Unit 2 Algorithms and Problem Solving

2.0 Overview

In this unit we are going to:

- investigate the types of problems that are suitable for computer solution
- find out what an algorithm is
- look at using pseudocode
- establish a systematic problem solving method
- discuss the types of error that can occur in a computer program
- discuss the of copyright of programs and the rights and responsibilities associated with writing and owning software
- look at how mp3s and open source software are changing the face of copyright.

Keywords:

algorithm	analysis	assignment
call	circumvention measure	coding
commercial software	compression algorithm	copyright
cracker	definition	design
digital agenda	download	encrypt
ethics	evaluation	external documentation
first digitisation	heuristics	implementation
infringement	intellectual property	intent error
interface	internal documentation	license
look and feel	manual	module
mp3	on-line help	open source software
patent	pirated	procedure
pseudocode	public domain	reverse engineer
right of communication	RMI	royalties
runtime error	shareware	skip
software development cycle	software house	specification
sub-routine	syntax	technological protection
technology neutral	trace	trade secret
unambiguous	user	user proof
variable	work for hire	

2.1 Computer suitable problems

A computer program is just a series of steps that have been arranged in the correct order to complete a process. Usually the process is to solve a problem (e.g. what are my total sales this year, how do I get this image to display, etc.). To write programs we are going to have to find out how to solve problems that have a possible step by step solution.

Solving problems

Anyone who has attempted to solve a problem (e.g. in Maths) knows it is not easy to know what to do, where to start, or even if the problem has a solution. Some people are good problem solvers, others are not.

When we have a problem most of us want to dive in and have a go at solving it without really knowing what we are doing. This method is called *trial and error*. We try the first thing that

occurs to us; if it does not work we will try something different, and keep going like this until we hit on the right answer, or give up. This method will work with simple problems but it is inefficient. It will not work with larger problems. There are more systematic ways that can be used.

The *analytical approach* is systematic way to solve problems. This method can be used where a decision has to be reached. All the advantages and disadvantages of each alternative are listed and then weighed against each other. An example might involve deciding to buy a new car or not. The advantages (reliability, prestige, comfort) and the disadvantages (cost, depreciation, etc.) are determined and the decision is made on the side that carries the most weight.

Working backwards from a goal is a another methodical way of finding a solution that is used for example by code breakers or computer hackers. They know what they want to achieve and systematically de-construct what is in place. Reverse engineering is a form of this method.

Sometimes we do not even work logically to solve a problem.

The *'Eureka Experience'* is used to describe the feeling we get when the answer to a difficult problem suddenly occurs to us. Archimedes supposedly jumped out of his bath yelling "*Eureka!"* when the solution of how to detect the presence of lead in a gold crown occurred to him. August Kekule, after months unsuccessfully trying to work out the structure of benzene, dreamed of snakes. He woke up when one of the twisting snakes grasped its own tail, and suddenly realised a ring like structure would solve the puzzle.

In these cases the problem solver has reached the solution unconsciously. The ability of the human mind to be creative, to initiate an original idea to solve a problem is unique, something that may never be duplicated in a computer.

Humans can also make *judgements* to solve problems. (How could a computer ever decide if a landscape painting is beautiful, or make a value judgement such as *is he evil?*) Using our past experience and what we have learned we can compare the evidence with what is expected and decide if it fits the requirements. From this we evaluate the criteria and make a judgement.

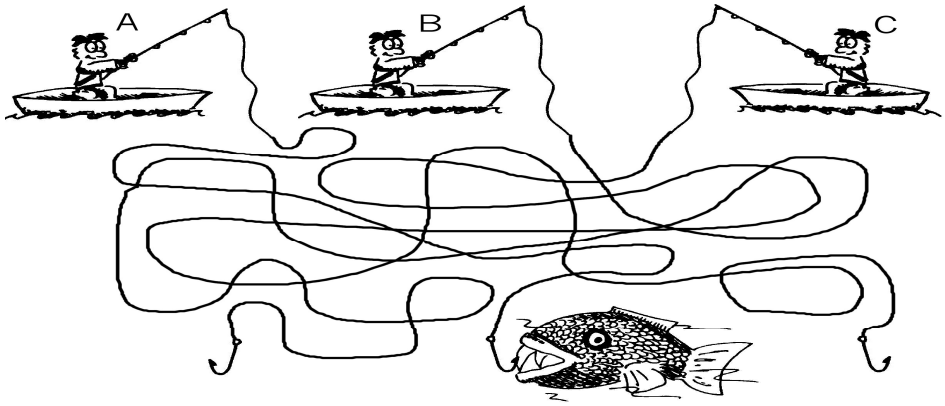
All of these problem solving methods of trial and error, creative or intuitive insight or evaluative judgements will work in their own area. However in solving computer problems something more methodical and systematic is needed. If we have no plan of where we are going in a program we will get confused, or leave out something, or even if we do get it right, will find it has taken us three times as long to get a solution.

Algorithmic vs heuristic methods

Generally computer suitable problems fit into the category where an answer (*output*) has to be produced, usually after some data has been supplied (*input*), and the method of solving the problem has to be determined (*process*). The human problem solver takes the problem, analyses it and plans a solution. The plan for solving the problem is called an *algorithm*. The computer is then coded from the algorithm to produce an answer or solution to the problem.

Computers excel in solving problems of the type that can be expressed as a series of steps. Humans on the other hand are better at solving problems heuristically. *Heuristics* are all the bits and pieces of experience we have stored away throughout our lives ready to be called upon to solve a problem.

For example if faced with the following puzzle can you find not only which fisherman has caught the fish, but also the quickest way of detecting him?:



When you were younger you might have spent time tracing each line down from the boats, but based on past experience you now probably realise it is quicker to trace from the fish up to the boat. This is an example of a heuristic solution, one based on the experience of having come across similar problems in the past.

A story may explain the difference between a heuristic and an algorithmic approach to problem solving:

> *Mike while visiting his aunt had gone out to see a movie. When he came out it was dark, and he could not remember how to get back to his aunt's house. Fortunately he remembered she lived near the beach. Looking at the road in front of him he saw it sloped from left to right. Assuming most roads near the coast sloped down toward the sea he followed it. At the end of it he chose the most downward sloping road, and so on. Eventually he reached the coast and soon his (by then angry) aunt.*

This is an example of a heuristic solution to a problem. Mike had no clear idea of how to get to his destination, but calling upon his past experiences he was able to come up with a workable method of getting closer to his aunt's.

Now imagine a robot was given the same problem of getting to a destination when the way was not known. Programming the automaton would involve exploring all the possibilities and combinations of moving to the end of a road, of turning left and right etc., and checking if the destination had been found. Eventually by investigating every possibility the algorithm would get the robot to the right place.

Some problems are best solved algorithmically, some heuristically. Humans (i.e. you) usually think heuristically. Computers on the other hand work algorithmically. The biggest problem you are going to have with learning programming is in having to discipline yourself to an algorithmic approach.

Activity 2.1 – Problem solving

1. Which of the following problems do think would be suitable for algorithmic solution, and which for heuristic solution?:
   - a calculating sales tax
   - b choosing the winner of a beauty contest
   - c establishing the surface area of a sphere
   - d calculating the area of a wall

e determining the meaning of life

f choosing the most likely winner of the Melbourne Cup

g finding out if a painting is fake.

2. a What types of problem are best solved by computer?

b What types of problem are best solved by people?

c Give two examples (of your own) of each type of problem.

3. Heuristics can also be described as rules of thumb, or ways you have found of doing things that have worked in the past. You may have experimented before and found a way that works for you, so when you come to a similar problem you apply these "rules of thumb" and develop a solution. You have been using heuristics.

   What are your heuristics for playing the game noughts and crosses (tic-tac-toe)? You might need to play several games with a friend to explore how you actually do play this game.

4. Imagine you are a TV executive producer and you have to plan a typical evening's entertainment. You can choose from the following available programs:

   - *The Tender Trap* (USA; $15 000; situation comedy; 24 min; rating .20)
   - *News and Weather* (Aust; $25 000; 22 min; rating .45)
   - *Perspective* (UK; $5 000; documentary; 54 min; rating .14)
   - *Funny Money* (Aust; $25 000; game show; 23 min; rating .62)
   - *What's New* (Aust; $18 000; current affairs; 24 min; rating .33)
   - *Midweek Movie* (USA; $1 500: various; approx. 82 min; rating .18)
   - *George's Dream* (UK; $4 000; situation comedy; 25 min; rating .16)
   - *Who Knows?* (Aust; $1 500; discussion programme; 55 min; rating .06)
   - *Cricket* (Aust; $55 000; live coverage; 5.30 - 8.30 only; rating .18)
   - *Going, Gone* (UK; $3 000; antiques show; 25 min; rating .06)
   - *Home Ties* (Aust; $30 000; soapie; 54 min; rating .56)
   - *Alive Oh!* (USA; $26 000; situation comedy; 21 min; rating .43)

   The duration of the program indicates how many adverts can be fitted in e.g. a 24 min program will allow 6 min of adverts in a half hour time slot. The ratings indicate how popular the program is – for example **.**45 means 45% of people watching at that time are tuned into this program.

   As producer you have some constraints on what you can do:

   - your time slot to fill is 5 p.m. to 10 p.m.
   - your budget is $130 000 (less if possible)
   - advertising revenue is highest during peak viewing times of 6 - 8 p.m.
   - you must maintain at least 50% Australian content of programs.

   Prepare the evening viewing achedule.

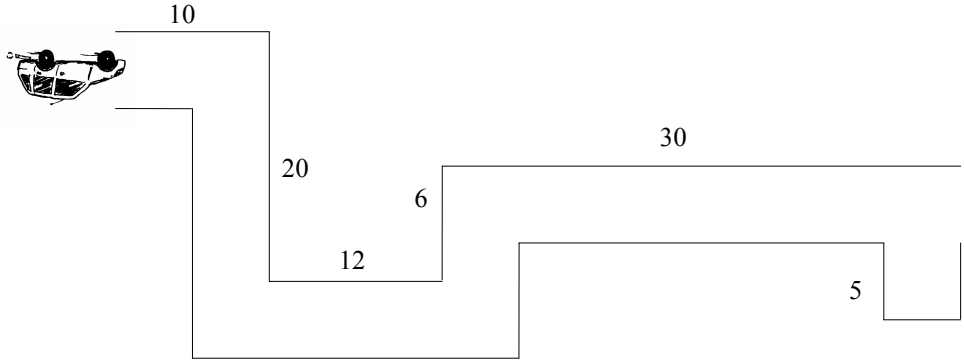
5. Your younger brother has a programmable car that understands the following instructions:

   left right forward x cm back x cm

   e.g. forward 10 cm
   left

   would move it forward 10 cm and then turn it to the left.

Program it to follow the following course and then reverse into the garage at the end:



6. Henri the new French chef has purchased a programmable microwave oven but is having a problem getting it to work. The instruction book lists six Pascal commands: on off high x min med x min low x min wait x min (The *wait* command is to let food stand to allow the heat to penetrate.)

   e.g. To scramble eggs, cooked on high for 5 min the program is:

   ```
   on
   high 5 min
   off
   ```

   a Describe to Henri what the following program would do:

   ```
   on
   high 2 min
   off
   wait 2 min
   on
   low 3 min
   off
   ```

   b Write a program for Henri to cook a frozen steak to the following requirements:

   defrost on low for 10 min, then cook on high for 7 min.

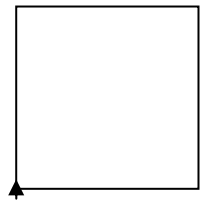
   c Write a program for Henri to cook a souffle to the following requirements:

   cook on high for 3 min, stand for 2 min, cook on low for 2 min, stand for 1 min, cook on high for 2 min.

7. Logo is a simple programming language developed by Seymour Papert to introduce children to programming and problem solving. Logo uses simple commands (similar to the car in Q5) to move a 'Turtle' around the floor. Computerised versions employ a triangle on screen to represent the turtle. The turtle is equipped with a pen to draw lines as it moves.

   The following commands would draw a square:

   ```
   pendown
   forward 100
   right 90
   forward 100
   right 90
   forward 100
   right 90
   forward 100
   right 90
   penup
   ```

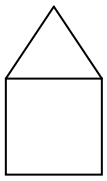


The *forward* command moves it forward 100 spaces and the *right* command turns the turtle right the given number of degrees (here 90^o).

For simplicity, a repeat command can be used to avoid writing the same lines over again.

e.g.
```
pendown
repeat 4(forward 100, right 90)
penup
```

would have the same effect to draw the square.



a Write a Logo program to draw an equilateral triangle. (Care: it is not as obvious as it seems.)

b Write a Logo program to draw a simple house.

8. You have just been appointed managing director of the Hitz Music Company. This is an old company that was very inefficient. To save the company from bankruptcy you will need to make many changes, including introducing computers.

You draw up the following list of some of the things you must do:

- select an assistant to help you
- stock take
- choose a new computer setup
- reorganise the management
- select which are the workers to be replaced
- establish a set of records of employees
- promote the company through advertising.

a List the tasks that can be accomplished using a computer to assist you, saying how the computer would be used.

b List the tasks that can only be done by a person, saying why a computer could not be used.

2.2 Algorithms

A mathematician living in Baghdad over 1100 years ago wrote a book that's title starts "*Kitab al-jabr...*". From this we get our word *algebra*. The man was named *al-Khowarizmi* in Arabic but this was converted into *Algorismus* in the Latin used by academics of the day. From his name, in turn, we get our word *algorithm*. (Don't let this put you off, algorithms are a lot more fun, and a lot easier than algebra.)

An algorithm is a list of the steps to follow to solve a problem, or to complete a process. Algorithms need to be unambiguous (clear), finite (will end), and general (can be used in a variety of situations).

Here is an algorithm for boiling an egg:

Boiling an egg
put water in a pan
put pan on stove
turn on the stove
wait till water boils
place egg in water
wait 5 minutes
remove egg from water
turn off stove.