CONCRETE Programming

Vlad Estivill-Castro

&

Brendan Bartlett

CONCRETE PROGRAMING FOR PROBLEM-SOLVING SKILLS

--- A project for improvement in the outcomes of first year IT students

CONCRETE Programming

Thanks for your interest

CONCRETE Programing

Ways of working

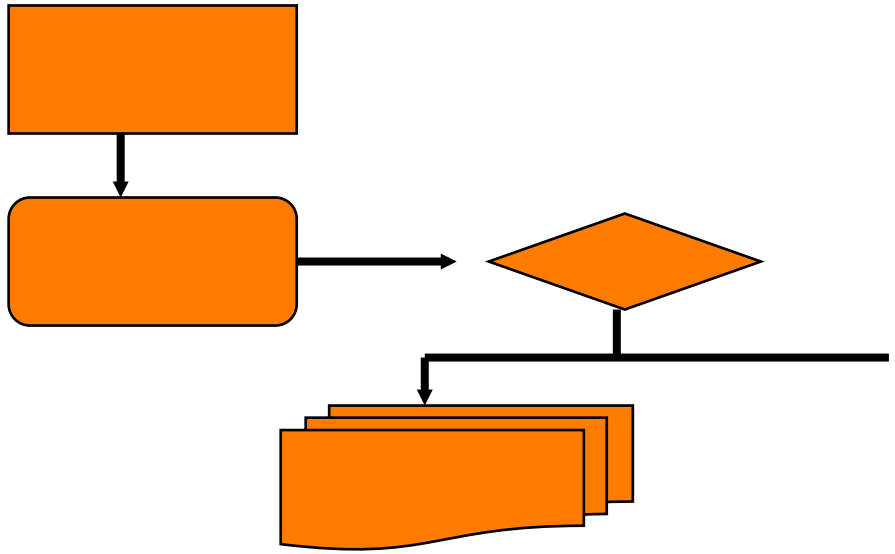
- plan activities and investigations to explore concepts
- plan strategies to solve mathematical questions, problems and issues
- evaluate thinking and reasoning,
- thinking and working and reasoning that can be applied to solve problems in real-life and abstract investigations

Problem Solving

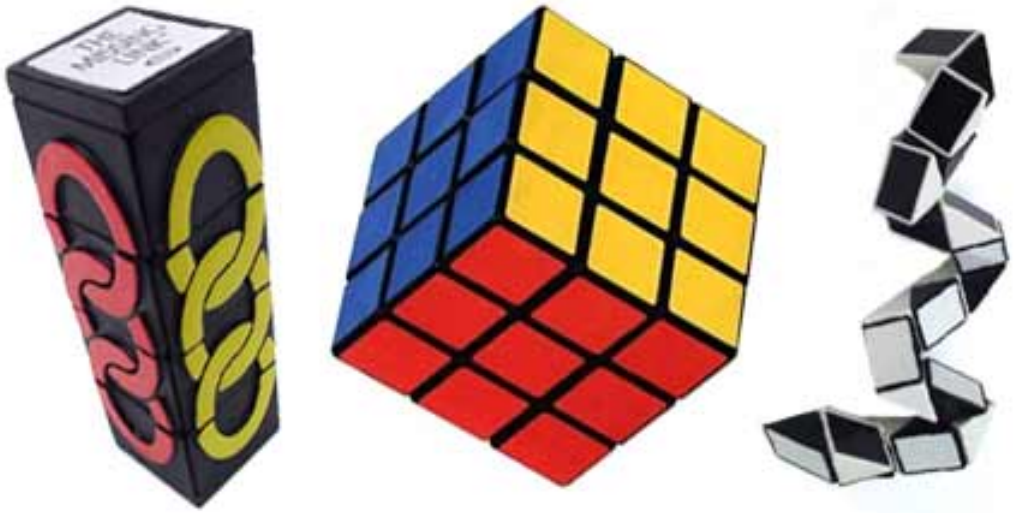
- Fundamental skill for the Software Developer
- We not only solving a problem
- We design a method (algorithm) for solving the problem
- Encoded in a language so an automata (a computer) can solve it repeatedly many times

Algorithms

- The underlying common element that brings all areas of computer science together is that for every task we want computers to do, we must describe an algorithm.



Example

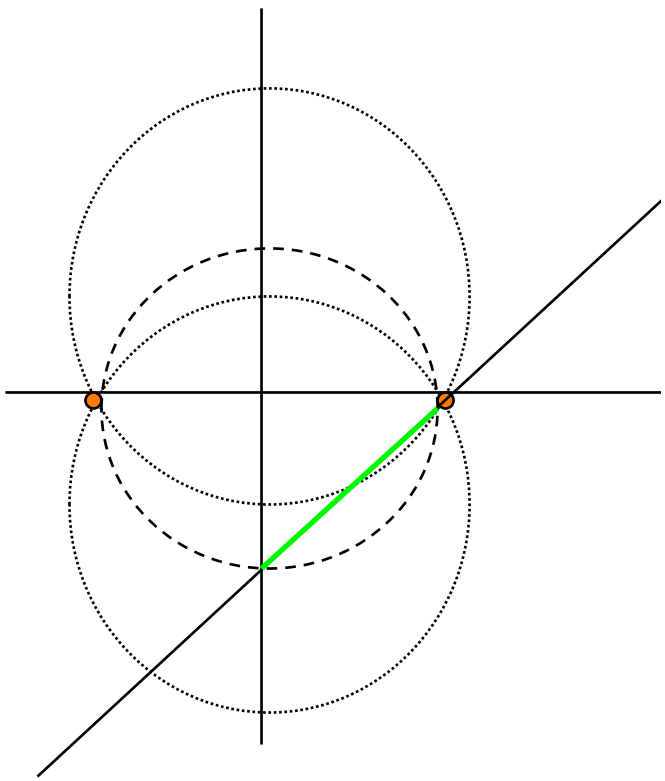


- Rubick cube
  - Charge one (1) for each rotation
    - the model of computation
- Problem:
  - find an algorithm
    - the steps to follow in order to sort it
  - prove it is best (no other can do faster)

To make a circle

To make a straight line

Constructing $\sqrt{2}$



1. Draw a line (with the ruler) *L*
2. Draw two equal circles (with the compass) that have their centre on the line and cut each other twice
3. Draw a line (with the ruler) that passes through the two points that intersect the circles
4. Draw a circle (with the compass) with centre the intersection of the two straight lines and through the intersection of the circles
5. Draw the line (with the ruler) between two points in this new circle that intersect different lines
6. The segment has length square root of 2.

CONCRETE Programing

Numbers, symbols and language are all information

- Numbers (as well as INFORMATION) are abstract concepts we talk about
- We can represent them exactly in some languages
  - In some *Models of Computation*
  - We cannot represent them exactly with other *Hardware*

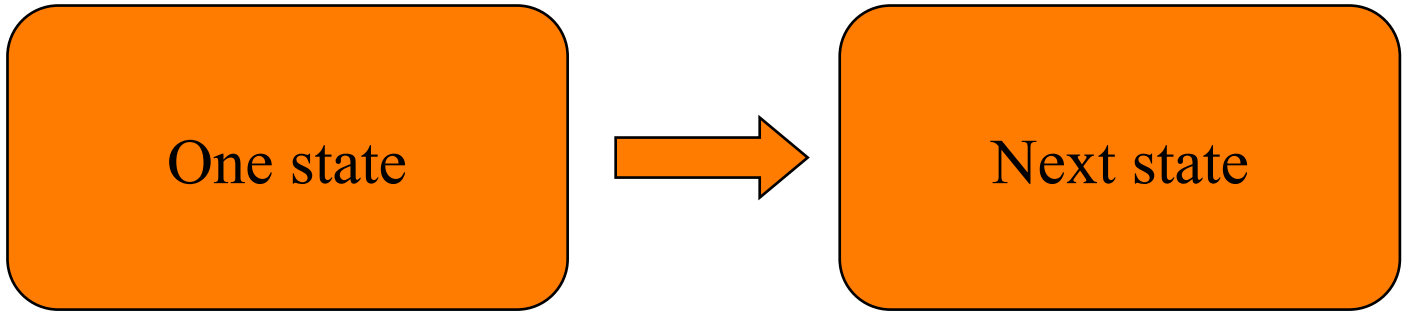
THE HARDWARE

THE SOFTWARE

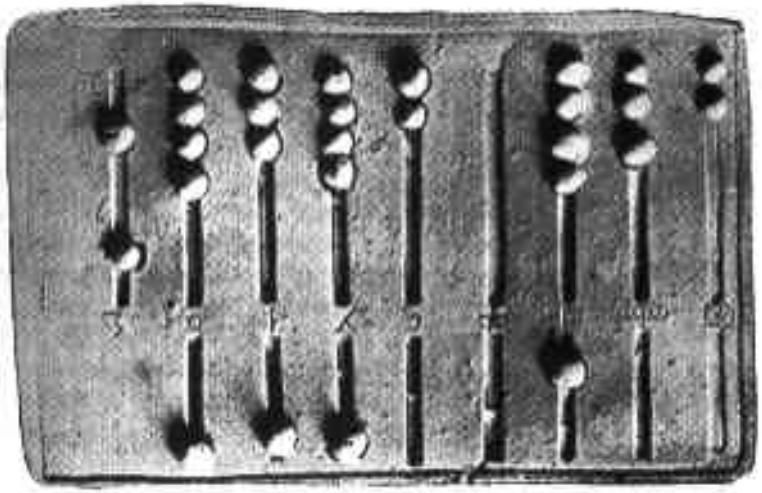
1. Draw a line (with the ruler) *L*
2. Draw two equal circles (with the compass) that have their centre on the line and cut each other twice
3. Draw a line (with the ruler) that passes through the two points that intersect the circles
4. Draw a circle (with the compass) with centre the intersection of the two straight lines and through the intersection of the circles
5. Draw the line (with the ruler) between two points in this new circle that intersect different lines
6. The segment has length square root of 2.

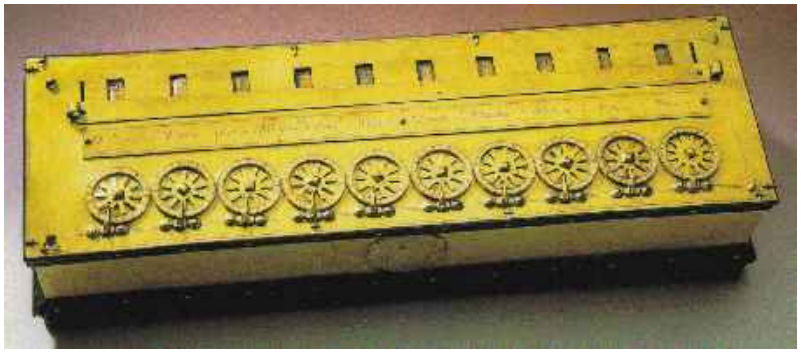
Algorithm

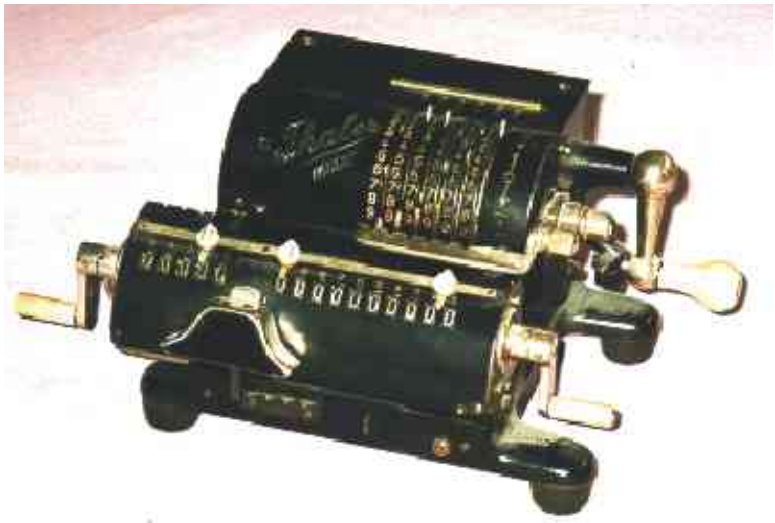
- The effective procedure that defines (without ambiguity) how to take the hardware from one state to the next state
  - until the desired final output is achieved
  - and always terminate in a finite number of steps.



Old Hardware / Old devices for computation

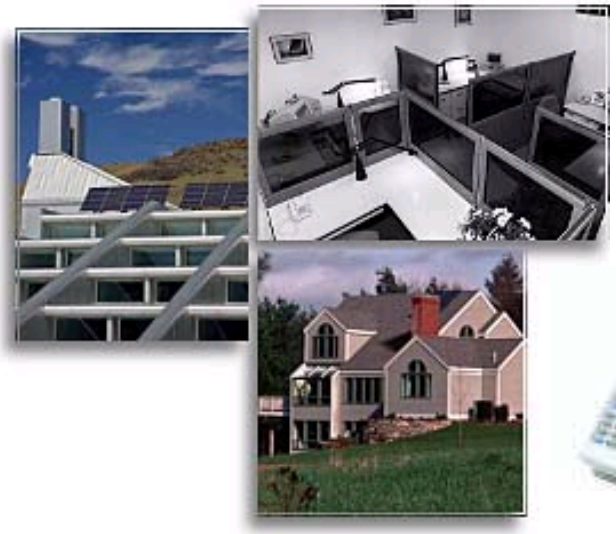






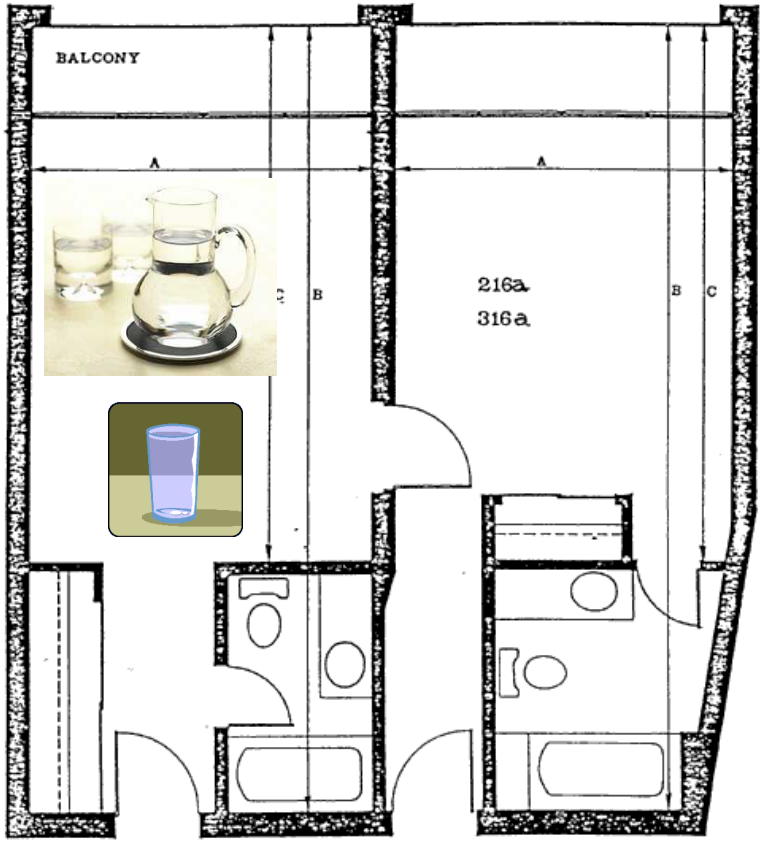
CONCRETE Programing

Now it is faster and smaller - and everywhere



The spin doctor

- When asked for something, change the question.



Basic Problem Solving

1) First, you have to understand the problem.
2) After understanding, then make a plan.
3) Carry out the plan.
4) Look back on your work. How could it be better?

First principle: Understand the problem

- What are you asked to find or show?
- Can you restate the problem in your own words?
- Can you think of a picture or a diagram that might help you understand the problem?
- Is there enough information to enable you to find a solution?
- Do you understand all the words used in stating the problem?
- Do you need to ask a question to get the answer?

CONCRETE Programing

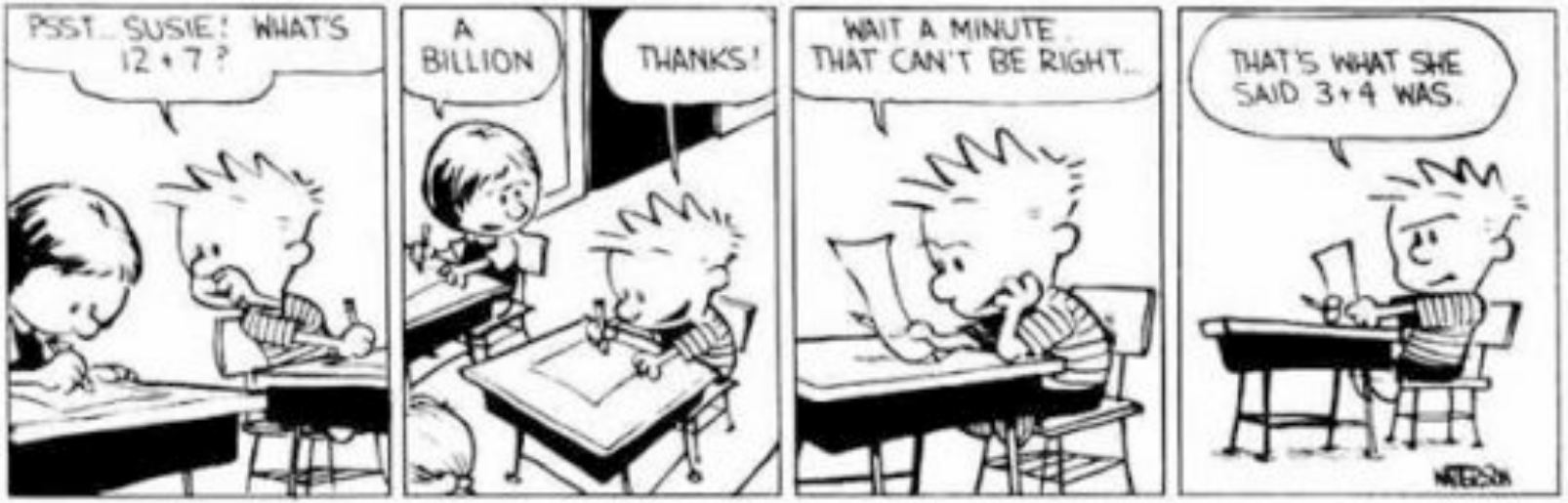
Second principle: Devise a plan

- Guess and check
- Make an orderly list
- Eliminate possibilities
- Use symmetry
- Consider special cases
- Use direct reasoning
- Solve an equation

Second principle: Devise a plan

- Look for a pattern
- Draw a picture
- Solve a simpler problem
- Use a model
- Work backward
- Use a formula
- Be creative
- Use your head

CONCRETE Programing



-Psst Susie
What's 12+7?

-A billion

-Thanks

-Wait a minute.
That can't be right…

-That's what she said 3+4 was.

CONCRETE Programing

-Pst! What's 7+6"

-Three hundred billion gazillion

-Oh! Thanks for the big help!

-That's a three followed by 85 zeros

- Ah! I knew that

Activities associated with curriculum

- Sorting Networks
  - The sorting cows activity.

SMALLER LARGER

The sorting bridges / gates --- building blocks

BLUE places larger up

GREEN places larger down

RED always swaps

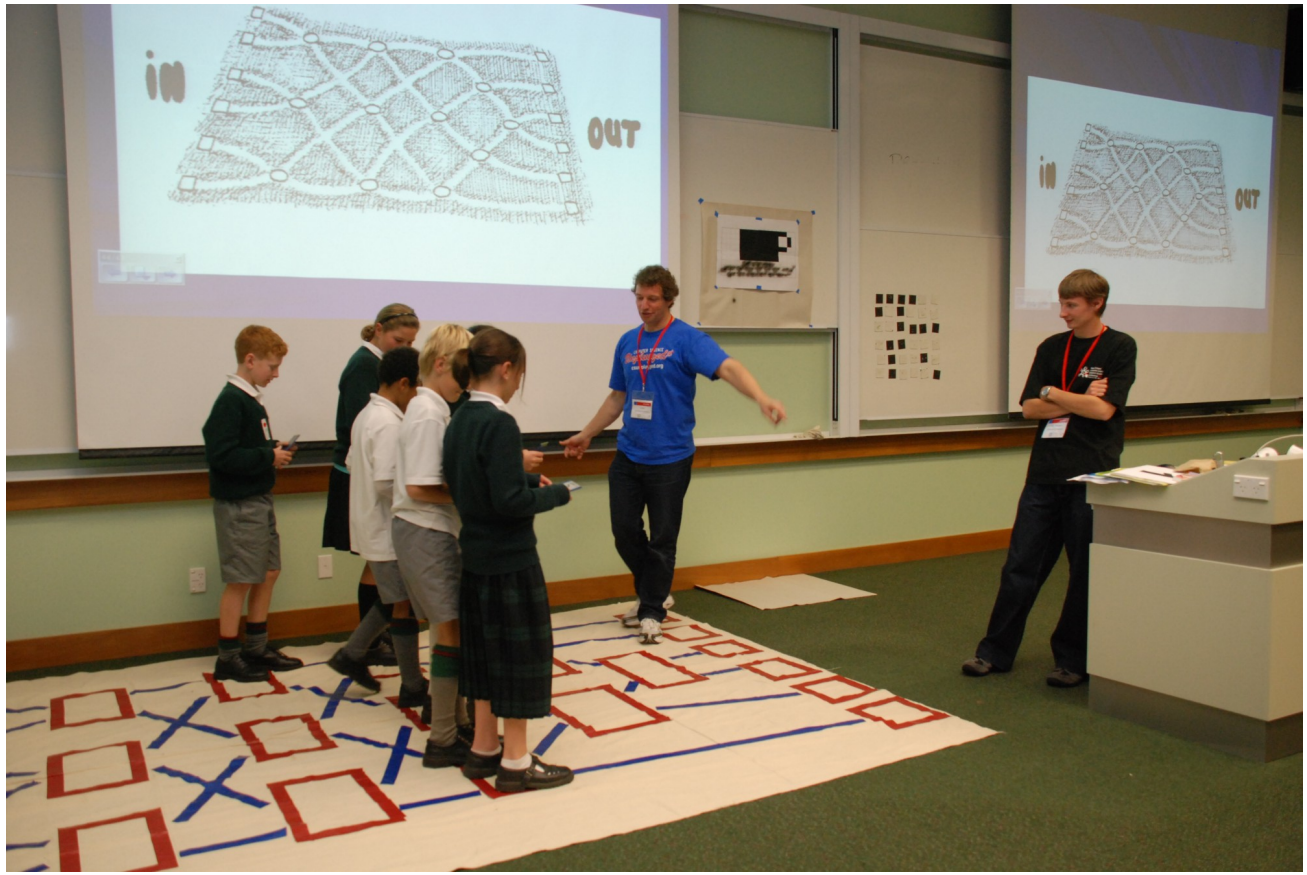
The problem (s)

- Build rail/gate/ bridge structures that sort
  - Find the largest cow
  - Find the smallest cow
  - Find the largest and the smallest
  - Sort the cows in ascending order
- Parameters
  - Size of network
  - Type of bridges
  - Cost of bridges

CONCRETE Programing

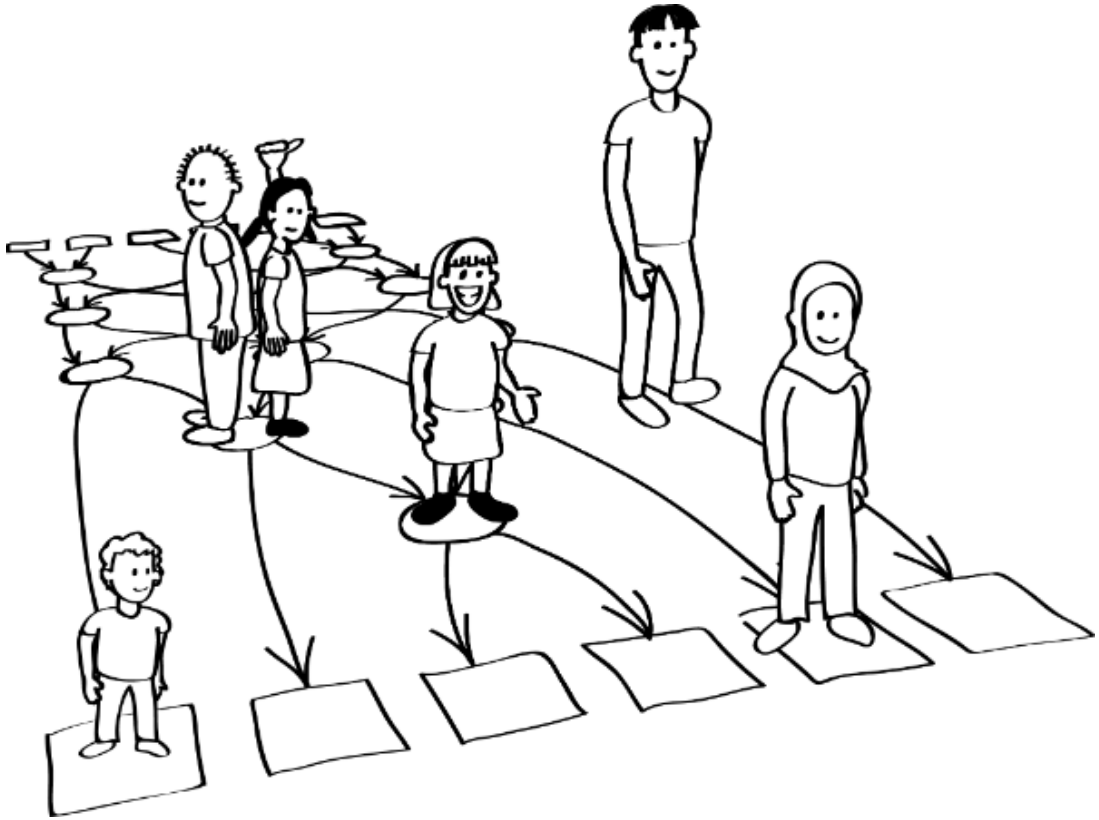
Stage 1:

- The physical experience



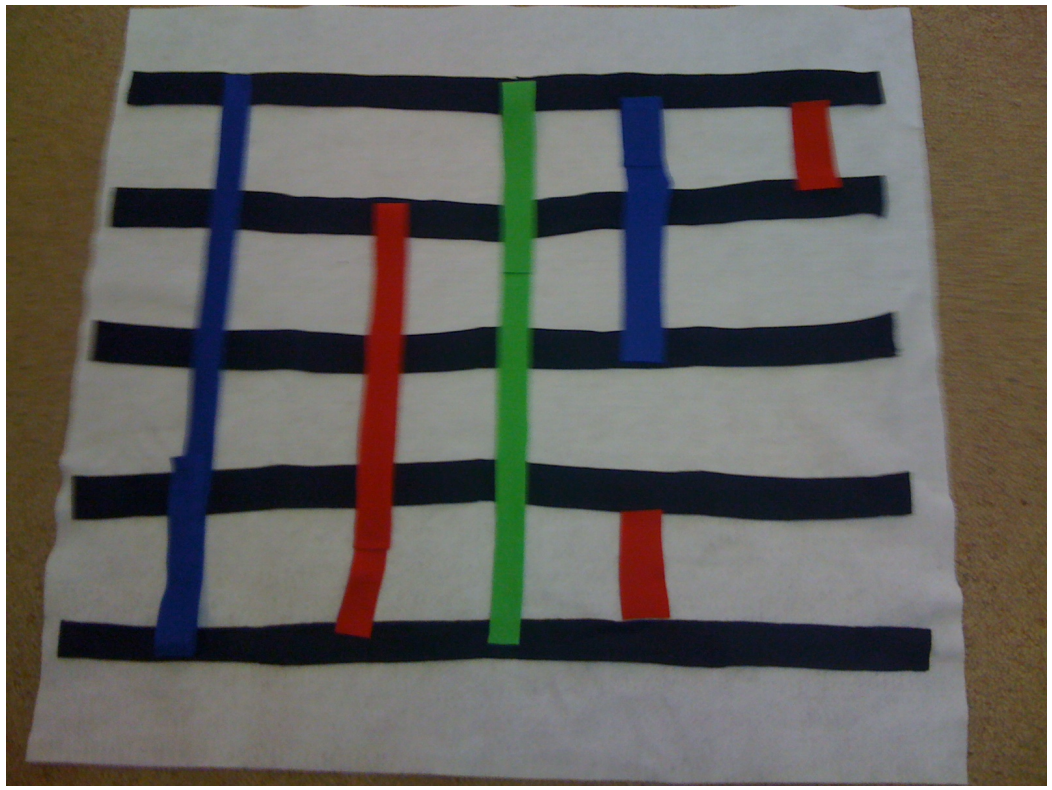
A concrete experience

- Build the network and walk on it

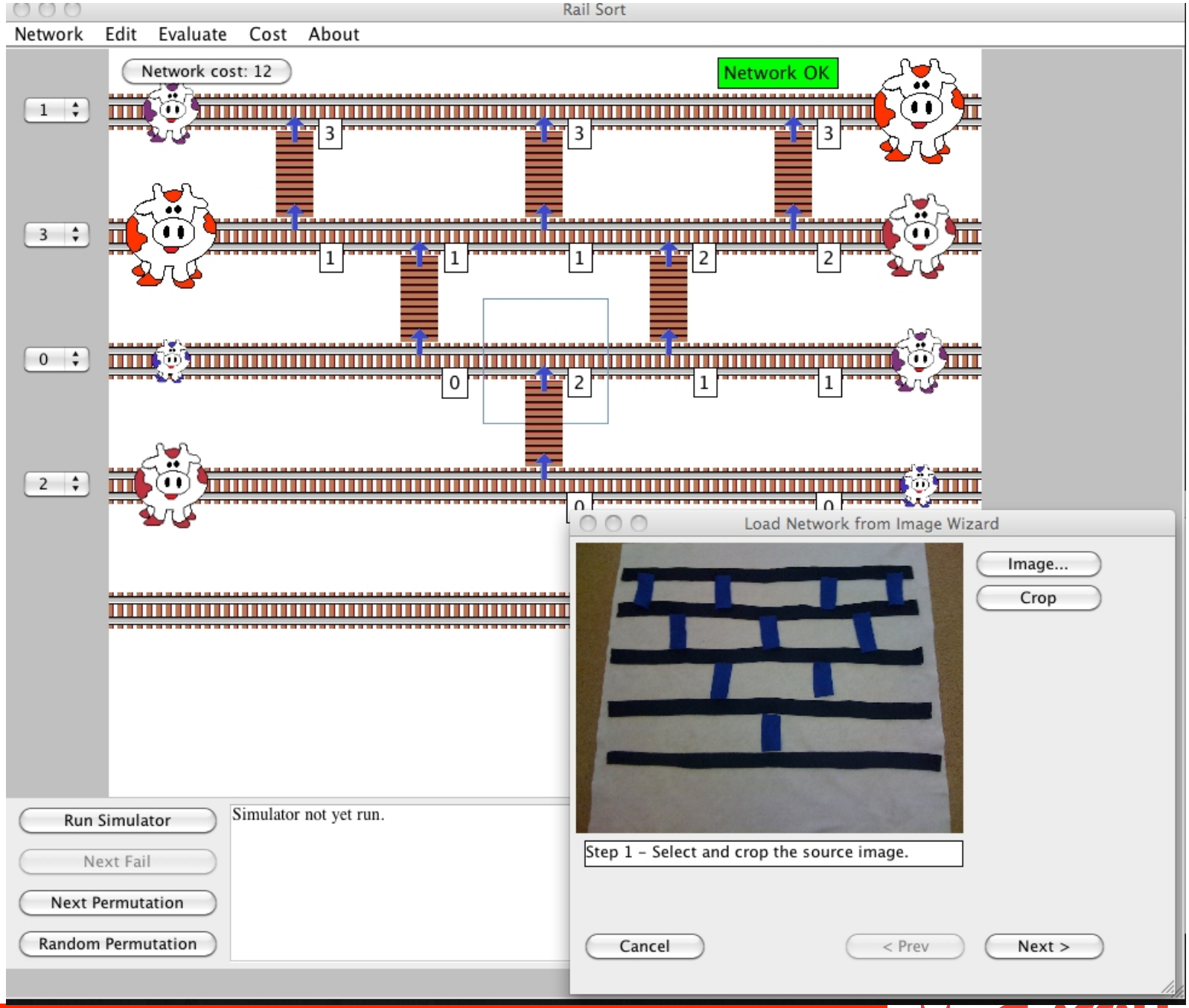


Stage 2:

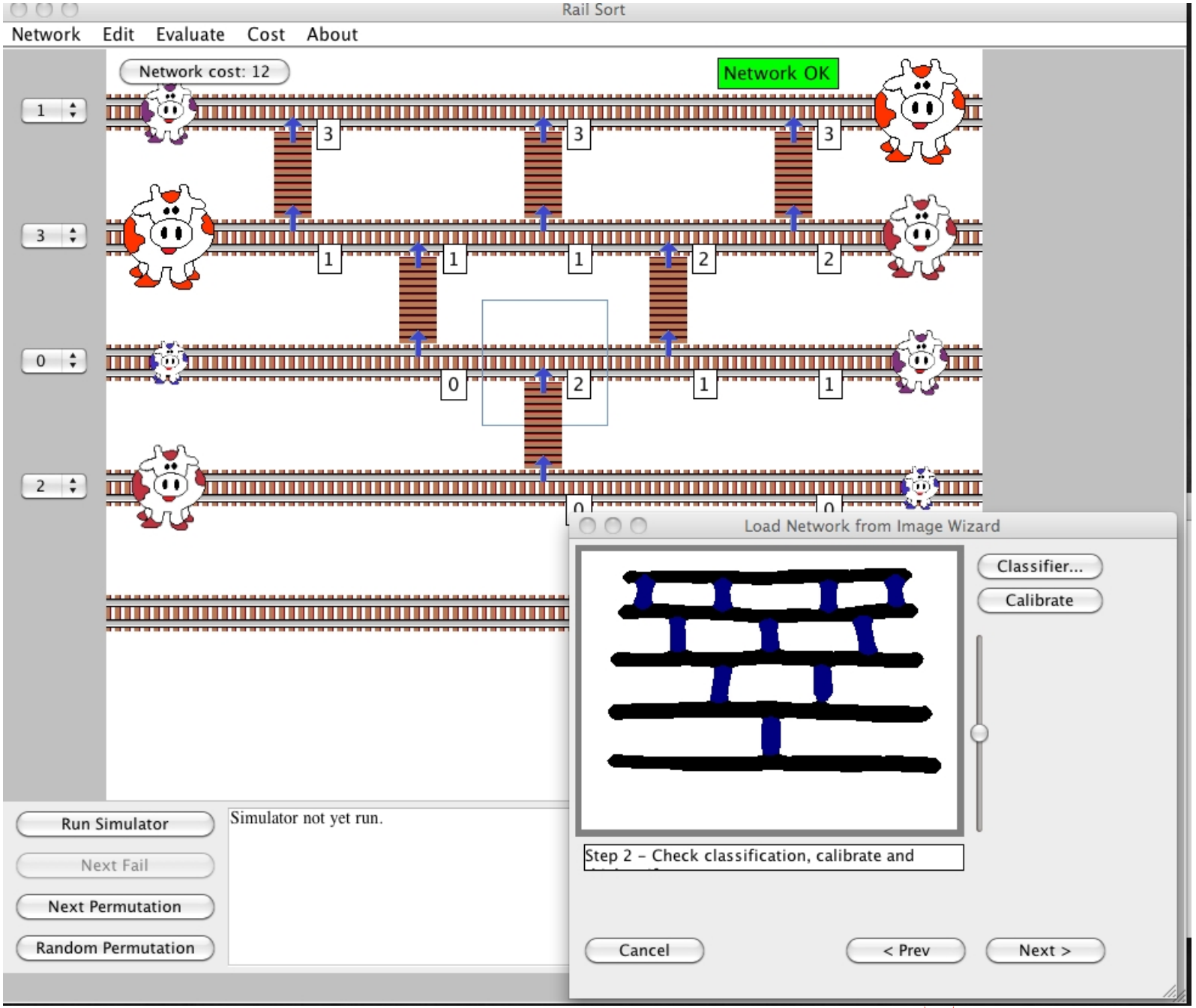
- Construct the networks for several problems and experiment, evaluate, interact



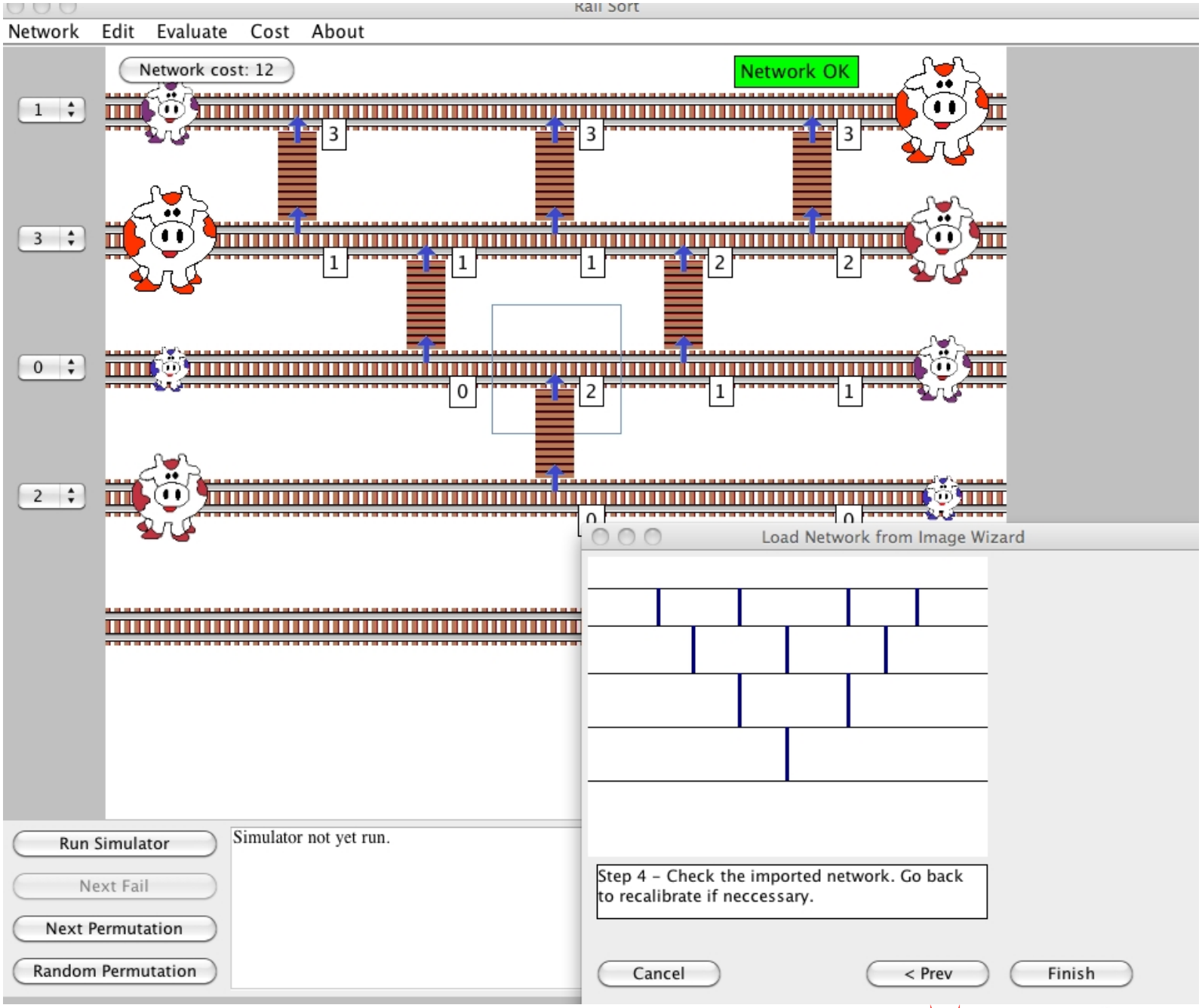
CONCRETE Programming



CONCRETE Programming



CONCRETE Programing



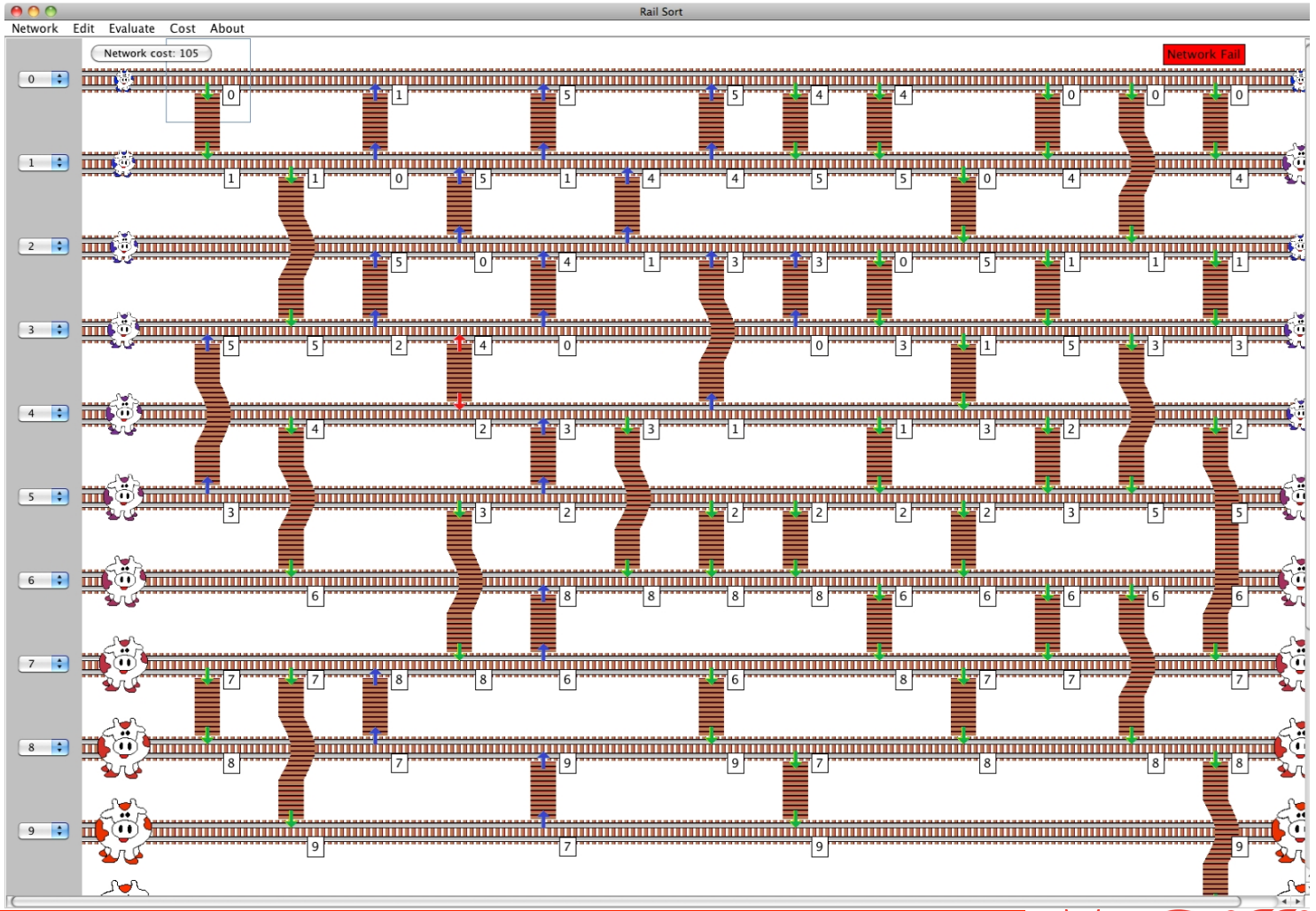
Feedback

- **Overall Network Score:** <A measure of the success of this network.>
  ---- SUCCESS ----
  # Runs: <The total number of inputs tested.>
  # Success: <The number of inputs that the network succeeded on, based on its current evaluation measure.>
  # Fails: <The number of inputs that the network failed on, based on its current evaluation measure.>

  ---- COST ----
  Cost of this network: 10
  - time: 5
  - operations: 5
  - parallel: 0
  - bridges: 0

CONCRETE Programming

Stage 3: Experiment in virtual environment



CONCRETE Programming

Introduce and illustrate concepts

- Discover patterns
- Use previous solutions as tools for new problems
  - Finding largest => Sorting
- Consider algorithmic strategies
  - Recursion
  - Divide and Conquer
- Illustrate correctness
  - Induction

CONCRETE Programming

Stage 4: Work on abstract / textual programming environment

- Use the **`MaSH`** programming tool
  - Making Stuff Happen [Andrew Rock, 2008]
- Stage 4
  - Level: statements

- Concepts:
  - identifier
  - method invocation
  - sequential control
  - comments
  - abstract execution
  - displaying output

CONCRETE Programming

Stage 4: Textual programming environment

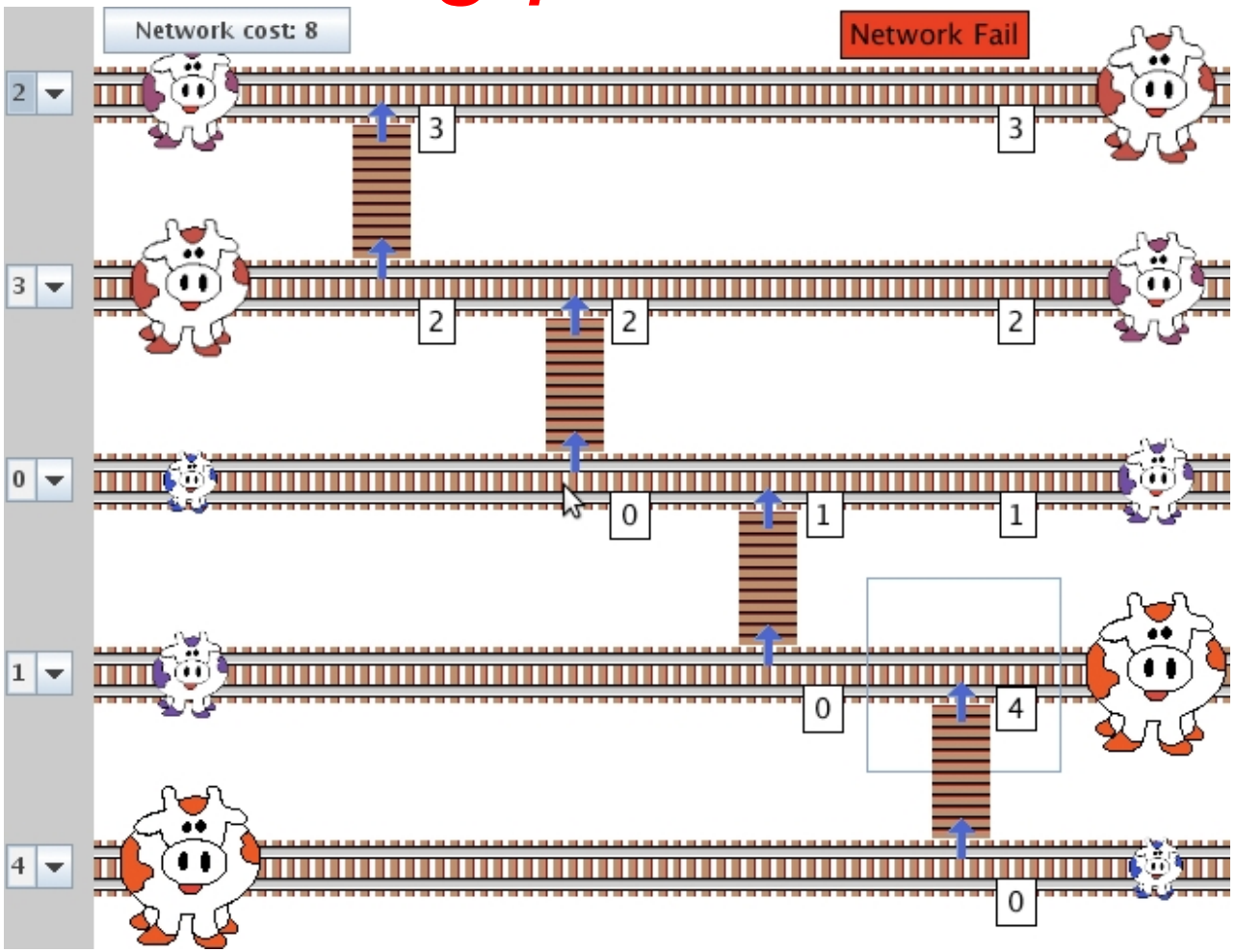
- **MaSH**
  - Level: statements

```
import sortingNetwork;

createRails(2);// create two rails with data in random order

printOrder();  // print the data

redSwap(0,1);  // Swap values in position 0 and 1

printOrder();  // print the data
```

output:

```
0, 1,
1, 0,
```

An interesting pattern:



- With **blue bridges**, it places smallest at the bottom

CONCRETE Programming

Stage 4: Textual programming environment

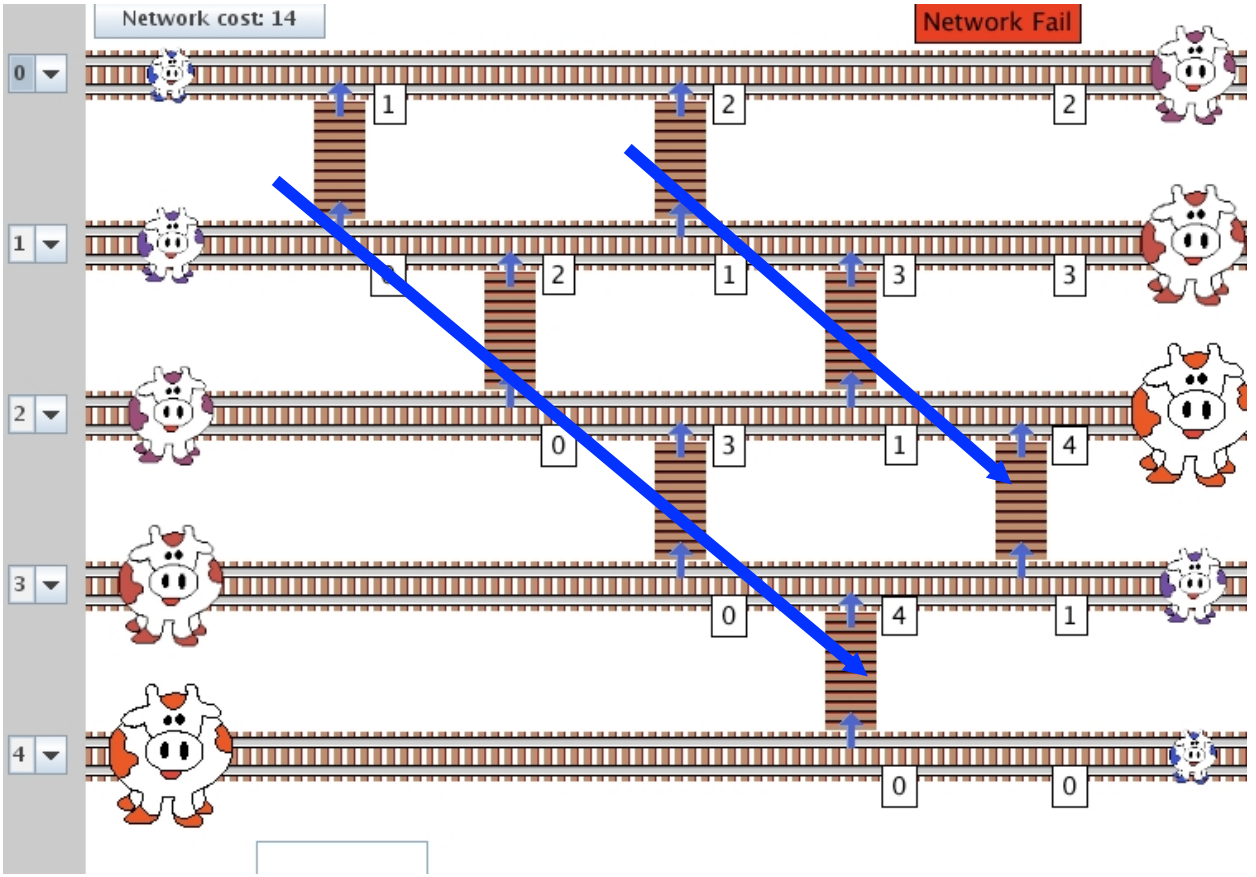
- **MaSH**
  - Level: control

```
import sortingNetwork;

createRails(5);// create 5 rails with data in random order

printOrder();  // print the data

       // cyclically shift one position to the left
for (int i=0; i<4; i=i+1)
       redSwap(i,i+1);

printOrder();  // print the data
```

output:

```
0, 1, 2, 3, 4,
1, 2, 3, 4, 0,
```

Repeat the pattern



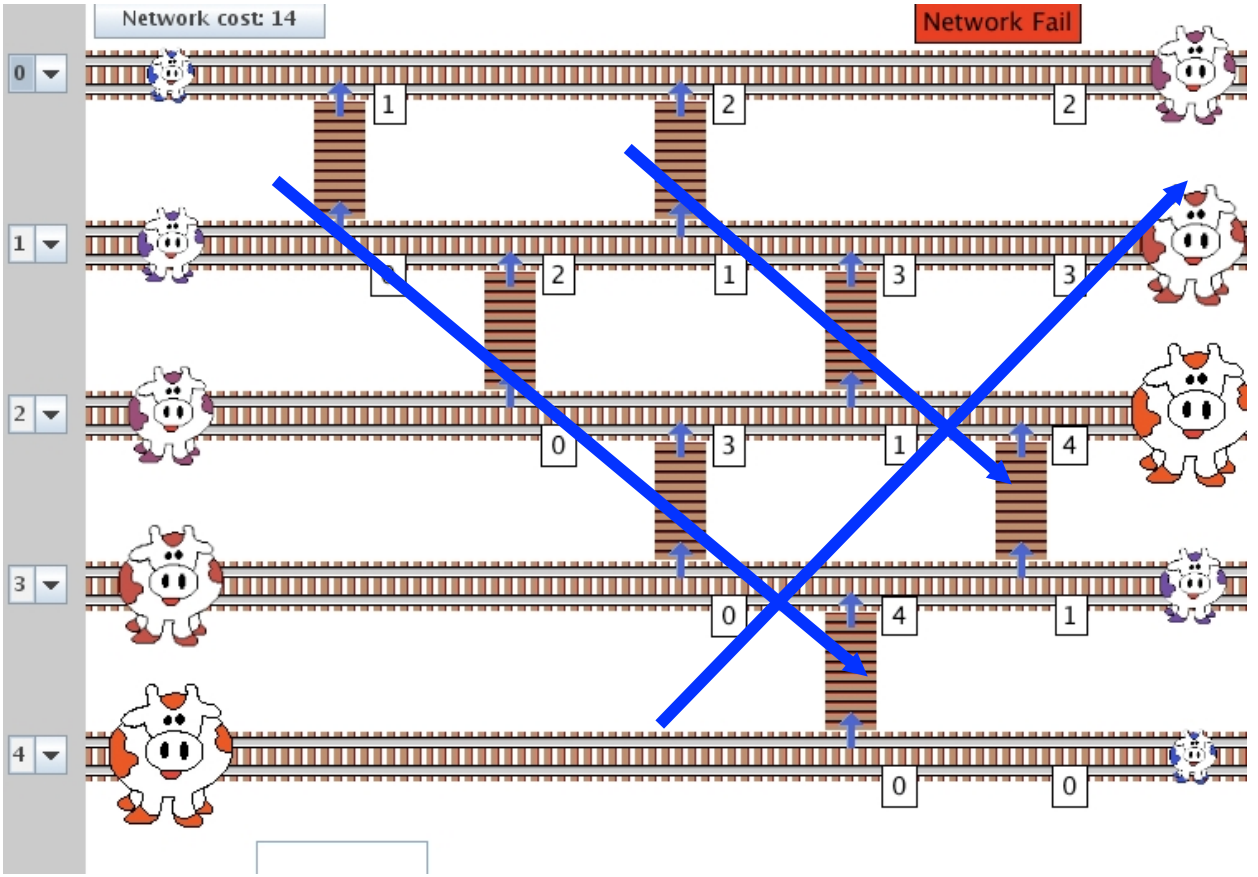
- This places the smallest at the bottom and second smallest just one above

Stage 4: Textual programming environment

- **MaSH** Level: **methods**

```
import sortingNetwork;

void select (int place) {
        for (int i=0; i<place; i=i+1)
                blueLargerUp(i,i+1);
}

void main() {
   createRails(5); // create five rails with data in random order

   printOrder();   // print the data

        // Select and place smallest at the bottom
   select(4);
   select(3);
   select(2);
   select(1);

   printOrder();   // print the data
}
```

A pattern in the pattern



- This places the smallest at the bottom and second smallest just one above

Stage 4: Textual programming environment

- **MaSH** Level: **methods**

```
import sortingNetwork;

void select (int place) {
        for (int i=0; i<place; i=i+1)
                blueLargerUp(i,i+1);
}
void sort (int place) {
        for (int i=place; i>0; i=i-1)
                select(place);
}

void main() {
   createRails(5); // create five rails with data in random order

   printOrder();   // print the data

   sort(4);       //Sort

   printOrder();   // print the data
}
```

CONCRETE Programming

Other concepts of textual programming

- With **MaSH** and the virtual tool
  - constants
  - arrays
  - sorting algorithms
  - complexity
- Proceed to object-orientation
  - class / object
  - encapsulation

CONCRETE Programing

Stage 5: What if?

- Links to research and open questions
  - What if the cost of building blocks varies?
  - What if sometimes the bridge does not succeed
  - What is the fastest with only blues?
  - Is the fastest with blues and red as costly as only with reds?
  - What is the cost gap between finding the median and sorting?
  - Can artificial intelligence techniques optimize the cost?

Robots and people

THANK YOU

CONCRETE Programing

Can research exist without Socratic (rational) thinking?

Thinking Tools - Philip Cam

Introductory tools

Suggestions, Reasons, Agreement/Disagreement, Examples, Distinctions, Borderline Cases, Target, Thought Experiments

Intermediate Tools

Agendas, Counterexamples, criteria, generalisations

Advanced Tools

Fact, Value, Concept, Deductive Reasoning, Reasoning diagrams, assumptions,