Setting the scene – a bit of the story

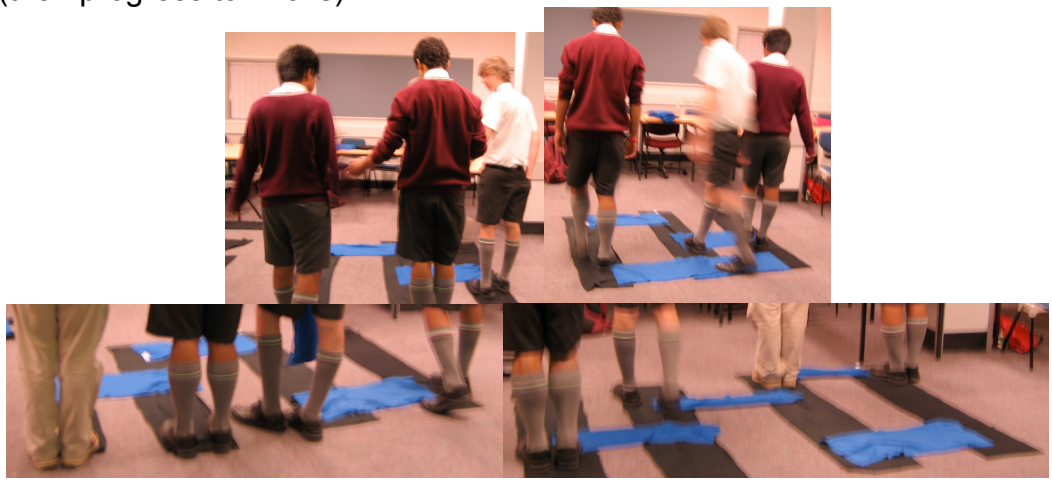
You are a framer with a herd of cows and you need to organize those that are large enough to take to the market. You would like to select your 10 largest cows, or perhaps the smallest, the largest and the median, to get an idea of the overall size of the heard (an estimate of size like the average). You don't have a mechanism to weight cows, but you can put them trough rails and sorting gates. In fact, you would like to build a sorting network with rails and gates to regularly examine the size of herds. How can you construct these rails so they use the least number of comparison gates?

The Activity

This activity is a competition between 4 or 5 teams of up to 5 participants-each. The activity requires each team to create a sorting network of parallel rails (masking tape/ribbon/duct tape) and comparisons gates (different colour tape/ribbon) on a flat surface and test if several selection or sorting arrangements can be obtained no matter in what order the cows are lined up at the start of the network.

The activity has several phases:

1. Phase 1: Participants should be able to act as the cows in the network. They earn points by correctly selecting the smallest cow among 3 rails (then progress to 4 rails).

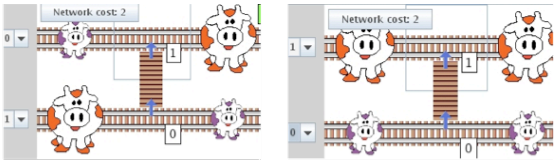


Your team shall experiment to create your own network. You earn points by illustrating their construction. These constructions are made with only the black rails and only the blue gates. These blue gates place the larger cow to the left. Specific problems to solve are:

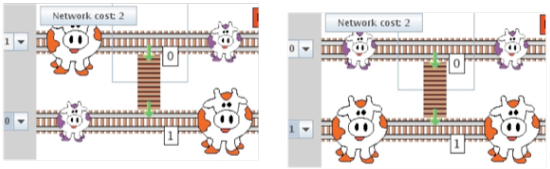
a. Find the largest cow (the largest cow is in a pre-determined rail no matter the initial order)
b. Find the smallest cow.
c. Find the smallest and the largest cow, not necessarily sorting.
d. Give the fist problem solving set of exercises
    i. Can gates that do not cross rails solve the above questions?
    ii. What is a plan to check that the network is correct?
    iii. If there are 5 rails, would it be possible to find the smallest cow with 3 rails, why yes or why no? In general, if there are *n* rails, is it possible to find the largest cow with *n*-2 or less gates?

2. Phase 2: Later you will find other types of gates. Green gates always place the smaller of the two cows on the left (or as per diagram above, the smaller cow goes up).

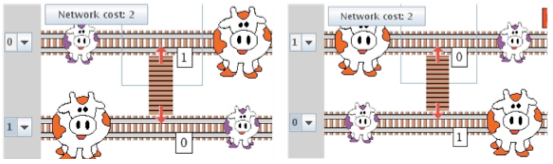
BLUE places larger up



GREEN places larger down



RED always swaps



With the mats, black stripes and red, blue and green gates solve the questions and problems as above but now with networks of 8 rails. You can work on the floor or on the table to build larger networks.

The parameters of the problems may change (size of the network, cost of the network, varying cost of gates)

3. Phase 3:The instructor will capture an image of each construction and uses the computer tool RailSort to
    a. Capture the constructions by the students
    b. Validate solutions and award them a score for different problems.

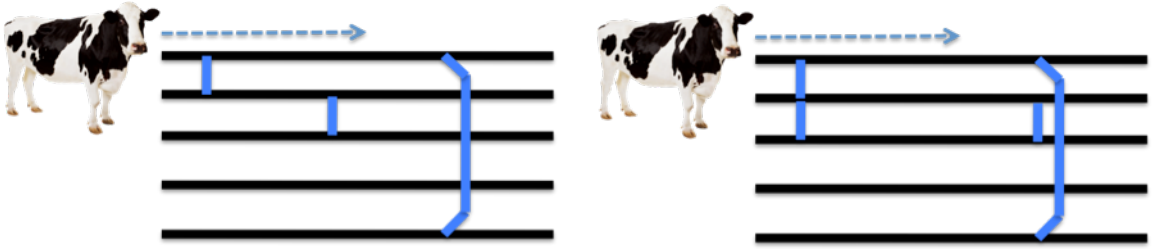
Overall Network Score: <A measure of the success of this network.>
---- SUCCESS ----
 # Runs: <The total number of inputs tested.>
 # Success: <The number of inputs that the network succeeded on, based on its current evaluation measure.>
 # Fails: <The number of inputs that the network failed on, based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

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

4. The instructor will indicate how this activity translates to real world applications (Design of circuits) and IT (problem solving, algorithm design, teamwork, etc.)
5. If time permits, the instructor discusses the possibility of the sorting network becoming textual programs (with the API for this activity and the programming language **`MaSH`**). Alternatively, can illustrate building even larger sorting networks with RailSort.

Rules for creating the network (3 or 4 rails walking on it):

- Each backcloth represents a rail where only one cow can travel. The rails are laid out parallel to each other. Gates are laid out orthogonally.
- Two gates cannot connect in the same position for the same rail
- A gate can connect across rails that are not adjacent, but it may cost more.



CORRECT INCORRECT

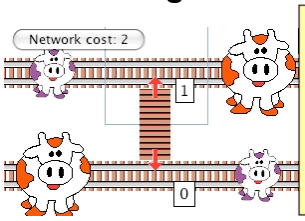
Rules for creating a network on a mat with more than 4 rails:

- Same as for those where people work on, but now there are also red gates and green gates.

OPTIONAL PHASE – Construct a description of your network with textual programming language.

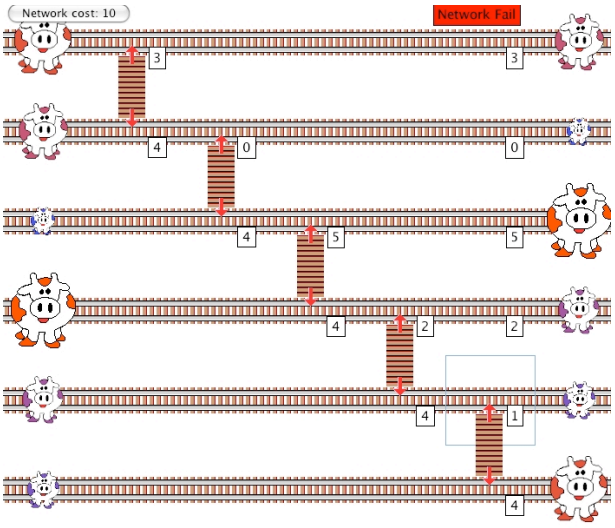
See documentation of `MaSH` sortingNetwork environment.

Code for one gate – 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
```

Code for a pattern – Mash level control



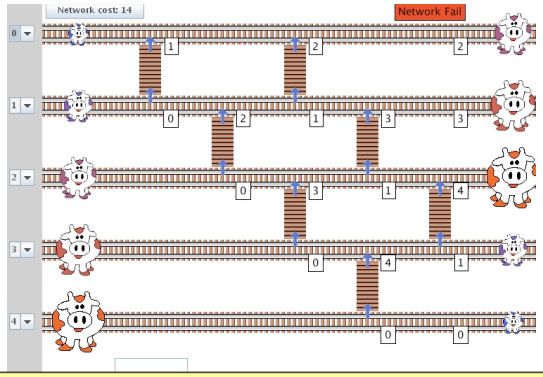

```
import sortingNetwork;

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
```

Explain what does the pattern above do

1. When all gates are red gates.
2. When all gates are blue gates
3. When all gates are red gates

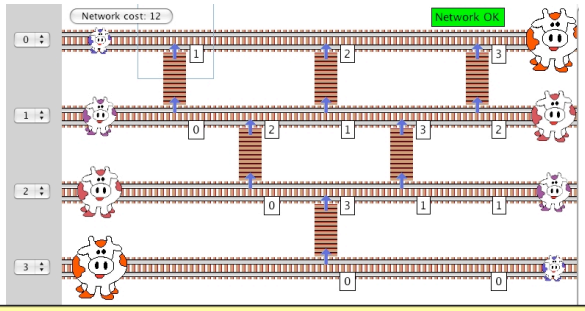
SUMMARY - Sort the Cows

Repeat a Pattern




```
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
}
```

`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
}
```