1. Module number	*SET09122*
2. Module title	*Artificial Intelligence*
3. Module leader	*Ben Paechter*
4. Tutor with responsibility for this Assessment Student's first point of contact	*Ben Paechter*
5. Assessment	*Report and Program*
6. Weighting	*50% of overall module total:*
7. Size and/or time limits for assessment	*None*
8. Deadline of submission Your attention is drawn to the penalties for late submission	Permission to submit program required to be obtained at practical classes 1-21 November. Midnight 3rd December 2019: Final submission on Moodle.
9. Arrangements for submission	Moodle

10. Assessment Regulations All assessments are subject to the University Regulations.	
11. The requirements for the assessment	*Please see attached document*
12. Special instructions	*See attached document*
13. Return of work	*within 3 weeks of submission.*
14. Assessment criteria	*See attached document* *Normal academic conventions for acknowledging sources should be followed.*

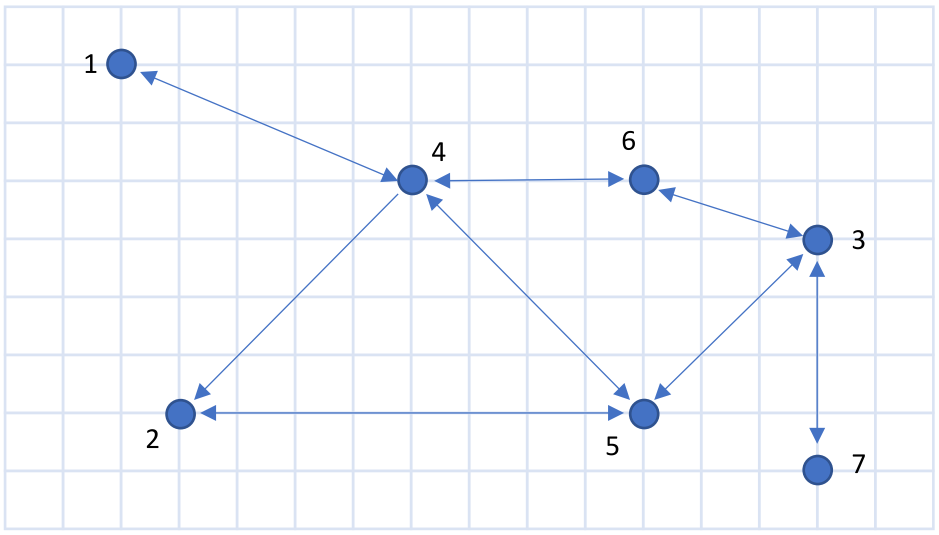
Artificial Intelligence

Coursework

Application

A robot has to navigate through a series of small underground caverns connected by straight tunnels. Some tunnels can only be navigated in one direction. The robot is given a map of the caverns and tunnels which is given as the coordinates of the centre of each cavern, plus a binary matrix showing which caverns can be reached from which other caverns.

For example, the following map:



is represented by the following coordinates for caverns:

(2,8) (3,2) (14,5) (7,6) (11,2) (11,6) (14,1)

and the following matrix to showing the connections:

From

To

	1	2	3	4	5	6	7
1	0	0	0	1	0	0	0
2	0	0	0	1	1	0	0
3	0	0	0	0	1	1	1
4	1	0	0	0	1	1	0
5	0	1	1	1	0	0	0
6	0	0	1	1	0	0	0
7	0	0	1	0	0	0	0

The connection matrix is given in the same order as the coordinates.

The task of the robot is always to navigate from the first cavern in the list to the last cavern in the list – or to identify that the route isn't possible

The distance between any two caverns is the Euclidean distance between the two coordinates:

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Input File format

Cavern maps are stored in .cav files which take the following format:

The file is a text file which contains a series of integers separated by commas.

The first integer gives the number of caverns - N.

The next N*2 integers give the coordinates of each of the caverns – each value is non-negative.

The final N*N integers give the connectivity of the tunnels. 1 means connected, 0 means not connected. Remember that some tunnel are one-way.

The order of the connectivity matrix is a follows:

Connectivity of Cavern 1 to Cavern 1

Connectivity of Cavern 2 to Cavern 1

Connectivity of Cavern 3 to Cavern 1

Connectivity of Cavern 4 to Cavern 1

Connectivity of Cavern 5 to Cavern 1

.

.

.

Connectivity of Cavern 1 to Cavern 2

Connectivity of Cavern 2 to Cavern 2

Connectivity of Cavern 3 to Cavern 2

.

.

.

So the file for the above example would be:

7,2,8,3,2,14,5,7,6,11,2,11,6,14,1,0,0,0,1,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,1,1,1,0,0,0,1,1,0,0,1,1,1,0,0,0,0,0,1,1,0,0,0,0,0,1,0,0,0,0

Output File Format

Routes through caverns are stored in the output file by having a series of integers denoting the order of visiting the caves. Caves are numbered starting at 1 – as in the diagrams above. The integers should be separated by spaces.

For example, the output from the file above might be:

1 4 2 5 3 7

Because you must always start at the first cave in the input file and end at the last cave in the input file, where a route has been found, the first number will always be 1 and the last number will always be the number of the last cave.

Where the algorithm identifies that no route can be found, it should place a single 0 in the file.

What to do:

1. **(12 Marks – maximum two pages)** Write a description of Dijkstra's pathfinding algorithm with enough detail for someone to be able to implement it to solve the robot/cave problem described. Marks will be given for diagrams which help understand the text, and particularly where the diagram is directly relevant to the problem being solved. You can use diagrams from the web, but these will not attract as many marks if they are not directly relevant to your text and the problem. Include an evaluation of the algorithm explaining the situations where it is more or less useful.
2. **(13 Marks – maximum one page)** Identify by name (but don't describe) a good algorithm to use for solving the given problem (even if you didn't use it for your program below) and explain why it is good for this particular problem. If the problem were to change so that the cost of taking a particular tunnel in a particular direction is given in the file – and is no-longer related to the distance travelled, explain how this might affect your choice of algorithm and identify any problem that might occur in this situations.

3. **(25 marks)** Write a computer program which runs on the windows command line with the parameter <file> which opens a cave map file called <file>.cav with up to 10,000 caverns and finds a path from the first cavern to the last cavern writing this to a solution file <file>.csn

   Your final program should not write any information to the screen (although you may wish to do this when developing the software). You can choose any language that can produce an executable to be run from the command line as above. Note that some languages will produce code than runs faster than others.

   You must then adapt the windows batch file provided on Moodle so that when

   caveroute <file> is typed on the command line your program executes reading in <file>.cav and outputting <file>.csn and the time taken is displayed.

   The main aim of the program is to find the shortest route possible.

   A secondary aim is to find the route as quickly as possible, by use of an efficient algorithm. Your program must run in less than a minute on each files provided on the machines in D2.

   Note that the time taken may vary depending out what else is happening of the machine and how much of the program is in memory. To get accurate results you should repeat the procedure and ensure that nothing resource intensive is running on the machine. On small files the time taken will not be measured accurately as it will be too short.

   Important – Permission to Submit In order to submit the program (rather than the alternative with lower possible marks) you **MUST** obtain a *permission to submit slip.* These can be obtained during the practical classes between 1 November and 21 November. In order to obtain the permission slip you must bring a USB storage device containing a zip file in the right format for the hand-in. You must be able to show that you have a program which runs in the method specified and reads and writes files in the correct format. The routes produced need not be correct or optimal – but you will receive feedback on this.

Program Marking Scheme (Total Marks 25)

Your programs will each be tested on unseen problem instances of differing sizes. Each submission will be scored on each instance. If a program takes more than a minute to run on an instance it will be stopped and no mark will be awarded for that instance (instances will be chosen to make this a reasonable limit).

There will be 10 unseen problem instances and scores will be awarded as follows.

Best route found or *no-route-possible* correctly identified : 2 marks
Valid route found: 1 marks
No route found where there is one, invalid route found, program crashes, program doesn't finish in one minute: 0

Where a submission finds the shortest route on the most difficult problem instance, an additional 5 bonus marks are available. The award of these marks will depend on the speed of solving the most difficult problem instance.

Alternative to Program Submission (maximum 15 out of 25 marks)

Where you have been unable to produce a working version of the code, you may submit up to five pages describing how you would have implemented a solution in a programming language of your choice. You will need to give detailed descriptions of the algorithms, with diagrams, so that you can show that you fully understand them and understand why they are suitable for solving the specific problem. You can include pseudo-code, but should ensure that this is written to by relevant to the specific problem being solved (rather than generic pseudocode taken unaltered from the web). You can also submit any program code you have written – even if it doesn't work. **Those taking this option will receive a maximum mark of 18 out of 30.**

What to hand in on Moodle:

- A single pdf file containing your answers to questions 1 and 2. The file should include your matriculation number on the front page.

  Then either
- A zip file containing the caveroute.bat file and whatever else is needed to run the program from the command line (usually just the executable) . Please don't include anything else in the zip. Don't include source code or input files – but retain your source code. **You must ensure that your program works from the command line or you will lose marks.** This should be set up so that if all the files are extracted to the same directory, typing the command line caveroute <file> (or caveroute2 <file>) in that directory will run your program on <file>.cav

  OR
- The alternative submission as a single PDF file.

Plagiarism Check

Programs submitted will be checked for plagiarism. Anyone submitting a program which is the same as someone else's is liable to investigation for plagiarism. Be careful not to allow anyone else access to your program. Please keep your source code. Those with identical submissions, and a random sample of others, will undergo an interview with their source code to confirm authorship.

Deadline: Midnight 3rd December 2019: Final submission on Moodle.