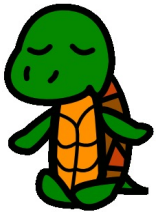
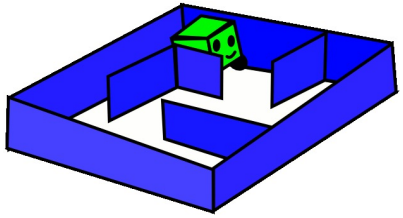
Projects





=

Due dates:
Milestone on 11:59 PM on Friday, December 3
Final project is due by 5 PM on Friday, December 10

Milestone – 60 pts
Final – 90 pts

Grading:
20% design
60% functionality
20% "documentation"

Pairs ok.

Project options:
- Keeping the Strains Straight
- Finding the Best Regulatory Network
- The Evolution of Picobot

What's coming next…

- After break
  - –11/30, 12/2 and 12/7 lecture in BECKMAN B126 (big Beckman)
  - –Class material: The limits of computation!
  - –12/9 we're back "home" in Shan 2460 for a final lecture
  - –Work on your project (milestone + final project)
  - –Labs are just for working and getting help on projects (will be at normal time and place with the three of us)

Project Choice 1

Keeping the strains straight

Strains were sequenced by the Broad center

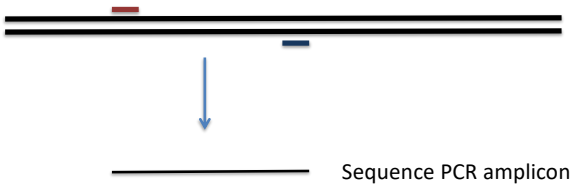
etc.

Question: is this the strain we think it is?

We could sequence the entire genome to check.

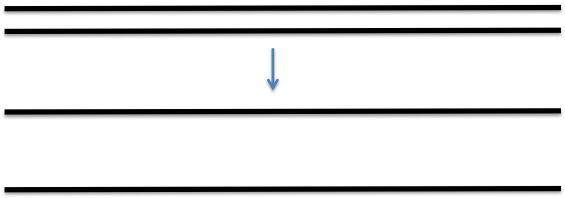
Disadvantage: expense

Alternative: use PCR to amplify diagnostic regions of a strain's genome



Project goal: design PCR primers to distinguish the strains

PCR in a nutshell



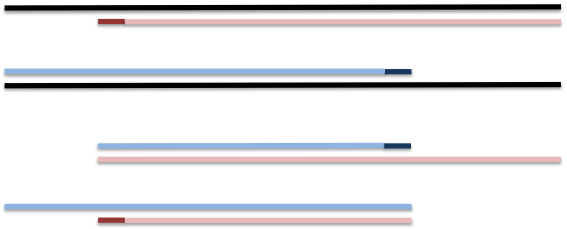
Cycle 1

Primers come in pairs

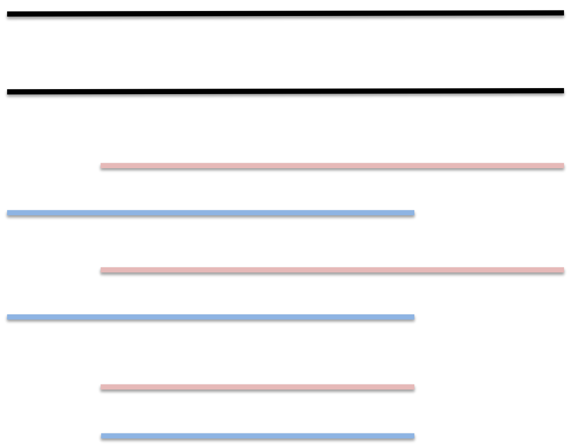
Cycle 1

Cycle 2

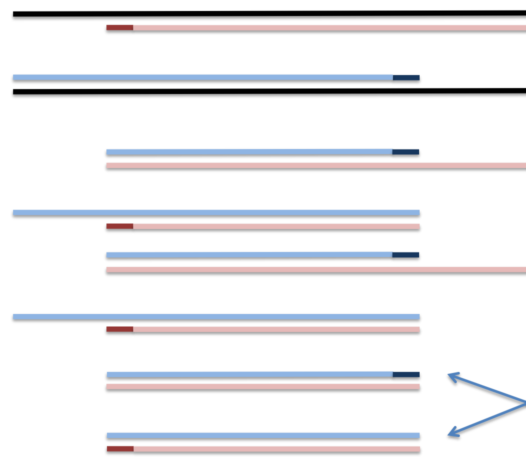
Cycle 2



Cycle 3



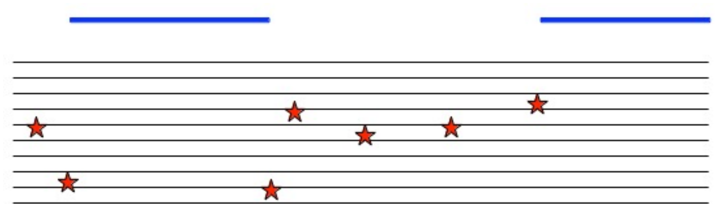
Cycle 3



The data: aligned coding sequences

```
>>> from ecoli10 import *
>>> seqs10L[4]
(
ATGACACAATTCGCTTCTCCTGTTCTGCACTCGTTGCTGGATACAGATGC...,
ATGACACAATTCGCTTCTCCTGTTCTGCACTCGTTGCTGGATACAGATGC...,
ATGACACAATTCGCTTCTCCTGTTCTGCACTCGTTGCTGGATACAGATGC...,
ATGACACAATTCGCTTCTCCTGTTCTGCACTCGTTGCTGGATACAGATGC...,
ATGACACAATTCGCTTCTCCTGTTCTGCACTCGTTGCTGGATACAGATGC...,
ATGACACAATTCGCTTCTCCTGTTCTGCACTCGTTGCTGGATACAGATGC...,
ATGACACAATTCGCTTCTCCTGTTCTGCACTCGTTGCTGGATACAGATGC...,
ATGACACAATTCGCTTCTCCTGTTCTGCACTCGTTGCTGGATACAGATGC...,
ATGACACAATTCGCTTCTCCTGTTCTGCACTCGTTGCTGGATACAGATGC...,
ATGACACAATTCGCTTCTCCTGTTCTGCACTCGTTGCTGGATACAGATGC...,
)
>>> seqs10L[5]
(
TTGCAACCAGCGTTGTCAGGGAACTTATCAACACAACAGGTGATTATGCG...,
---------------------------------------------ATGCG...,
---------------------------------------------ATGCG...,
---------------------------------------GTGATTATGCG...,
---------------------------------------GTGATTATGCG...,
---------------------------------------------ATGCG...,
---------------------------------------------ATGCG...,
---------------------------------------------ATGCG...,
---------------------------------------------ATGCG...,
---------------------------------------------ATGCG...,
)
```

Where to put primers



Primers themselves should bind in regions where all strains identical.

Amplicons: picking pairs of primers

- Primer pairs should make amplicons (200-500 bp long)
- What makes one amplicon better than another?

CAG
TGG
ACG
CTG

An example amplicon from 4 strains.
Minimum pairwise difference: 1

A bigger minimum pairwise difference is better.

Steps…

- Find places primers could bind
- Find pairs of primers which make good amplicons
- Two data sets:
  - 10 strain
  - 88 strain

Computational ideas used here…

- Breaking a larger problem into smaller easy-to-solve parts
- Optimization in a large space of possibilities
- Opportunity to develop an algorithm

Biological ideas used here…

- Problem solving with genomic data
- Opportunity to solve a real biological problem

Project Choice 2
(Chapter 13 in our book!)

Gene Regulatory Networks and the Maximum Likelihood Method

This project was adapted from materials generously provided by Professor Russell Schwartz, Department of Biological Sciences and Lane Center for Computational Biology, Carnegie Mellon University

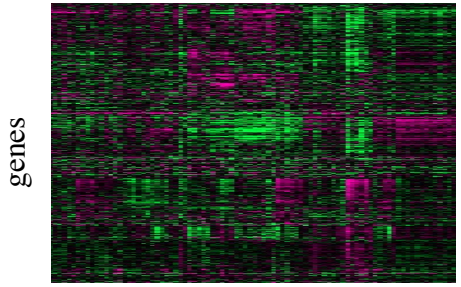
Prof. Russell Schwartz

Genes…

- We know how to find genes!
- Some genes produce proteins that in turn promote or inhibit the production of other genes!
- Those interactions generally depend on the conditions in the cell, e.g., the concentrations of other substances.
  - Yeast activates genes that convert sugar to alcohol, depending on concentration of sugar.
  - Therapies for fighting disease by altering regulation of certain genes.

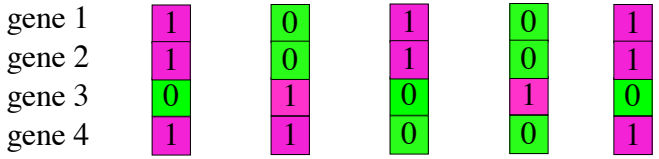
Courtesy of Prof. Russell Schwartz

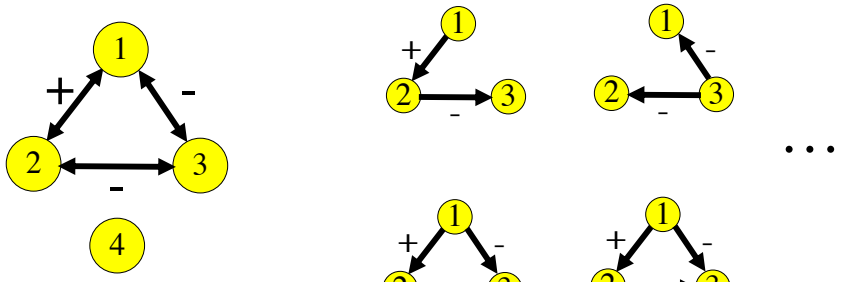
- Some genes encode transcription factors that promote or inhibit the expression of other genes
- Purple is highly expressed, green is not expressed



Courtesy of Prof. Russell Schwartz

Intuition Behind Network Inference





correlated expression implies common regulation

that intuition still leaves a lot of ambiguity

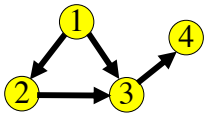
Courtesy of Prof. Russell Schwartz

Assuming a Binary Input Matrix

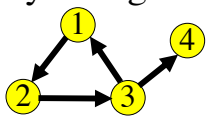
- We will assume that genes only have two possible states: 0 (off) or 1 (on)

	conditions							
gene 1	1	1	0	0	1	1	1	0
gene 2	0	1	0	1	1	1	1	0
gene 3	0	0	1	0	0	0	0	1
gene 4	0	0	0	0	0	1	0	1

- We will also assume that we want to find directionality but not strength of regulatory interactions
- We will exclude the possibility of regulatory cycles:



OK



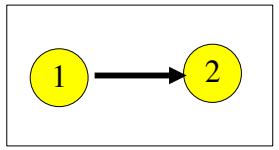
NOT OK

Courtesy of Prof. Russell Schwartz

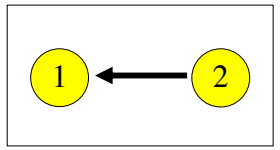
A Simple Case: Two Genes

	conditions							
gene 1	1	1	0	0	1	1	1	0
gene 2	0	1	0	1	1	1	1	0

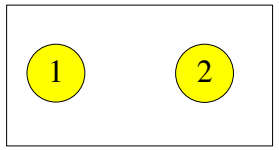
- Only three possible models to consider



model 1
"G1 regulates G2"



model 2
"G2 regulates G1"



model 3
"G1 and G2 are independent"

Courtesy of Prof. Russell Schwartz

What is the Probability of a Microarray?

- We can describe the probability of a microarray as the product of the probabilities of all of its individual measurements:

Pr{ 1 1 0 0 1 1 1 0 }

Pr{ 1 }x Pr{ 1 }x Pr{ 0 }x Pr{ 0 }x Pr{ 1 }x Pr{ 1 }x Pr{ 1 }x Pr{ 0 }

Courtesy of Prof. Russell Schwartz

What is the Probability of One Measurement on a Microarray?

- We can estimate Pr{ 1 } and Pr{ 0 } by counting how often each individual value occurs
  - Pr{ 1 } = 5/8
  - Pr{ 0 } = 3/8
- Therefore:

Pr{ 1 1 0 0 1 1 1 0 }
=Pr{ 1 }x Pr{ 1 }x Pr{ 0 }x Pr{ 0 }x Pr{ 1 }x Pr{ 1 }x Pr{ 1 }x Pr{ 0 }
=5/8 x 5/8 x 3/8 x 3/8 x 5/8 x 5/8 x 5/8 x 3/8
= 0.00503

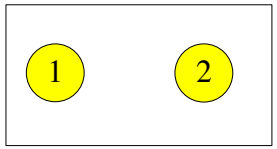
Courtesy of Prof. Russell Schwartz

Evaluating One Model

data D =

gene 1	1	1	0	0	1	1	1	0
gene 2	0	1	0	1	1	1	1	0

model M =



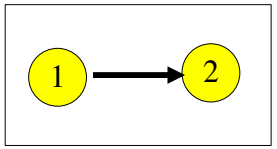
Pr{D|M} = Pr{ 1 1 0 0 1 1 1 0 } x
Pr{ 0 1 0 1 1 1 1 0 }

= 0.00503 x 0.00503 = 2.5×10^{-5}

Courtesy of Prof. Russell Schwartz

Adding in Regulation

- How do we evaluate output probabilities for a regulated gene?



gene 1	1	1	0	0	1	1	1	0
gene 2	0	1	0	1	1	1	1	0

- We need the notion of *conditional probability*: evaluating the probability of gene 2's output given that we know gene one's output:

Pr{G2= 0 |G1= 1 } = 1/5 Pr{G2= 0 |G1= 0 } = 2/3

Pr{G2= 1 |G1= 1 } = 4/5 Pr{G2= 1 |G1= 0 } = 1/3

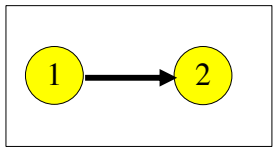
Pr{G2= 0 |G1= 1 } = 1/5 Pr{G2= 0 |G1= 0 } = 2/3

Pr{G2= 1 |G1= 1 } = 4/5 Pr{G2= 1 |G1= 0 } = 1/3

data D =

gene 1	1	1	0	0	1	1	1	0
gene 2	0	1	0	1	1	1	1	0

model M =



gene1

Pr{D|M} = Pr{ 1 1 0 0 1 1 1 0 } x
Pr{ 0 1 0 1 1 1 1 0 | 1 1 0 0 1 1 1 0 }

gene2 gene1

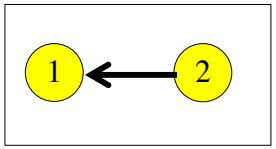
= 0.00503 x
(1/5 x 4/5 x 2/3 x 1/3 x 4/5 x 4/5 x 4/5 x 2/3)
= 6.1×10^{-5}

Evaluating Another Model

data D =

gene 1	1	1	0	0	1	1	1	0
gene 2	0	1	0	1	1	1	1	0

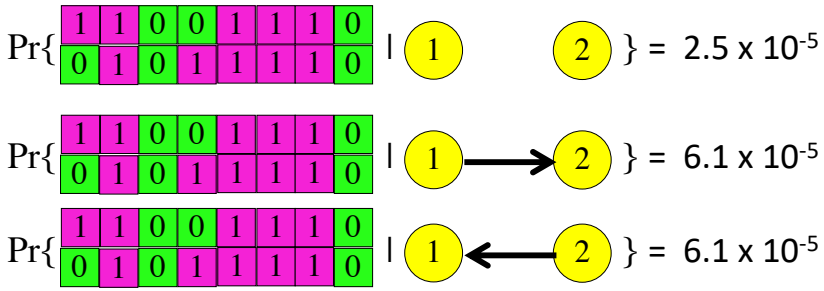
model M =



Pr{D|M} = Pr{ 1 1 0 0 1 1 1 0 | 0 1 0 1 1 1 1 0 }
x Pr{ 0 1 0 1 1 1 1 0 }

= (1/3 x 4/5 x 2/3 x 1/5 x 4/5 x 4/5 x 4/5 x 2/3) x
0.00503
= 6.1×10^{-5}

Comparing the Models for Two Genes



Conclusions:

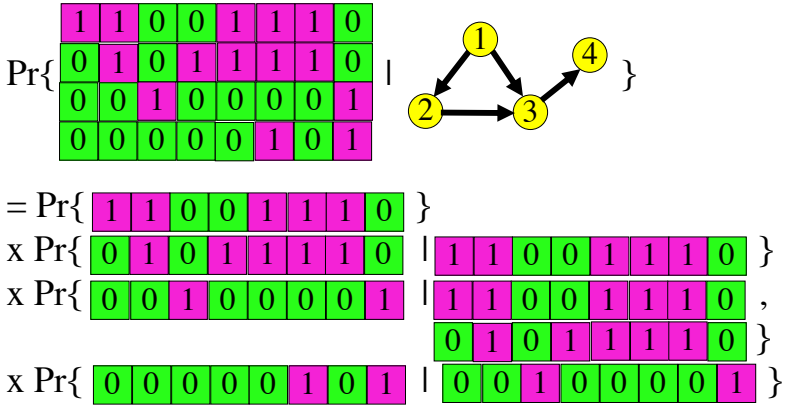
-Knowing the expression of gene 1 helps us predict the expression of gene 2 and vice versa

-We can suggest there should be an edge between them but cannot decide the direction it should take

Courtesy of Prof. Russell Schwartz

Generalizing to Many Genes

- The same basic concepts let us evaluate the plausibility of any regulatory model



The Project

- Take binary expression data as input
- Find the regulatory network with the highest likelihood
- Display the network somehow

Computational ideas used here…

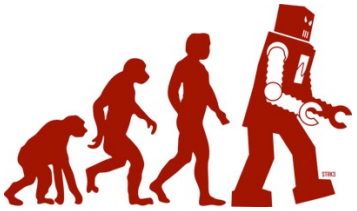
- Representing networks computationally
- Visualizing the networks
- Breaking a larger problem into smaller easy-to-solve parts
- Maximum likelihood method

Biological ideas used here…

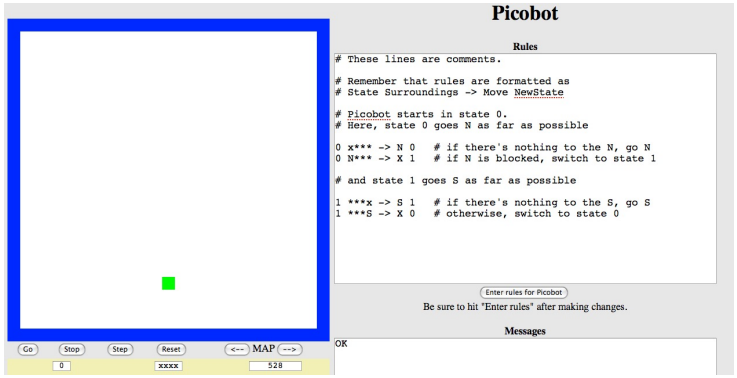
- The concept of gene regulatory networks
- Problem solving with gene expression data

Project Choice 3

Evolution of Picobot



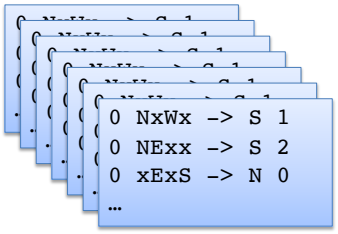
Remember Picobot?




```
NEWS
0 NxWx -> E 0
0 NExx -> S 1
```

NEWS format!

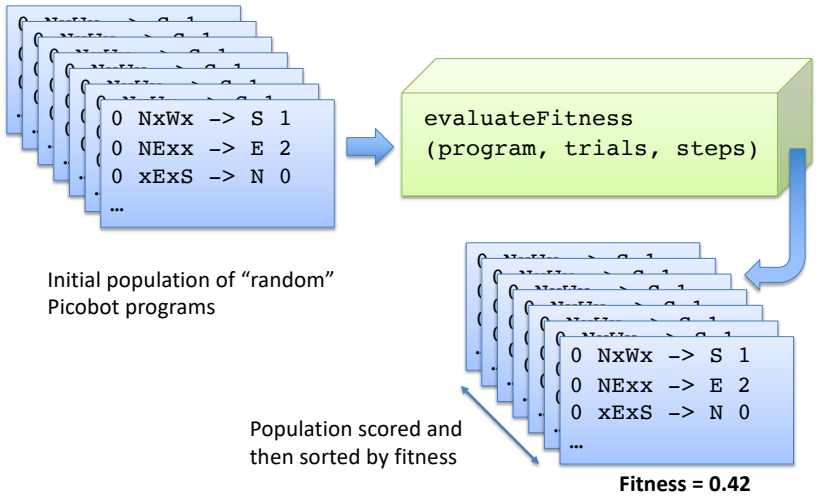
Evolving Programs through Simulated Evolution!



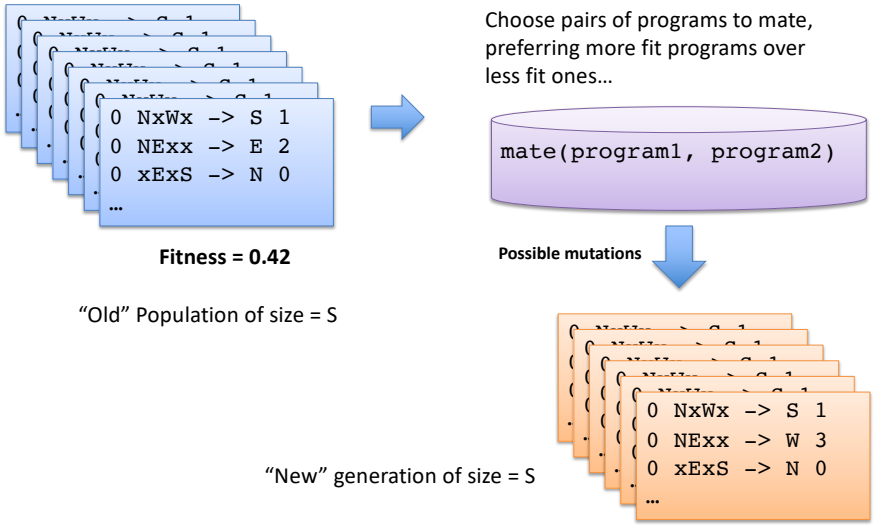
Initial population of "random" Picobot programs

How do we measure the fitness of a program?

Evolving Programs through Simulated Evolution!



Evolving Programs through Simulated Evolution!



Programs, Parents, Offspring

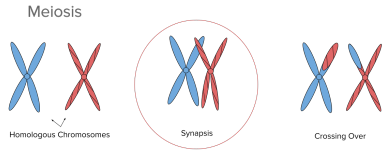
```
0 xxxx -> N 2          0 xxxx -> S 3
0 Nxxx -> S 4          0 Nxxx -> N 2
0 NExx -> W 0          0 NExx -> S 2
…                      …
0 xExx -> S 3          0 xExx -> W 4
0 xxWx -> E 1          0 xxWx -> E 4
1 xxxx -> S 4          1 xxxx -> N 2
…                      …
1 xxWx -> N 0          1 xxWx -> E 4
…                      …
4 xxxx -> S 3          4 xxxx -> N 2
…                      …
4 xxWx -> E 1          4 xxWx -> S 4
```

Parent Program 1 Parent Program 2

```
xxxx
Nxxx
NExx
NxWx
xxxS
xExS
xxWS
xExx
xxWx
```

In a rectangular room, these are the only patterns we need to consider.!

Crossover




```
0 xxxx -> N 2          0 xxxx -> S 3
0 Nxxx -> S 4          0 Nxxx -> N 2
0 NExx -> W 0          0 NExx -> S 2
…                      …
0 xExx -> S 3          0 xExx -> W 4
0 xxWx -> E 1          0 xxWx -> E 4
1 xxxx -> S 4          1 xxxx -> N 2
…                      …
1 xxWx -> N 0          1 xxWx -> E 4
…                      …
4 xxxx -> S 3          4 xxxx -> N 2
…                      …
4 xxWx -> E 1          4 xxWx -> S 4
```

Random crossover point (between two different states)

Parent Program 1 Parent Program 2

Offspring!

```
0 xxxx -> N 2
0 Nxxx -> S 4
0 NExx -> W 0
…
0 xExx -> S 3
0 xxWx -> E 1
1 xxxx -> S 4
…
1 xxWx -> N 0
…
4 xxxx -> S 3
…
4 xxWx -> E 1
```

Parent Program 1

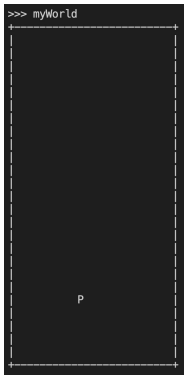
```
0 xxxx -> S 3
0 Nxxx -> N 2
0 NExx -> S 2
…
0 xExx -> W 4
0 xxWx -> E 4
1 xxxx -> N 2
…
1 xxWx -> E 4
…
4 xxxx -> N 2
…
4 xxWx -> S 4
```

Parent Program 2

```
0 xxxx -> N 2
0 Nxxx -> S 4
0 NExx -> W 0
…
0 xExx -> S 3
0 xxWx -> E 1
1 xxxx -> N 2
…
1 xxWx -> E 4
…
4 xxxx -> N 2
…
4 xxWx -> S 4
```

Offspring

Computational ideas used here…



- Object-oriented programming!

```
class Program:
  def __init__(self):
  def randomize(self):
  def getMove(self, state, pattern):
  def mutate(self):
  def crossover(self, other):
  def __repr__(self):

class World:
  def __init__(self, initial_row, initial_col, program)
       self.row = initial_row
       self.col = initial_col self.state = 0
       self.program = program
       self.room = [[' ']*WIDTH for row in range(HEIGHT)]
  …
```

Biological ideas used here…

- Demonstration of “power” of evolution
- Exploration impact of mating and mutation on fitness

Demo