Clever Stuff For Common Problems

Going beyond simple algorithms

Data Structures Matter

Classroom Resources

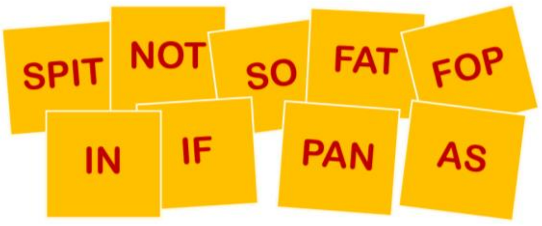
Data Structures Matter

Teacher Notes to support Tenderfoot Unit 2: Clever Stuff For Common Problems – Going beyond simple algorithms

This activity demonstrates that different ways to hold data (data structures) can have a profound impact on our ability to solve a problem.

Preparation required:
1 set of 9 Spit Not So cards, guillotined for each pair of students.
1 'Top Secret' aid for each pair.

Spit Not So



Doing things well depends on the way we order information. That is the key intended learning outcome from this activity. Split group into pairs. Each pair has a set of 9 cards spread randomly between them. Display the rules on the board using the slide included. Select one person in each pair to take the first turn.

Taking alternate turns, each student takes a card. The objective of the game is to be the first player to collect three cards with words that contain a common letter. For example, SPIT, NOT and FAT all contain the letter T. Each time a player wins a game, they are awarded 1 point. All cards are returned to the middle and a new game starts. The players take turns to be the player taking the first card.

Each pair should play as many games as possible in a few minutes, keeping score. Try to encourage a fast pace between games, maybe setting a competitive element to see which pair can complete 'x' number of games first. When you stop the activity ask those who are winning in each pair to put their hands up. If a pair have an equal number of wins, play one more game to ensure one is in the lead, whilst you discuss the game with the group, asking if they found it easy / difficult etc.

The Secret

NOT	IN	PAN
SO	SPIT	AS
FOP	IF	FAT

Explain you are going to give those currently losing some top secret help. Select those who didn't put up their hand and gather them around your desk, or ideally just outside the room. Establish if they have played noughts and crosses. If not, demonstrate the game. Hand each a copy of the Top Secret aid and insist on them keeping their instructions hidden in their hand. They will need a pencil to keep a note on their aid. The aid arranges the words in a 3 by 3 grid, just like a noughts and crosses board. Words containing the same letter form a line, either vertically, horizontally or diagonally. By recording the words they take, and their opponents, using X and O, children can now play Spit Not So as if it was a game of noughts and crosses. They may not always win, but most children know how to force at least a draw in noughts and crosses.

The students should return to their pairs. Explain to the class that you have given them some secret training and that we are going to play again. Start with a blank score and give students several more minutes to play some more.

When you stop the groups again, ask those with the Top Secret aid whether it has changed their fortunes, before discussing as a class. Ask the children to explain why the Top Secret aid helped them.

Reinforce the point that the way we structure data has a very big bearing on how successful we might be. We shouldn't underestimate the challenges facing children when learning to code. At KS3, repeated exposure to single variables and one dimensional lists or arrays is probably about right, but by introducing more complex data structures (without worrying about how we implement them in code) we can begin to open their eyes to clever algorithms that can solve everyday problems.

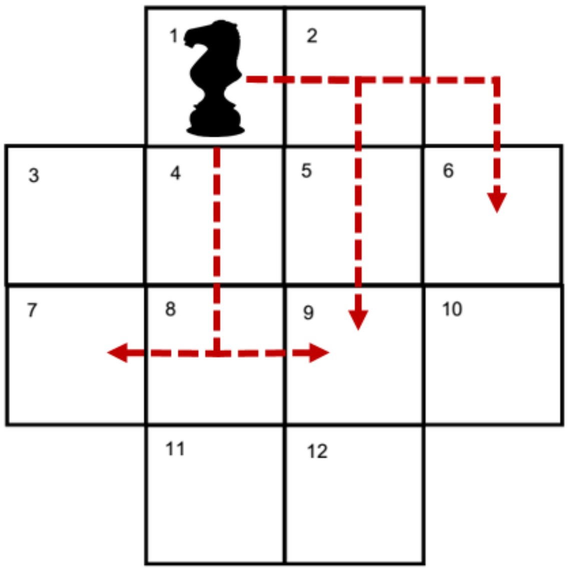
Problem Representation and Abstraction

Teacher Notes to support Tenderfoot Unit 2: Clever Stuff For Common Problems – Going beyond simple algorithms

These two class activities are designed to demonstrate how the representation of a problem can help or hinder the solution. They focus on representing problems through graphs, graph traversals and using abstraction to draw out the essential elements of a problem.

Preparation required:
Knights Tour resources (board, solution sheet and Tour Guide map) + 1 counter per student.
Plain paper and pencils.

A Knights Tour



A Knights Tour is a puzzle, recently been reproduced as a cs4fn booklet free to download from http://teachinglondoncomputing.org

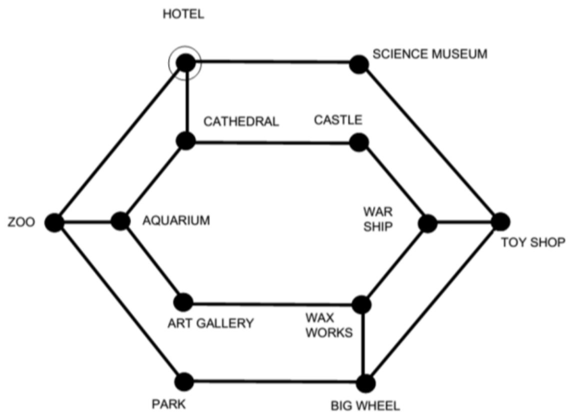
Give each student a board and Knight piece (counter). This can be cut from the board if needed. Place the Knight on square 1. Check everyone is familiar with the way a Knight can move: from square 1, the piece could move 3 possible ways, to squares 7, 9 or 6. The challenge is to work out a series of moves so the knight visits each square once (and once only) and ends up back at its original position.

The Knights Tour: Solution

Move	From Square Numbered	To Square Numbered
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		

Give out the sheets to record the moves. You could perhaps offer a small prize for the first to solve it. There are many solutions, so if a child solves it quickly, challenge them to find as many different ways as possible. It isn't easy, however, so allow perhaps 15 minutes for this activity and then stop everyone. Establish if anyone / how many did succeed and discuss how hard they found it.

Tour Guide



If the Knights Tour was a little challenging, helping a Tour Guide plan an outing to visit every attraction might be easier. Ask a child to give out the maps as the students will solve this one very quickly. Alternatively project onto a whiteboard and ask children to come and outline their solutions, or simply read the order of attractions to visit.

There are many solutions, one is highlighted on the slides provided. Solving the problem isn't the point of the exercise. Discuss with the class why it was much easier to solve this problem, compared to the Knights Tour. Answers may include the fact that you can trace a trail, having a map makes it easy to see etc.

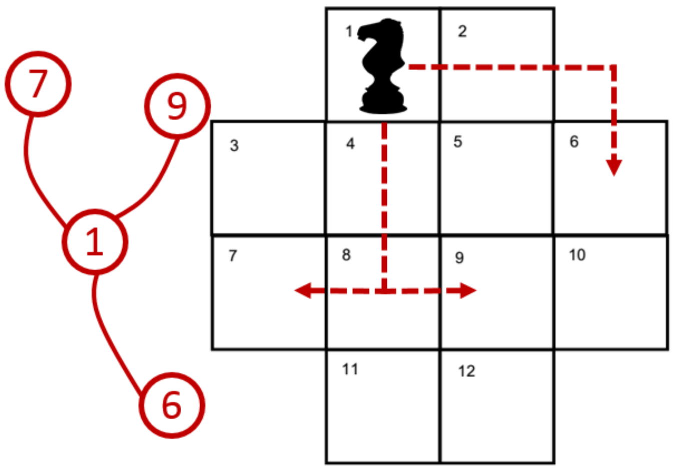
In essence, the way the problem is represented makes it easy to solve. The map or diagram is an example of a structure computer scientists call a graph. A graph consists of a series of nodes connected by edges and they are remarkably useful for representing a range of problems.

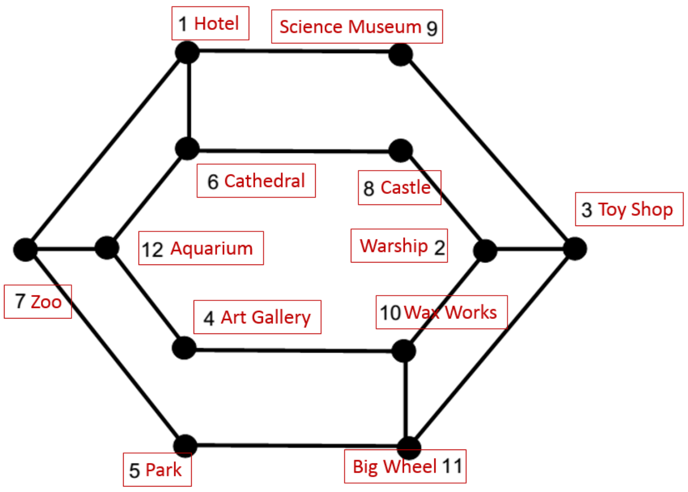
A Knights Tour Revisited

Deriving a graph from a problem statement is a good example of representational abstraction. The graph captures all the relevant information required to solve the problem, and remove s any superfluous detail.

Discuss with the class whether we could use a graph to represent the Knights Tour. What superfluous detail could we remove? Do we need to know which squares, for example, are adjacent to each other?

All we really need to know, to solve this problem is which squares the Knight can reach from any location. A node can represent each square, linked by an edge to every other square the Knight can reach, so node 1 is connected to node 7 and to node 9 and node 6 as shown.





Armed with that knowledge, give out plenty of rough paper and challenge the students to complete the graph representation. The answer is available on a slide for students to check their attempts against. Ensure they check carefully.

Often they may not look alike, but a little re-arranging of nodes will show it is the same. The layout is secondary to the connections. By arranging it in the way shown on the slide should make something obvious. Do they notice anything about the graph? It is identical to the London Tour Guide problem!

By writing the specification of the problem in general terms, we can begin to see similarities. Once we see the similarities in problems, we can apply a general solution to both. This notion of generalisation, of recognising patterns is a central concept in Computational Thinking.

The solution to both these problems, visiting every node and returning to complete a circuit is known as a Hamiltonian Cycle. It is named after the Irish physicist and Mathematician, William Rowan Hamilton (1805-1865), who devised a puzzle to visit every corner of a dodecahedron, as shown on the final slide. The twelve sides of the dodecahedron can be represented as a graph and a cycle traced through it.

Tracing a route through a graph can make good homework exercises, but the difficulty depends on the size of the graph. Determining whether a graph contains a Hamiltonian Cycle is one of a class of very hard problems for computers. Visiting every node of a graph is just one of several common operations which collectively can be called graph traversals. As you might expect various algorithms have been developed to perform different types of traversals.

Developing an efficient way of finding some types of traversal remains a key challenge in computer science. The key point for us though, is to stress that by abstracting away details and recognising similarities we have the potential to use a general algorithm to solve a particular problem.