PROGRESS OUTCOME 7

Solving unsolvable problems

Context

The students have been investigating possible answers to the question "How could you solve an unsolvable problem?" An algorithm is a set of instructions for solving a problem, so students were asked to consider whether people could solve all their problems using computer-based algorithms.

Kalani has decided to investigate how well algorithms can solve the complex problem of efficient international parcel delivery.

Insight 1: Complex problems

I did some research and discovered that because of their complexity, some problems are intractable. In an ideal world, computers would always be able to use an algorithm to find the best solution to any problem within moments. However, in reality many problems are so complex that there are billions of possible solutions and a computer cannot solve these problems in a useful amount of time.

From my research, I determined that the complexity of a problem is often described using the Big O notation. This describes the performance of the algorithm in the worst-case scenario by using an expression of n, where n can be the number of objects within a problem or the number of steps a computer must take to solve the problem.

Insight 2: Problem-solving mechanisms

Computers do not approach problems in the same way as people. Because a computer doesn't have the ability to be 'spatially aware', it cannot approximate a solution in the same way we can. To solve a problem, a computer has to 'brute-force' its way through the problem by calculating every possible candidate for a solution.

'Brute-force searching' a problem will deliver the best solution, but it usually takes impractical amounts of time to do so. For example, in an algorithm set up to efficiently manage an international delivery system, for every new city added to the list of delivery addresses, the number of solutions increases by another order of magnitude, as there are n! (1 ×2 ×3 × ... n) solutions for a problem of size n (where n = the number of addresses in the system).

Insight 3: Application and impacts

I decided to see whether algorithms could improve efficiencies in the transport industry and have a positive impact by reducing emissions. For example, logistics and distribution companies regularly need to improve the efficiency of the algorithms they use for their deliveries. A multinational courier company delivering hundreds of thousands of mail items every day must manage its fleet of courier vans and aeroplanes so mail is delivered as quickly and cheaply as possible to the correct addresses. To do this, the routes the couriers take must be organised to minimise the total distance travelled.

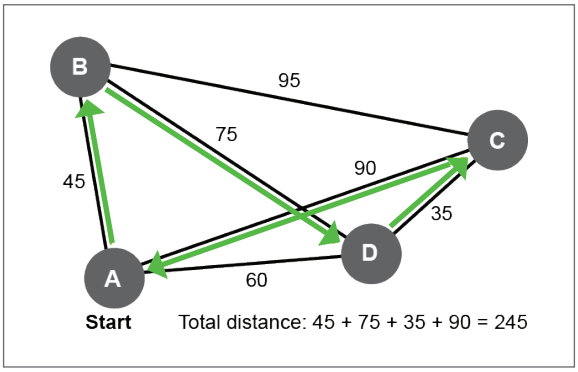
The less distance a courier travels, the lower the cost of fuel and wages and the more profit the company will make from its deliveries. An added benefit is that fewer fuel emissions will cause less harm to the environment. Any cost savings will compound over time into major increases in profit, putting the company ahead of its competitors. However, the courier company can't wait for a computer to take months or years to find the best route. At the same time, relying on human interpretation to find the best route is unlikely to be as beneficial as using a computer.

Insight 4: Solving the unsolvable

I researched this concept and learnt about heuristics, which in computer science are techniques designed to help computers solve problems efficiently and quickly. The "nearest neighbour" heuristic is commonly used to find a solution to complex problems. Nearest neighbour is a "greedy" algorithm, where the algorithm attempts to make the best possible choice at each stage, rather than looking for an overall best solution.

In the example below, a company has to send a courier from the depot (A) to three different houses (B, C and D) to deliver parcels and then return to the depot. Using the nearest neighbour heuristic, the total distance travelled is 245 units. This route appears to be efficient, but we don't know whether this is the optimal route because the other routes have not been calculated.

The nearest neighbour heuristic is one of the fastest algorithms for solving intractable problems like this, because at each step it simply calculates the distance from a point to all other points and chooses the point the minimum distance away.

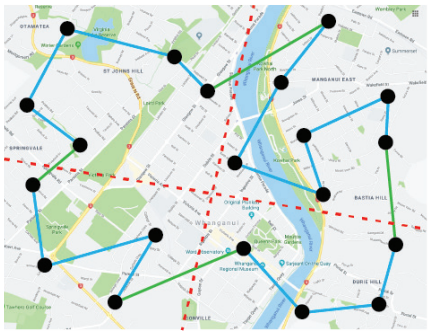


Insight 5: The wider implications

In real-world cases, typically more than one heuristic is used on a problem to ensure the best possible solution is found within a short amount of time.

For example, a company needs to deliver to the 20 addresses shown on this map. If it were to brute-force this problem, approximately 2.4×10^{18} obtained solutions would need to be calculated. This would take an impractical amount of time, and so the company needs to use a better way to find a solution.

In this image , the company has used the divide and conquer heuristic to divide the 20 addresses into smaller problems of more manageable size. The company can then brute-force these individual sections and put them back together. However, with a larger problem, brute-forcing could again run into time issues. In that case, the company could use other heuristics to find an efficient solution for each small sub-problem.



Once all the small solutions are joined together, the company has a final solution to its initial problem. This efficient and optimal problem-solving approach enables the company to deliver their parcels more quickly, decreasing their running costs and increasing profits.

This method of problem solving has implications for many kinds of organisations. It helps them reduce costs and solve ‘unsolvable’ problems, as well as benefitting the environment.

ISBN: 978-1-77669-239-2

MINISTRY OF EDUCATION
TE TĀHUHU O TE MĀTAURANGA

PROGRESS OUTCOME 7

Looking at cybersecurity

Context

Hamiora has been investigating the concept of encryption and its impact on society. He has learnt that encryption is an important aspect of computer science as it impacts upon all our online activities, including online shopping, banking and financial transactions.

Insight 1: The concept of encryption

Most people take cybersecurity for granted, yet few understand exactly how encryption works. I discovered that encryption is the conversion of data into a form that is unreadable to anyone except the intended recipients. This is important in our society because we all have private information that we don't want the world to see. This includes the things we expect to remain private, like our online searches and the messages we send our friends, but also the things we expect to remain secure, like our banking information when we use internet banking or the credentials for our Wi-Fi networks. Unfortunately, there are malicious people who break into individuals' systems for personal gain. Encryption helps protect against this.

Insight 2: An example of encryption

A simple example of encryption is an XOR cipher used on text, which is stored as binary using the ASCII encoding standard. This relies on a constant binary 'key' that is applied to the input. For simplicity, the binary key is exactly 8 bits long (the length of a byte).

The example below uses the string "Testing testing" (but in binary) as the input and the binary 00010111 as the key.

Input string:	Testing testing
Key:	00010111
Output of XOR cipher (gibberish):	Crdc~yp7crdc~yp

The XOR cipher works by applying the XOR binary operator to each bit of each byte. The binary operator takes two bits and returns true (represented as 1) if the two inputs are different, or false (represented as 0) if the two inputs are the same. One of the two bits is from the key, while the other is from the input string.

Insight 3: Key problems and issues

A major problem with using an XOR cipher on text it that it becomes vulnerable to frequency analysis. This is a technique that uses the known frequency of each letter in a language to guess which letter corresponds to each encrypted character. For example, since letters like e and t are far more common than z and q in the English language, it's highly probable that two of the characters that appear most commonly in an encrypted message are e or t. Using this information, a hacker can guess the code and decrypt the message, thus "cracking" the cipher.

Insight 4: The key distribution problem

Two forms of encryption ensure that only the intended recipient can read a message: symmetric cryptography and asymmetric cryptography. Symmetric cryptography relies on the sender and receiver sharing or exchanging a key securely. Most real-world systems use asymmetric cryptography, or public-key cryptography, which uses two keys instead of one. The first key (the "public key") encrypts a message and can be shared widely. The other key (the private key") decrypts the message and is only known to the recipient of the message.

With symmetric cryptography, the one key must be kept absolutely secret between the two parties. However, in asymmetric cryptography the public key used to encrypt messages can be shared widely without fear of it being used to decrypt messages, because decryption requires the private key.

Insight 5: How asymmetric cryptography is used

A popular form of asymmetric cryptography is RSA (named after its inventors: Rivest, Shamir and Adleman), which uses a public key and a private key. RSA is used on the internet to securely establish a connection between computers and websites. As RSA uses asymmetric cryptography, the website and the computer can send their public keys to each other, without having to worry about communicating secretly.

RSA encryption is used in many everyday internet tasks, including internet banking, checking email and browsing online. It uses what is known as a 'trapdoor calculation', where the key is created by multiplying two very large prime numbers. To crack the key, you would need to factorise the resulting number, which is an intractable problem that would take many years to solve.

ISBN: 978-1-77669-240-8

MINISTRY OF EDUCATION
TE TĀHUHU O TE MĀTAURANGA

PROGRESS OUTCOME 7

Ordering smoothies

Context

Jordan's class has been learning how to develop a computer program that uses a GUI (graphical user interface) and responds to users' inputs from button clicks, text input, or selections from drop-down menus or radio buttons.

The students have been given a scenario of developing an app for ordering smoothies from the school canteen. They have had to think about the design of the GUI and the program, and the testing needed to ensure the program functions accurately.

Insight 1: Designing the GUI

We asked the school canteen about the process when students and teachers order smoothies. They gave us the following information about smoothie choices:

- three dairy choices (customers can only pick one)
- three size choices (customers can only pick one)
- six mixture choices (customers can only pick one).

I decided that the GUI would need three radio buttons for the dairy choices, three radio buttons for size choices and six radio buttons for the mixture. There would also need to be a button for clearing the order, a button to submit the order and a text box to display the order on the screen. I sketched out the layout and realised that there was no place to enter a name for the order, so I added another text box for input.

Insight 2: Decisions about variables and collections

Looking at my design and thinking about how the program should function, I decided that I would need integer variables to store the choices for dairy, size, and mixture, and a string variable to store the name. I decided to store the different options for each choice in arrays so it would be easy to build the order. I could have built the order by hard-coding the output with a series of switch statements, but I thought this would take longer to code, which could introduce more errors and would be harder to update if the choices changed. Using arrays, I could loop through them to find the choices and build the order.

Insight 3: Documenting and testing

After I had used a form design tool to build my GUI and initialised my variables and collections, I decided to program and test each event to make sure it worked as I expected. As each event should work in a similar way, I thought it would be more efficient to have one functioning correctly before programming the rest. I made sure all my variable names were clear and used the camelCase convention we had been taught in class. I then wrote some pseudocode for the first radio button click event and used that for my code comments.

I began testing by selecting a dairy choice and checking whether the correct choice displayed in the order output on the screen. At first this didn't work as I expected, but then I remembered that arrays are 0-indexed and I had set my choices to start with 1. Finding this out early on enabled me to get the codes to work correctly for the other radio button click events.

Insight 4: Testing and debugging the order output

My order output had the correct information, but it wasn't readable because all the choices were joined together, such as in "Charlietrimlargebananarama". I had used string concatenation to build the output, but I had forgotten to add in carriage returns and newline commands. I added these commands and tested that the output was easier to read.

ISBN: 978-1-77669-241-5

MINISTRY OF EDUCATION
TE TĀHUHU O TE MĀTAURANGA