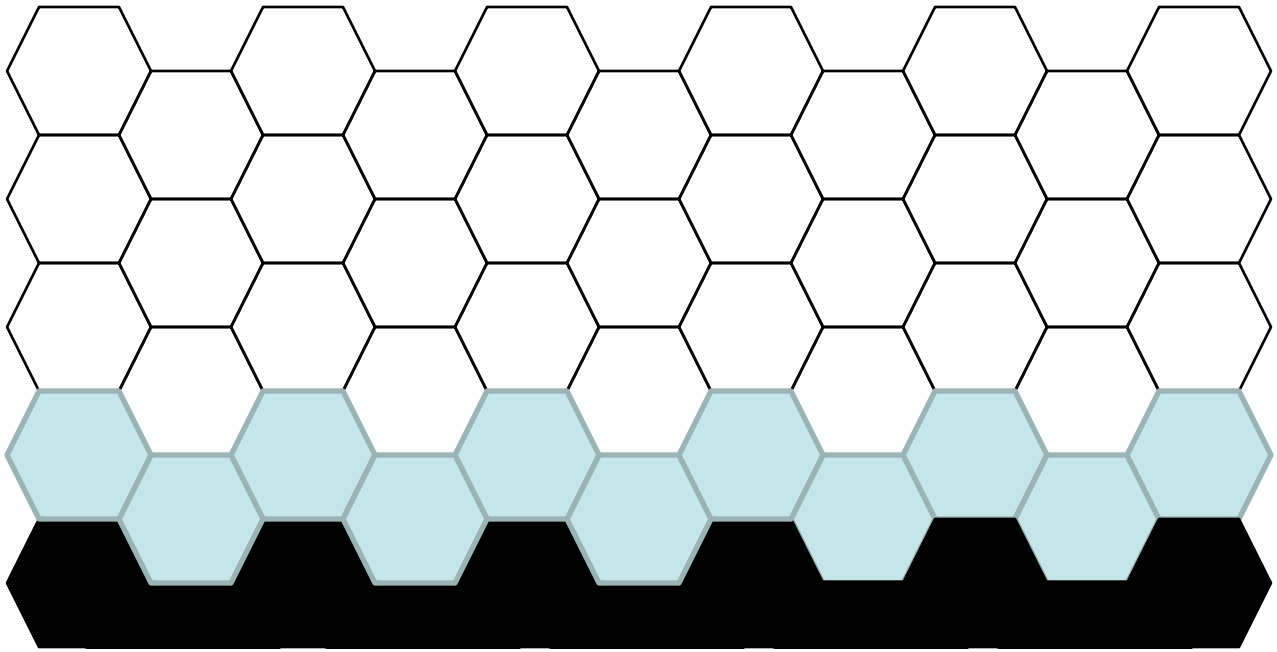
Covering Problem

Infinite surface: eventually, just one thin layer of particles.



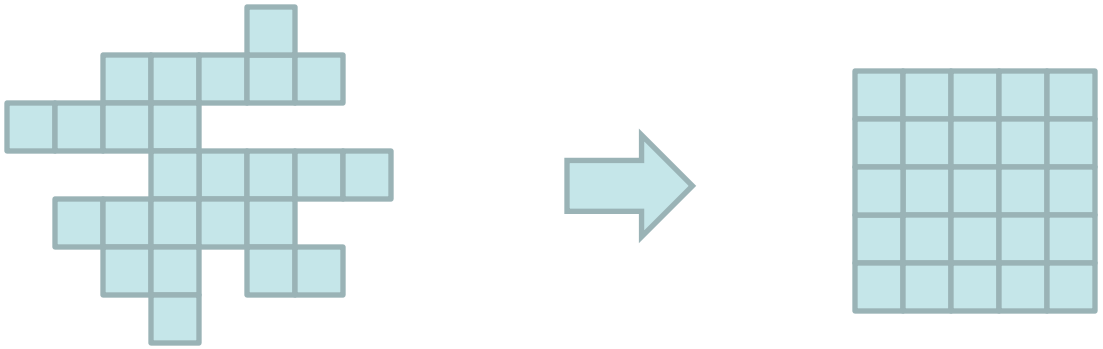
Covering Problem

Infinite surface: eventually, just one thin layer of particles.



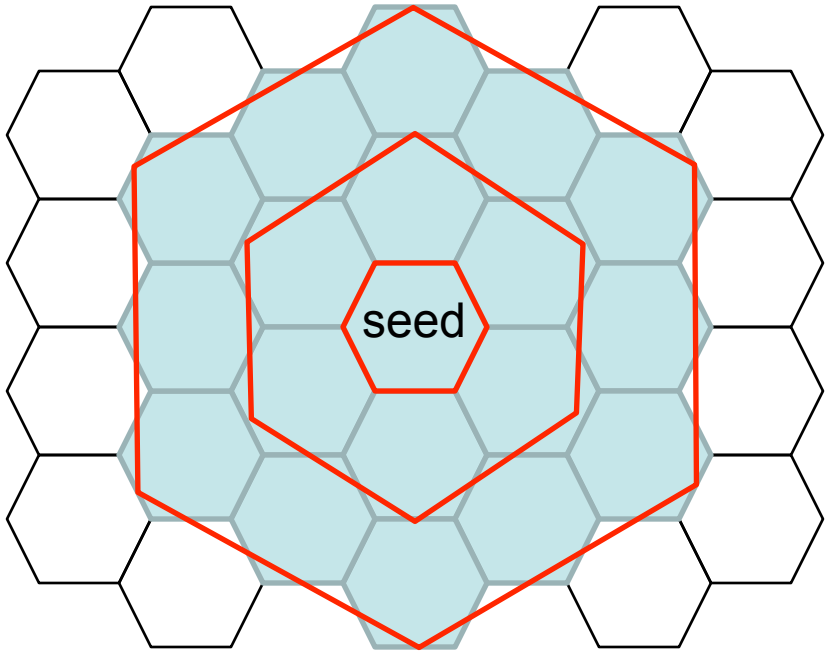
Predefined Shape

Problem: Given a predefined shape and a connected set of particles, form the largest possible structure of that shape with these particles.



Predefined Shape

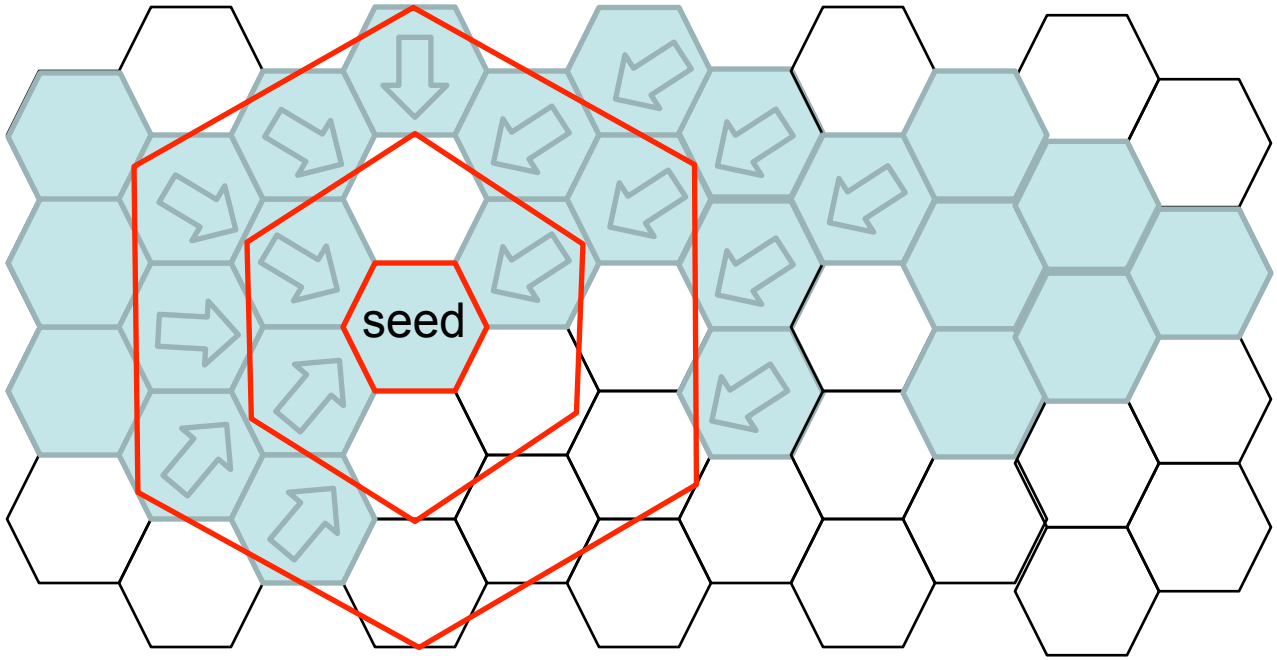
Idea: use solution to covering problem to establish predefined shape



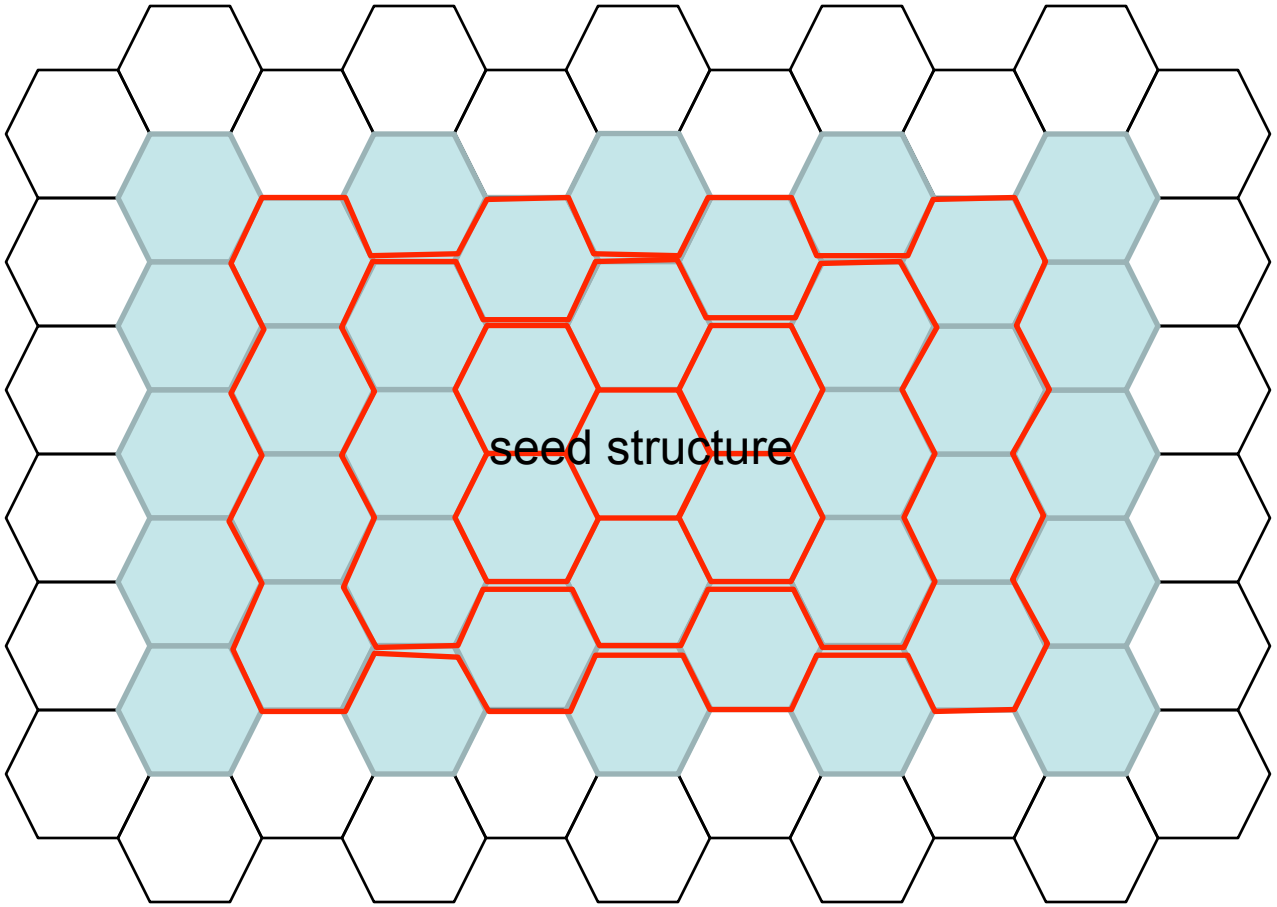
Add layer by layer to seed structure

Predefined Shape

Idea: use solution to covering problem to establish predefined shape



Predefined Shape



Predefined Shape

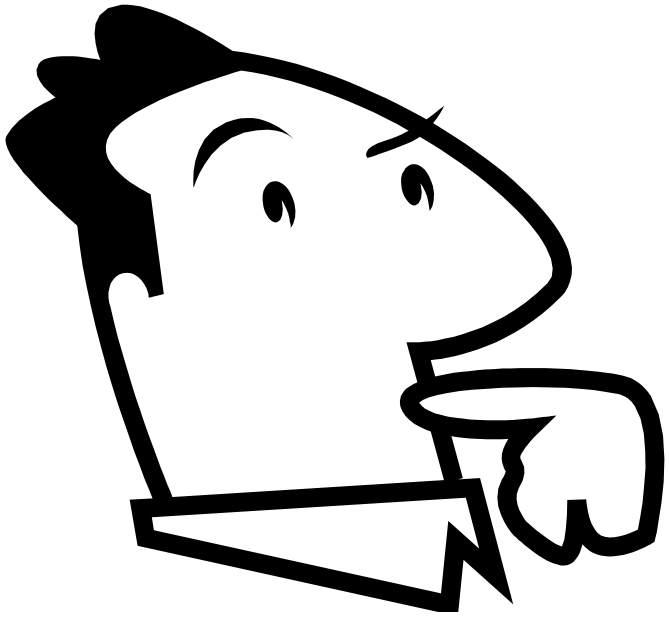
More complex convex shapes:

- Start with irreducible structure
- Grow from there layer by layer

Nonconvex shapes:

- Still unclear how to do that

Self-assembling Nano-Sensors



Questions?