**Introduction to 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:

1

6

4

3

5

7

2

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:

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 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.

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

**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 1 to Cavern 2

Connectivity of Cavern 1 to Cavern 3

Connectivity of Cavern 1 to Cavern 4

Connectivity of Cavern 1 to Cavern 5

.

.

.

Connectivity of Cavern 2 to Cavern 1

Connectivity of Cavern 2 to Cavern 2

Connectivity of Cavern 2 to Cavern 3

.

.

.

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

**What to do**

1. 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. One of the methods should be the method used by your program below. You can look at the visualisation software for the week 6 practical for ideas about new methods to research. Identify the advantages and disadvantages of each method (20 marks)
2. Write a computer program of your choice which opens a cave map file called input.cav with up to 20 caverns and finds a path from the first cavern to the last cavern.

You can choose any language that can produce a windows .exe file. We suggest that you use Java as we can provide the most support for this language.

The program should have two modes. The first mode should allow the search to be stepped through by pressing a button of key to advance by one step. At each step information about the current state of the search should be displayed (at least the current cavern/tunnel being considered, but also possibly open lists, closed lists, heuristic information etc depending on the search method). When a solution is found the final route and its length should be displayed in its entirety.

The second mode should run as fast as possible and only display the final route and its length.

Extra marks will be awarded for programs that find shorter (less distance travelled) routes or work faster.

Extra marks will be awarded if the program provides a visualisation of the search.

You must attend the week 10 practical to demonstrate your program and test against previously unseen map files. (30 marks)

If for any reason you cannot get the program to run, as an **alternative** you can do what is below for a **maximum of 20 marks** **out of 30** (don’t do this is your program runs OK):

* Produce pseudo-code for a program which reads cavern data from a .cav file and finds a path from the first cavern to the last cavern. Pseudo code which always finds a path if there is one will achieve a pass mark. Extra marks will be given if consideration has been given to finding solutions faster, using less memory, or finding shorter paths. (12 Marks)
* Explain how the search method you used would be implemented in your program and any measures you would take to ensure that the program runs quickly (8 marks)

**What to hand in on moodle:**

* A pdf document containing (a) Answers to questions (b) instructions for running the program (c) source code for your program
* a windows .exe executable file
* instructions for running the program in a pdf document.

**Deadlines:**

Practical class Week 10 (week beginning 11th March 2018) : Demonstration and live testing of program on new data.

5.00pm 23rd March 2018: Final submission on Moodle.