1. Module number	*SET09122/SET09822*
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	*Program*
6. Weighting	*60% 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	11.00pm 24 November 2023 – Hand in 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 document below*
12. Special instructions	*See document below*
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 A (only do one of the courseworks)

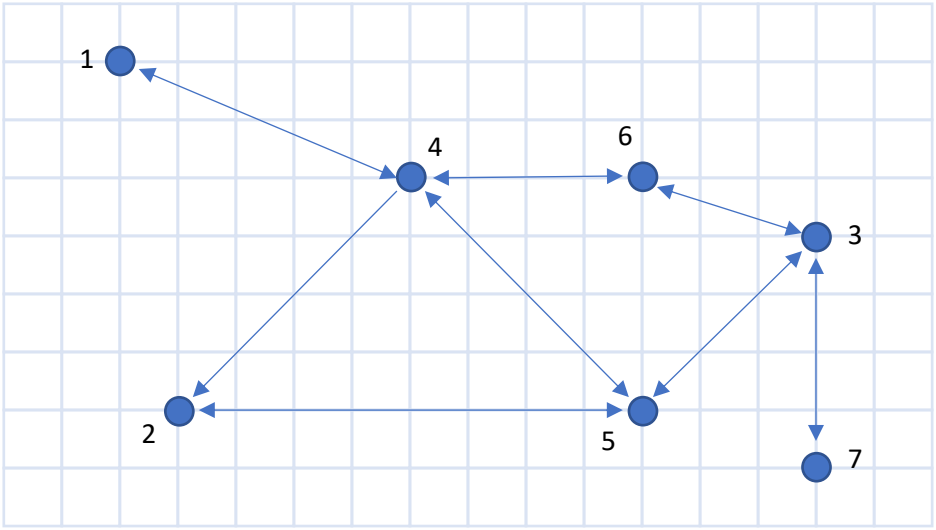
Plagiarism Check

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

Caverns Routing 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:

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 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:

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

Cavern Map 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 tunnels are one-way.

The order of the connectivity matrix is as 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

Be sure to check that you are reading the format correctly – with the ordering (from/to) of the tunnels the right way around. Some people find the ordering counter-intuitive.

Solution File Format

Read this carefully – getting the format exactly correct as important as the solutions are marked automatically by a program that is not forgiving about incorrect formatting.

Routes through caverns are stored in the output file by having a series of integers denoting the order of visiting the caverns. Caverns 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 cavern in the input file and end at the last cavern 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 cavern.

Where the algorithm identifies that no route can be found, it should place a single 0 in the file (not a message saying no route is possible) .

What to do:

Write a computer program to solve this problem which runs on the Windows command line. If you don't have access to a Windows computer, please contact Ben Paechter so that appropriate arrangements can be made.

The program should open a cavern map file with up to 5,000 caverns and find a path from the first cavern to the last cavern writing this to a solution file

The program should have one parameter which should be the name of the cave map file without the .cav extension. So, if the parameter is "banana" the cave map file read should be "banana.cav". The cave map file will be in the current folder.

The program should write the solution file to the current folder with the name being the parameter plus the extension .csn.

So, if the parameter is "banana" you program should open file banana.cav and write the solution to file banana.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.

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, with the example files given, on a reasonably modern computer. If the program takes too long on one of the test files you will not receive a mark for that file.

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.

It is important that you adhere to the following instruction very carefully – marks will be deducted if your program does not fit with the exact specification:

- Your final program should not write any information to the screen (although you may wish to do this when developing the software).
- Your final program must not require any kind of keyboard or mouse input – it should simply run from the command line and open the cave map file from the current folder and write the solution file to the current folder.
- 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.

  So, for example, typing:

  caveroute banana

  should read the cave map file banana.cav (stored in the current folder) and write the solution to the solution file banana.csn (saved in the current folder)

You will find test input files, some with the best-known solution, on Moodle. You can use these to check that you have solutions that are at least quite close to the best found. If you can't find a route when one is possible then maybe you are reading the input file wrongly.

Program Marking Scheme (Total Marks 60)

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 of varying difficulty. Two of these will not have any valid route. Scores will be awarded as follows.

Quality of Solutions:

At least one valid route found: 20 marks

For each of the ten test cases (max mark 26):

For the two cases where no route is possible:

No-route-possible correctly identified (0 in output file): 1 mark

For the eight cases where a route is possible:

Best route found: 3 marks
Valid route found, but not the best route: 1 mark

No route found where there is one, invalid route found, program crashes, program doesn't finish in one minute: 0 marks

Speed on hardest problem:

Where a submission finds the best route on the most difficult problem instance, up to an additional 10 bonus marks are available. The award of these marks will depend on the speed of solving the most difficult problem instance. Programs that run faster will receive more of the 10 marks.

Adhering to the specification:

For example, the output file format; the requirements of running the program; the folder in which the program must run; the function of the .bat file; correct files submitted; the programming running without the need for additional DLLs etc.

4 marks for adhering to the specification exactly.
2 marks for mainly adhering to the specification but with minor deviations.
0 marks for significant deviation from the specification resulting in the automatic testing not working.

If, at the end of this process, your mark is less than 24 out of 60 – I will look at your source code to see if the mark can be brought up to 24.

What to hand in on Moodle:

- 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 or equivalent) . Please don't include anything else in the zip. Don't include source code or input files. **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> in that directory will run your program on cavern map file <file>.cav.
- A separate zip file containing your source code. This won't normally be marked but may be marked if your program does not run well. It will also be part of the plagiarism check.
- **Deadline:** 11pm, 24 November 2023 on Moodle.