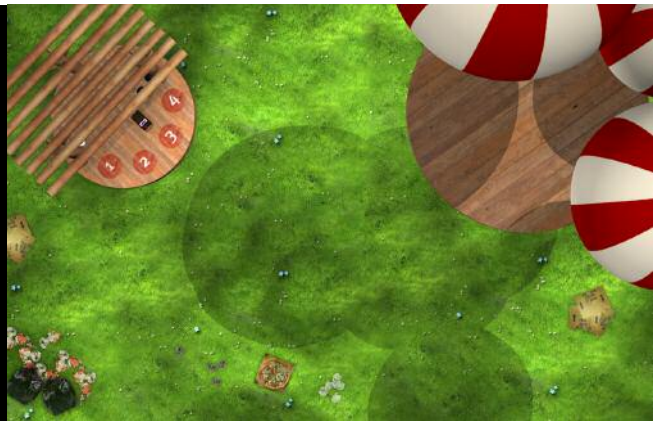
Christmas Lectures 2008

Get it sorted!



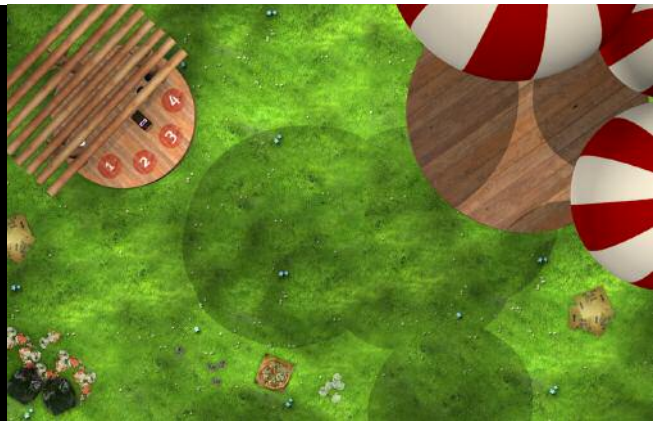
Computers don't just automatically know how to do things – they need software programmed for them by people. A mobile phone, for example, has software installed to tell the on-board computer how to carry out different functions. This includes running the menus, digitising your voice to transmit it, and even playing music or games. The software breaks down these functions into individual tasks, each of which is processed by a particular *algorithm*. 'Algorithm' may sound technical, but it is just a way of describing how to do something. An algorithm is a list of instructions, like following a recipe to bake a cake.

Computer programmers write algorithms on how to accomplish all the different things a computer needs to do. Computers constantly sort items into order. For example, every time you check your list of emails by date received or scan through your MP3 tracks alphabetically, the computer uses an algorithm to sort the items into that order. Just as there are different recipes for baking a carrot cake, there are lots of different algorithms that a computer can use to do the same thing (for example, sorting numbers into ascending or descending order). The difference is that some algorithms work faster or require less computer memory to run than others.

Here we'll show you how to try two different algorithms to sort numbers. You can race the algorithms against each other in teams with your friends at home or at school!

Christmas Lectures 2008

Get it sorted!



SORTING ALGORITHMS

BubbleSort and QuickSort are both sorting algorithms that you can try out either one after the other or in a race against each other. Count how many steps each algorithm uses to sort your list of numbers. Which algorithm uses fewer steps and is therefore more efficient? Which one can sort the list of numbers the fastest?

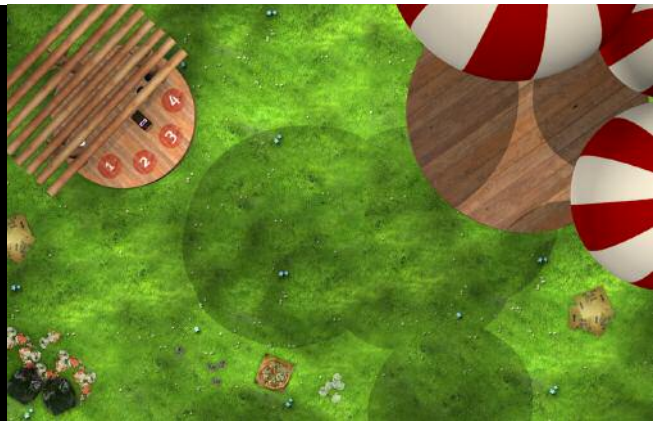
To run both algorithms you need to number each team member from 1 upwards. Write the numbers on A4 sheets and give one to each team member. One person needs to act as team controller. They operate the algorithm and actually sort the numbered people into ascending order. The numbered team members represent the computer data and the team controller the processor running the software using the data.

BubbleSort

BubbleSort is easy to understand and program a computer with. To try the BubbleSort algorithm, line the team up shoulder to shoulder facing the controller. Shuffle the numbered sheets (to put the data into a random order) and then hand one to each person in the line. Ask everybody to hold their number up so that the controller can see it.

1. The controller starts at their left-hand side of the line of data. They look at the first two numbers, if the larger number is on the left the two numbers need to be swapped. The two people move so that the smaller number is on the controller's left and the bigger number on the right. If the larger number is already on the right then the people stay in place in the line.
2. The controller then takes one step to the right and looks at the next pair of numbers (i.e. the rightmost number from before is now the leftmost one of the pair and a new number is the right of the pair). The controller repeats step 1: swap the two numbers if the larger one is on the left otherwise leave them alone.
3. The controller continues like this all the way along the line, swapping pairs of numbers if necessary. Once they reach the right-hand end of the line they run back to the beginning and start moving along it again. Each time the controller moves along the line the numbers get steadily more organised. When the controller can walk all the way along the line without making any swaps the numbers are in ascending order and BubbleSort has finished!

Get it sorted!



QuickSort

QuickSort can be faster than BubbleSort, but is more complicated and uses more computer memory to run. To try the QuickSort algorithm, again get the team to line up shoulder to shoulder facing the controller, shuffle the numbered sheets to randomise the data and then hand one to each team member.

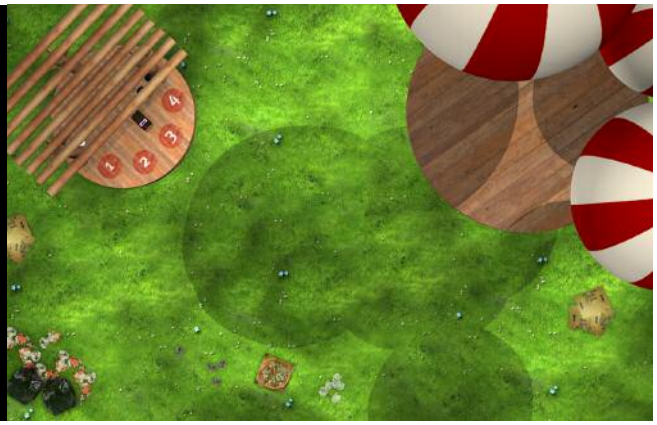
1. The controller picks a random person in the list and asks them to put one hand on their head. This number is the *pivot* (like the middle of a see-saw). Then the controller walks along the numbers from their left to right and tells anyone whose number is lower than the pivot number to move to the left of the pivot (from the point of view of the controller) and anyone who is higher to stand on the right of the pivot. When the numbered people move they are not allowed to arrange themselves into order – they must stand wherever they arrive. At the end of this step you should have a line of numbers with the pivot somewhere in the middle (but don't worry if there aren't any numbers to the left or right of the pivot – the controller may have picked the smallest or biggest number to be the pivot). The pivot number must keep their hand on their head so that the controller knows they have already acted as a pivot.
2. Now the numbers to the left and the numbers to the right of the pivot form two different groups. For both groups the controller picks a new pivot number. These two pivots put their hand on their head, and again within each of the subgroups the controller tells numbers to move to the left or right of their group's pivot. Also, all of the pivots must keep their hand on their head afterwards.
3. This process is repeated with smaller and smaller groups. When all of the numbered people have their hand on their head QuickSort has finished. If the controller has sorted them properly, the list should be in ascending order.

Now that you've tried both methods:

What did you find? Which algorithm was faster: BubbleSort or QuickSort?

Despite its name, QuickSort is not always faster than BubbleSort. BubbleSort can order a handful of numbers in just a few passes whereas QuickSort might take longer because it needs a bit more reorganising of the numbers. But for longer lists of data QuickSort becomes much faster than BubbleSort.

Get it sorted!



Trying other data

When you've run both sorting algorithms with numbers, try seeing how they work on other data. Get each team member to write their name on their sheet of paper, ask them to line up in random order and then use both algorithms to sort them into alphabetical order. You can also use BubbleSort and QuickSort on other data, for example, increasing order of height or decreasing order by age. Also, how would you adapt the two algorithms to sort numbers into descending order?

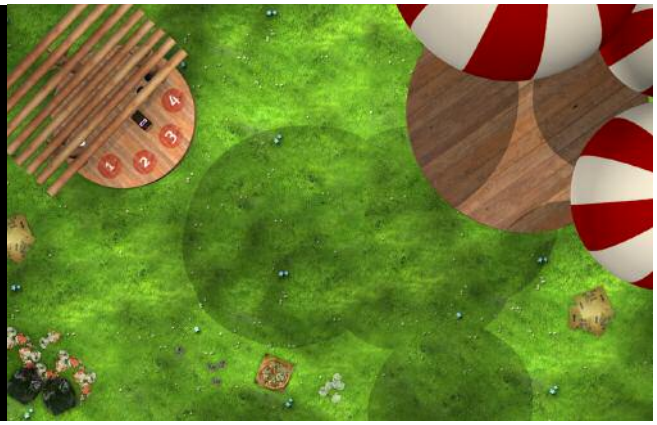
Now, let's take a look at a different approach to ordering data, called a sorting network.

SORTING NETWORKS

The main reason why both of these algorithms take a while to sort even a short list of numbers is because the controller can only do one thing at a time. For example, in the BubbleSort algorithm the controller can only look at one pair of numbers at each step before moving onto the next pair. Exactly the same is true of computers – a lot of the time the processor just sits around waiting for one function to finish before it can move on. So why not speed it up by letting it calculate more than one thing at a time? This idea is called *parallel processing*. The computer breaks up jobs into lots of little tasks so that many of them can be done at the same time by multiple computer chips. Sorting algorithms can be turned into parallel processing using a *sorting network*.

The diagram on the last page shows a network that operates like a parallel processing version of BubbleSort for quickly sorting six numbers. The numbers are placed in random order in the squares at the start. Each number moves along to the next column on the right, following the arrows. At each circle, two numbers meet and are compared with each other. The smaller number moves left (up the page) and the larger number moves right (down the page). By the time the numbers reach the right-hand side of the network they will have been sorted into ascending order.

Get it sorted!



To try this for yourself: print or copy the sorting network. Number squares of paper 1-6 and place them in random order one number to each square on the left. Now push the numbers through the network of comparison circles, following the rules outlined above.

Why not ask your teacher if you can do this on a larger scale at school?
Find some chalk or masking tape and copy the sorting network on the playground or classroom floor. Number six A4 sheets and give one to each person.
Now get each person to walk through the giant-size sorting network.

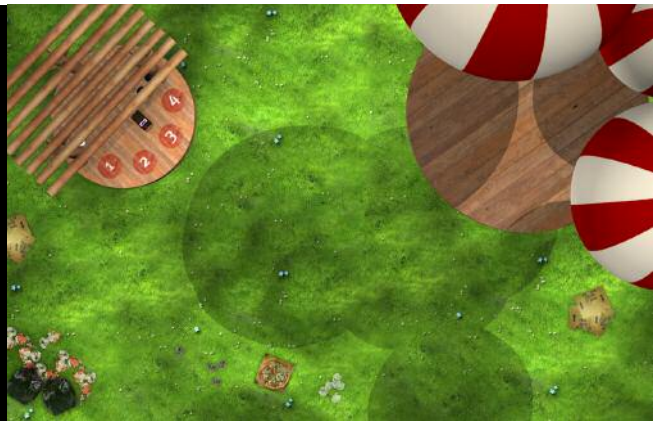
CS Unplugged

View a video of a sorting network being run at a school:

http://csunplugged.org/index.php/en/videos-mainmenu-139

Christmas Lectures 2008

Get it sorted!

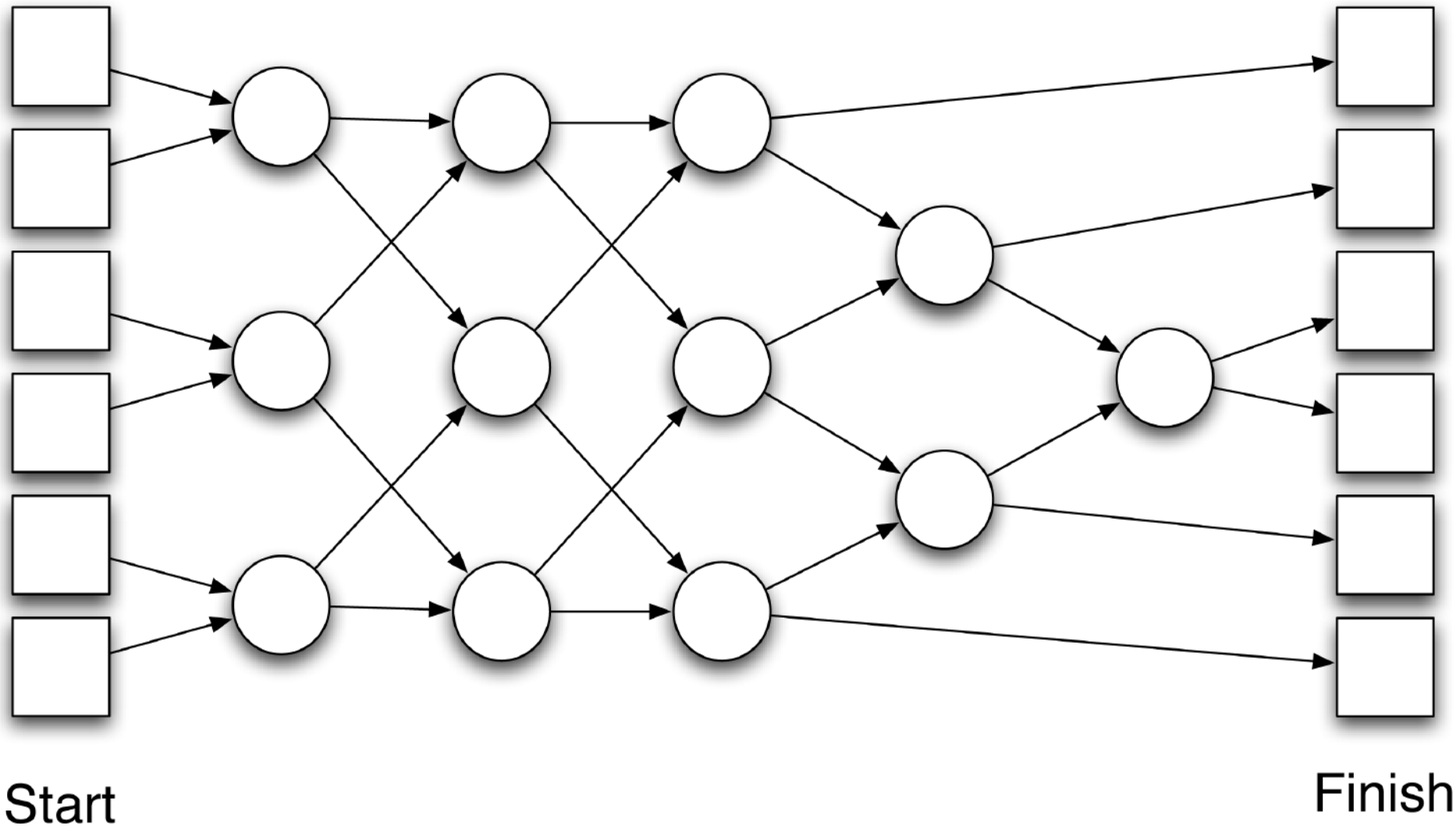


Try racing the sorting network against the BubbleSort and QuickSort algorithms – which is fastest?

The reason that this network is so quick at sorting numbers is that at each step three (rather than one) comparisons are made at the same time.

Why not try this sorting network for organising people into alphabetical order by name or into order of height or age?

Finally, try and work out what the sorting network would look like for sorting 10 numbers, or 20, or more. Where do the circles and lines need to go for bigger sorting networks? How does the number of comparison circles change as you add more numbers to be sorted?


Start
Finish