Learning Aims

Solving problems using:

- Backtracking
- Data mining
- Heuristics
- Performance modelling
- Visualisation
- Pipelining
- Divide and conquer

Introduction

The main computational methods to solve a problem follow the process of:

Problem Recognition – Decomposition - Abstraction
(pattern recognition and algorithmic thinking)

But there are some situations where other computational methods may be required…

Various Other Computational Methods:

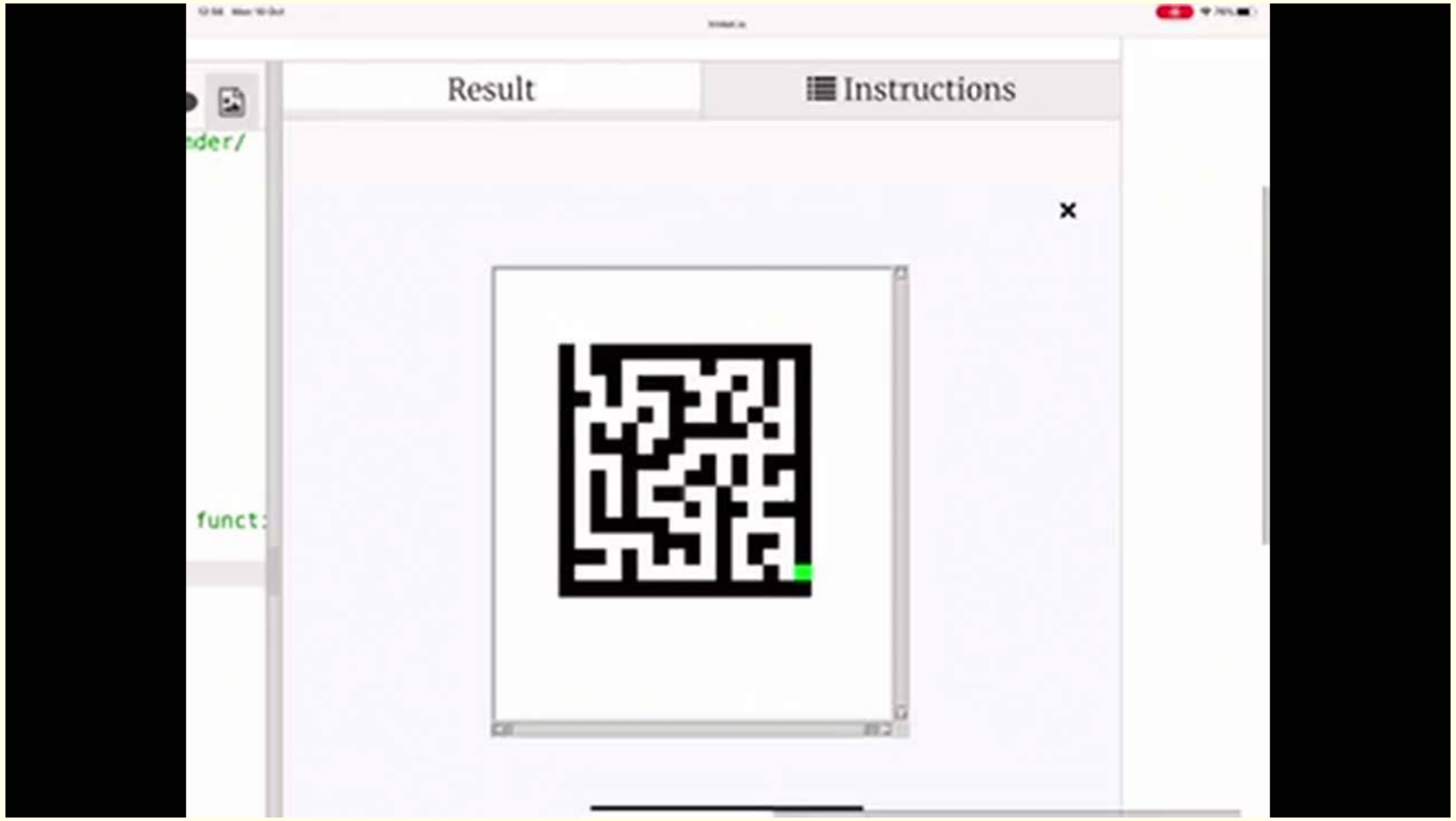
- data mining
- heuristics
- performance modelling
- pipelining
- visualisation to solve problems.
- backtracking

Backtracking

- ***Backtracking** can be defined as a general algorithmic technique that considers searching every possible combination in order to solve a computational problem.*
- Problem-solving technique implemented using algorithms , often **recursively**
- Best described as an "**Organised Brute Force**"
- It works by methodically visiting each path and building a solution based on the paths found to be correct .
- If a path is found to be invalid at any point, the algorithm backtracks to the previous stage and visits an alternate path.
- Backtracking can be used to solve mazes, journey planning, Magic Square Solver, sudoku …

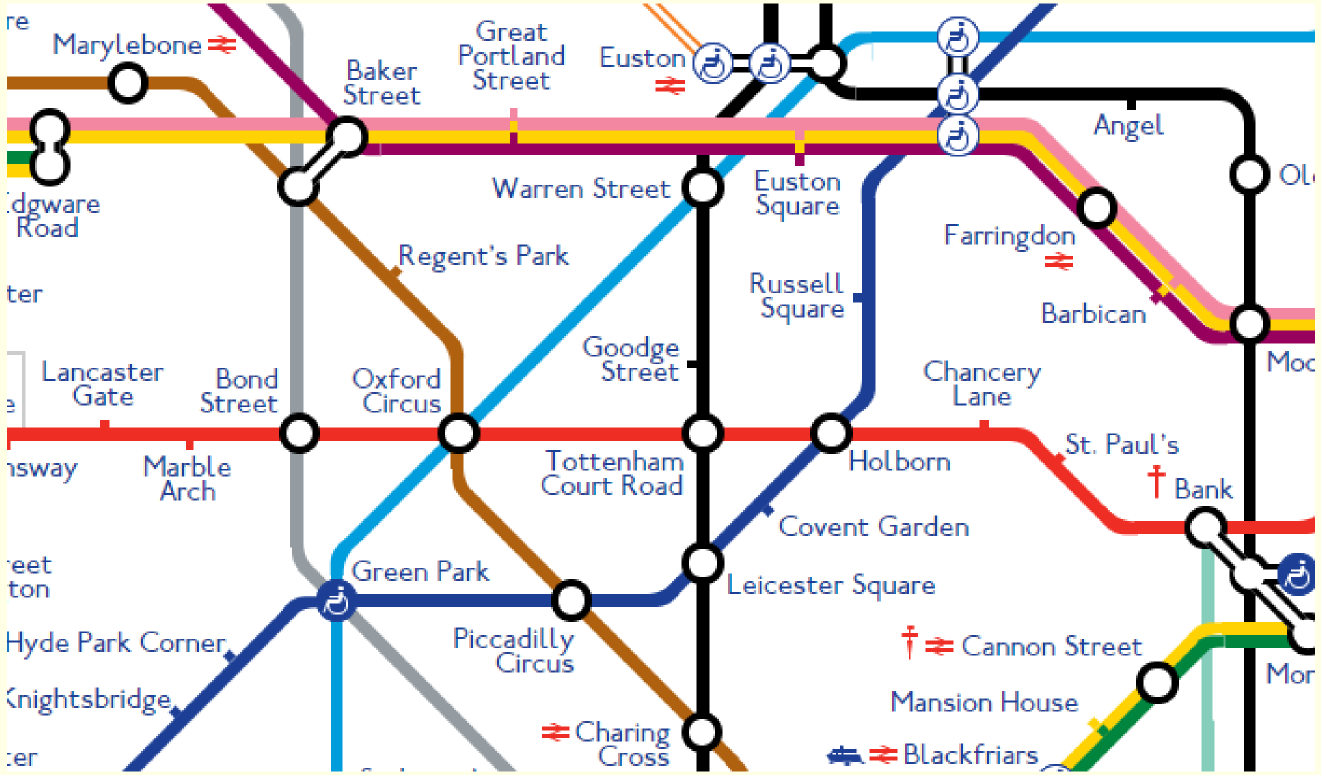
Example of backtracking - Maze

- A **maze** can be solved using backtracking by visiting each path and, if a path leads to a dead end, returning back to the most recent stage where there are a selection of paths to choose from. The maze can be represented as a graph and a Depth-first Graph traversal can be used to explore possible paths.



London Underground – Journey Planner

- A Map can be represented as a graph.
- Finding a route from start to destination can be found using backtracking
- An algorithm that uses a **backtracking approach** to explore different paths.
- It does stop as soon as a path has been found.
- Note that the path that will be found first may not be the shortest path though.

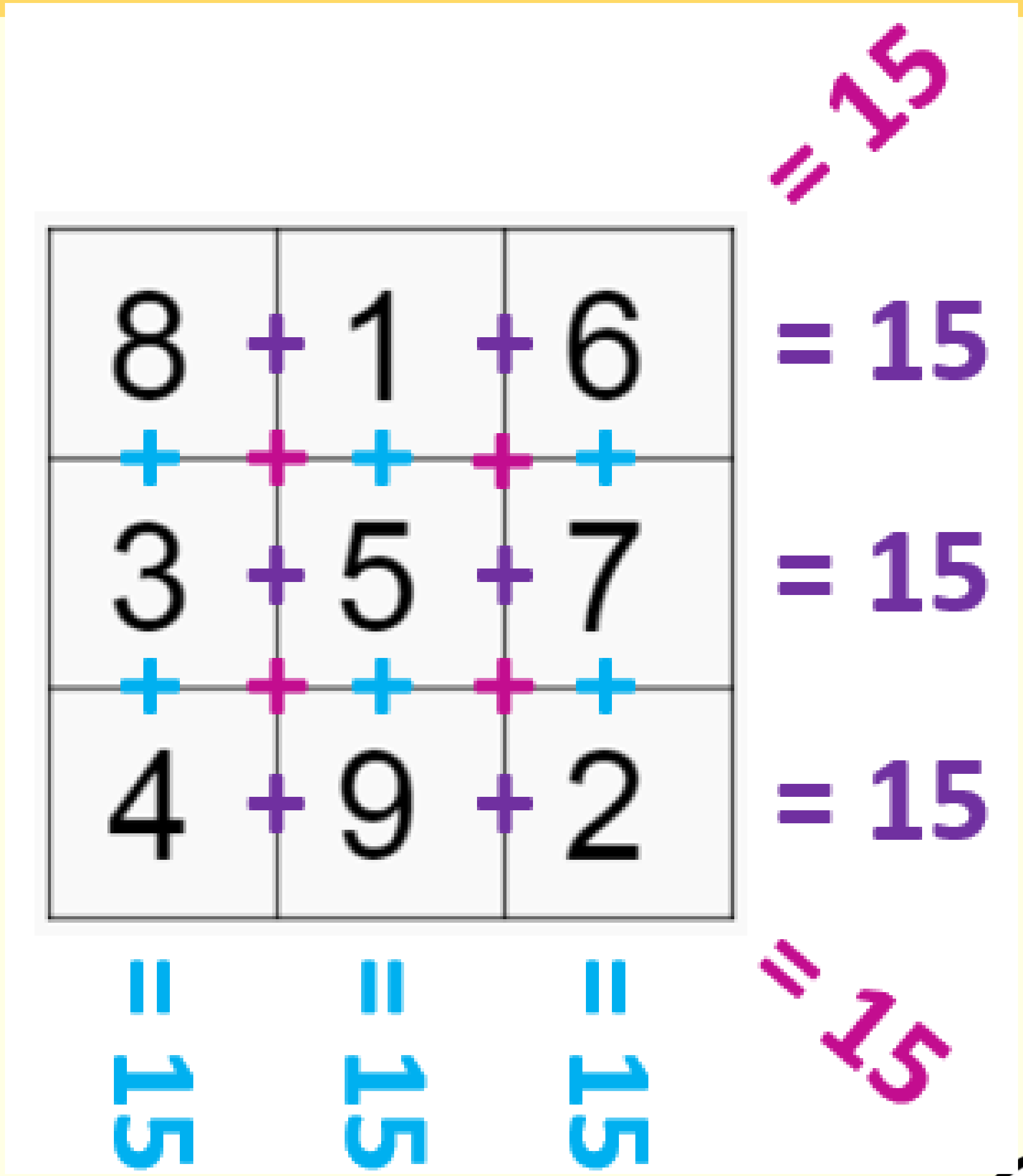


Magic Square Solver

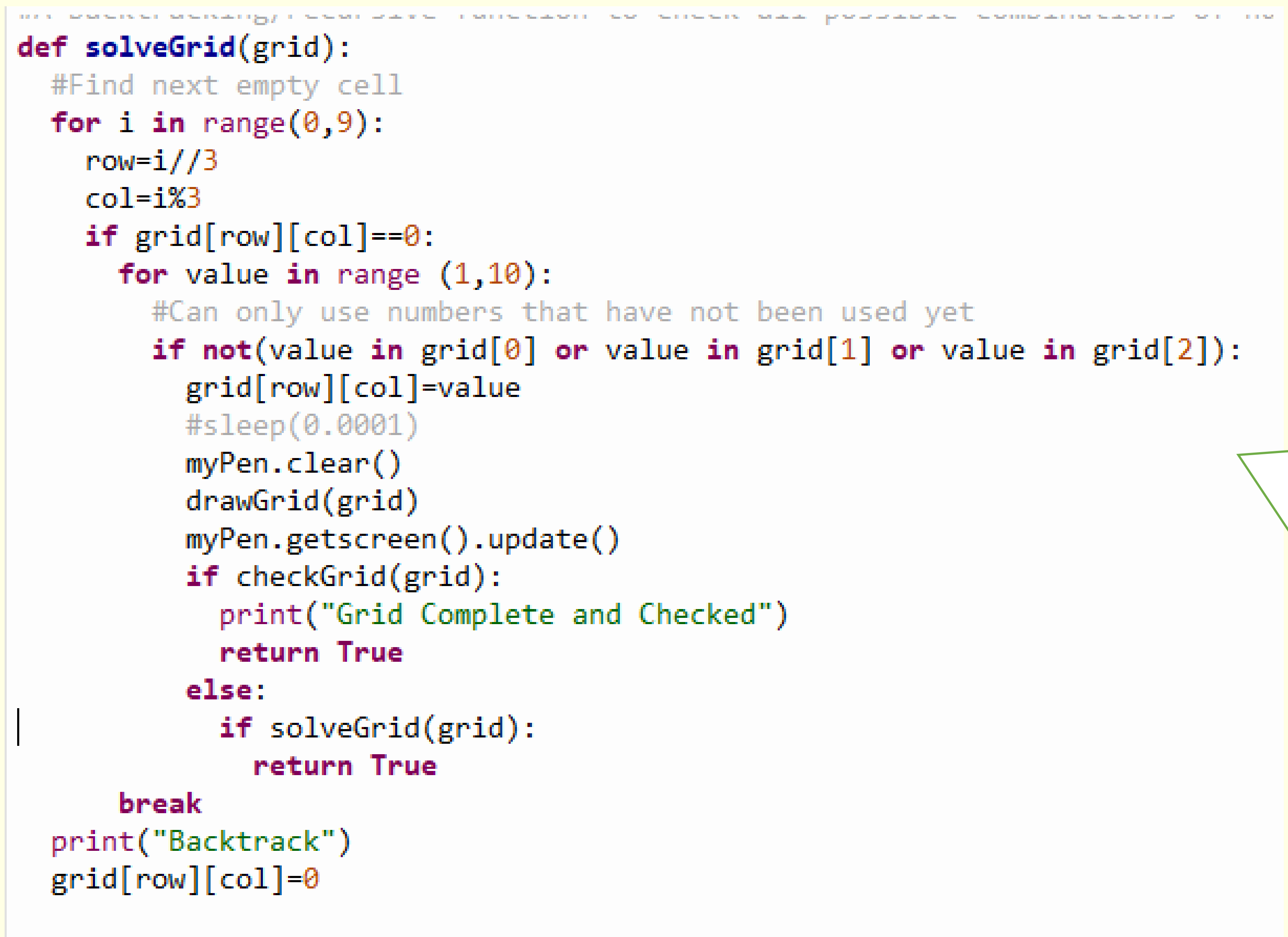
The purpose of this Python challenge is to demonstrate the use of a **backtracking algorithm** to solve a Magic Square puzzle.

Did You Know?
A 3×3 magic square is an arrangement of the numbers from 1 to 9 in a 3 by 3 grid, with each number occurring exactly once, and such that the sum of the entries of any row, any column, or any main diagonal is the same.

The algorithm works by backtracking/recursive function to check all possible combinations of numbers until a solution is found.



Algorithm



```
#A backtracking/recursive function to check all possible combinations of nu
def solveGrid(grid):
  #Find next empty cell
  for i in range(0,9):
    row=i//3
    col=i%3
    if grid[row][col]==0:
      for value in range (1,10):
        #Can only use numbers that have not been used yet
        if not(value in grid[0] or value in grid[1] or value in grid[2]):
          grid[row][col]=value
          #sleep(0.0001)
          myPen.clear()
          drawGrid(grid)
          myPen.getscreen().update()
          if checkGrid(grid):
            print("Grid Complete and Checked")
            return True
          else:
            if solveGrid(grid):
              return True
      break
  print("Backtrack")
  grid[row][col]=0
```

It can be solved by one by one assigning numbers to empty cells.

Before assigning a number, check whether it is safe to assign.

It will enter a value into the grid using a value that has not already been used in the row.

It then will checkGrid(grid) – every row and column is checked = 15

If False then it tries again using a different number

Data Mining

- Data mining is the process of analysing large sets of data to find **patterns, trends, relationships, or outliers** that are not immediately obvious.

Key points:

- Often used with **Big Data** and **machine learning** to interpret huge, complex datasets.
- Helps businesses predict future behaviour, such as which products sell at certain times of year.
- Retailers use it to understand **shopping habits**, often using loyalty card data.
- Can combine different data sources (e.g., sales + weather data) to improve decision-making.
- Useful for **crime prediction**, **virus checking**, **weather modelling**, **medical research**, and **market research**.
- Helps spot **errors or unusual data** that may indicate problems.
- Gives companies insights that guide stock management, marketing, and planning.

Positives and negatives of data mining

+ Can identify areas to focus attention

+ Save time and money by identifying areas that are not popular/used

+ New features targeted at specific groups

- But care would need to be applied to **privacy issues / GDPR** and potential impact on the users

Consider a social network that is used for listening to music. Some of the users' activities will be:

- Placing songs into playlists
- Rating songs, e.g. by using a five-star rating system
- Marking songs, playlists, or artists as 'favourites'
- Sharing songs or playlists with other users
- Becoming 'friends' or 'followers' of other users or artists

Using data mining, a company can interrogate the data that is gathered as a result of the above activities and find out the answers to the following questions:

- Which songs or artists are the most/least popular?
- Which features of the platform are used the most/least?
- What are the characteristics of the users (e.g. age)?
- Is there a correlation between the various types of users and their activity or preferences? For example, is there a genre of music that is very popular among young people?
- What is the level of engagement of the users? For example, are there any 'inactive' users that need to be re-engaged?

Heuristics

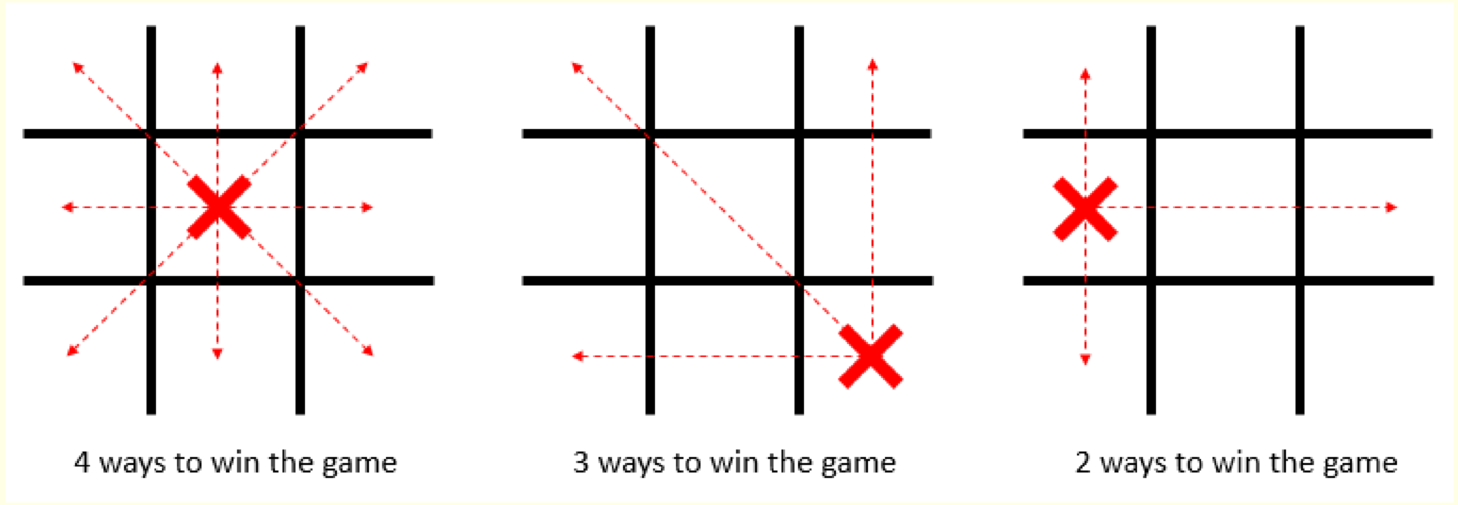
- Heuristics are **quick, rule-of-thumb methods** used to find an **approximate** solution when a perfect solution would take too long or use too many resources.

Key points:

- Not guaranteed to be perfect, but the solution is "**good enough**."
- Useful for **shortest-path problems**, **machine learning**, and **language recognition**.
- Often used when problems are **intractable** (take too long to solve exactly).
- Use **human insight** to shortcut the problem.
- Example: Instead of calculating every route between many cities, you can quickly rule out obviously long routes to estimate a shorter one.

Noughts & Crosses

- To help the computer make a decision as to where to place a token on a 3×3 noughts and crosses grid, a basic heuristic algorithm should be based on the rule of thumb that some cells of the grid are more likely to lead to a win:



Game of Chess

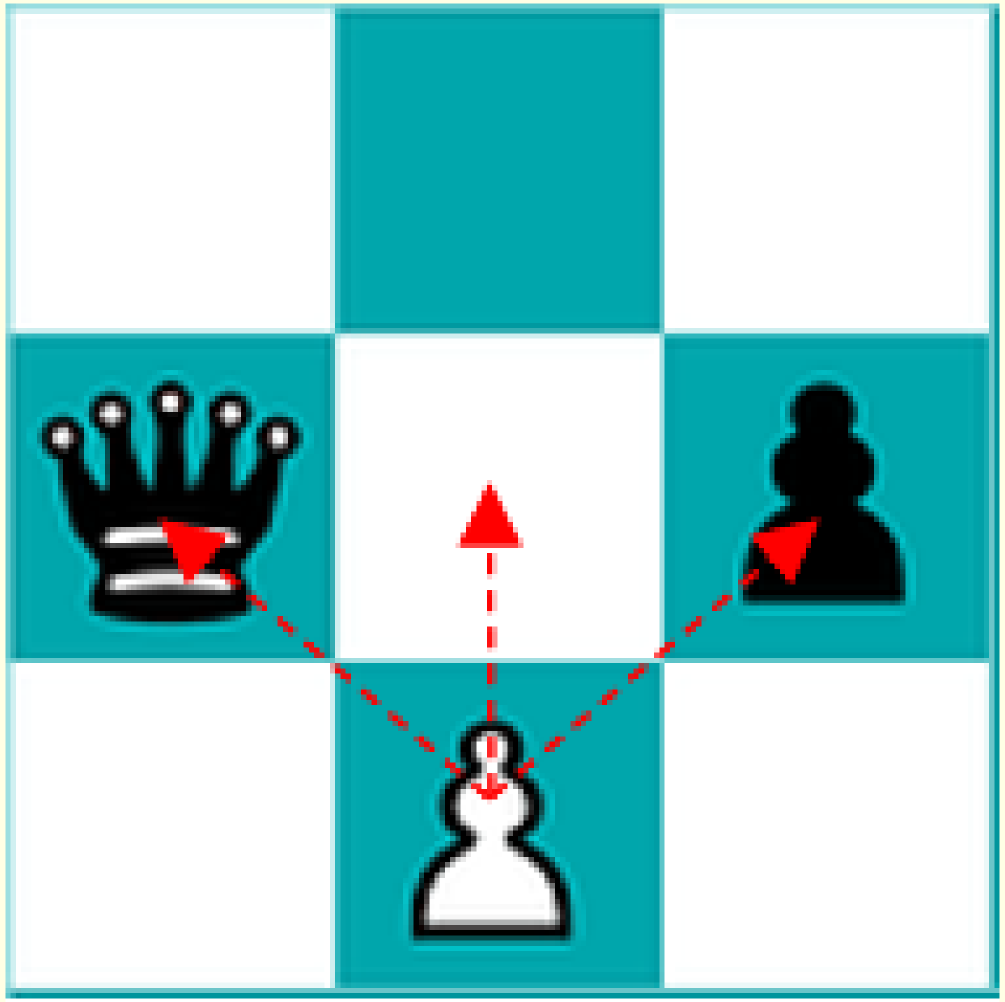
When playing a game of chess, expert players can "see" several moves ahead. It would be tempting to design an algorithm that could investigate every single possible move that players can make and investigate the impacts of such moves on the outcome of the game.

However such an algorithm would have to investigate far too many possible moves and would quickly become too slow and too demanding in terms of memory resources in order to perform effectively.

It is hence essential to use a heuristic approach to quickly discard some moves which would most likely lead to a defeat while focusing on moves that would seem to be a good step towards a win!

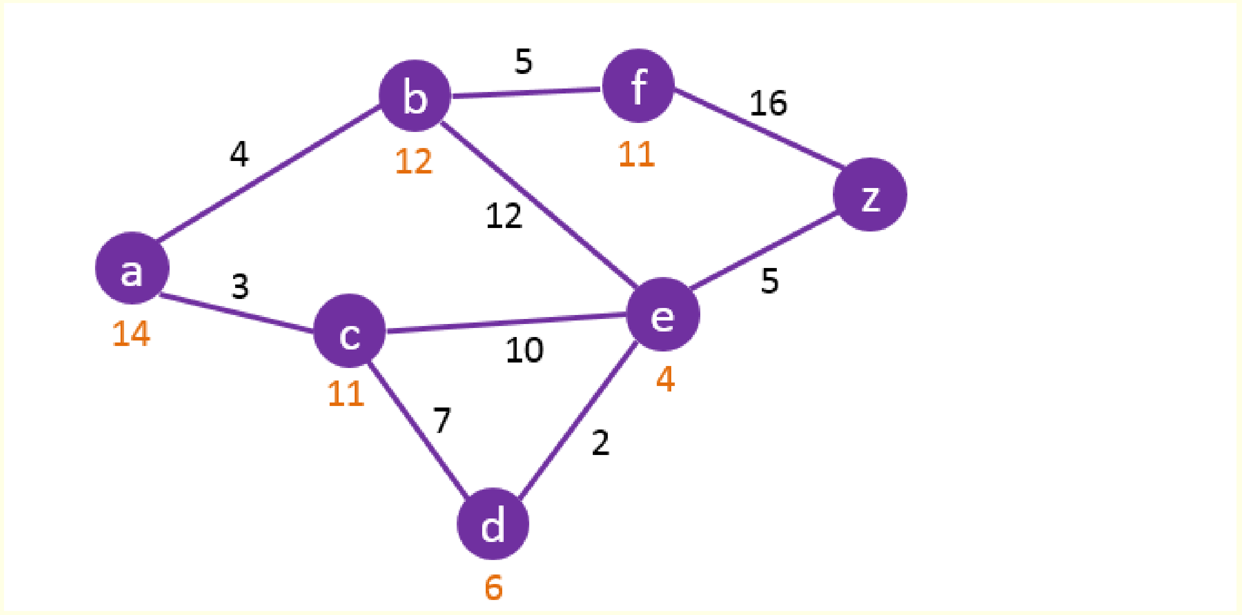
Let's consider the above scenario when investigating all the possible moves for this white pawn.

Can the computer make a quick decision as to what would most likely be the best option?



Shortest Path Algorithm

- used by GPS systems and self-driving cars also use a heuristic approach to decide on the best route to go from A to Z.
- This is for instance the case for the A* Search algorithm which takes into consideration the distance as the crow flies between two nodes to decide which paths to explore first and hence more effectively find the shortest path between two nodes.



Further reading …

Shortest Path Algorithms

Used by GPS systems and self-driving cars also use a heuristic approach to decide on the best route to go from A to Z (e.g. A* Search Algorithm). More advanced algorithms can also take into consideration a range of factors including the type of roads, the speed limits, live traffic data, etc. which can result in extremely complex algorithms./ A* path finding algorithms

AI When a computer algorithm plays a game of Chess (e.g. Deep Blue) or a game of Go (e.g. AlphaGo), the computer cannot investigate every single move that can be played. Instead it will apply a few rules of thumb to quickly discard some moves while focusing on key moves that are more likely to lead to a victory.

Language recognition

When analysing the meaning of a user input, for instance when a user asks a question on a search engine or interacts with a Google Home or Amazon Echo device, an algorithm tries to make sense of the user inputs. Word associations, analysis of context, previous searches history and current trends/search engine statistics can be used in a heuristic algorithm to speed up the search process.

Big Data Analysis

When there are large amounts of data to analyse. This is the case for search engines to return search results very efficiently, profiling algorithms which can be used by a marketing department to identify members of a target audience and their behaviours, data analysis in scientific research (e.g. medicine to identify cause/effect correlations between large data sets).

Machine Learning

Artificial Intelligence algorithms based on machine learning where the computer builds up a knowledge base from previous experiences is another application of heuristic algorithms. In this case the algorithm uses a self-maintained knowledge base to inform decisions and make "educated guesses" based on previous experiences.

Find out more https://www.101computing.net/heuristic-approaches-to-problem-solving/

Performance modelling

- Performance modelling uses **mathematical models and simulations** to test how a system will behave, without needing to test it in real life.
- Key points:
- Cheaper, faster, and safer than real-world testing.
- Important for **safety-critical systems**, such as those in aircraft.
- Helps companies judge whether a system is capable, reliable, and safe before implementation.
- Uses realistic data to model real-world conditions.
- Example: Simulating a new traffic-light system on a computer instead of testing it on a real roundabout, preventing congestion and safety risks.

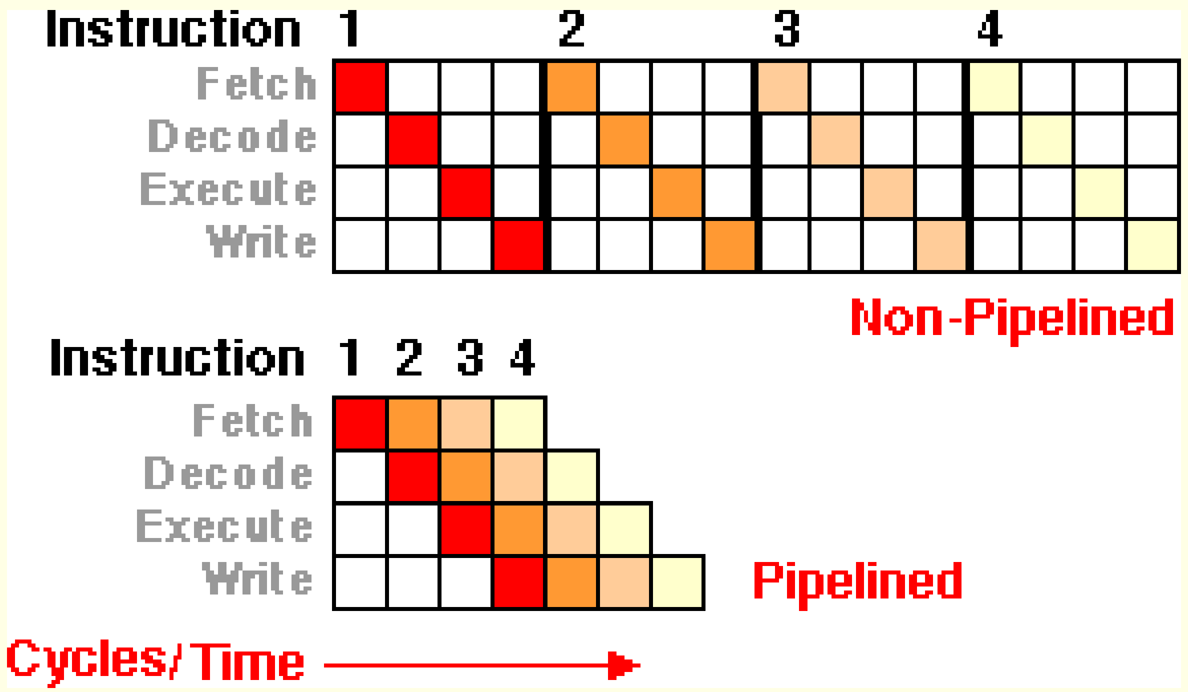
Examples

Simulations – flight simulation, re-routing trains in the London Underground

Big-O notation – algorithm complexity for time efficiency and memory space

Pipelining

- Process of taking a task and splitting it up into smaller tasks and then overlapping the processing of each sub task to speed up the overall process
- Commonly used in RISC processors : different sections of the Fetch-Decode-Execute cycle are performed simultaneously



Computational Methods

Pipelining

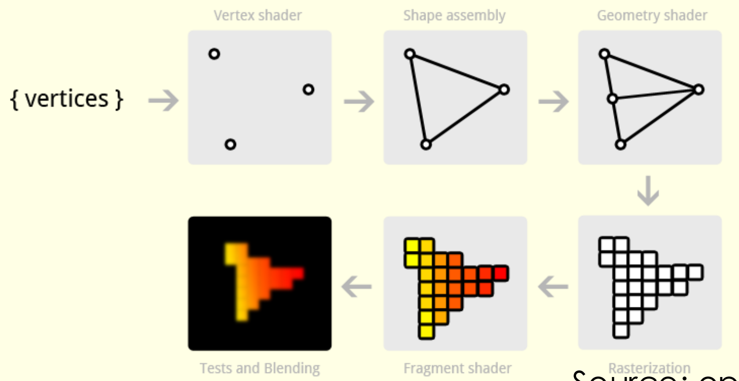
Pipelining has many uses in computer systems.

For example, **instruction pipelining** is used in CPUs to speed up processing time.

As one instruction is executed, the next can be decoded and the next can be fetched and in this way, instructions can be queued up reducing processing time.

Software pipelining allows the output of one program to be in the input of the next.

Graphics pipelining allows various graphics to be queued up for the various stages of graphic rendering.



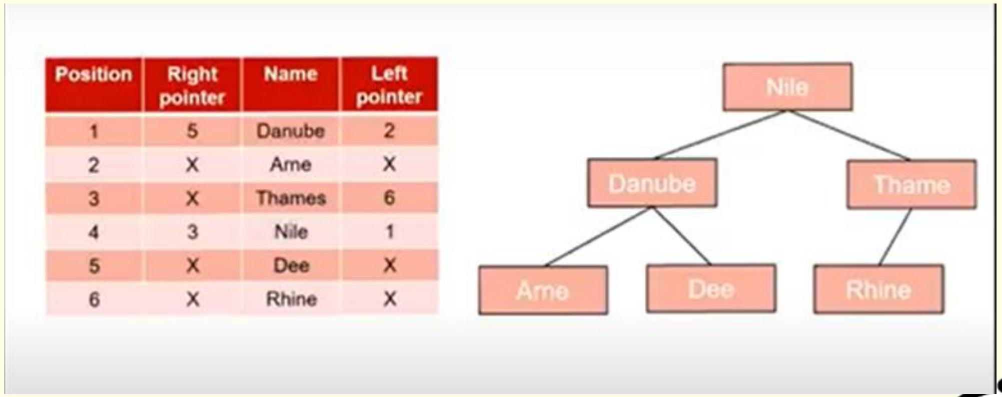
Source: open.gl

Visualisation

- Data can be shown in different ways to make it easier to understand and help solve problems. Depending on the type of data, it might be displayed as **graphs, charts, tables, or trees**.
- **Visualisation** is a method where computer scientists turn data into pictures or graphics so patterns and trends are easier to spot.

- Large sets of complex data can be hard to interpret in number form, but visual displays allow us to quickly see important information that might otherwise be missed.

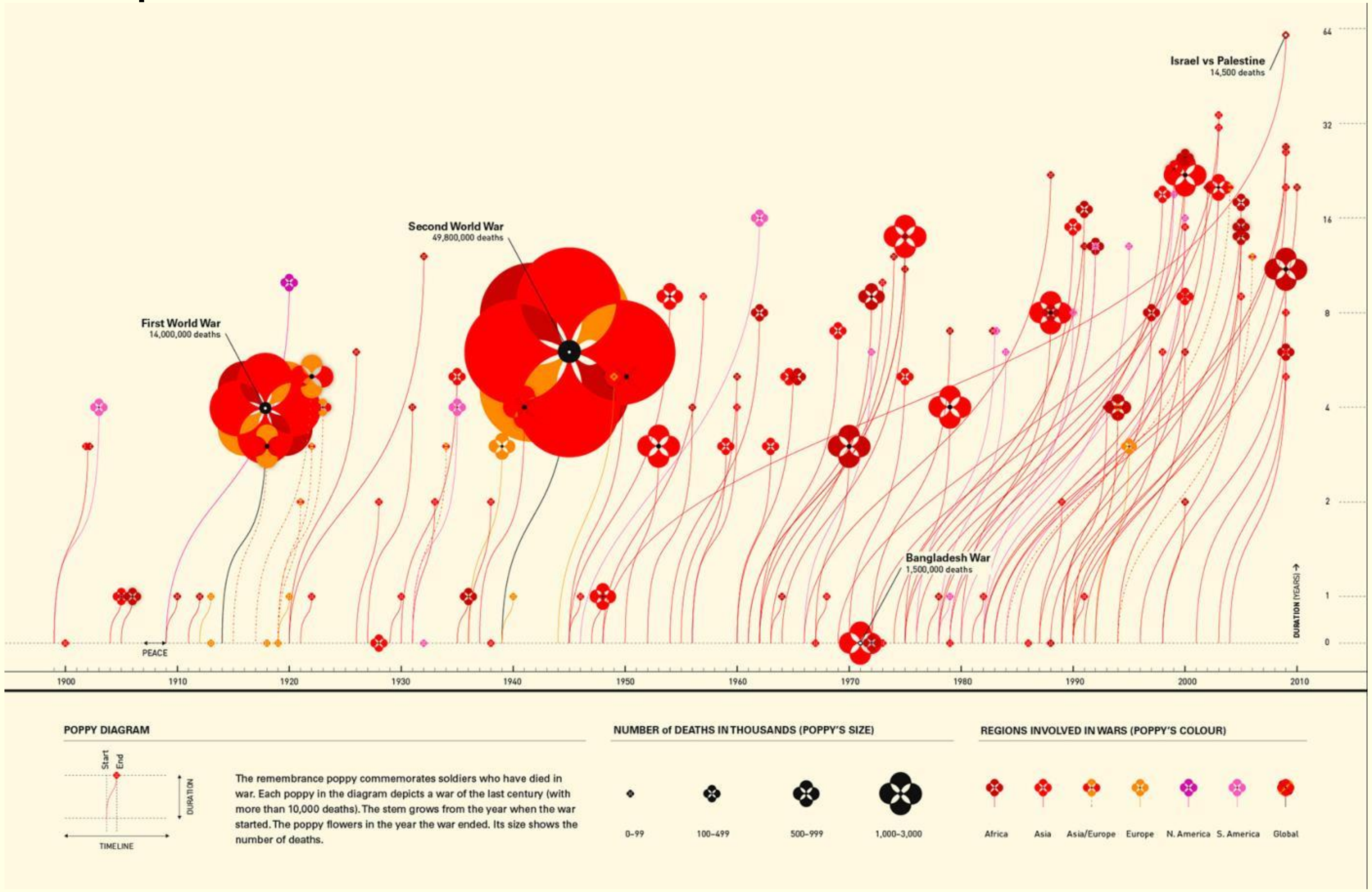
- Flow diagrams to represent an algorithm



Position	Right pointer	Name	Left pointer
1	5	Danube	2
2	X	Arne	X
3	X	Thames	6
4	3	Nile	1
5	X	Dee	X
6	X	Rhine	X

Computational Methods

Visualisation – An Example



The remembrance poppy commemorates soldiers who have died in war. Each poppy in the diagram depicts a war of the last century (with more than 10,000 deaths). The stem grows from the year when the war started. The poppy flowers in the year the war ended. Its size shows the number of deaths.

Source: www.datafloq.com

Examples

- Flow charts
- Table to represent a 2D array
- Binary Trees

Divide and conquer.

- Common divide and conquer algorithms are **binary search, merge and quick sort**
- The biggest advantage of using divide and conquer to solve problems is that the size of the problem is halved with each iteration which greatly simplifies very complex problems .
- This means that as the size of a problem grows, the time taken to solve it will not grow as significantly.
- For example, the number of recursive calls made by a function halves with each iteration.
- Therefore, the time complexity of algorithms that use divide and conquer is O(log n) .
- However, As divide and conquer mostly makes use of recursion, it faces the same problems that all recursive functions face: stack overflow will cause the program to crash and large programs are very difficult to trace .