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	*Report – 1 page per description of algorithm 1 page for evaluation*
8. Deadline of submission Your attention is drawn to the penalties for late submission	Practical class week beginning 19th November 2018 : Demonstration and live testing of program on new data. 5.00pm 30th November 2018: Final submission on Moodle.
9. Arrangements for submission	Moodle (+demo in lab)

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.*

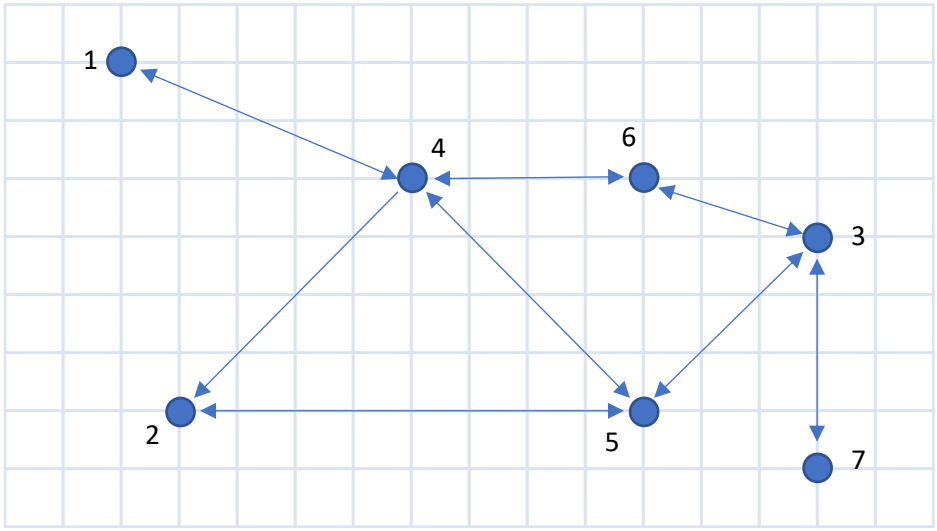
Artificial Intelligence

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:



Might be 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:

To \ From	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 coordinates of the caverns are not given in any particular order, but 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. (15 marks) Research and describe three methods that could be used to search through the map for a route. One of the methods should be a method not covered in the lecture material (but might have been covered in the tutorials or practicals). One of the methods should be a method used by your program below. You can look at the visualisation software for the week 5 practical for ideas about new methods to research. Extra marks will be awarded for diagrams which help to explain the search method. Please don't include diagrams or text taken from the internet without referencing these appropriately. Maximum one page per method.

2. (10) Evaluate each of the methods you described with respect to solving search problems in general and this particular problem in particular. Maximum one page total.

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 50 caverns and finds a path from the first cavern to the last cavern writing this to a solution file <file>.csn

   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.

   You must attend the practical in the week beginning 19th November to demonstrate your program working with the amended batch file and test it against previously unseen map files.

   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.

Program Marking Scheme

Your program will be tested on 10 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).

The score will be as follows:

Best route found or *no-route-possible* correctly identified : 2 marks
Valid route found: 1 mark
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 three largest files, an additional 5 bonus marks are available. The award of these marks will depend on the efficiency of the algorithm used.

Where the scoring method above produces a score of less than 10 out of 25, the code will be examined in order to give a score reflective of the quality of the code written for solving the problem – the maximum mark in this case is 10.

What to hand in on Moodle:

- A pdf document containing answer to question (a)
- A zip file containing the caveroute.bat file and whatever else is needed to run the program bit not any input or output files. This should be set up so that if all the files are extracted to the same directory, typing the command line caveroute <file> in that directory will run your program on <file>.cav

Deadlines:

Practical class week beginning 19th November 2018 : Demonstration and live testing of program on new data.

5.00pm 30th November 2018: Final submission on Moodle.