Introduction

slides © 2021, David Doty

ECS 232: Theory of Molecular Computation

UC Davis

Theory of molecular computation = Theory of computation ∩ nanotechnology

In principle, any nanotechnology…
In practice, DNA nanotechnology. Why?

- limitations of my own expertise
- DNA is naturally information bearing… because of recent technological breakthroughs, also information <u>processing</u>
- More experimentalists in DNA nanotech interested in the idea of “making molecules compute” than in other nanotech fields.

Potential DNA nanotechnology applications

nonbiological:

- **art**

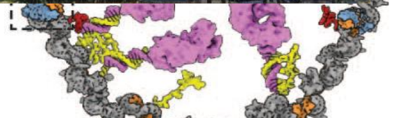
oscale resolution surface placement

pens to deliver antibody

e of two protein antigens

a

fr



oper

ple

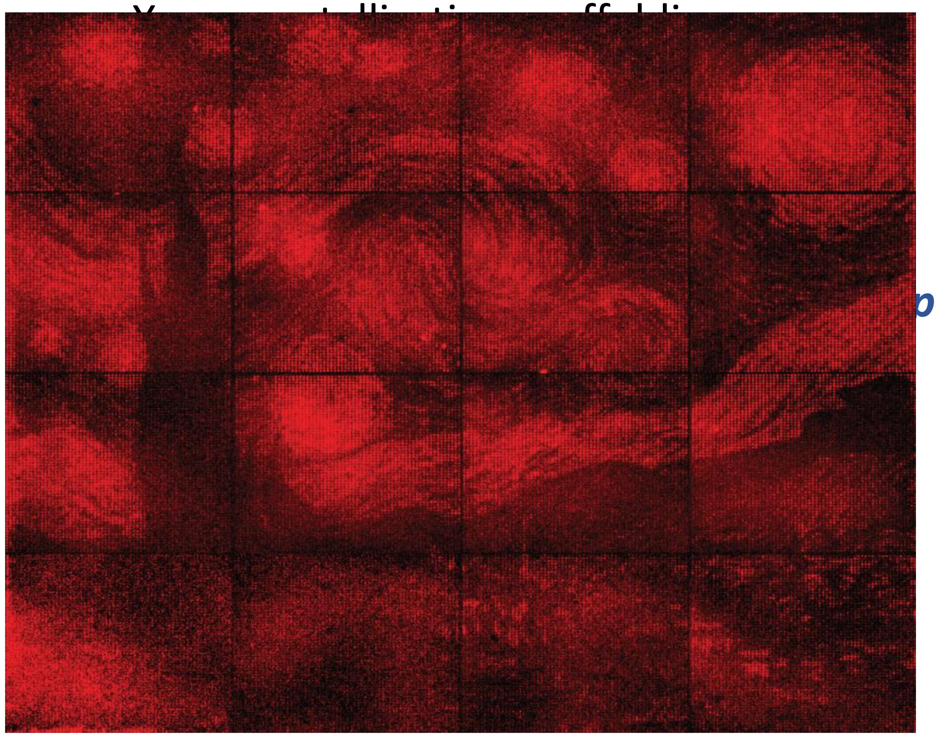
ec

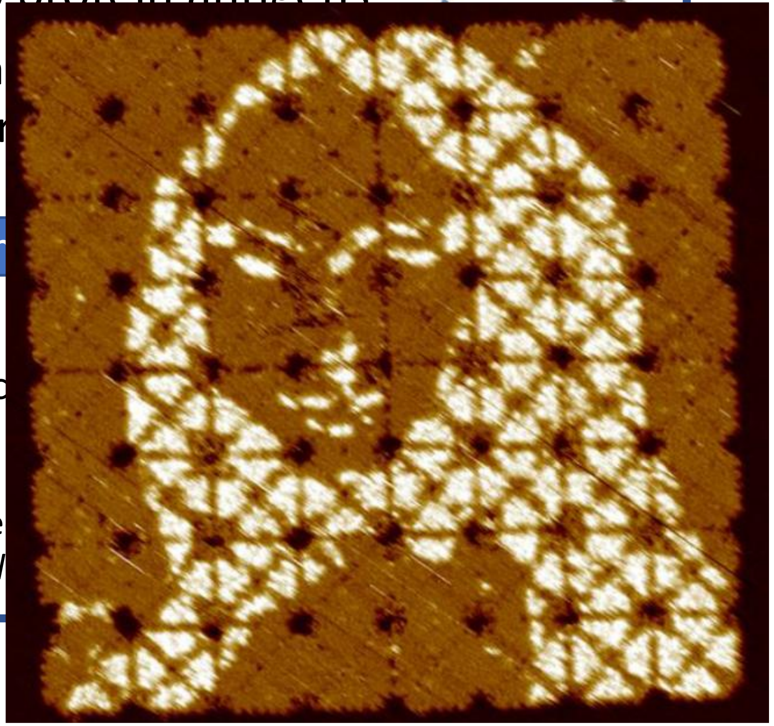
Shawn

for tar

Ge

cul





Ashwin Gopinath, Evan Miyazono, Andrei Faraon, Paul Rothemund. *Engineering and mapping nanocavity emission via precision placement of DNA origami*, Nature 2016

Grigory Tikhomirov, Philip Petersen, and Lulu Qian. *Fractal assembly of micrometre-scale DNA origami arrays with arbitrary patterns*. Nature 2017.

Theory of computation: Bird’s eye view

- Church-Turing thesis: *For any physical device that can be built in our universe that can ‘reasonably’ said to compute a function f*: $\{0,1\}^* \rightarrow \{0,1\}^*$, *f is also computable by a Turing machine.*
- Not a mathematical theorem; it is a physical hypothesis.
  - Physicists call such hypotheses “laws of nature” (always the optimists).
  - It’s the connection between the physical world of *voltages*, *transistors*, and *time in seconds*, and the abstract ideas of *digital data*, *Boolean logic*, and *number of steps in an algorithm*.
- If no Turing machine can solve a computational problem, then no physical device can solve it either.
- Some problems *are* inherently difficult (Boolean satisfiability) or impossible (Halting problem) for computers to solve.
  - Fact about the problem itself, not about a lack of human ingenuity.

Theory of computation: Bird's eye view

- Many models of computation:
  - Turing machine
  - finite state machine
  - polynomial-time Turing machine
  - Boolean circuit
  - distributed network with limited communication between nodes but unlimited computational ability at each node
- None is more or less "correct" as a model of all computing devices.
  - Each is an abstraction useful in some contexts, like Newtonian spherical, frictionless cows.
- Consider: no finite state machine can decide if a binary string has equal 0s and 1s
  - Overly literal interpretation: Your laptop has finite memory, so cannot solve this problem.
  - Better interpretation: If you write an algorithm to solve this problem, then it must, somewhere, use unbounded memory (list, stack, recursion, etc.)

Theory of molecular computation: Bird’s eye view

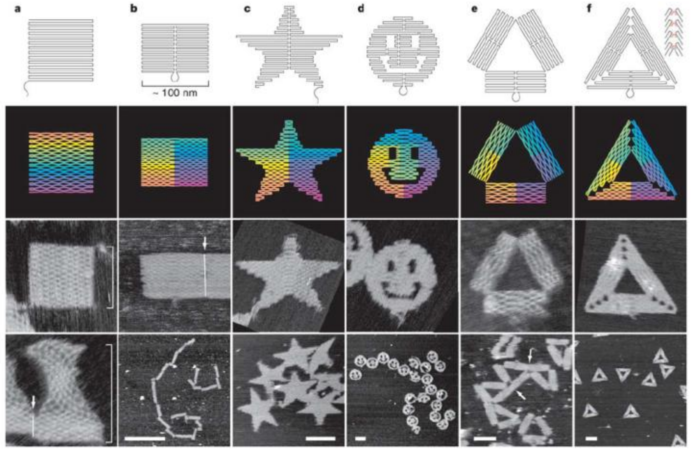
Goal of course: Apply the Theory of Computing “lens” to molecular engineering.

Topics we’ll cover:

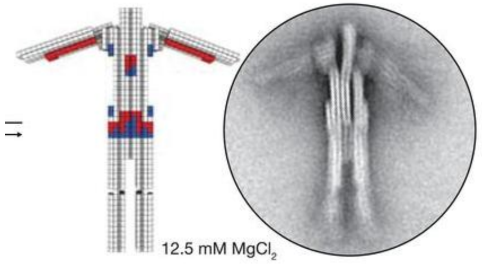
- Basic experimental background
  - structural DNA nanotechnology (building things from molecules)
    - DNA tile self-assembly
    - DNA origami
  - dynamic DNA nanotechnology (reconfiguring molecules)
    - DNA strand displacement
- Algorithmic tile self-assembly
- Chemical reaction networks (and relation to distributed computing)
- Thermodynamic binding networks
- DNA sequence design (classical algorithmic problem helpful in experiments)

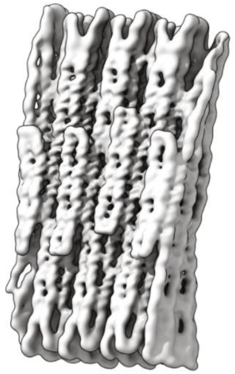
Basic experimental background: Structural DNA nanotechnology

DNA origami



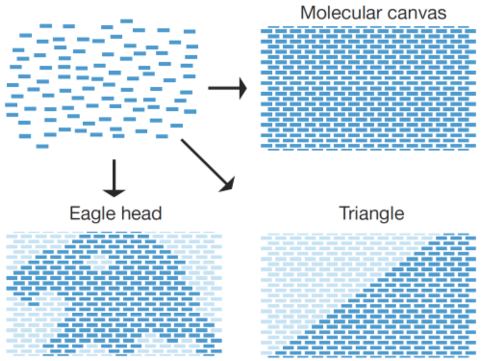
https://www.nature.com/articles/nature04586





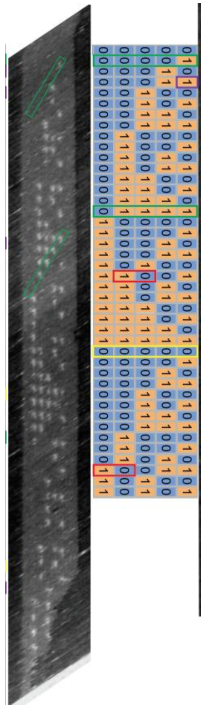
https://science.sciencemag.org/content/347/6229/1446.full
https://www.nature.com/articles/s41467-020-20020-7

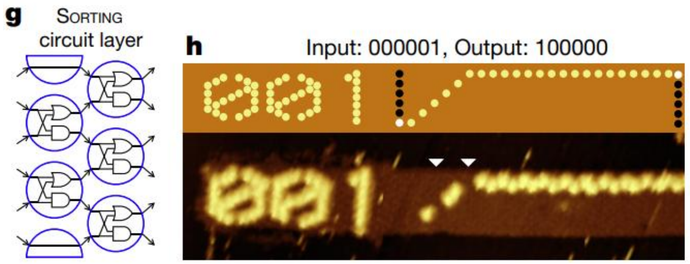
DNA tile self-assembly

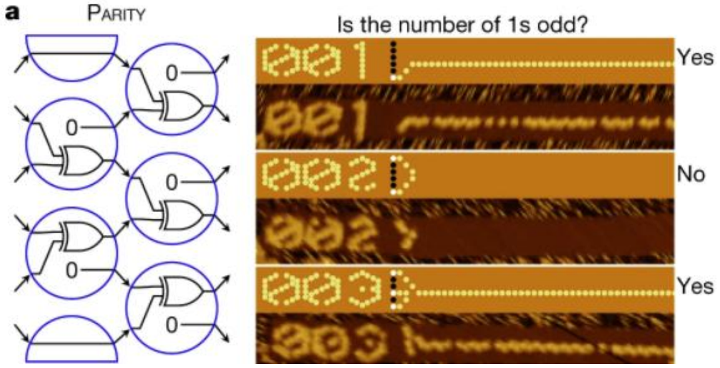


https://www.nature.com/articles/nature11075

algorithmic DNA tile self-assembly



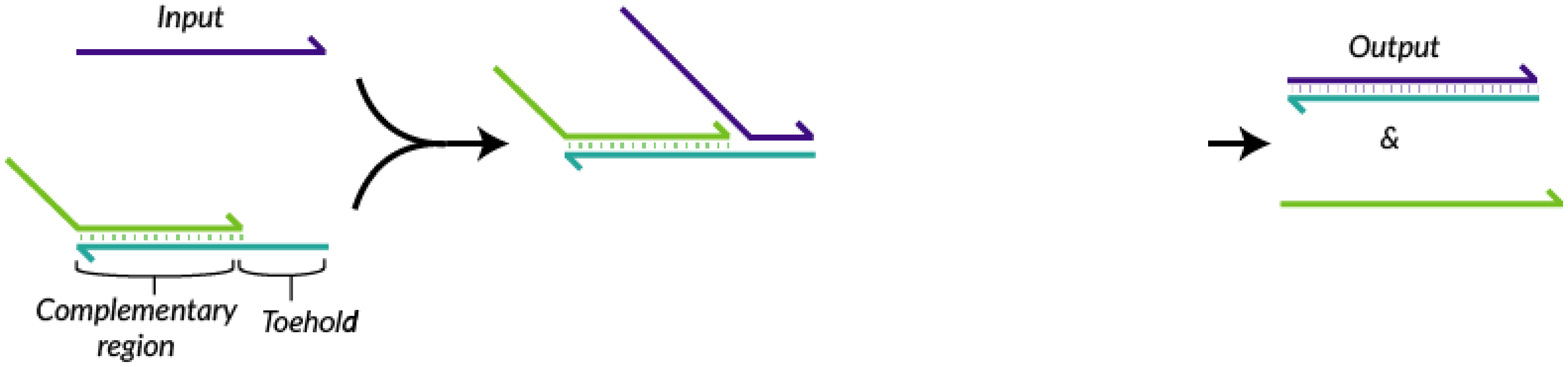




https://www.dna.caltech.edu/Papers/cge-thesis2014.pdf
https://www.nature.com/articles/s41586-019-1014-9

Basic experimental background: Dynamic DNA nanotechnology

<u>DNA strand displacement</u>: DNA reconfiguring itself without enzymes

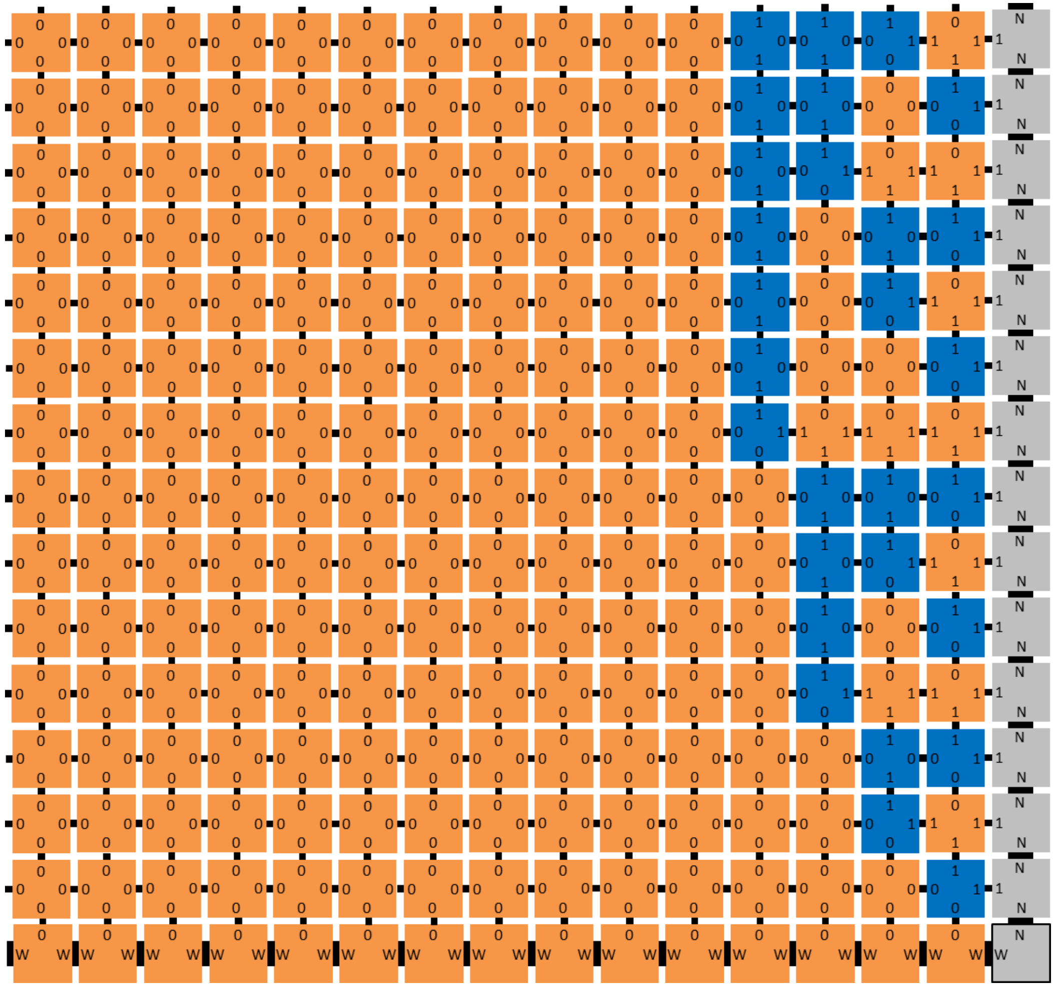


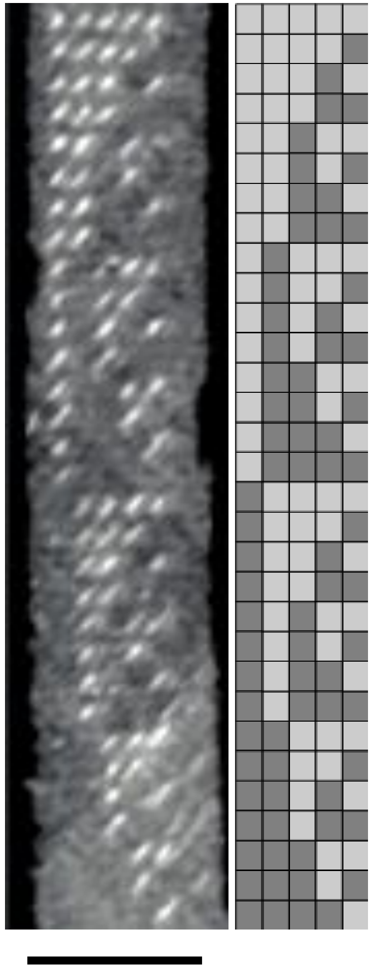
http://parts.igem.org/Part:BBa_K1761002:Experience

DNA strand displacement

https://www.microsoft.com/en-us/research/video/dna-strand-displacement/

Algorithmic tile self-assembly





Chemical reaction networks

reactant(s)	$R \longrightarrow P_1 + P_2$	*product(s)*
monomers	$M_1 + M_2 \longrightarrow D$	*dimer*
catalyst	$C + X \longrightarrow C + Y$	

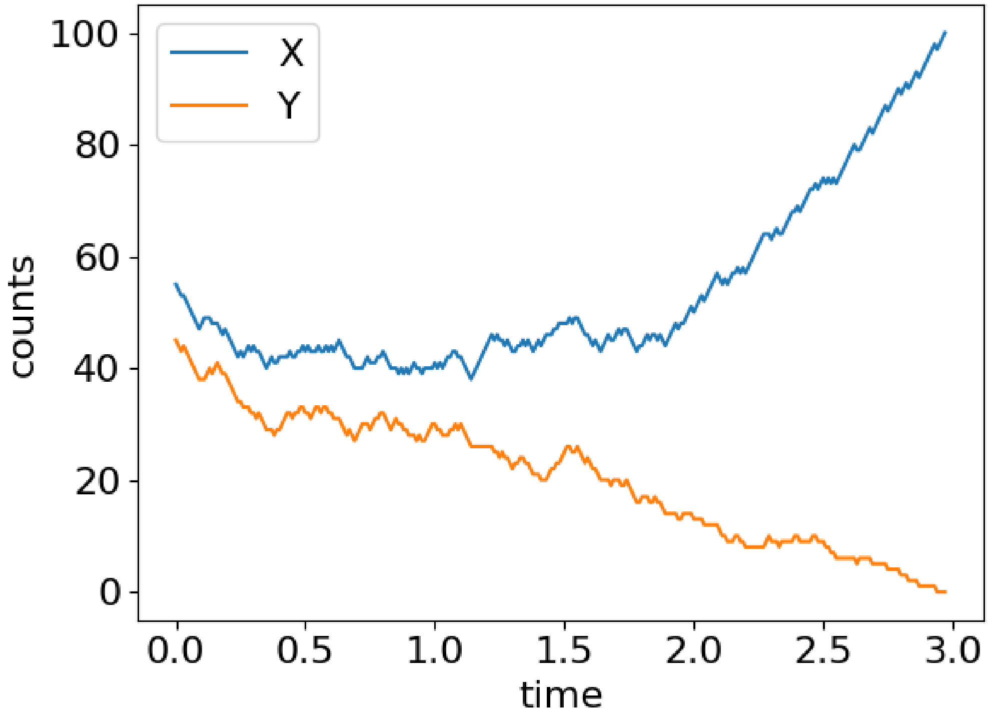
Traditionally a descriptive modeling language…
Let’s instead use it as a prescriptive programming language

Chemical caucusing

$$X+Y \longrightarrow U+U$$

$$X+U \longrightarrow X+X$$

$$Y+U \longrightarrow Y+Y$$



distributed algorithm for *"approximate majority"*:
initial majority (X or Y) quickly overtakes whole population

[Angluin, Aspnes, Eisenstat, A simple population protocol for fast robust approximate majority, *DISC* 2007]

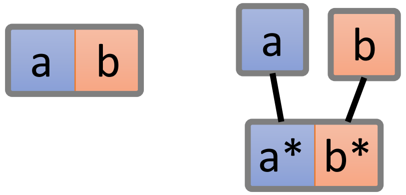
Thermodynamic binding networks

Goal of model: abstract away geometry of DNA to understand effect of two contributions to energetics:

1. number of bonds formed (“enthalpy”)
2. number of separate complexes (“entropy”)

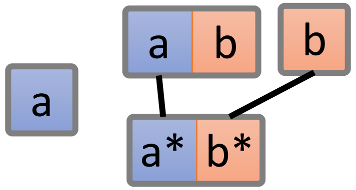
#bonds = 2

#complexes = 2



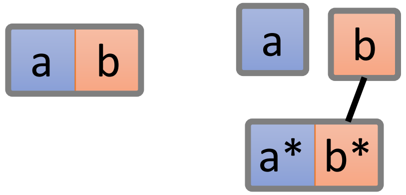
#bonds = 2

#complexes = 2



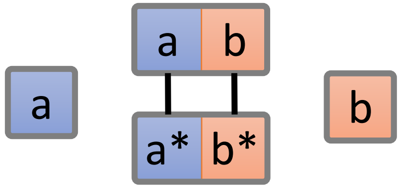
#bonds = 1

#complexes = 3



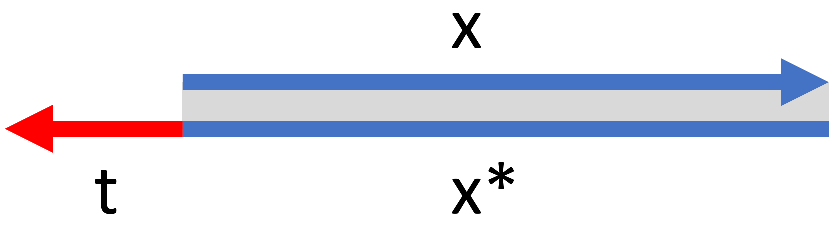
#bonds = 2

#complexes = 3



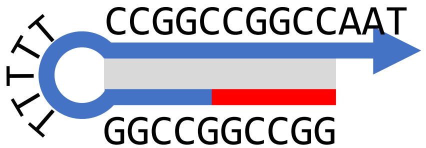
DNA sequence design

Goal: given strands with abstract “binding domains”

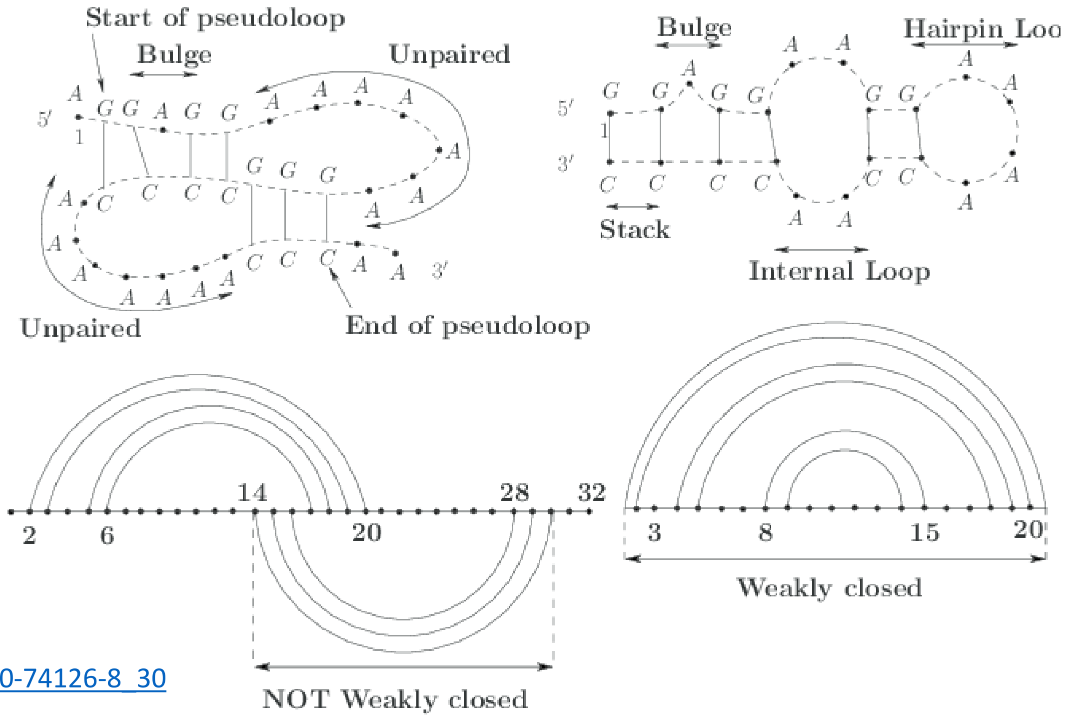


assign to domains concrete DNA sequences:

while preventing unwanted interactions (e.g., strand folds up on itself)



using combinatorial predictive models of DNA energy:



https://link.springer.com/chapter/10.1007/978-3-540-74126-8_30

Logistics

Grading

- 40% homework
- 30% final project
- 30% in-class participation

Prerequisites

- Undergraduate theory of computation (ECS 120)
  - experience with proofs, formal definitions, discrete math (*sets, sequences, finite strings, graphs, big-O notation*)
  - finite automata, Turing machines, computability (*halting problem, decidable versus computably enumerable languages*) and complexity theory (*polynomial-time,* ***NP****-completeness*)
- Probability
  - events
  - random variables
  - expected value