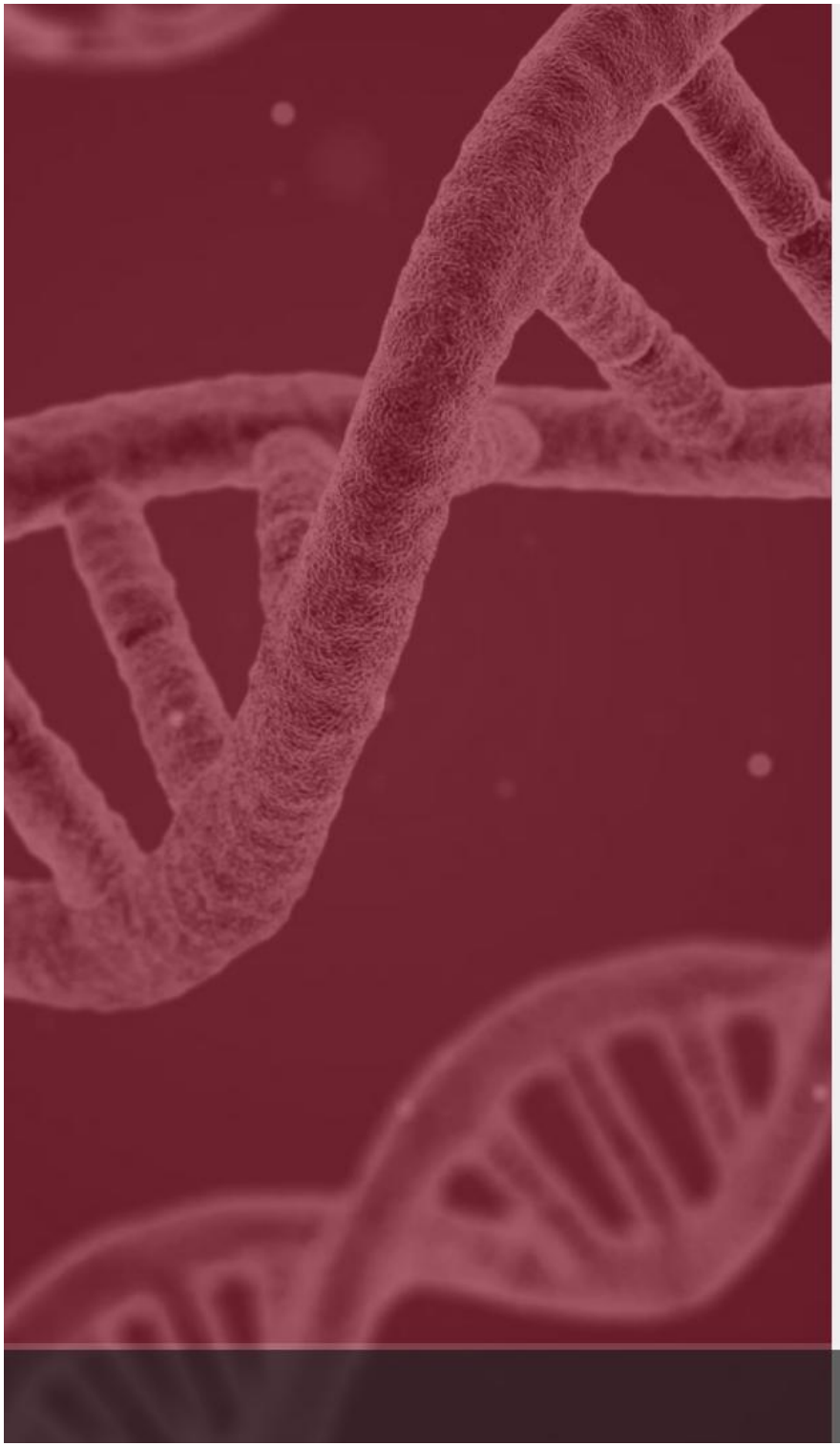


RUKMANGADH SAI MYANA
EKANTHESWAR BANDARUPALLI
ANDREW SALAZAR BEDON

Bioinformatics and Computational Biology

Presentation Overview

LIST OF KEY CONCEPTS

- What is Bioinformatics?
- Nucleic Acid Sequencing
- Needleman-Wunsch Algorithm
- Nucleic Acid Secondary Structure
- Nussinov Algorithm
- Cellular Automata
- Natural selection
- Simulation

What is Bioinformatics?

COMBINING COMPUTER SCIENCE AND BIOLOGY

Bioinformatics involves the application of computational techniques to analyze, interpret and manage biological data.

Why is it important?

BETTER UNDERSTANDING OF THE BIOLOGY FIELD

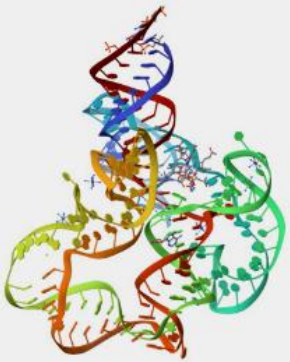
- Data Handling
- Genome Sequencing
- Disease Research
- Biological Databases

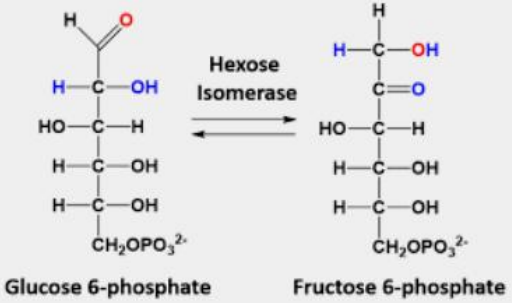
Central Dogma of Molecular Biology

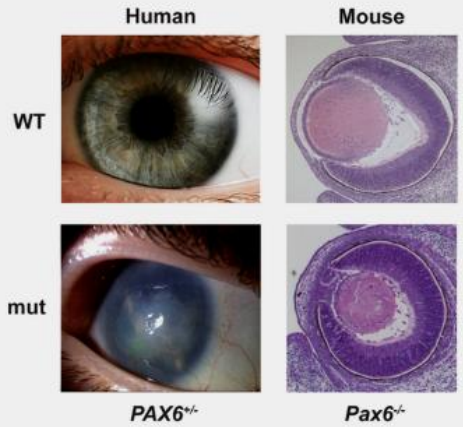
Central Dogma of Bioinformatics

Genetic Information → **Molecular Structure** → **Biochemical Function** → **Phenotype / Symptoms**

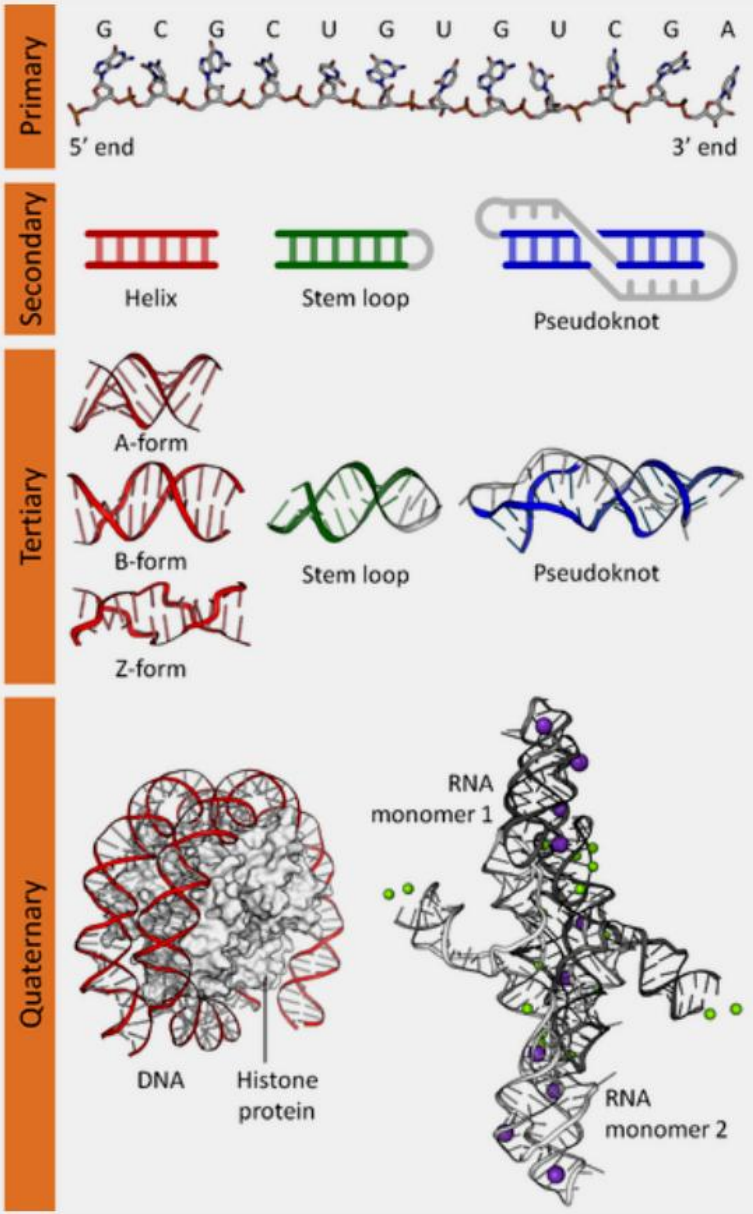
CATGACGT
CTGGGGCT
ATTTTCTG
TTTTTCGA
CGATAAAA
CCCTGGGT
ATCCAAGC
AGCCCAGT
GGTTAAGG
TCGCGGTT
AGACAACC
CAATGATA





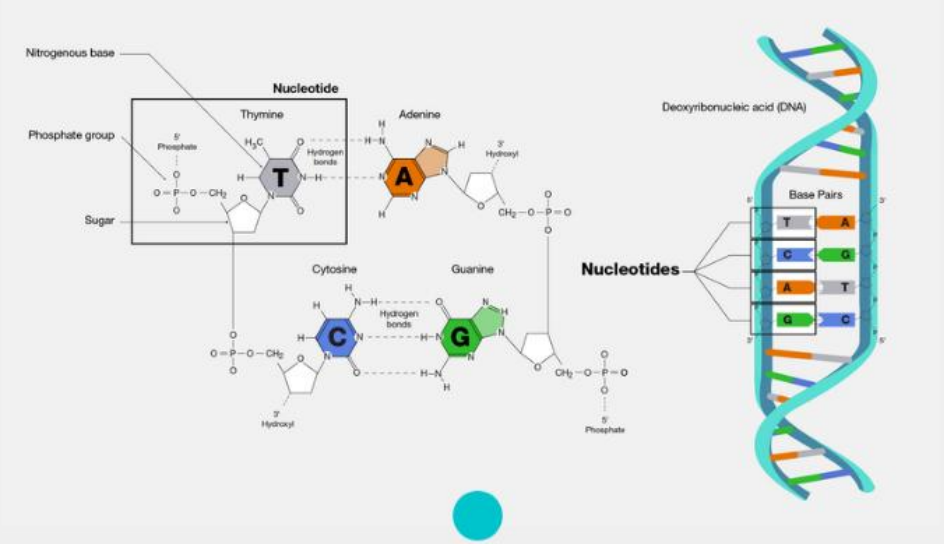


Structure Hierarchy



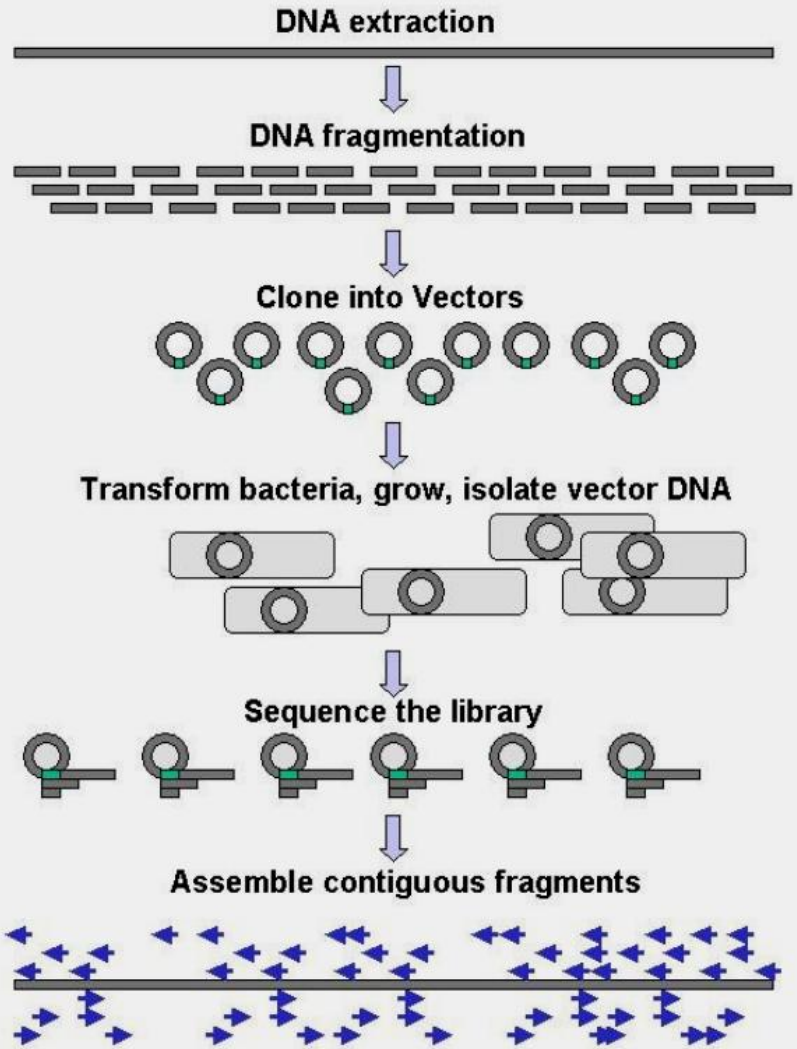
Primary structure

- **Nucleotide:** Basic building blocks of a nucleic acid chain
- Four types of nucleotides:
  - Adenine
  - Guanine
  - Cytosine
  - Thymine (DNA) / Uracil (RNA)



Sequencing

DETERMINING THE PRIMARY STRUCTURE

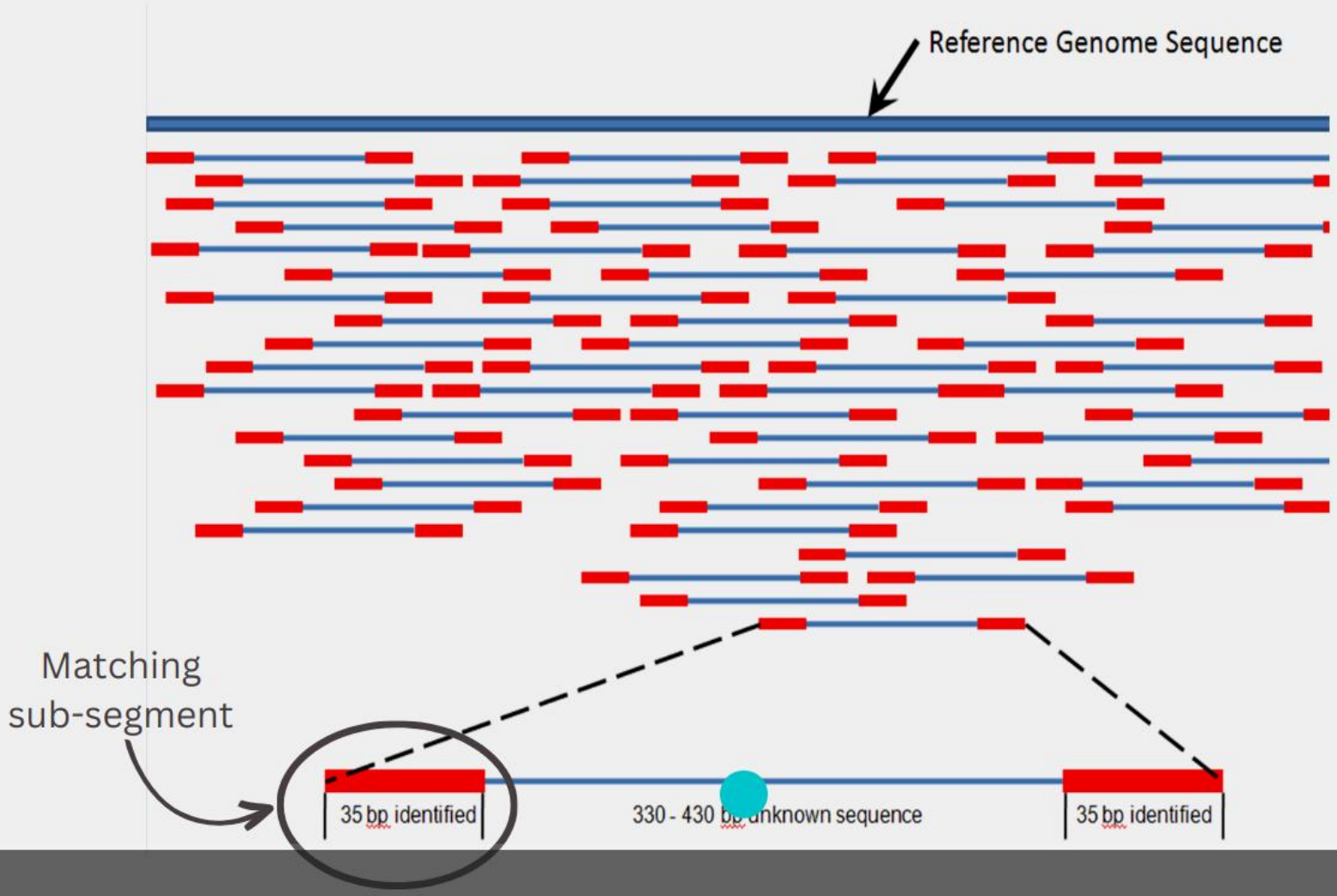


Sequence Assembly

- Approaches are classified as follows:
  - **Reference-guided:** Assembling fragments using a template
  - **De-novo:** Assembling fragments from scratch

Template FTW!!

REFERENCE GUIDED ASSEMBLY



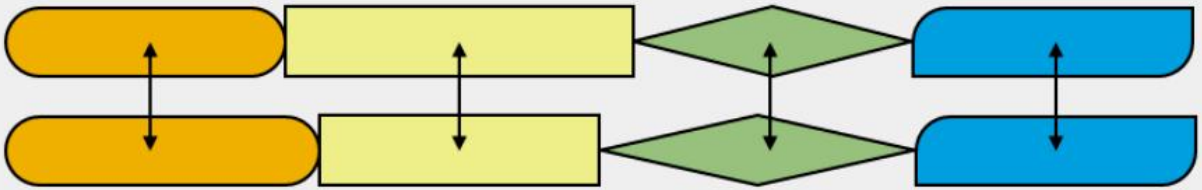
How is Matching Done?

SEQUENCE ALIGNMENT

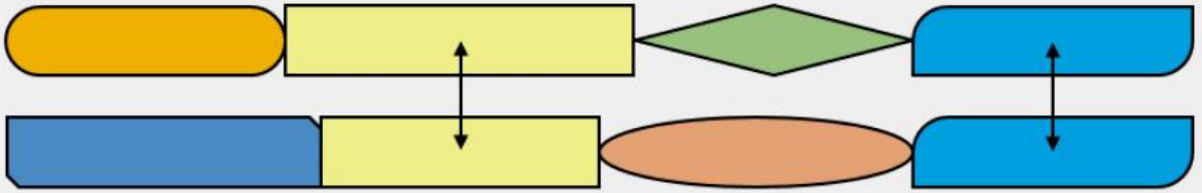
- Two popular sequence alignment algorithms:
  - **Smith-Waterman Algorithm** (Local Alignment)
  - **Needleman-Wunsch Algorithm** (Global Alignment)
- Local alignment
  - Longer segment lengths
  - Less similarity between segments is expected
- Global Alignment
  - Smaller segment lengths
  - More similarity between segments is expected

Whats the difference?

LOCAL VS GLOBAL ALIGNMENT



Global Alignment

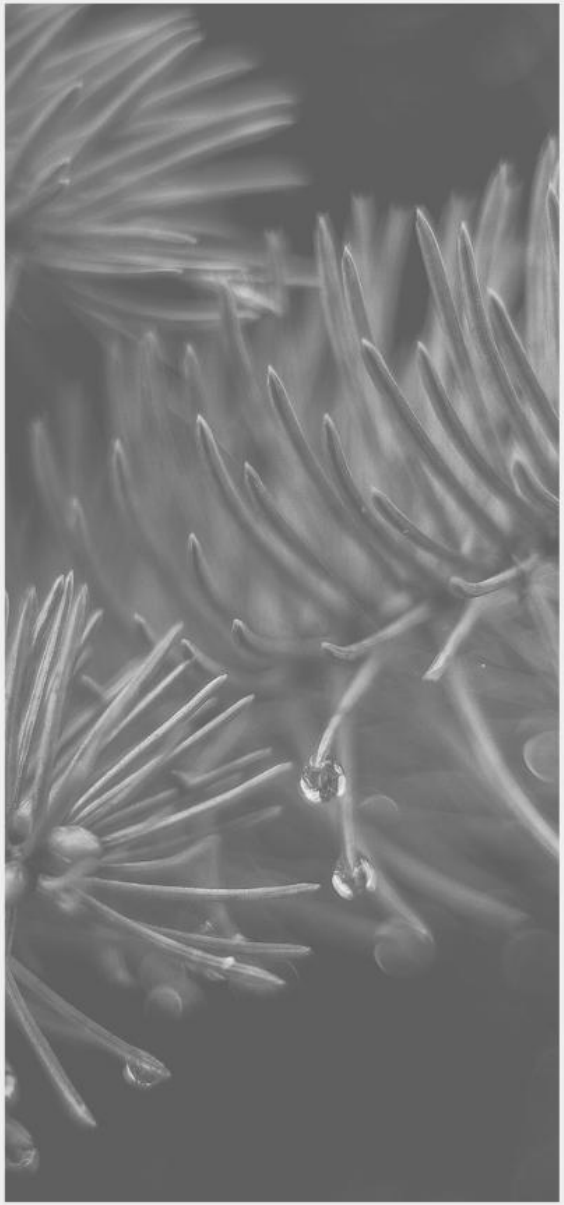


Local Alignment

Dynamic Programming

NEEDLEMAN WUNSCH ALGORITHM

- Programming paradigm for solving optimization problems
- Need two conditions:
  - Optimal Substructure
  - Overlapping Sub-problems



Setting the Context

SUBSTITUTION MATRIX

	C	T	A	G
C	2	1	-1	-1
T	1	2	-1	-1
A	-1	-1	2	1
G	-1	-1	1	2

GAP PENALTY

$$\gamma(n) = -kn$$

Penalty for Gap of length - (n)

- Example of a Real-life scoring system: BLOSUM62

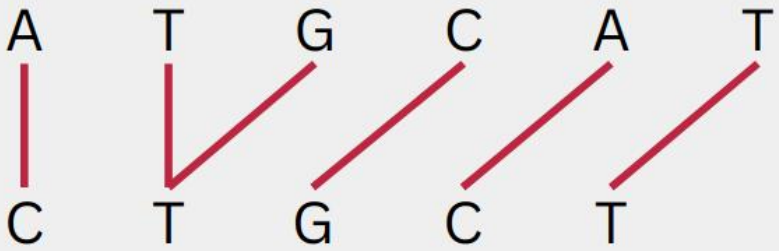
An Example

A	T	G	C	A	T	
C	T	$-$	G	C	T	$=-3$
-1	1	-2	-1	-1	1	

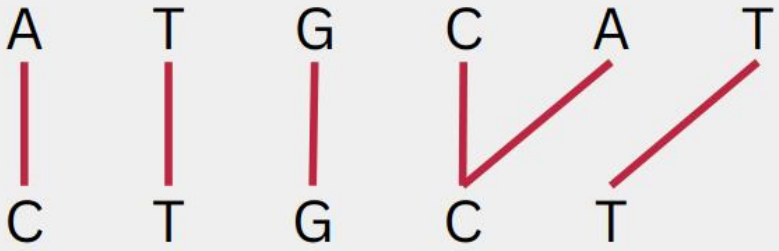
A	T	G	C	A	T	
C	T	G	C	$-$	T	$=1$
-1	1	1	1	-2	1	

Reimagining the problem

ANOTHER WAY TO LOOK AT THE SAME EXAMPLE



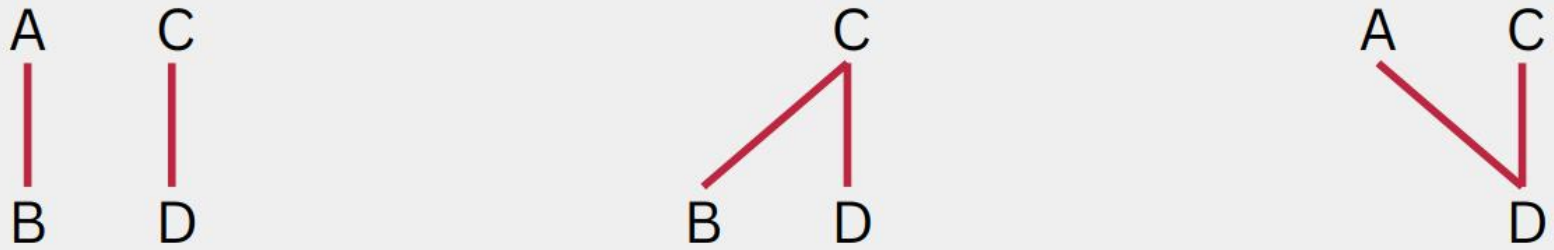
Score = -3



Score = 1

Optimal Substructure

OPTIMIZING SUBPROBLEMS OPTIMIZES THE WHOLE



$$D(i,j) = \max \begin{cases} D(i-1, j-1) + s(x_i, y_j) \\ D(i-1, j) + g \\ D(i, j-1) + g \end{cases}$$

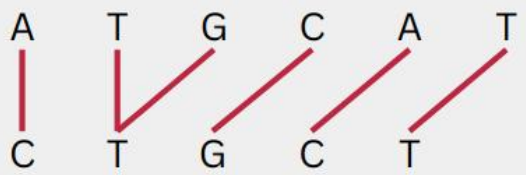
Cost of mapping the **i'th** element in the first sequence to the **j'th** element in the second sequence

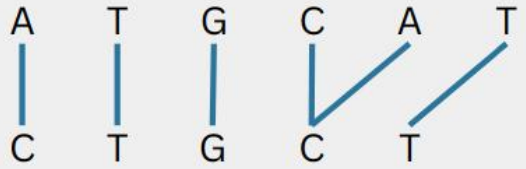
Writing DP Code

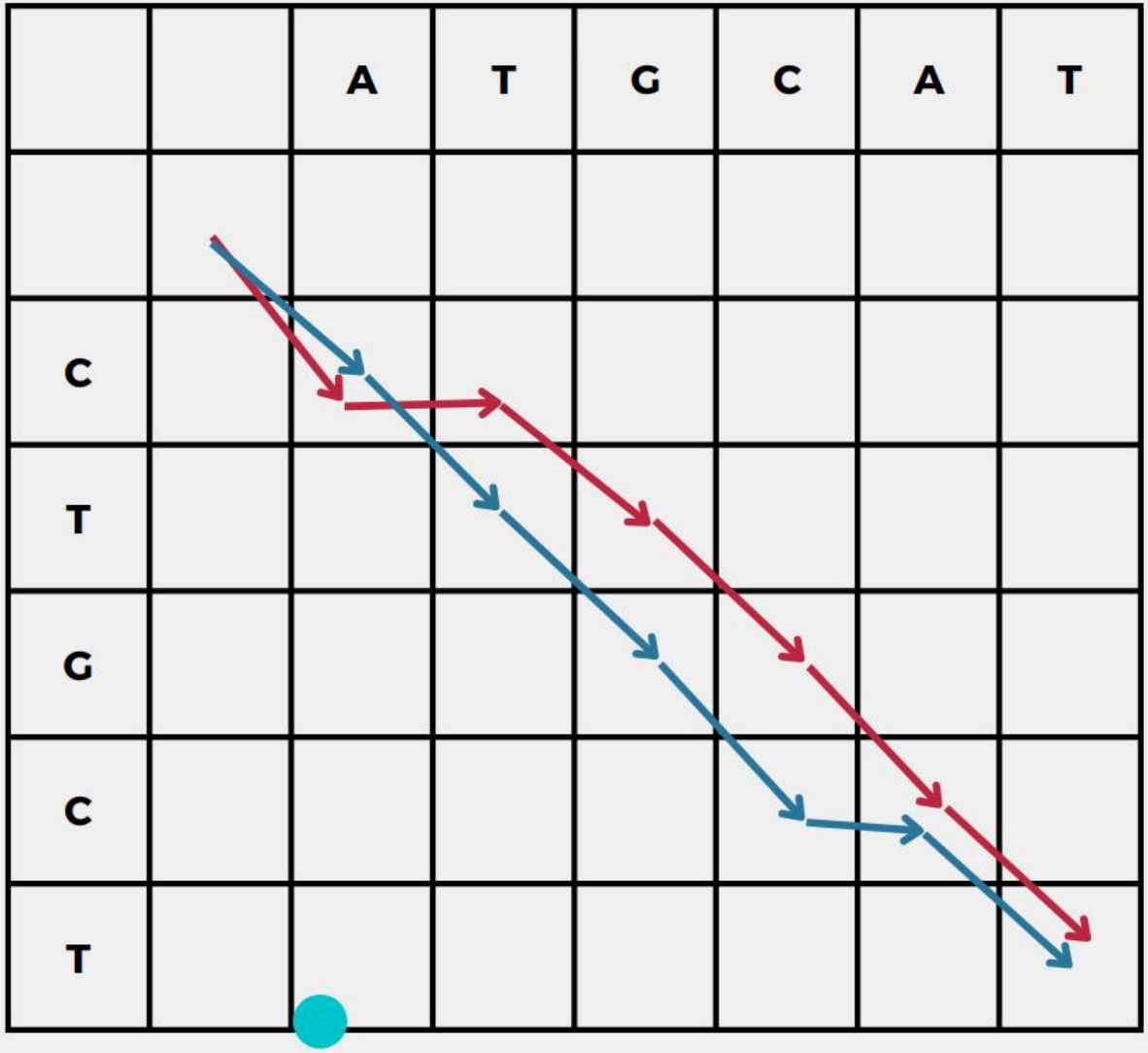
TABULATION APPROACH

- Use a table/matrix to compute the sub-problems
- **Bottom-up approach:** Compute smaller problems before larger problems
- Calculate M(a,b) (a < i), (b < j) ; before calculating M(i,j)
- **Intuition:** An alignment is a path on the matrix

Matrix Intuition







Example - Tabulation

		A	T	G	C	A	T
	0	-2	-4	-6	-8	-10	-12
C	-2						
T	-4						
G	-6						
C	-8						
T	-10						

Example - Tabulation

		A	T	G	C	A	T
	0	-2	-4	-6	-8	-10	-12
C	-2	-1	-3	-5	-5	-7	-9
T	-4	-3					
G	-6	-5					
C	-8	-7					
T	-10	-9					

Example - Tabulation

		A	T	G	C	A	T
	0	-2	-4	-6	-8	-10	-12
C	-2	-1	-3	-5	-5	-7	-9
T	-4	-3	0	-2	-4	-6	-6
G	-6	-5	-2				
C	-8	-7	-4				
T	-10	-9	-6				

Example - Tabulation

		A	T	G	C	A	T
	0	-2	-4	-6	-8	-10	-12
C	-2	-1	-3	-5	-5	-7	-9
T	-4	-3	0	-2	-4	-6	-6
G	-6	-5	-2	1	-1	-3	-5
C	-8	-7	-4	-1			
T	-10	-9	-6	-3			

Example - Tabulation

		A	T	G	C	A	T
	0	-2	-4	-6	-8	-10	-12
C	-2	-1	-3	-5	-5	-7	-9
T	-4	-3	0	-2	-4	-6	-6
G	-6	-5	-2	1	-1	-3	-5
C	-8	-7	-4	-1	2	0	-2
T	-10	-9	-6	-3	0		

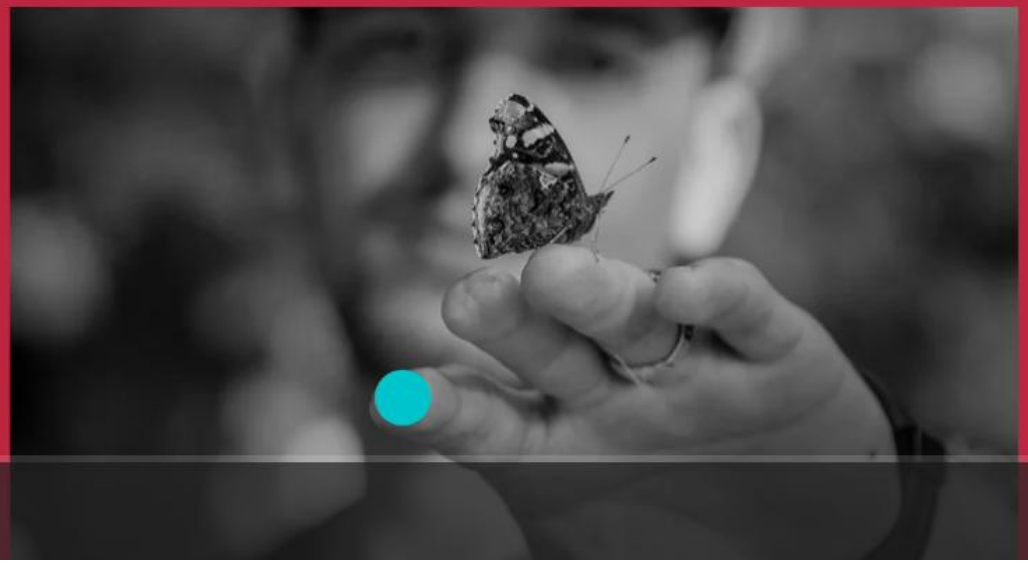
Example - Tabulation

		A	T	G	C	A	T
	0	-2	-4	-6	-8	-10	-12
C	-2	-1	-3	-5	-5	-7	-9
T	-4	-3	0	-2	-4	-6	-6
G	-6	-5	-2	1	-1	-3	-5
C	-8	-7	-4	-1	2	0	-2
T	-10	-9	-6	-3	0	1	1

Example - Traceback

		A	T	G	C	A	T
	0	-2	-4	-6	-8	-10	-12
C	-2	-1	-3	-5	-5	-7	-9
T	-4	-3	0	-2	-4	-6	-6
G	-6	-5	-2	1	-1	-3	-5
C	-8	-7	-4	-1	2	0	-2
T	-10	-9	-6	-3	0	1	1

Demo



Smith - Waterman Algorithm

$$H_{ij} = \max \begin{cases} H_{i-1,j-1} + s(a_i, b_j), \\ \max_{k \geq 1} \{H_{i-k,j} - W_k\}, \\ \max_{l \geq 1} \{H_{i,j-l} - W_l\}, \\ 0 \end{cases} \qquad (1 \leq i \leq n, 1 \leq j \leq m)$$




Algorithm Visualized

Initialize the scoring matrix

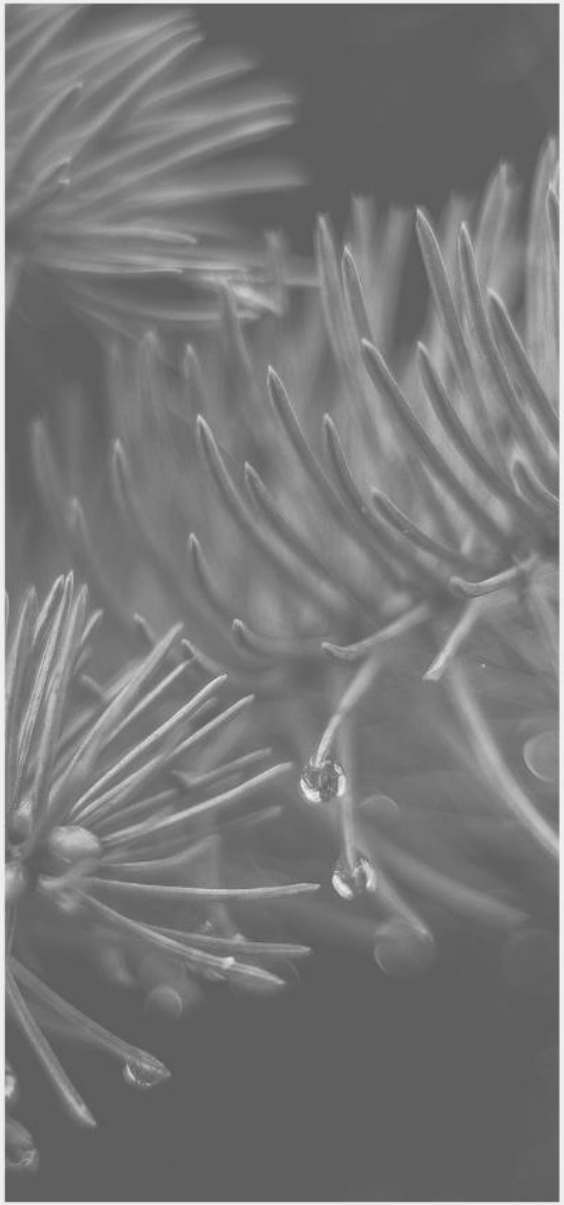
		T	G	T	T	A	C	G	G
	0	0	0	0	0	0	0	0	0
G	0								
G	0								
T	0								
T	0								
G	0								
A	0								
C	0								
T	0								
A	0								

Substitution matrix: $S(a_i, b_j) = \begin{cases} +3, & a_i = b_j \\ -3, & a_i \neq b_j \end{cases}$

Gap penalty: $W_k = kW_1$

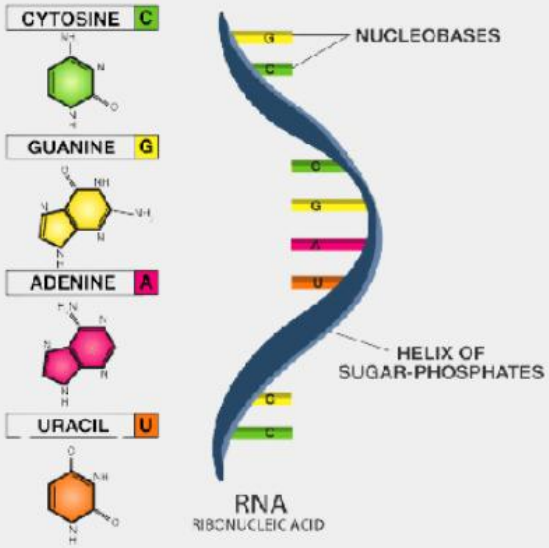
$W_1 = 2$

Nussinov Algorithm



SECONDARY STRUCTURE

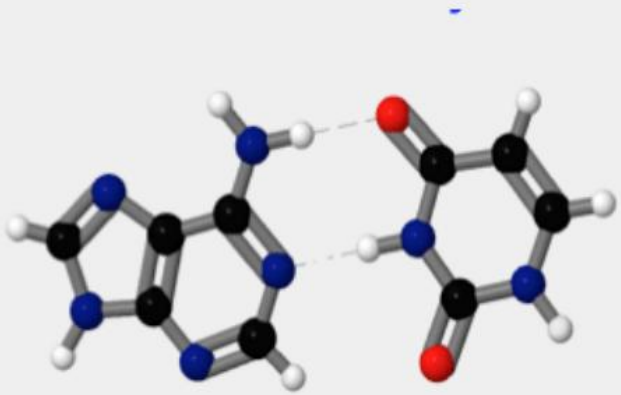
- 3D arrangement of base pairs
- Reveals nucleotides fold and interact
- Deep understanding

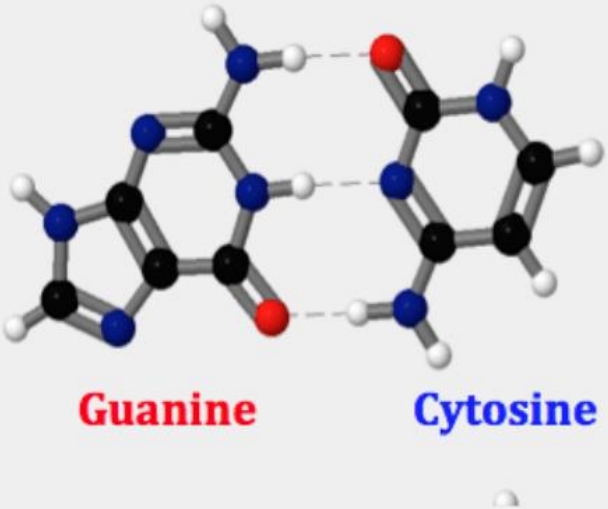


BASE PAIRING

Base pairing is a fundamental principle in RNA secondary structure.

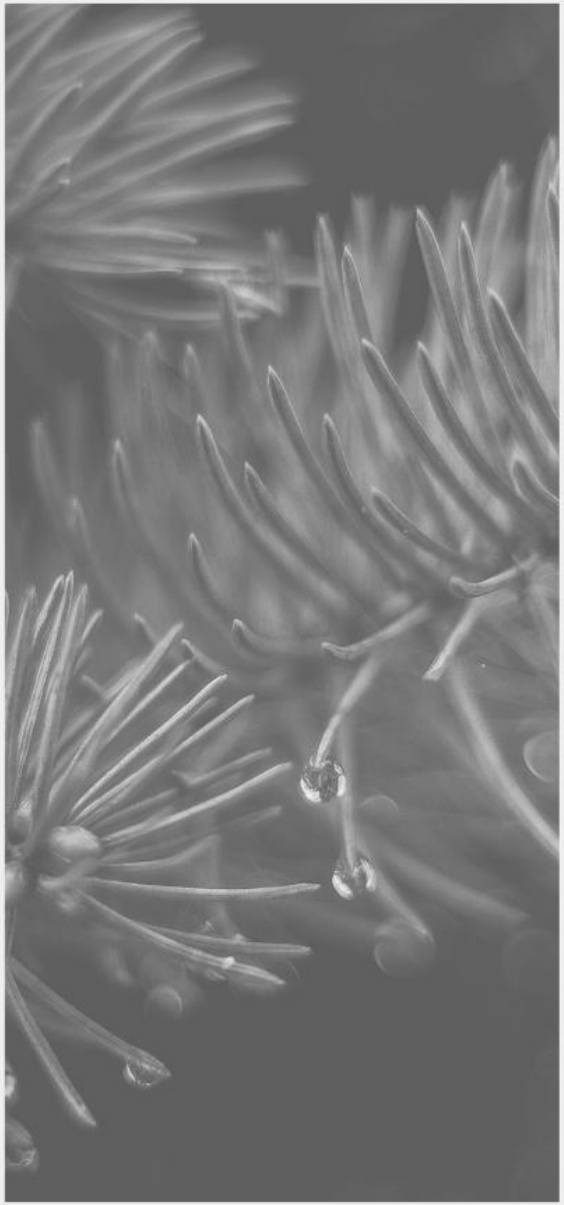
In RNA, Adenine (A) pairs with uracil (U), and guanine (G) pairs with cytosine (C).





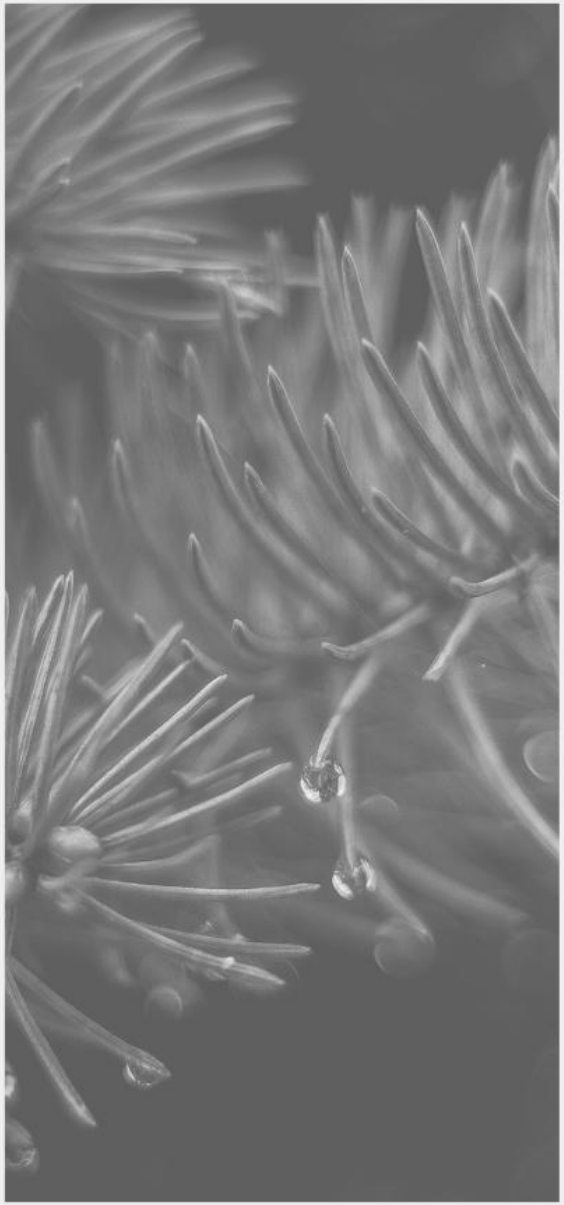
NON-CROSSING

- Fundamental aspect of RNA secondary structure
- Base pairs do not cross over each other
- Base pairs are nested and do not overlap



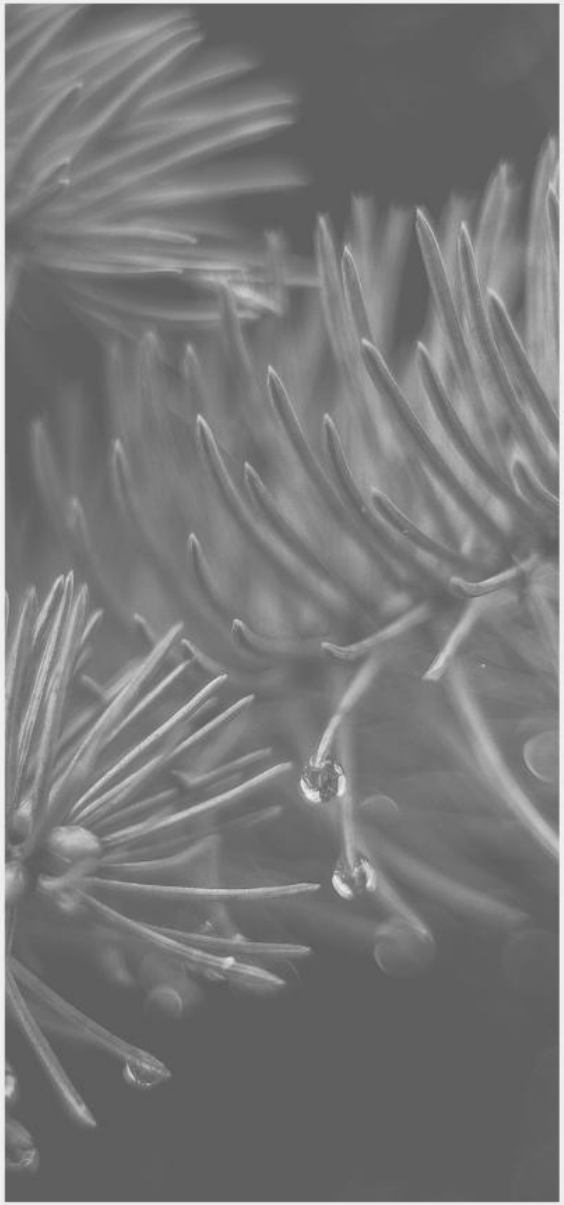
PREDICTING RSS

- RNA molecules are incredibly dynamic
- Secondary structures can vary greatly



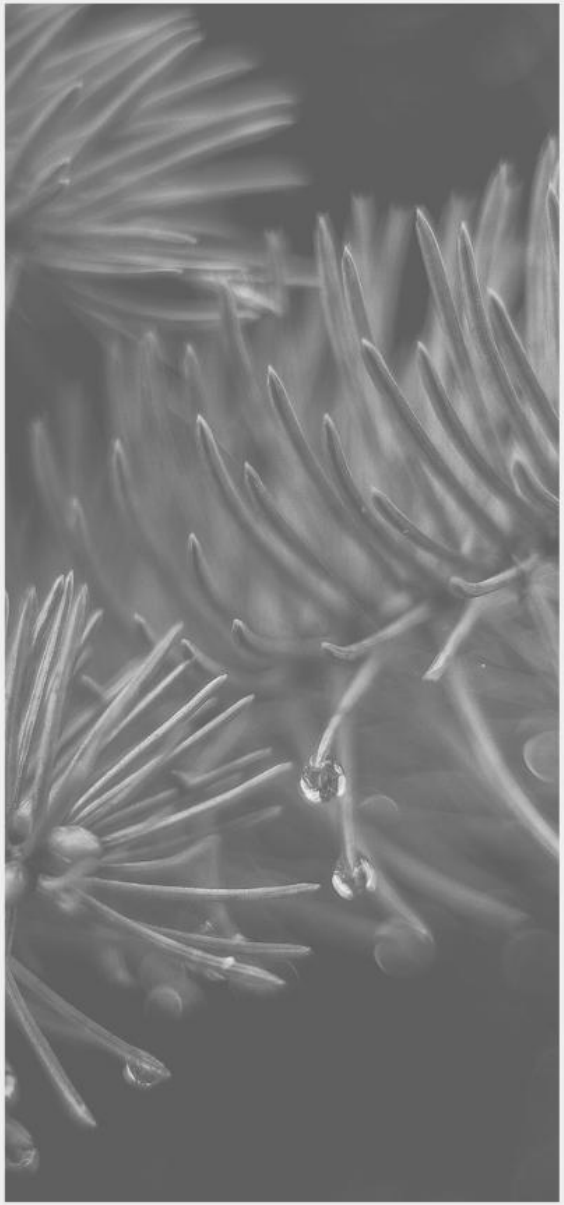
IMPORTANCE

- Drug Design
- Disease Mechanisms
- RNA Engineering



NUSSINOV ALGORITHM

- Dynamic programming algorithm
- Breaks down the complex task into sub problems
- Identifies patterns of complementary base pairs

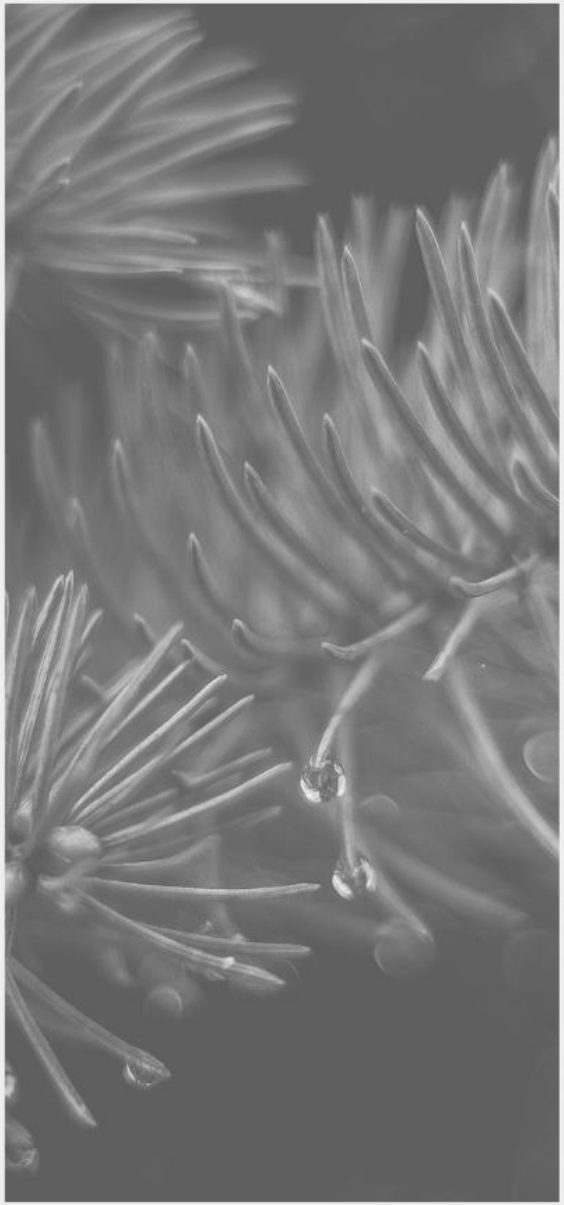


HISTORY

- Named after its creator Ruth Nussinov
- A breakthrough in the late 1970s.
- Crucial step in understanding how RNA molecules fold into stable, functional structures

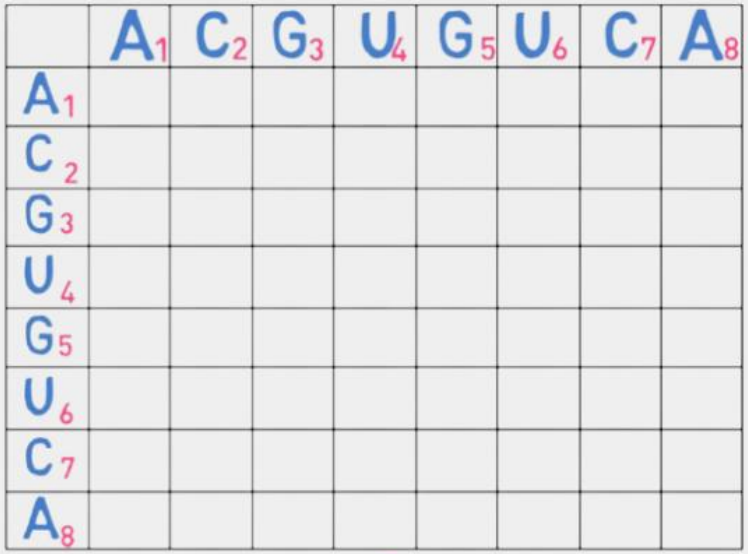
SIGNIFICANCE

- Significant in the study of RNA
- Earliest algorithm
- A foundational tool in RNA research.
- Adopted in multiple fields
- Numerous advancements



INITIALIZATION

- Matrix is set up as a two-dimensional table, with rows and columns corresponding to positions within the RNA sequence.



FILL THE MATRIX

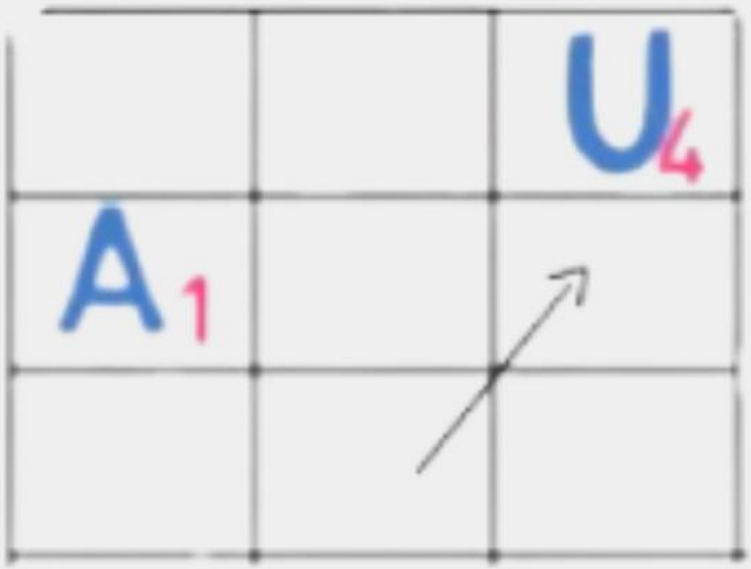
- Matrix is filled with values of maximum number of base pairs
- Core of this lies in the recursive calculations

SCORING SCHEME

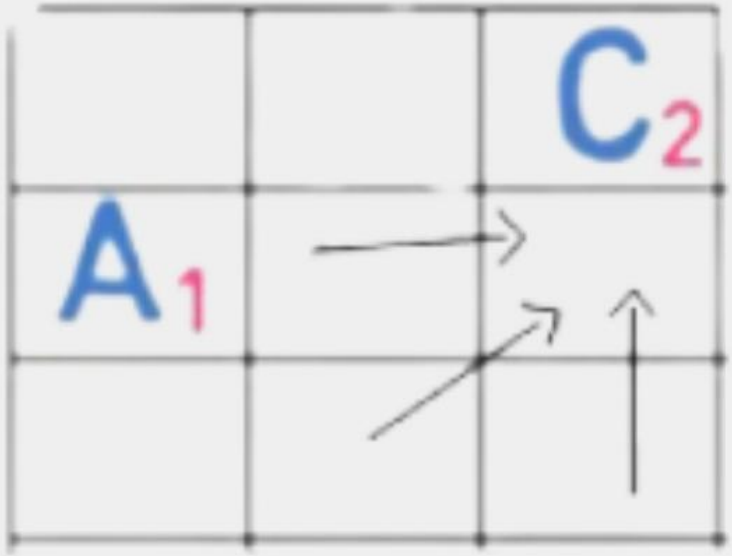
COMPLEMENTARY BASES: 1

NON-COMPLEMENTARY BASES: 0

COMPLIMENTARY

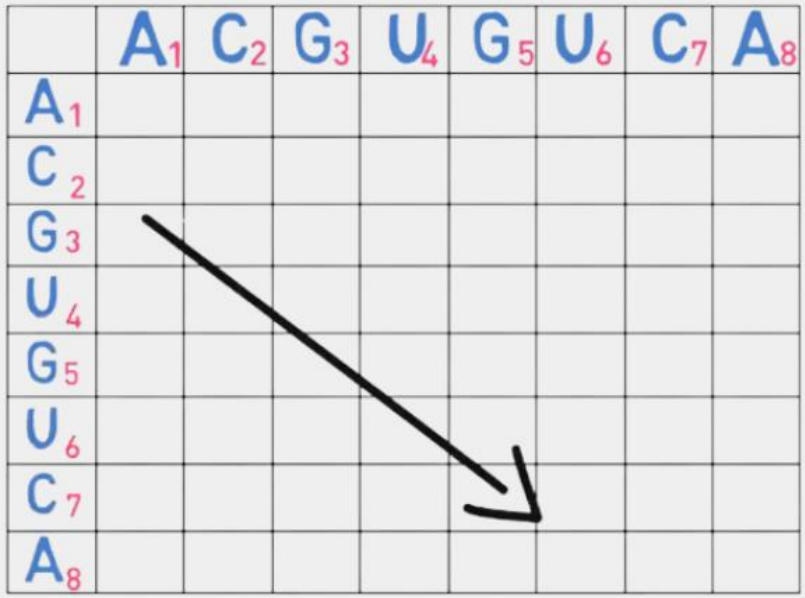


NON-COMPLIMENTARY



FILL THE MATRIX

- The matrix is typically initialized to zero, and diagonal cells (i.e., those where i equals j) are set to 0 as well.



FILL THE MATRIX

	A_1	C_2	G_3	U_4	G_5	U_6	C_7	A_8
A_1	0							
C_2		0						
G_3								
U_4								
G_5								
U_6								
C_7								
A_8								

FILL THE MATRIX

	A_1	C_2	G_3	U_4	G_5	U_6	C_7	A_8
A_1	0							
C_2	0	0						
G_3	0	0	0					
U_4	0	0	0	0				
G_5	0	0	0	0	0			
U_6	0	0	0	0	0	0		
C_7	0	0	0	0	0	0	0	
A_8	0	0	0	0	0	0	0	0

FILL THE MATRIX

- Using the scoring scheme start filling the matrix.

	A_1	C_2	G_3	U_4	G_5	U_6	C_7	A_8
A_1	0	0						
C_2	0	0						
G_3	0	0	0					
U_4	0	0	0	0				
G_5	0	0	0	0	0			
U_6	0	0	0	0	0	0		
C_7	0	0	0	0	0	0	0	
A_8	0	0	0	0	0	0	0	0

FILL THE MATRIX

	A_1	C_2	G_3	U_4	G_5	U_6	C_7	A_8
A_1	0	0	1	2				
C_2	0	0	1	1	2			
G_3	0	0	0	1	1			
U_4	0	0	0	0	1	1		
G_5	0	0	0	0	0	1	1	
U_6	0	0	0	0	0	0	0	1
C_7	0	0	0	0	0	0	0	0
A_8	0	0	0	0	0	0	0	0

FILL THE MATRIX

	A_1	C_2	G_3	U_4	G_5	U_6	C_7	A_8
A_1	0	0	1	2	2	3	2	2
C_2	0	0	1	1	2	2	2	2
G_3	0	0	0	1	1	2	2	2
U_4	0	0	0	0	1	1	1	2
G_5	0	0	0	0	0	1	1	1
U_6	0	0	0	0	0	0	0	1
C_7	0	0	0	0	0	0	0	0
A_8	0	0	0	0	0	0	0	0

TRACEBACK

- Find the highest value in the whole matrix

	A_1	C_2	G_3	U_4	G_5	U_6	C_7	A_8
A_1	0	0	1	2	2	3	2	2
C_2	0	0	1	1	2	2	2	2
G_3	0	0	0	1	1	2	2	2
U_4	0	0	0	0	1	1	1	2
G_5	0	0	0	0	0	1	1	1
U_6	0	0	0	0	0	0	0	1
C_7	0	0	0	0	0	0	0	0
A_8	0	0	0	0	0	0	0	0

TRACEBACK

- Start tracing back the path from where the present value is derived from.

	A_1	C_2	G_3	U_4	G_5	U_6	C_7	A_8
A_1	0	0	1	2	2	3	2	2
C_2	0	0	1	1	2	2	2	2
G_3	0	0	0	1	1	2	2	2
U_4	0	0	0	0	1	1	1	2
G_5	0	0	0	0	0	1	1	1
U_6	0	0	0	0	0	0	0	1
C_7	0	0	0	0	0	0	0	0
A_8	0	0	0	0	0	0	0	0

TRACEBACK

- Trace back till you encounter zero.

	A_1	C_2	G_3	U_4	G_5	U_6	C_7	A_8
A_1	0	0	1	2	2	3	2	2
C_2	0	0	1	1	2	2	2	2
G_3	0	0	0	1	1	2	2	2
U_4	0	0	0	0	1	1	1	2
G_5	0	0	0	0	0	1	1	1
U_6	0	0	0	0	0	0	0	1
C_7	0	0	0	0	0	0	0	0
A_8	0	0	0	0	0	0	0	0

SECONDARY STRUCTURE

- Join the matching pairs

	A_1	C_2	G_3	U_4	G_5	U_6	C_7	A_8
A_1	0	0	1	2	2	3	2	2
C_2	0	0	1	1	2	2	2	2
G_3	0	0	0	1	1	2	2	2
U_4	0	0	0	0	1	1	1	2
G_5	0	0	0	0	0	1	1	1
U_6	0	0	0	0	0	0	0	1
C_7	0	0	0	0	0	0	0	0
A_8	0	0	0	0	0	0	0	0

A1 — U6

SECONDARY STRUCTURE

- Join the matching pairs

	A_1	C_2	G_3	U_4	G_5	U_6	C_7	A_8
A_1	0	0	1	2	2	3	2	2
C_2	0	0	1	1	2	2	2	2
G_3	0	0	0	1	1	2	2	2
U_4	0	0	0	0	1	1	1	2
G_5	0	0	0	0	0	1	1	1
U_6	0	0	0	0	0	0	0	1
C_7	0	0	0	0	0	0	0	0
A_8	0	0	0	0	0	0	0	0

SECONDARY STRUCTURE

- Join the matching pairs

	A_1	C_2	G_3	U_4	G_5	U_6	C_7	A_8
A_1	0	0	1	2	2	3	2	2
C_2	0	0	1	1	2	2	2	2
G_3	0	0	0	1	1	2	2	2
U_4	0	0	0	0	1	1	1	2
G_5	0	0	0	0	0	1	1	1
U_6	0	0	0	0	0	0	0	1
C_7	0	0	0	0	0	0	0	0
A_8	0	0	0	0	0	0	0	0

SECONDARY STRUCTURE

- Join the matching pairs

	A_1	C_2	G_3	U_4	G_5	U_6	C_7	A_8
A_1	0	0	1	2	2	3	2	2
C_2	0	0	1	1	2	2	2	2
G_3	0	0	0	1	1	2	2	2
U_4	0	0	0	0	1	1	1	2
G_5	0	0	0	0	0	1	1	1
U_6	0	0	0	0	0	0	0	1
C_7	0	0	0	0	0	0	0	0
A_8	0	0	0	0	0	0	0	0

DOT BRACKET

- Secondary structure can also be represented using dot bracket notation
- Here every dot/period represents a isolated nucleotide.
- Every open parenthesis show that it is paired with a nucleotide after it
- Closing parenthesis shows that it is paired with nucleotide before it.

.()(())

Problem

- To predict the RNA secondary structure with a RNA sequence as a input.

- Input: RNA Sequence

- Output: RNA secondary structure in Dot bracket format

Java Implementation

NussinovAlgorithm.java

```java
import java.util.Scanner;


public class NussinovAlgorithm {

    public static void main(String[] args) {
        Scanner sc= new Scanner(System.in);
        String rnaSequence = sc.next();
        int n = rnaSequence.length();
        int[][] dp = new int[n][n];

        // Initialize the dynamic programming matrix
        for (int len = 1; len < n; len++) {
            for (int i = 0; i < n - len; i++) {
                int j = i + len;
                dp[i][j] = Math.max(dp[i + 1][j], dp[i][j - 1]);
                if (areComplementary(rnaSequence.charAt(i), rnaSequence.charAt(j))) {
                    dp[i][j] = Math.max(dp[i][j], dp[i + 1][j - 1] + 1);
                }
                for (int k = i + 1; k < j; k++) {
                    dp[i][j] = Math.max(dp[i][j], dp[i][k] + dp[k + 1][j]);
                }
            }
        }

        // Print the DP matrix
```

OUTPUT

```
C:\Users\Student\.jdks\openjdk-21\bin\java.exe "-javaagent:C:\Pr
ACGUGUCA
0   0   1   2   2   2   3   3
0   0   1   1   1   1   2   3
0   0   0   0   0   0   1   2
0   0   0   0   0   0   1   2
0   0   0   0   0   0   1   1
0   0   0   0   0   0   0   1
0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0

RNA Sequence: ACGUGUCA

Predicted Secondary Structure: .()(())

Process finished with exit code 0
```

DRAWBACKS

- The time complexity of the Nussinov is O(n3)

- The various reasons for the cubic complexity are:
  - Three nested loops
  - Consideration of all pairings
  - Cubic complexity per cell

LIMITATIONS

- Inability to handle pseudoknots
- Overestimation of stability
- Lack of consideration for energy parameters
- Sensitivity to sequence length

Using Cellular Automata

What is Automata?

- Automata theory is a branch of computer science and mathematics that studies abstract computational models known as automata.
- These models, which include finite automata, pushdown automata, and Turing machines, are used to understand how algorithms work and to solve problems in various domains.
- Automata are foundational in areas like formal language theory, compiler design, and theoretical computer science.

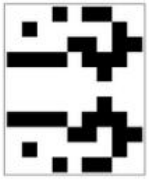
Correlation to Computational biology & bioinformatics?

- Cellular automata are mathematical models consisting of a grid of cells that evolve over discrete time steps according to a set of rules. Each cell's future state is determined by its current state and the states of its neighboring cells.
- Cellular automata are used in various fields such as physics, biology, and computer science to model complex systems and phenomena.

Cellular Automata

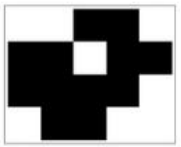
index 115
Glider
John Conway, 1970
period 4
displacement {1, 1}
size {3, 3}

index 10240
Hartmut Holzwart, 1992
period 4
displacement {2, 0}
size {9, 11}



index 13491
Dean Hickerson
period 5
displacement {2, 0}
size {10, 15}

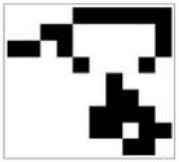
index 1062
Lightweight Spaceship
John Conway, 1970
period 4
displacement {2, 0}
size {5, 4}



index 11302
Turtle
Dean Hickerson
period 3
displacement {1, 0}
size {12, 10}

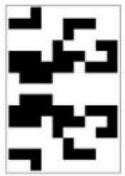
index 13881
Dean Hickerson
period 3
displacement {1, 0}
size {5, 16}

index 9153
Coe Ship
Tim Coe, 1995
period 16
displacement {8, 0}
size {10, 9}



index 13374
Schick Engine
Paul Schick, 1972
period 12
displacement {6, 0}
size {15, 9}

index 14105
Weekender
David Eppstein, 2000
period 7
displacement {2, 0}
size {10, 16}

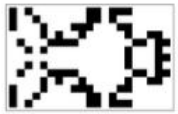


index 16485
period 2
displacement {1, 0}
size {12, 21}

index 16532
period 4
displacement {1, 0}
size {21, 13}

index 17361
Jason Summers, 2005
period 5
displacement {1, 1}
size {22, 22}

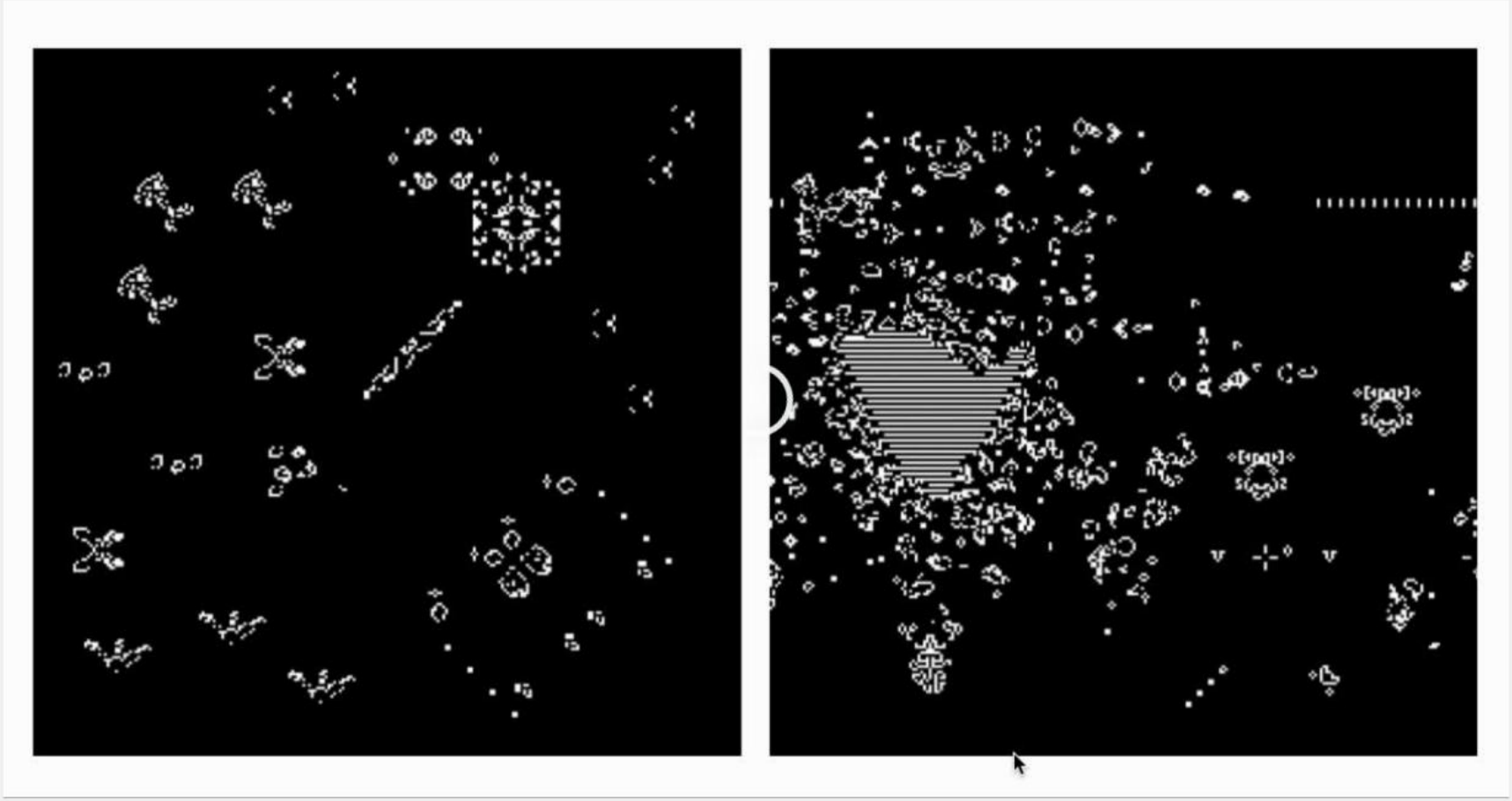
index 18216
Spider
David Bell, 1997
period 5
displacement {1, 0}
size {8, 27}



index 18788
Dragon
Paul Tooke, 2000
period 6
displacement {1, 0}
size {30, 14}

index 19739
Seal
N. Beluchenko, 2005
period 6
displacement {1, 1}
size {35, 34}

What can you do?



Rules?

1. **Birth: A dead cell with exactly three live neighbors becomes a live cell.**
2. **Survival: A live cell with two or three live neighbors remains alive.**
3. **Death:**
    - **Overpopulation: A live cell with more than three live neighbors dies.**
    - **Loneliness: A live cell with fewer than two live neighbors dies.**

Rules for computational biology

1. Self-replication
2. Blueprint
3. Inherit blueprint
4. Mutation
5. Selection

How it works

ACTION OUTPUTS

LPD = set long-probe distance
Kill = kill forward neighbor
OSC = set oscillator period
SG = emit pheromone
Res = set responsiveness

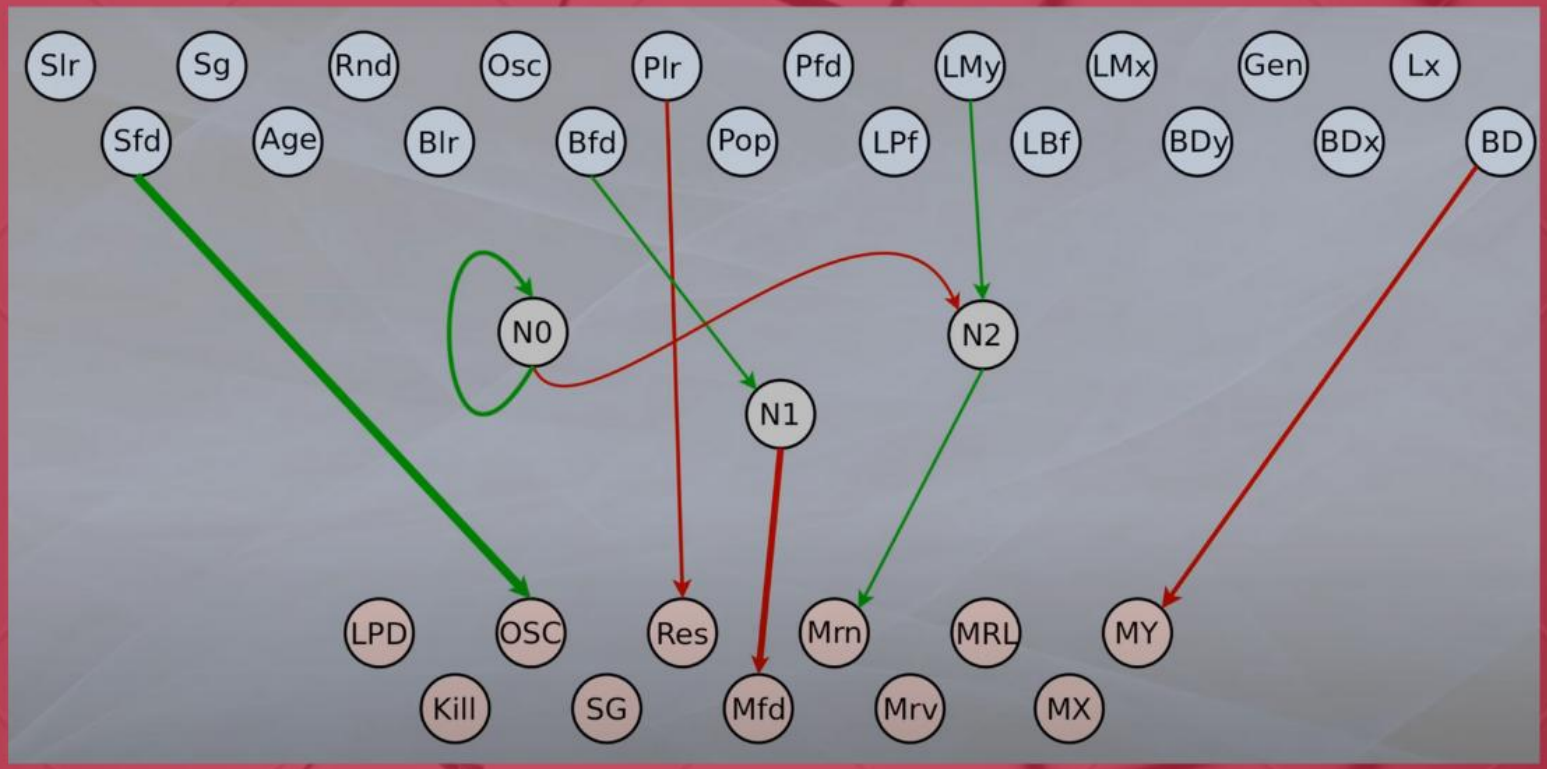
Mfd = move forward
Mrn = move random
Mrv = move reverse
MRL = move left/right (+/-)
MX = move east/west (+/-)
MY = move north/south (+/-)

SENSORY INPUTS

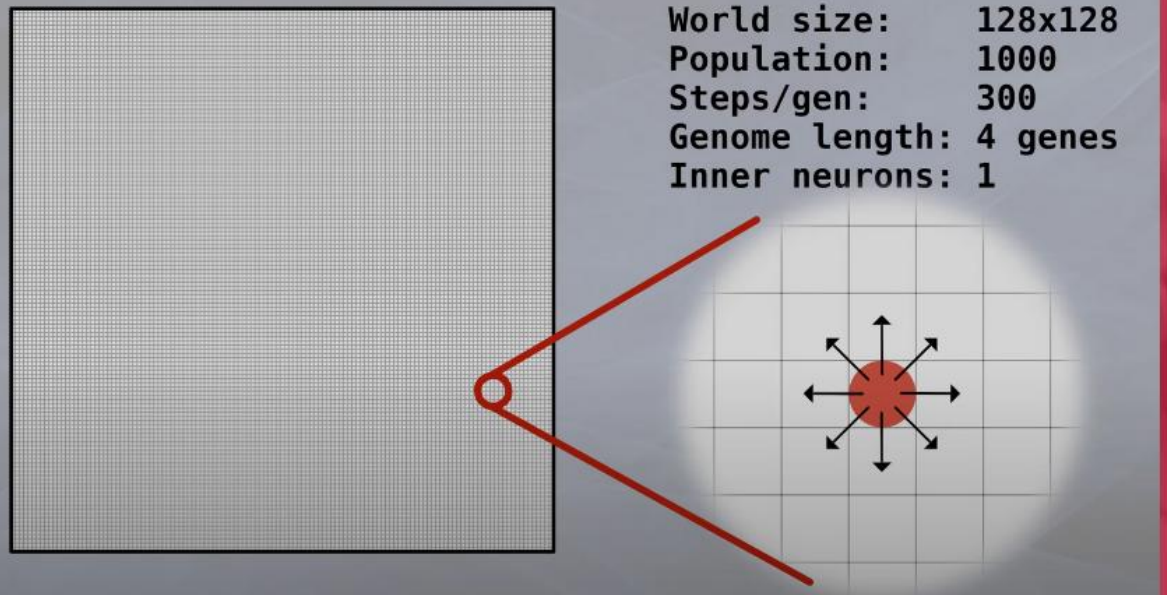
Slr = pheromone gradient left-right
Sfd = pheromone gradient forward
Sg = pheromone density
Age = age
Rnd = random input
Blr = blockage left-right
Osc = oscillator
Bfd = blockage forward
Plr = population gradient left-right
Pop = population density
Pfd = population gradient forward

LPf = population long-range forward
LMy = last movement Y
LBf = blockage long-range forward
LMx = last movement X
BDy = north/south border distance
Gen = genetic similarity of fwd neighbor
BDx = east/west border distance
Lx = east/west world location
BD = nearest border distance
Ly = north/south world location

Connections

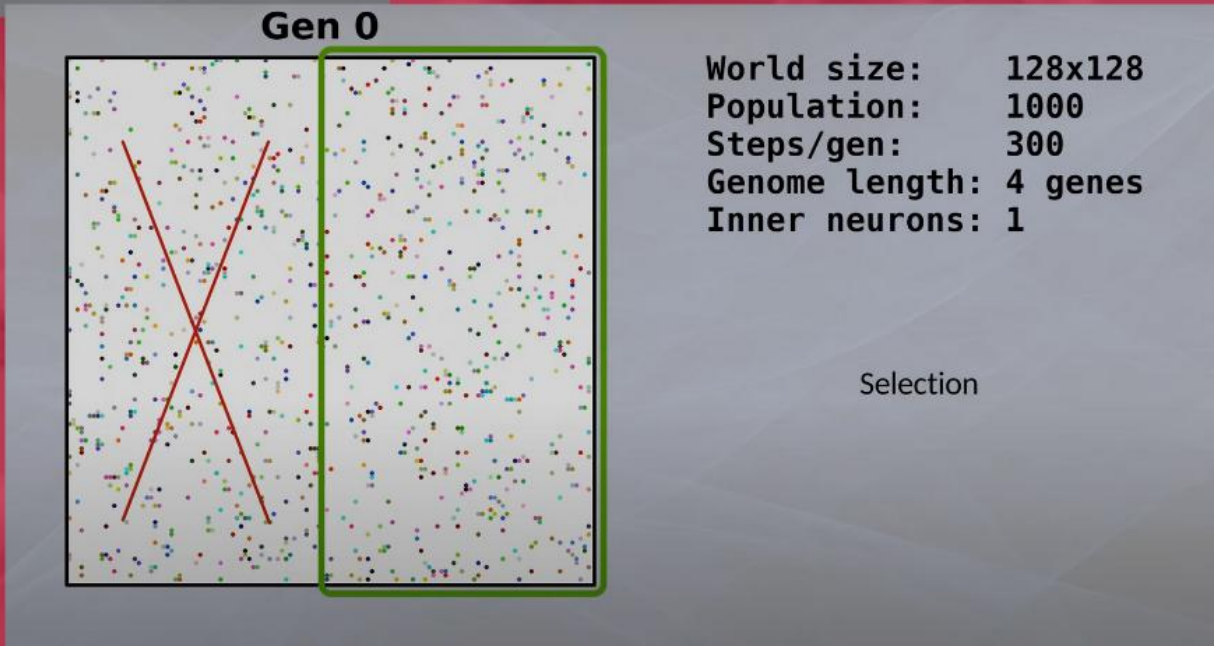


Create an environment

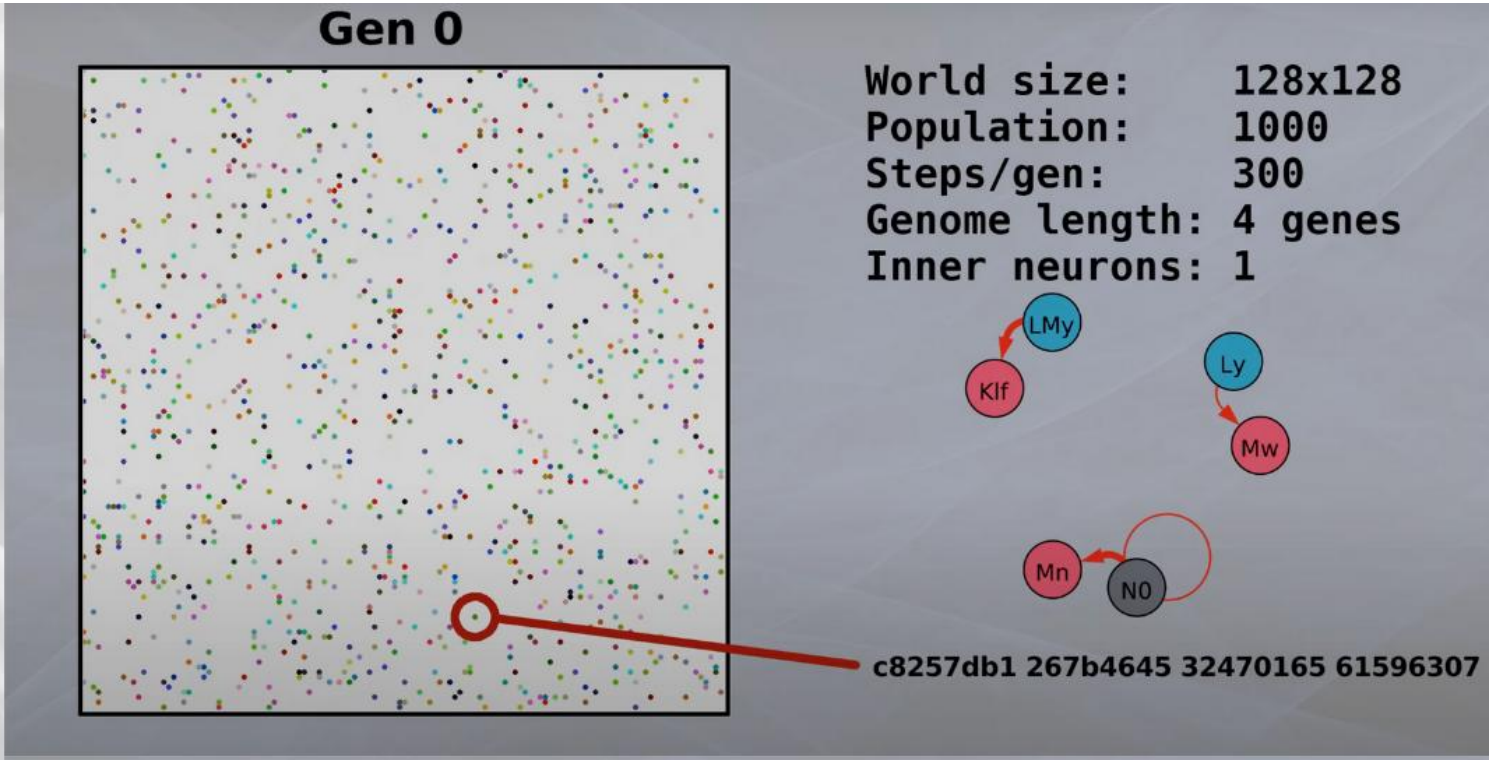


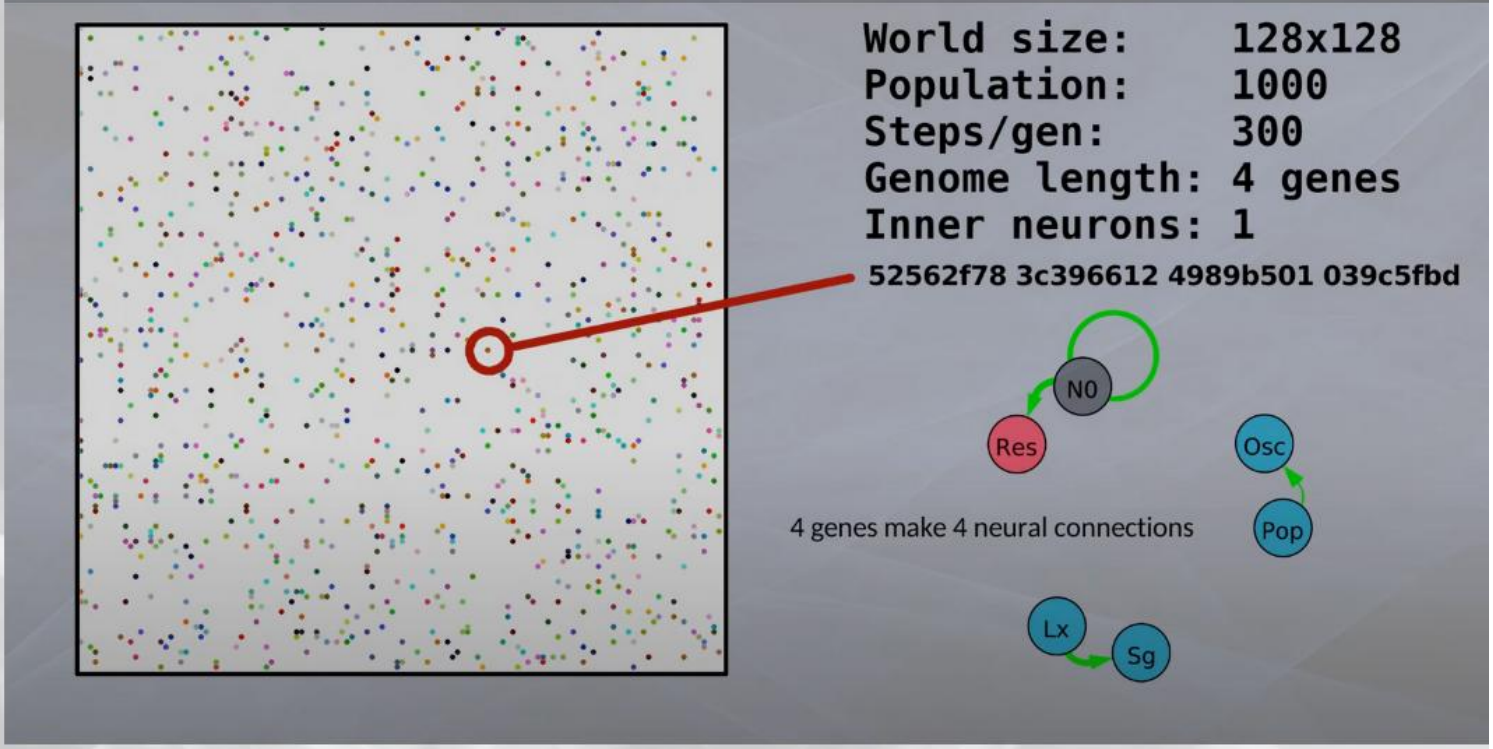
Set parameters

Introduce rules



Founding Generation



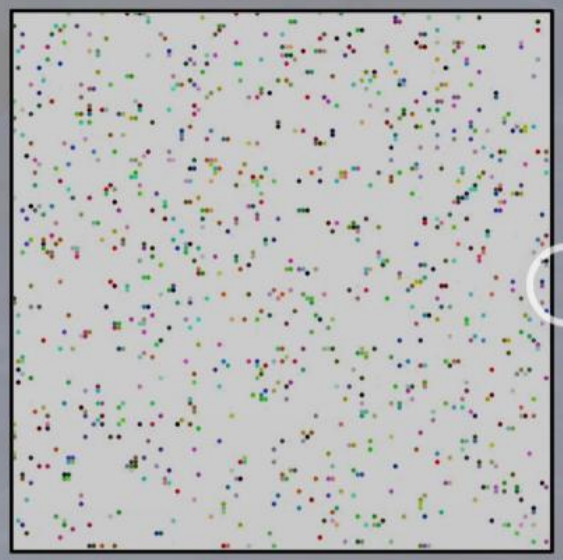


First simulations

Gen 0

```
World size:     128x128
Population:     1000
Steps/gen:      300
Genome length:  4 genes
Inner neurons:  1
```

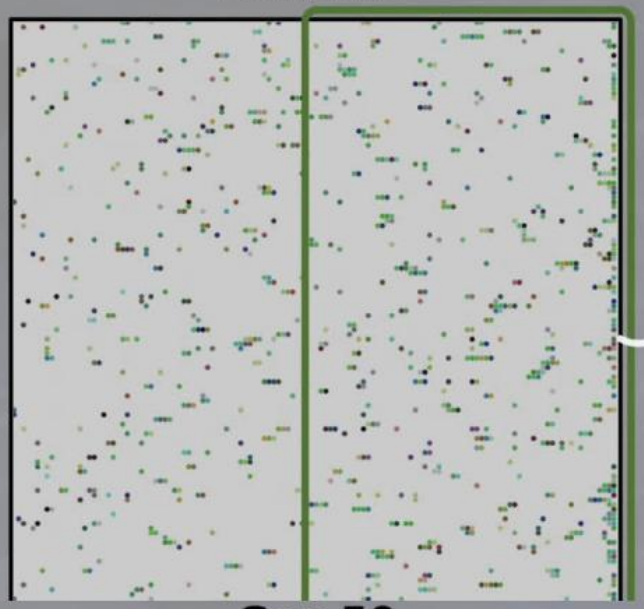
Gen 1



```
World size:     128x128
Population:     1000
Steps/gen:      300
Genome length:  4 genes
Inner neurons:  1
```

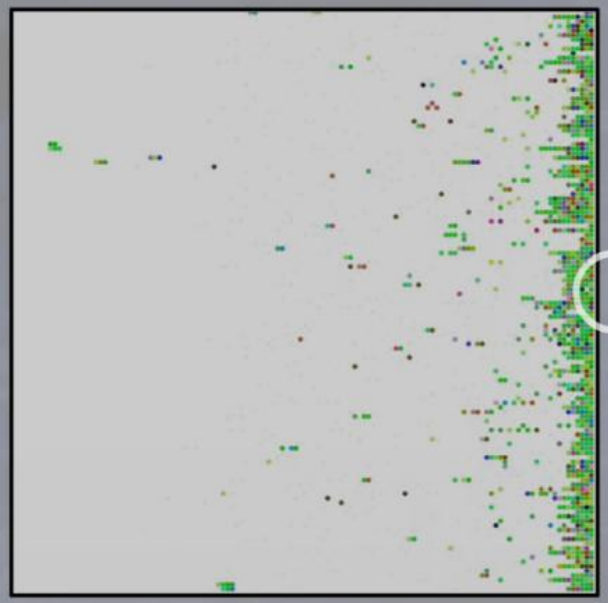
Generation 20 & 50

Gen 20



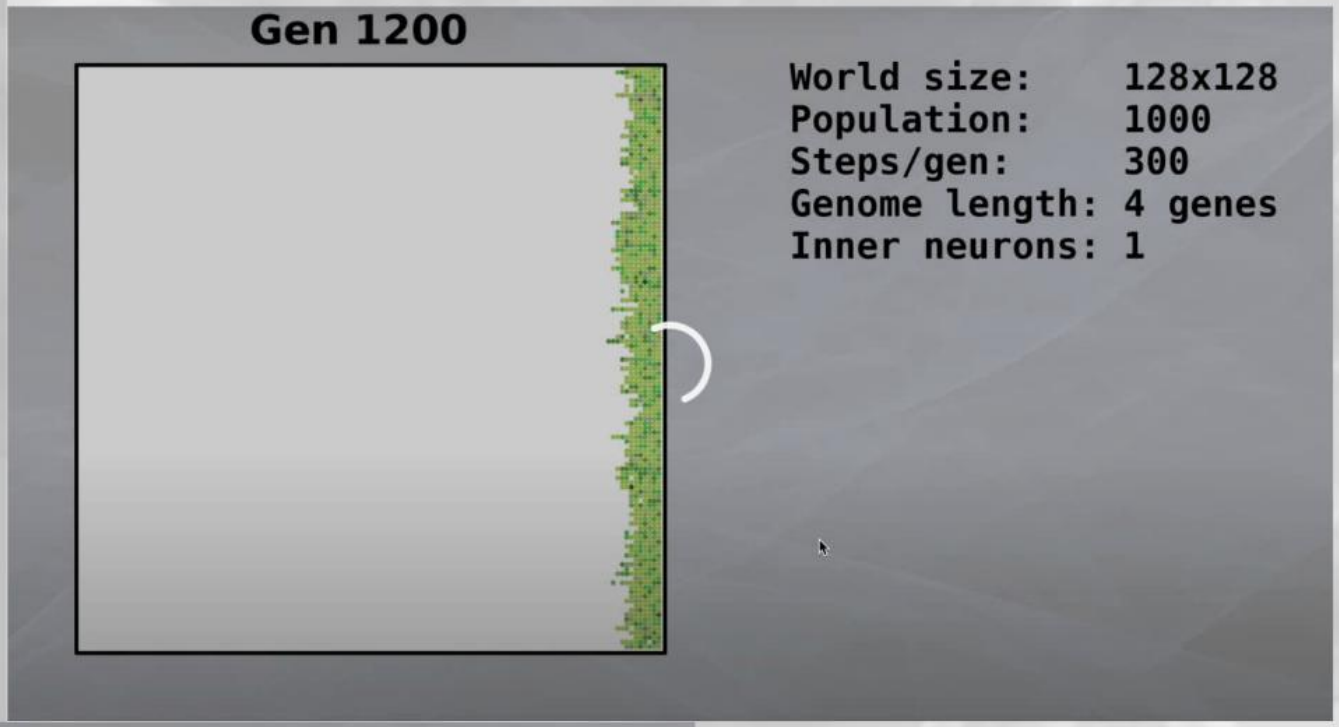
World size: 128x128
Population: 1000
Steps/gen: 300
Genome length: 4 genes
Inner neurons: 1

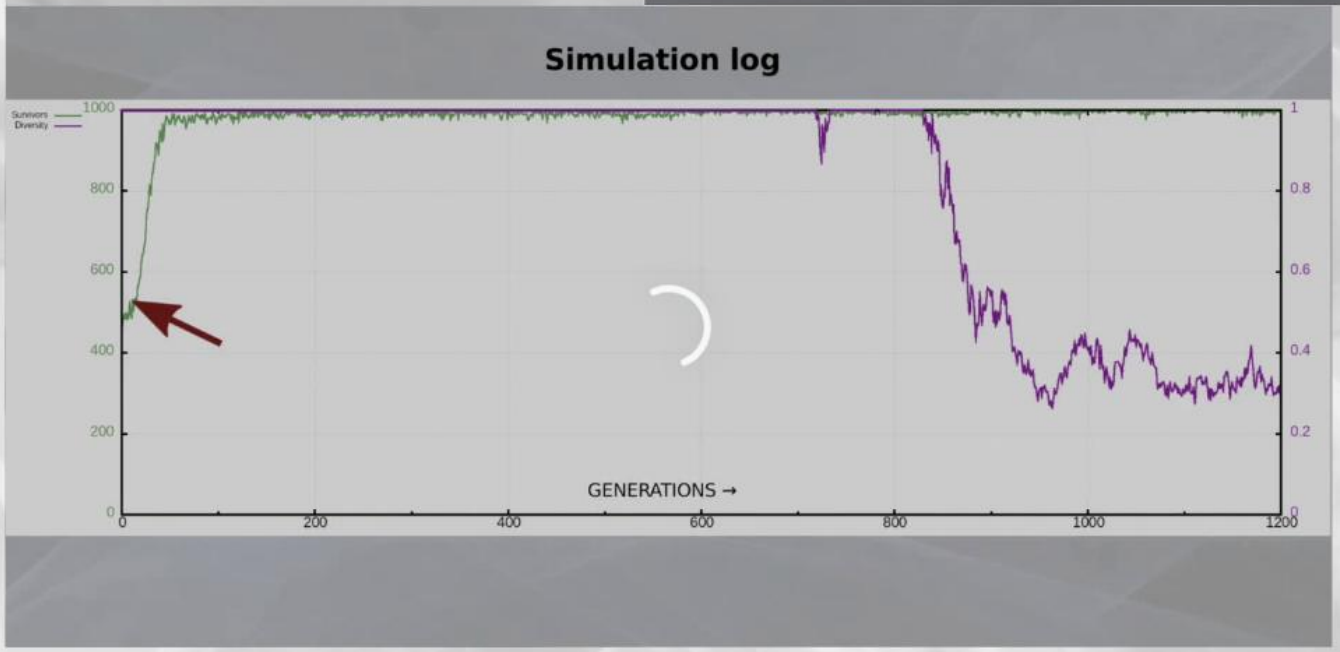
Gen 50



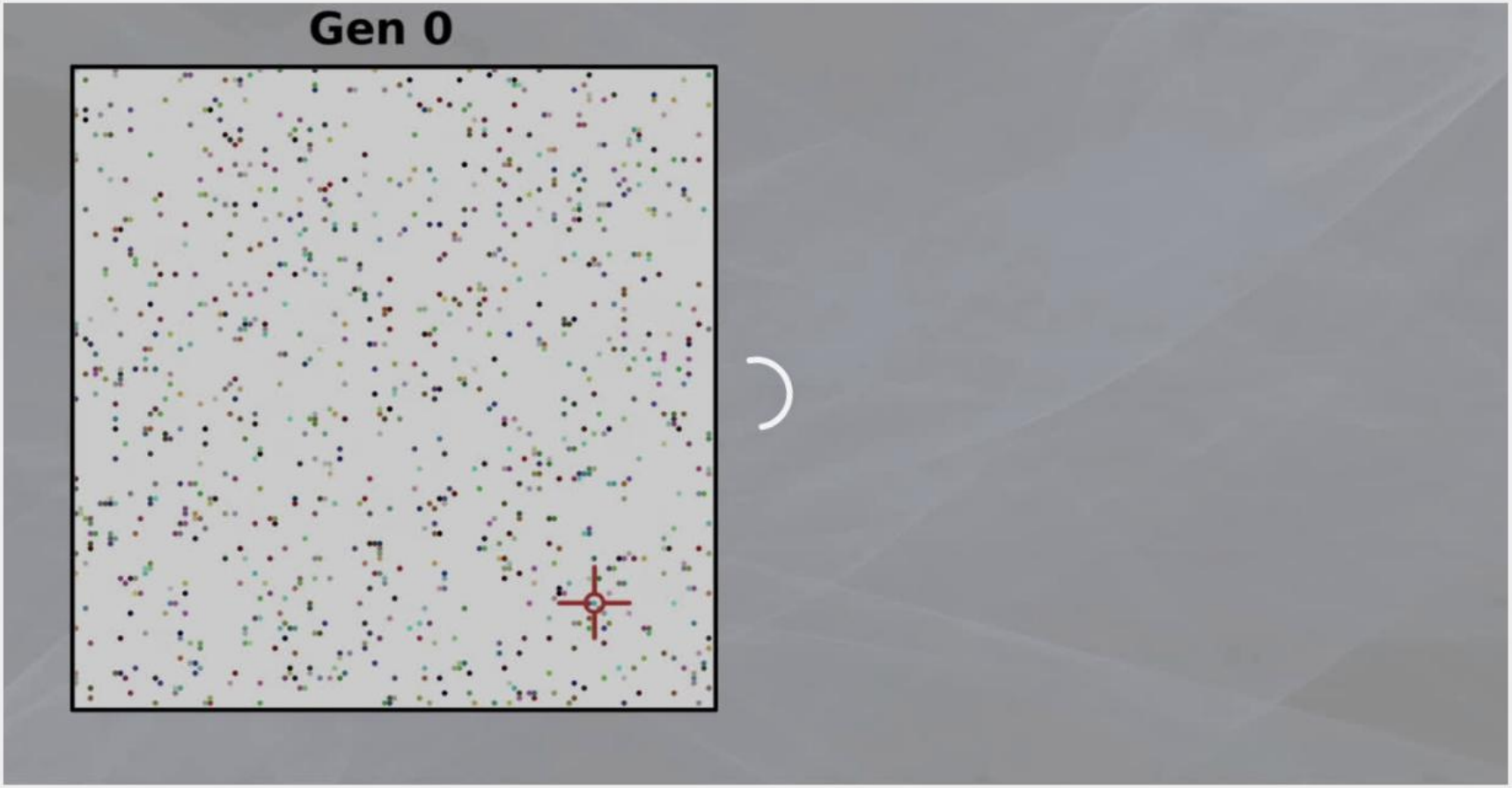
World size: 128x128
Population: 1000
Steps/gen: 300
Genome length: 4 genes
Inner neurons: 1

Conclusions



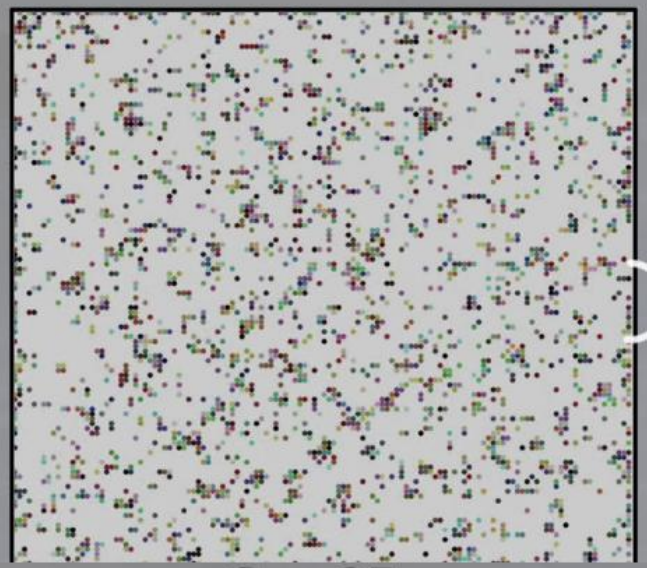


Natural selection



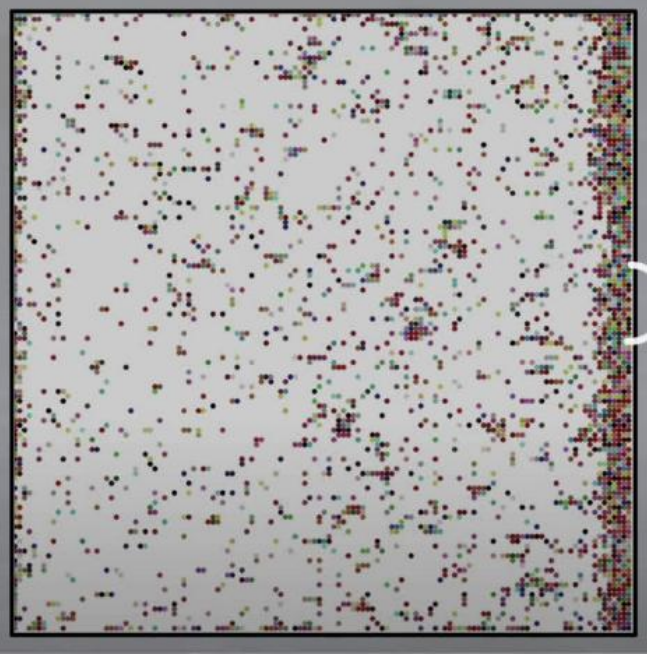
Simulation 2

Gen 0



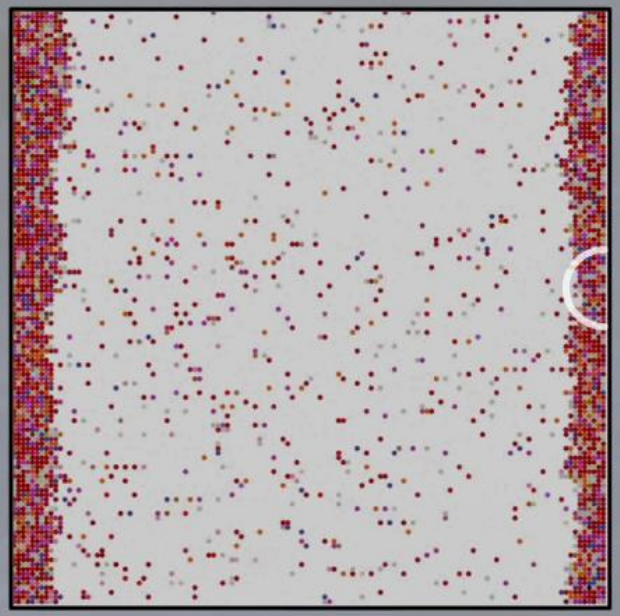
```
World size:    128x128
Population:    3000
Steps/gen:     300
Genome length: 24 genes
Inner neurons: 4
Mutation rate: 0.001
```

Gen 25



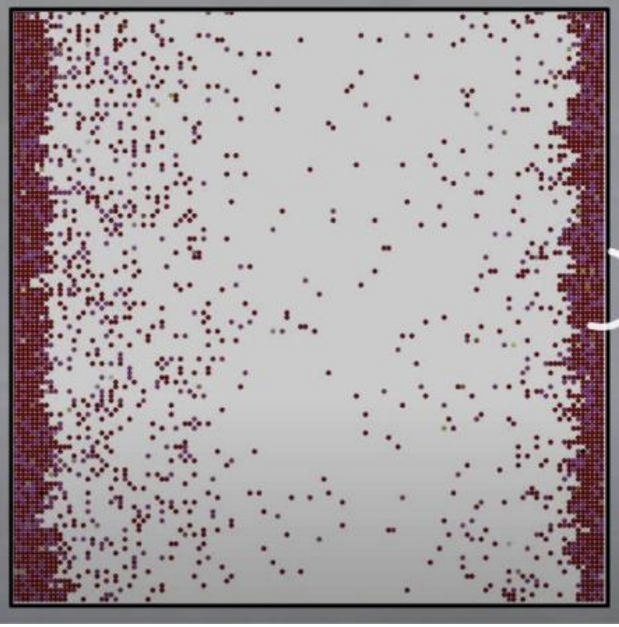
```
World size:    128x128
Population:    3000
Steps/gen:     300
Genome length: 24 genes
Inner neurons: 4
Mutation rate: 0.001
```

Gen 1000



World size: 128x128
Population: 3000
Steps/gen: 300
Genome length: 24 genes
Inner neurons: 4
Mutation rate: 0.001

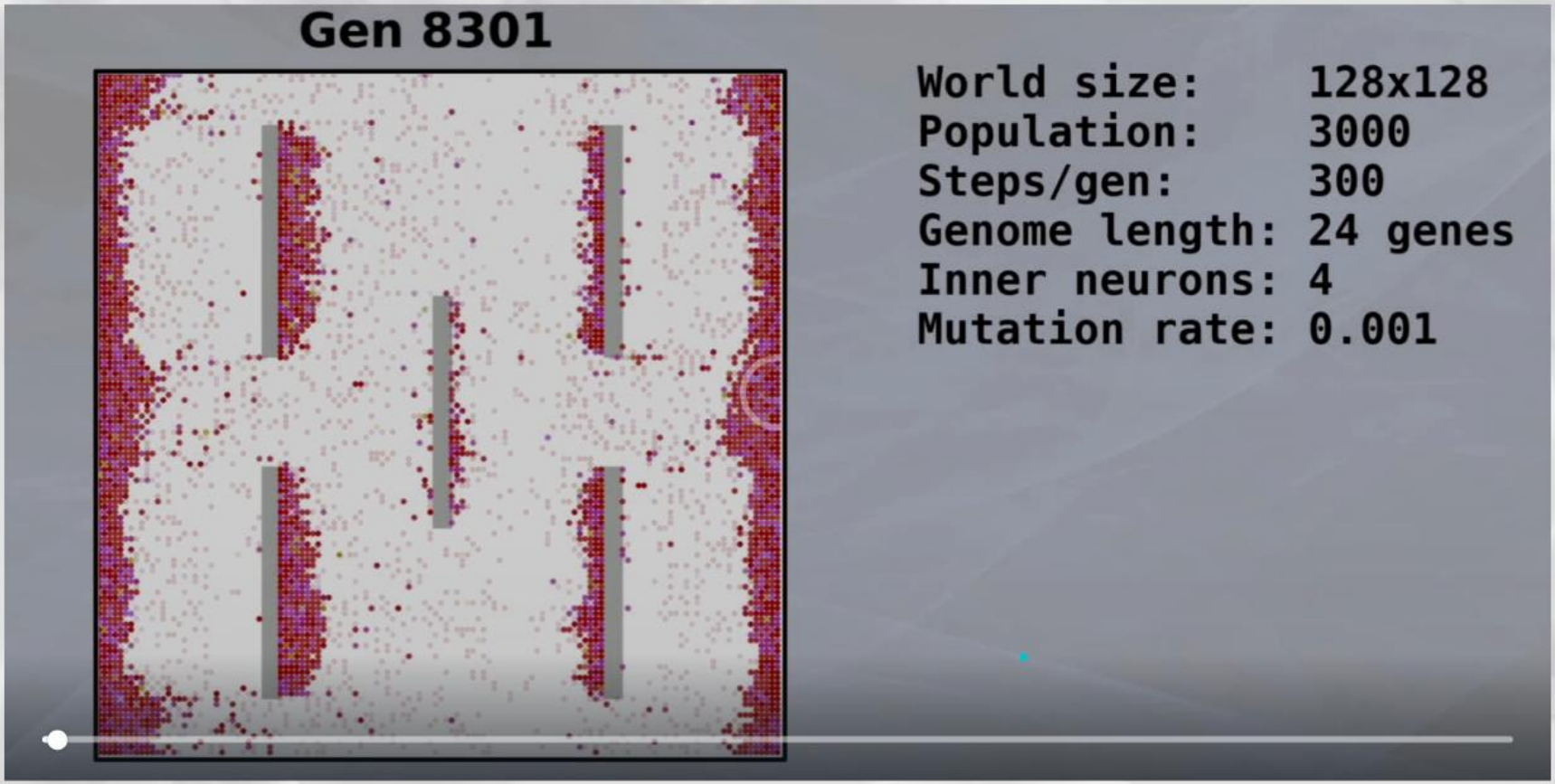
Gen 8300



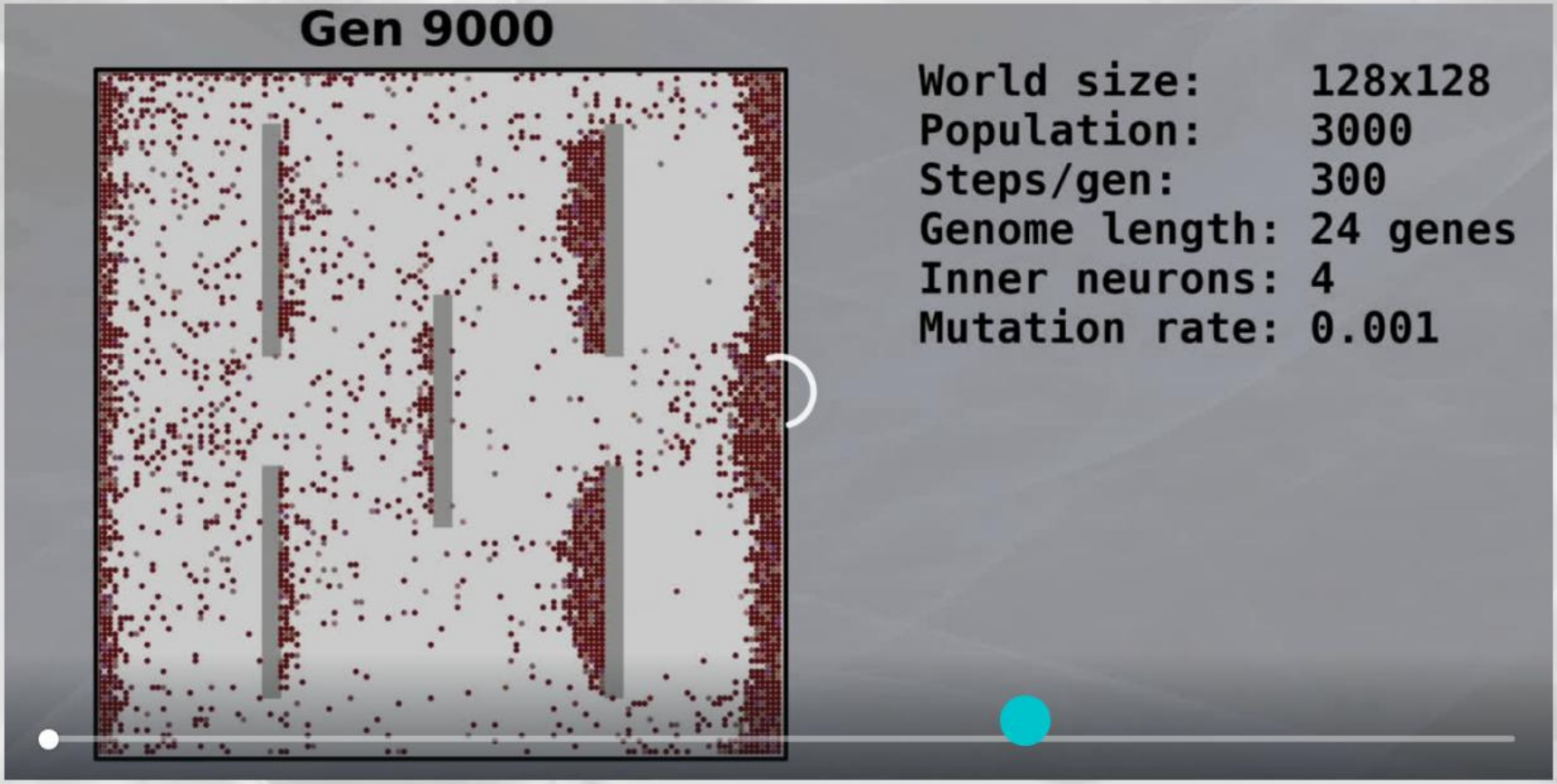
World size: 128x128
Population: 3000
Steps/gen: 300
Genome length: 24 genes
Inner neurons: 4
Mutation rate: 0.001

99.3%

Power of Mutations



After 700 Generations



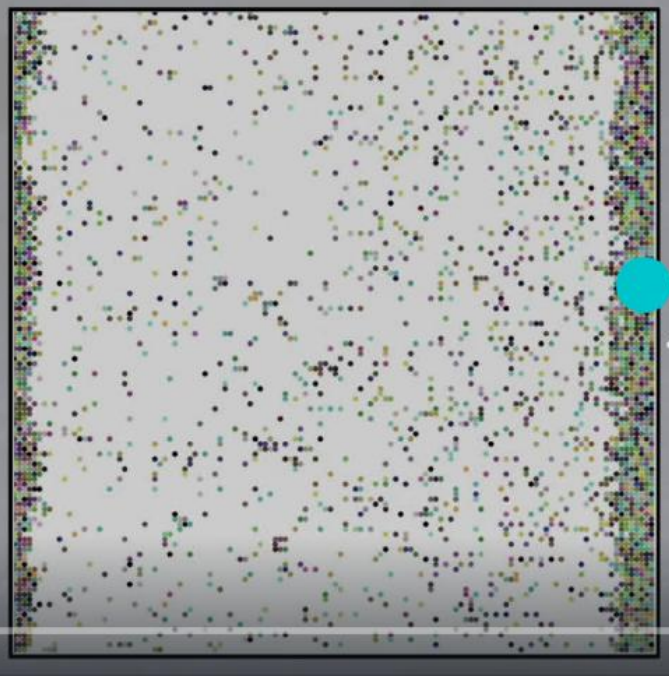
Gen 177,000

World size: 128x128
Population: 3000
Steps/gen: 300
Genome length: 24 genes
Inner neurons: 4
Mutation rate: 0.001

83%

The Beauty in Randomnesses

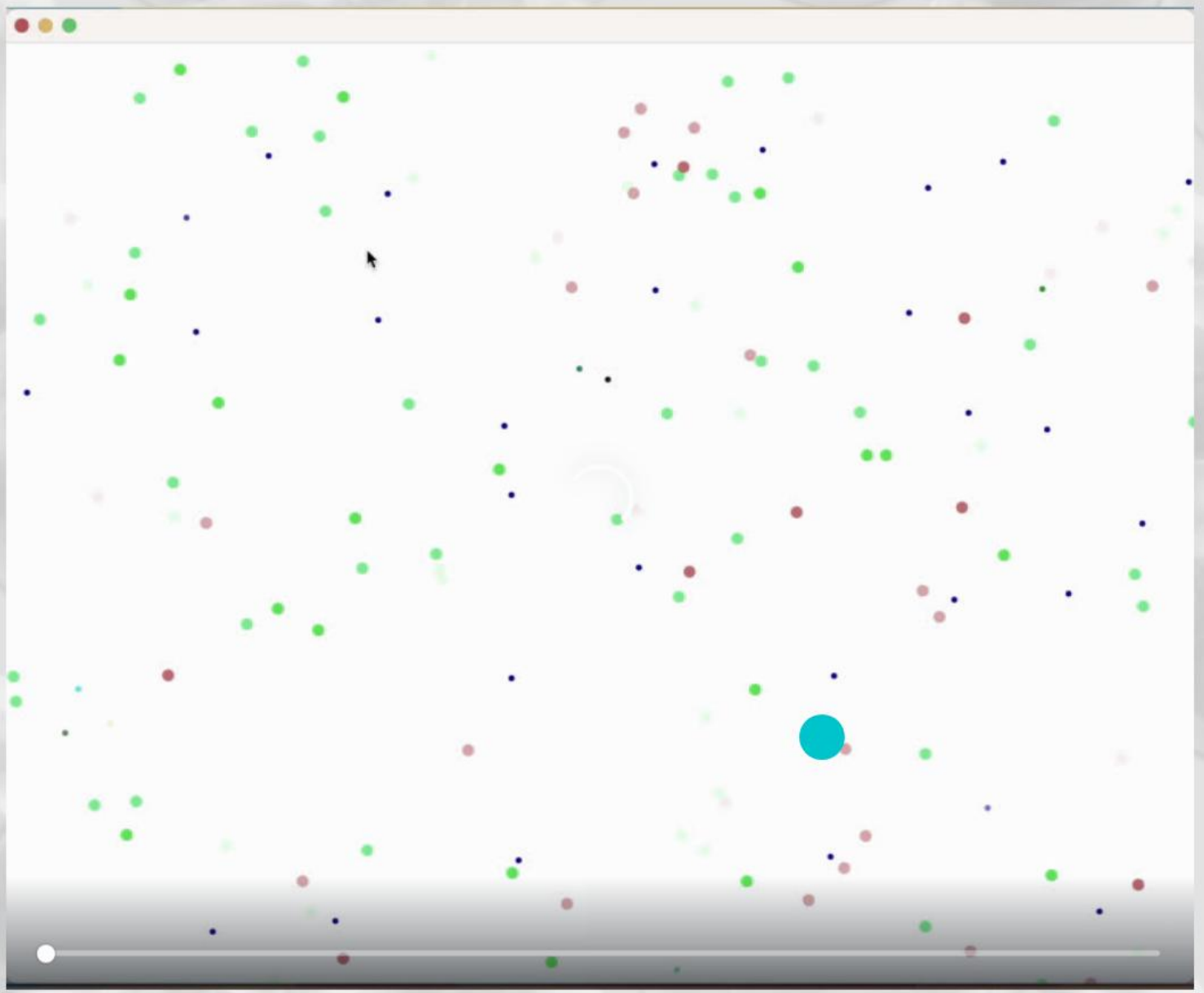
Gen 100



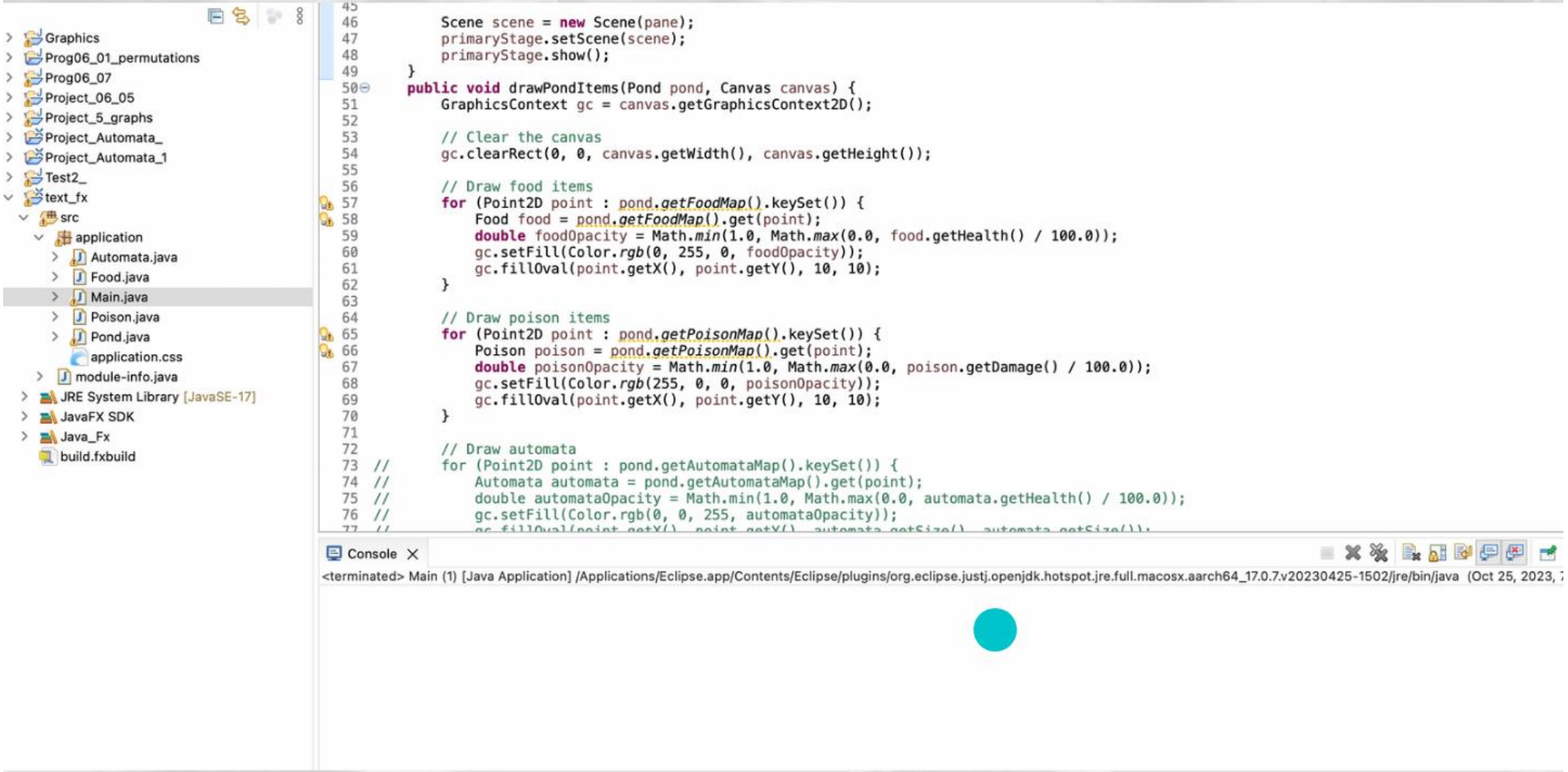
World size: 128x128
Population: 3000
Steps/gen: 300
Genome length: 24 genes
Inner neurons: 4
Mutation rate: 0.0

97%

Pond Simulation



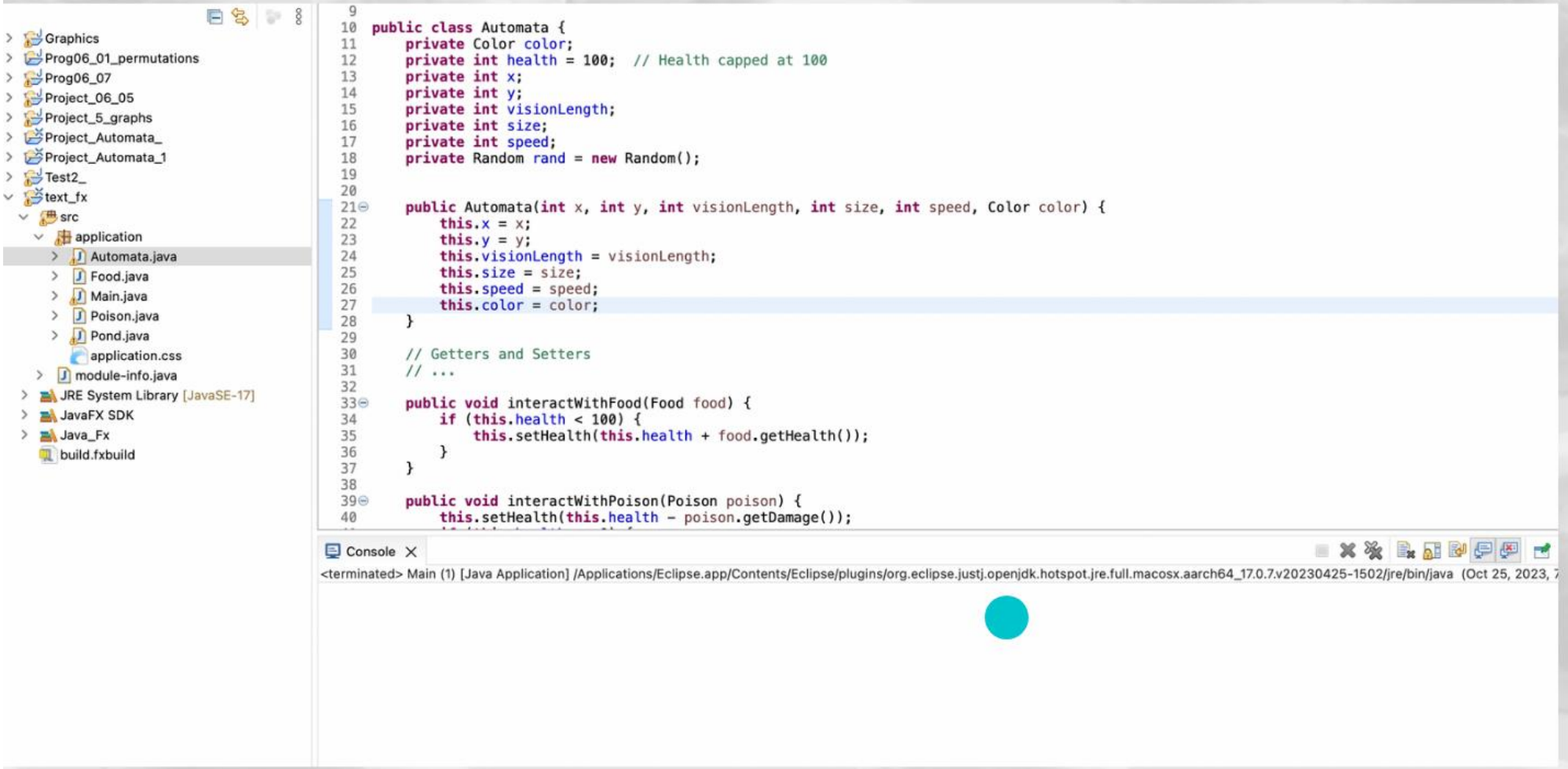
Main


Graphics
Prog06_01_permutations
Prog06_07
Project_06_05
Project_5_graphs
Project_Automata_
Project_Automata_1
Test2_
text_fx
src
application
Automata.java
Food.java
Main.java
Poison.java
Pond.java
application.css
module-info.java
JRE System Library [JavaSE-17]
JavaFX SDK
Java_Fx
build.fxbuild
Scene scene = new Scene(pane);
primaryStage.setScene(scene);
primaryStage.show();
}
public void drawPondItems(Pond pond, Canvas canvas) {
GraphicsContext gc = canvas.getGraphicsContext2D();
// Clear the canvas
gc.clearRect(0, 0, canvas.getWidth(), canvas.getHeight());
// Draw food items
for (Point2D point : pond.getFoodMap().keySet()) {
Food food = pond.getFoodMap().get(point);
double foodOpacity = Math.min(1.0, Math.max(0.0, food.getHealth() / 100.0));
gc.setFill(Color.rgb(0, 255, 0, foodOpacity));
gc.fillOval(point.getX(), point.getY(), 10, 10);
}
// Draw poison items
for (Point2D point : pond.getPoisonMap().keySet()) {
Poison poison = pond.getPoisonMap().get(point);
double poisonOpacity = Math.min(1.0, Math.max(0.0, poison.getDamage() / 100.0));
gc.setFill(Color.rgb(255, 0, 0, poisonOpacity));
gc.fillOval(point.getX(), point.getY(), 10, 10);
}
// Draw automata
// for (Point2D point : pond.getAutomataMap().keySet()) {
// Automata automata = pond.getAutomataMap().get(point);
// double automataOpacity = Math.min(1.0, Math.max(0.0, automata.getHealth() / 100.0));
// gc.setFill(Color.rgb(0, 0, 255, automataOpacity));
Console
<terminated> Main (1) [Java Application] /Applications/Eclipse.app/Contents/Eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.aarch64_17.0.7.v20230425-1502/jre/bin/java (Oct 25, 2023,

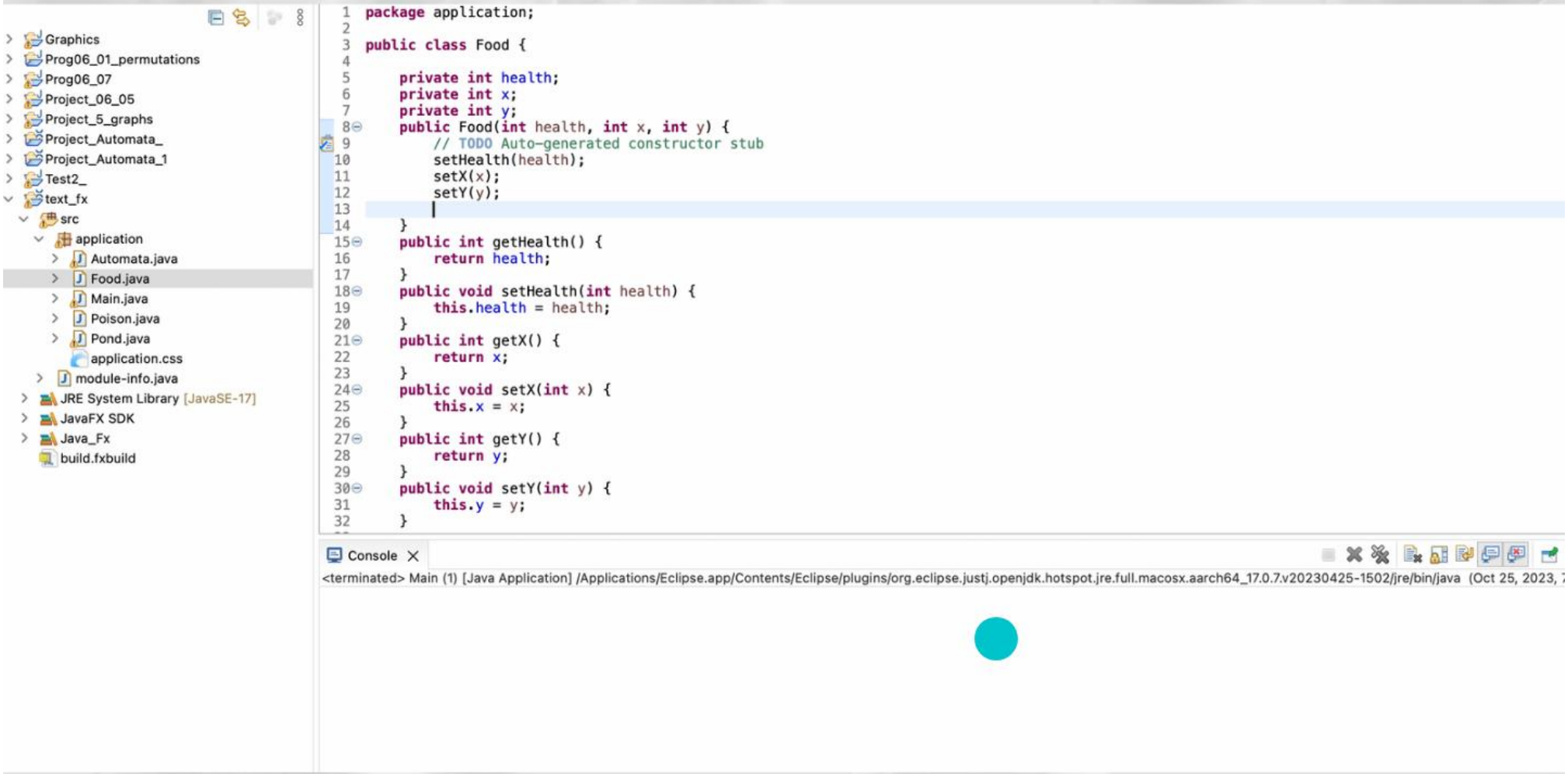
Pond


Graphics
Prog06_01_permutations
Prog06_07
Project_06_05
Project_5_graphs
Project_Automata_
Project_Automata_1
Test2_
text_fx
src
application
Automata.java
Food.java
Main.java
Poison.java
Pond.java
application.css
module-info.java
JRE System Library [JavaSE-17]
JavaFX SDK
Java_Fx
build.fxbuild
80
81 // Method to add random Automata
82 public void addRandomAutomata(int numOfAutomata) {
83 Random rand = new Random();
84 for (int i = 0; i < numOfAutomata; i++) {
85 int x = rand.nextInt(width);
86 int y = rand.nextInt(height);
87 Point2D point = new Point2D(x, y);
88 if (isWithinPond(x, y) && !automataMap.containsKey(point) && !foodMap.containsKey(point) && !poisonMap.containsKey(point)) {
89 Automata automata = new Automata(x, y, 50,5, 2,Color.rgb(0, 0, 255)); // visionLength=50, size=5, speed=2
90 automataMap.put(point, automata);
91 }
92 }
93 }
94 public void moveAndInteractAutomata() {
95 ArrayList<Point2D> pointsToRemove = new ArrayList<>();
96 ArrayList<Point2D> pointsToAdd = new ArrayList<>();
97 HashMap<Point2D, Automata> newAutomataPoints = new HashMap<>();
98
99 for (Point2D point : new ArrayList<>(automataMap.keySet())) {
100 Automata automata = automataMap.get(point);
101 automata.move();
102
103 // Remove this point from the map as the automata has moved
104 pointsToRemove.add(point);
105
106 // Check for food within the pond
107 for (Point2D foodPoint : new ArrayList<>(foodMap.keySet())) {
108 if (isTouching(point, foodPoint, automata.getSize())) {
109 automata.interactWithFood(foodMap.get(foodPoint));
110 foodMap.remove(foodPoint); // Remove the food that was consumed
111 break; // Exit the loop as the automata consumed one food item
Console
<terminated> Main (1) [Java Application] /Applications/Eclipse.app/Contents/Eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.aarch64_17.0.7.v20230425-1502/jre/bin/java (Oct 25, 2023,

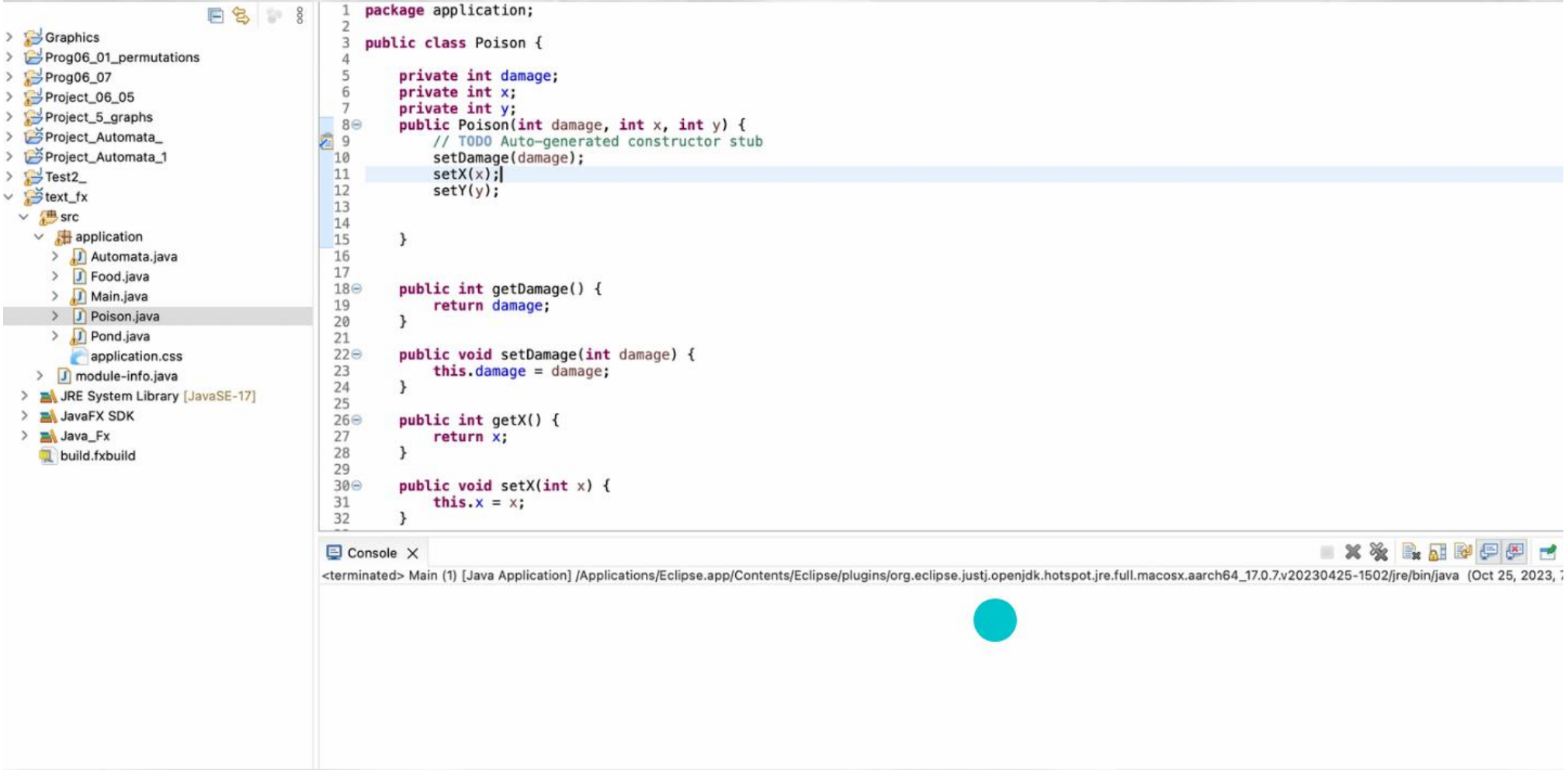
Automata


Graphics
Prog06_01_permutations
Prog06_07
Project_06_05
Project_5_graphs
Project_Automata_
Project_Automata_1
Test2_
text_fx
src
application
Automata.java
Food.java
Main.java
Poison.java
Pond.java
application.css
module-info.java
JRE System Library [JavaSE-17]
JavaFX SDK
Java_Fx
build.fxbuild
public class Automata {
private Color color;
private int health = 100; // Health capped at 100
private int x;
private int y;
private int visionLength;
private int size;
private int speed;
private Random rand = new Random();
public Automata(int x, int y, int visionLength, int size, int speed, Color color) {
this.x = x;
this.y = y;
this.visionLength = visionLength;
this.size = size;
this.speed = speed;
this.color = color;
}
// Getters and Setters
// ...
public void interactWithFood(Food food) {
if (this.health < 100) {
this.setHealth(this.health + food.getHealth());
}
}
public void interactWithPoison(Poison poison) {
this.setHealth(this.health - poison.getDamage());
Console
<terminated> Main (1) [Java Application] /Applications/Eclipse.app/Contents/Eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.aarch64_17.0.7.v20230425-1502/jre/bin/java (Oct 25, 2023,

Food


Graphics
Prog06_01_permutations
Prog06_07
Project_06_05
Project_5_graphs
Project_Automata_
Project_Automata_1
Test2_
text_fx
src
application
Automata.java
Food.java
Main.java
Poison.java
Pond.java
application.css
module-info.java
JRE System Library [JavaSE-17]
JavaFX SDK
Java_Fx
build.fxbuild
1 package application;
2
3 public class Food {
4
5 private int health;
6 private int x;
7 private int y;
8 public Food(int health, int x, int y) {
9 // TODO Auto-generated constructor stub
10 setHealth(health);
11 setX(x);
12 setY(y);
13
14 }
15 public int getHealth() {
16 return health;
17 }
18 public void setHealth(int health) {
19 this.health = health;
20 }
21 public int getX() {
22 return x;
23 }
24 public void setX(int x) {
25 this.x = x;
26 }
27 public int getY() {
28 return y;
29 }
30 public void setY(int y) {
31 this.y = y;
32 }
Console
<terminated> Main (1) [Java Application] /Applications/Eclipse.app/Contents/Eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.aarch64_17.0.7.v20230425-1502/jre/bin/java (Oct 25, 2023,

Poison


Graphics
Prog06_01_permutations
Prog06_07
Project_06_05
Project_5_graphs
Project_Automata_
Project_Automata_1
Test2_
text_fx
src
application
Automata.java
Food.java
Main.java
Poison.java
Pond.java
application.css
module-info.java
JRE System Library [JavaSE-17]
JavaFX SDK
Java_Fx
build.fxbuild
1 package application;
2
3 public class Poison {
4
5 private int damage;
6 private int x;
7 private int y;
8 public Poison(int damage, int x, int y) {
9 // TODO Auto-generated constructor stub
10 setDamage(damage);
11 setX(x);
12 setY(y);
13
14
15 }
16
17
18 public int getDamage() {
19 return damage;
20 }
21
22 public void setDamage(int damage) {
23 this.damage = damage;
24 }
25
26 public int getX() {
27 return x;
28 }
29
30 public void setX(int x) {
31 this.x = x;
32 }
Console
<terminated> Main (1) [Java Application] /Applications/Eclipse.app/Contents/Eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.aarch64_17.0.7.v20230425-1502/jre/bin/java (Oct 25, 2023,


Thank
You