The Swap Puzzle

Age group:	7 – adult
Abilities assumed:	Nothing
Time:	50-60 minutes,
Size of group:	8 to 30

Focus

What is an algorithm?
Testing
Efficiency of algorithms
Computational Thinking: algorithmic thinking

Syllabus Links

This activity can be used both as a general introduction to what an algorithm is and their efficiency from KS2 upwards, writing and testing programs is as well as introducing computational thinking.

Summary

You set a simple swap puzzle to be solved. The aim is not only to solve the puzzle though, but to come up with a set of instructions that anyone could follow to solve it in future. Those instructions should have the shortest possible number of steps so that the algorithm is as fast as possible. The class then try out their solutions, aiming to beat the clock, but with students as the pieces on a giant 'board'. They have to finish the puzzle correctly by following the instructions before their time runs out. The activity introduces the idea of the solution to a problem being a set of instructions that allow others to 'solve' it with no understanding. It also shows how different algorithms can solve the same problem but some may be better than others.

Technical Terms

Algorithm, computational thinking, algorithmic thinking, efficiency, testing.

Materials

For each team:
    Game board sheets
    Game pieces (cut out card or use bottle tops, buttons or similar)
    Instruction sheets and algorithm recording sheets
Sheets to label squares on the giant board
    Ideally print a set for each level on to red, white or blue paper as appropriate for the board of the level being played.
7 chairs
3 red team bibs and 3 blue team bibs
Countdown timer

What to do

The Grab:
Explain that the class are going to be solving a puzzle in a race against time, where they will be the pieces. To beat the clock, they will have to think like computer scientists.

The Set-up:
Split the class into pairs. Give each a set of puzzle boards, pieces, rules of the game and algorithm recording sheets. Have the chairs, labels, etc available to make the giant boards.

The activity:
Explain the aim of the game as well as the rules as below. Demonstrate the starting position, possible moves and winning position on the level 1 (3 square) board.

The aim of the game is to swap the position of the blue pieces with those of the red pieces so that all the red pieces end up on red squares and all the blue pieces end up on blue squares. You must do it in as few moves as possible.

There are two kinds of move:

1) Move a piece to an adjacent empty square (forwards or backwards).
2) Jump a single adjacent piece of any colour into an empty space (forwards or backwards).

Start the game with the red pieces at the blue end and the blue pieces at the red end, one piece per square. The square in the middle should be the only square left empty at the start.

The game can be played on different sized boards with different number of pieces, the bigger the board the harder the puzzle:

Level 1 Easy:	1 piece per colour,	3 squared board
Level 2 Medium	2 pieces per colour,	5 squared board
Level 3 Hard	3 pieces per colour,	7 squared board

Have everyone try and solve the level 1 puzzle to make sure they understand the rules. Work through the solution with them. They can then move on to the harder puzzles.

Explain that they not only have to solve the puzzles, but they must record the solution in a precise way so that anyone else could then solve it just by following their instructions. They will be writing down an ‘algorithm’ for solving the puzzle. They do this on the special algorithm recording sheet. They must write down the series of moves in the order they must be made using instructions of the form:

Square **????** GETS THE PIECE FROM Square **????**

They must first fill in the square the piece is to be moved **TO** and then the square it is to be moved **FROM**. For example, to move the piece from square 0 to square 1, you would write:

Square 1 GETS THE PIECE FROM Square 0

Emphasise that the first number is the place the piece is moved to. (We do it this way round to mirror the typical way assignment is done in programming languages).

Have the class work out and write down the algorithm for the level 1 puzzle.

Now explain that the ultimate aim is for each team to run the puzzle for real faster than the target time for that level. The target times (where we have given 5s per move assuming the optimum solution) are as follows. Adjust these times depending on the age of the class to make it a doable target but only provided the algorithm is good. A variation is to allow each team 3 attempts but reduce the target time for each.

Level 1 target time: 15 seconds

Level 2 target time: 40 seconds

Level 3 target time: 75 seconds

Set out three chairs in a line. Tape the labels naming the chairs on the floor below them as e.g., Square 0 (ideally using the colour as on the board. Choose two trustworthy class members and give one a blue bib and the other a red bib. Sit the blue person on square 0. Sit the red person on Square 2.

Now pick one team to try their algorithm. One person reads the instructions, a step at a time. Only when the move is completed can they move on. The other taps the appropriate person on the shoulder and points them to the seat to move to. They move. The reader then reads the next instruction. You use a countdown timer set to 15 s to time them. The team have to finish before the timer runs out.

In their pairs, the class should now try to solve the medium puzzle, writing down their moves, and so the algorithm as they go. They may wish to adopt a short hand like an arrow for the instructions (in doing so they are inventing their own programming language!).

Point out the importance of testing. Once solved they should check the algorithm by running through it to be sure it does work and they didn't make a mistake writing down any instruction. They do not want to run the algorithm in front of everyone and find it doesn't work.

Once they are sure it works, they should also think about whether there is a faster algorithm. In fact, for each level the most efficient solution never requires a piece to move backwards, so if a team's algorithm does that, then they can do better.

When a team is ready they can try the timing challenge. They should announce how many steps long their algorithm is. Set out more chairs to make the bigger board. Stop everyone to watch and get trustworthy students again to be the pieces.

Teams can move on to the level 3 puzzle, once they have solved the level 2 puzzle.

The explanation:

One of the core topics of Computer Science is the study of 'Algorithms'. What do we mean by an algorithm though? It is just a series of actions to perform and the order to do them to get a job done. The study of algorithms is about coming up with such sequences that guarantee particular jobs are done. Once you have solved a problem, you don't want to have to solve it again. If you write down the algorithm of the solution then you won't have to. Once you have an algorithm, you, anyone else or even a computer, can just blindly follow its instructions to solve the problem. That is what we have done with the puzzle. Computational thinking is about only accepting you have solved a problem when you have an algorithm to do it!

It's also about devising efficient ways of doing things. Efficiency is a big issue in computer science. The challenge is not just about coming up with an algorithm that works. The challenge is to come up with an algorithm that is 'efficient'. Being efficient can mean lots of different things. A factory could be hailed as 'efficient' if it uses as few resources (people, raw material, money) as possible - producing the goods with the bare minimum. Alternatively, it could be 'efficient' meaning producing the goods as fast as possible. That might or might not use more resources to achieve.

Algorithms can be efficient in different ways too. They can store as little data as possible or need vast resources. They can be fast or slow. There can be many different algorithms to do the same thing. Which you choose depends on what properties of the algorithm is most important to you at the time. Two different ways of doing something could both guarantee to get the job done but one may be quicker than the other and so better because for that job speed matters.

How fast an algorithm is depends on how quickly it is executed though. Rather than worrying about times, which in any case depend on the speed of the processor or even the particular way it is coded in a programming language, we compare algorithms by counting the major operations involved. An algorithm with more steps will generally be slower than one with fewer steps, however they are implemented.

It is very easy, if you are persistent, to get the puzzle out. Anyone can use trial and error to come up with a series of steps, an algorithm, which eventually does swap the pieces over. However, especially with the longer version of the puzzle, it is much harder to come up with a really efficient version. That takes some solid algorithmic thinking. It is important to look ahead before committing to a move – will it lead to you getting stuck and having to back track further down. It is also easy to overlook possible moves at each step. It's important that you don't. Attention to detail is an important part of algorithmic thinking and of computational thinking more generally.

The exercise shows the importance of testing. It is easy to make a mistake in writing down the algorithm you found that solves the puzzle. That is why it is vital that an algorithm is tested before being used for real. Testing is another important part of algorithmic thinking. You mustn't just assume it works. You must always check.

When we do try the algorithm out we are running the algorithm just like programs are run. The only difference is we ran it on a processor made of people rather than of silicon.

Solutions:
The following are efficient solutions for each level.

The level 1 puzzle can be solved in 3 moves as follows:

Step 1: **Square 1** GETS THE PIECE FROM **Square 0**

Step 2: **Square 0** GETS THE PIECE FROM **Square 2**

Step 3: **Square 2** GETS THE PIECE FROM **Square 1**

The level 2 puzzle can be solved in 8 moves as follows:

Step 1: **Square 2** GETS THE PIECE FROM **Square 1**

Step 2: **Square 1** GETS THE PIECE FROM **Square 3**

Step 3: **Square 3** GETS THE PIECE FROM **Square 4**

Step 4: **Square 4** GETS THE PIECE FROM **Square 2**

Step 5: **Square 2** GETS THE PIECE FROM **Square 0**

Step 6: **Square 0** GETS THE PIECE FROM **Square 1**

Step 7: **Square 1** GETS THE PIECE FROM **Square 3**

Step 8: **Square 3** GETS THE PIECE FROM **Square 2**

The level 3 puzzle can be solved in 15 moves as follows:

Step 1: **Square 3** GETS THE PIECE FROM **Square 2**

Step 2: **Square 2** GETS THE PIECE FROM **Square 4**

Step 3: **Square 4** GETS THE PIECE FROM **Square 5**

Step 4: **Square 5** GETS THE PIECE FROM **Square 3**

Step 5: **Square 3** GETS THE PIECE FROM **Square 1**

Step 6: **Square 1** GETS THE PIECE FROM **Square 0**

Step 7: **Square 0** GETS THE PIECE FROM **Square 2**

Step 8: **Square 2** GETS THE PIECE FROM **Square 4**

Step 9: **Square 4** GETS THE PIECE FROM **Square 6**

Step 10: **Square 6** GETS THE PIECE FROM **Square 5**

Step 11: **Square 5** GETS THE PIECE FROM **Square 3**

Step 12: **Square 3** GETS THE PIECE FROM **Square 1**

Step 13: **Square 1** GETS THE PIECE FROM **Square 2**

Step 14: **Square 2** GETS THE PIECE FROM **Square 4**

Step 15: **Square 4** GETS THE PIECE FROM **Square 3**

Variations and Extensions

Online version of the puzzle

Instead of doing a paper and pencil exercise, students do the online version of the puzzle (http://www.cs4fn.org/algorithms/swappuzzle/). It records a version of the algorithm for you as you play. This could be done with or without the follow-up role play part.

Short version

For a shorter lesson don't act out the algorithms just have teams check them on the paper board.

Assignment

Rather than have students fill out the algorithm recording sheet using the English version of the command have them write the algorithm using a more typical programming notation such as:

Set Square1 to Square0

or

Square1 := Square0;

or

Square1 = Square0;

Programming the puzzle in Scratch

Have the class program their own version of the puzzle in Scratch, using characters of their choice as pieces. In the simple version they have to just give instructions to move the pieces in the right order. A more complicated exercise is to create a Scratch game version of the puzzle, similar to the online program above, that allows a player to try to solve the puzzle themselves.

Programming a general solution (advanced)

Have more advanced members of the class write a program that can solve the swap puzzle for itself, whatever size board it is presented with.

Further Reading

The FUNdamentals of Algorithms

http://www.cs4fn.org/fundamentals/algorithms.php.

Links to other activities

The intelligent piece of paper

Take part in a test of intelligence against an intelligent piece of paper! This is a good introduction to what an algorithms is and how a computer program is just an algorithm. It can also be used to start a discussion on what it would mean for a computer to be intelligent. It can lead on to an unplugged programming activity creating winning instructions.

The Invisible Palming Trick

Teach a trick where the magician invisibly moves a card between 2 piles. This is a fun way to introduce the idea of an algorithm, showing how algorithms are a series of steps that if followed precisely lead to something (in this case magical) being guaranteed to happen – even if the person (or computer) following the algorithm doesn't know what they are doing.

Live demonstration of this activity

Teaching London Computing give live sessions for teachers demonstrating this and our other activities. See http://teachinglondoncomputing.org/ for details. Videos of some activities are also available or in preparation.

SUPPORTED BY
MAYOR OF LONDON

The Swap Puzzle:
Rules

Aim

The aim of the game is to swap the position of the blue pieces with those of the red pieces so that all the red pieces end up on red squares and all the blue pieces end up on blue squares.

You must do it in as few moves as possible.

Moves

There are two kinds of move:

1) Move a piece to an adjacent empty square (forwards or backwards).
2) Jump a single adjacent piece of any colour into an empty space (forwards or backwards).

Starting position

Start the game with the red pieces at the blue end and the blue pieces at the red end, one piece per square. The square in the middle should be the only square left empty at the start.

Board

The game can be played on different sized boards with different number of pieces. The bigger the board the harder the pieces:

Level 1 Easy:	1 piece per colour,	3 squared board
Level 2 Medium	2 pieces per colour,	5 squared board
Level 3 Hard	3 pieces per colour,	7 squared board

Start with the easy board and work up through the levels as you solve each.

Algorithm Recording Sheet

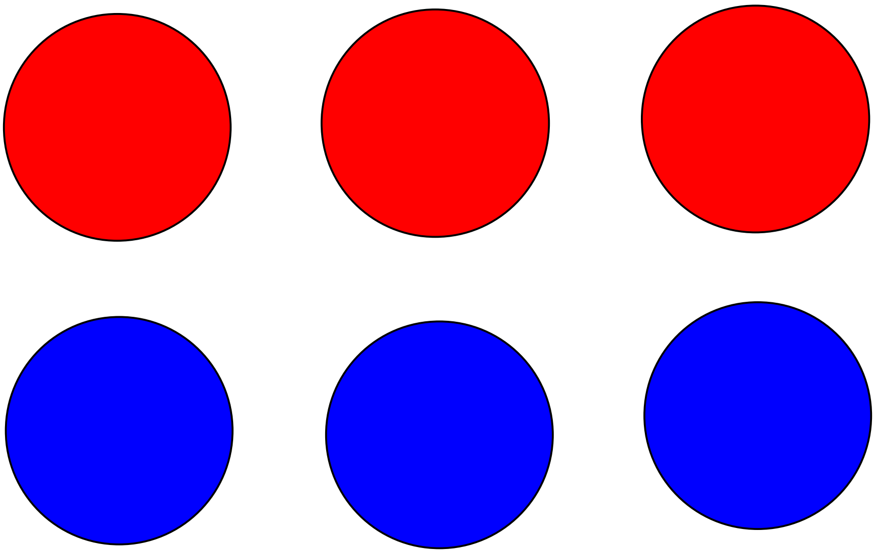
STEP	TO	COMMAND	FROM
1		GETS THE PIECE FROM	
2		GETS THE PIECE FROM	
3		GETS THE PIECE FROM	

STEP	TO	COMMAND	FROM

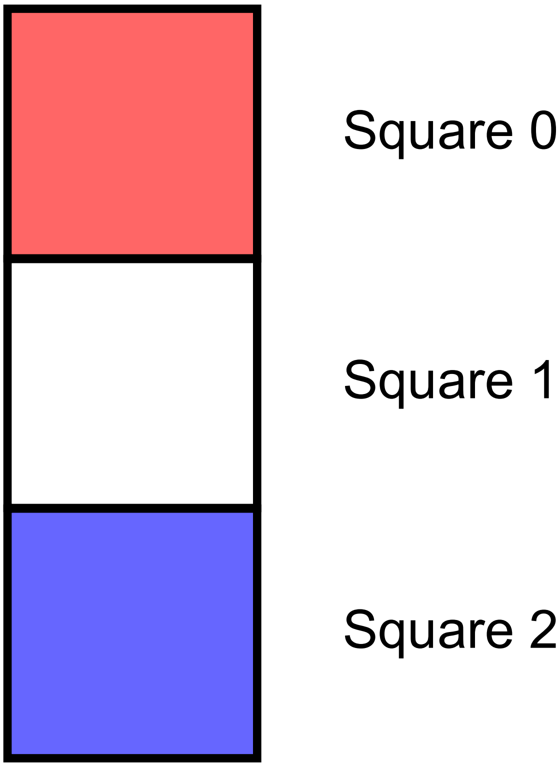
Example of how to fill in the algorithm recording sheet

STEP	TO	COMMAND	FROM
Step 1:	Square 1	GETS THE PIECE FROM	Square 0
Step 2:	Square 0	GETS THE PIECE FROM	Square 2
Step 3:	Square 2	GETS THE PIECE FROM	Square 1

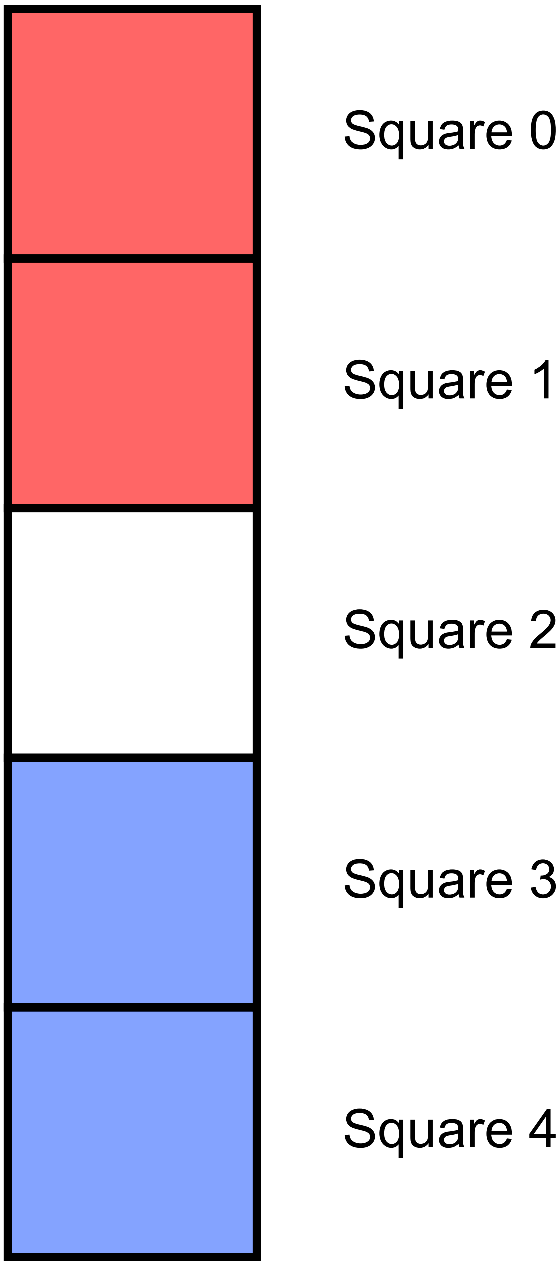
Swap Puzzle Pieces



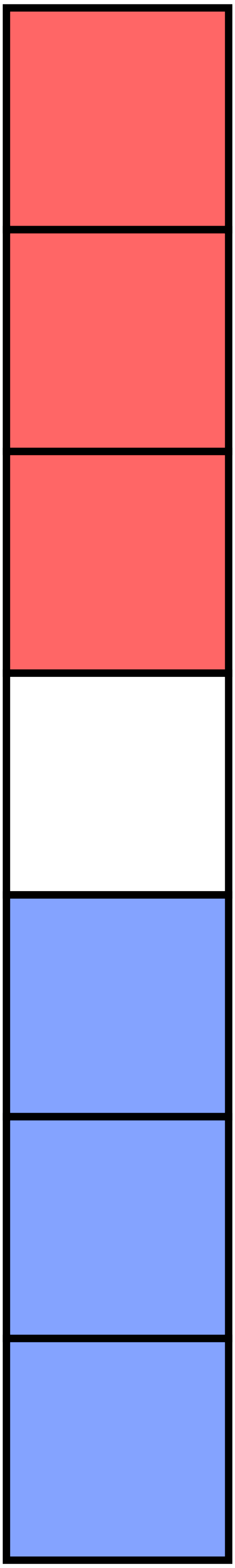
Board for Swap Puzzle
Level 1 (Easy)



Board for Swap Puzzle
Level 2 (Medium)



Board for Swap Puzzle
Level 3 (Hard)



Square 0

Square 1

Square 2

Square 3

Square 4

Square 5

Square 6

Square

0

Square

1

Square

2

Square

3

Square

4

Square

5

Square

6

Square

7

Square

8

Square

9