Student ID: 18249833
Name: Xhien Yi Tan (Xavier)

Document version: 1.1 (2015-11-15)

Curtin University – Department of Computing

Assignment Cover Sheet / Declaration of Originality

Complete this form if/as directed by your unit coordinator, lecturer or the assignment specification.

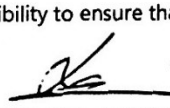
Last name:	Tan	Student ID:	18249833
Other name(s):	Xhien Yi		
Unit name:	Artificial and Machine Intelligence	Unit ID:	COMP 3006
Lecturer / unit coordinator:	Mihai	Tutor:	Stefan
Date of submission:	1 May 2017	Which assignment?	(Leave blank if the unit has only one assignment.)

I declare that:

- The above information is complete and accurate.
- The work I am submitting is *entirely my own*, except where clearly indicated otherwise and correctly referenced.
- I have taken (and will continue to take) all reasonable steps to ensure my work is *not accessible* to any other students who may gain unfair advantage from it.
- I have *not previously submitted* this work for any other unit, whether at Curtin University or elsewhere, or for prior attempts at this unit, except where clearly indicated otherwise.

I understand that:

- Plagiarism and collusion are dishonest, and unfair to all other students.
- Detection of plagiarism and collusion may be done manually or by using tools (such as Turnitin).
- If I plagiarise or collude, I risk failing the unit with a grade of ANN ("Result Annulled due to Academic Misconduct"), which will remain permanently on my academic record. I also risk termination from my course and other penalties.
- Even with correct referencing, my submission will only be marked according to what I have done myself, specifically for this assessment. I cannot re-use the work of others, or my own previously submitted work, in order to fulfil the assessment requirements.
- It is my responsibility to ensure that my submission is complete, correct and not corrupted.

Signature:  Date of signature: 1 May 2017

(By submitting this form, you indicate that you agree with all the above text.)

Student ID: 18249833
Name: Xhien Yi Tan (Xavier)

Artificial and Machine Intelligence - COMP3006

Assignment Semester 1, 2017

Student ID: 18249833

Name: Xhien Yi Tan (Xavier)

This submission includes the following file:

- { beam-search, alim-search }
- { BeamSearch, AStarSearch }.java
- { BeamSearchMain, AStarSearchMain }.java
- { FileIO, Node }.java
- report.pdf, README.txt
- Test Files

1. Introduction

For this Assignment, I have implemented two Informed Searches. These searches are Beam Search and Memory Limited A* Search.

1.1 Beam Search

For beam search, I choose to use linked list for nodes because it makes it easier to visualise and keep track for what I have. For example, if Node A has children of B and C, then I will make two lists for this, one list for A → B and the other list for A → C and put them in queue. So when I want to deal with the children of B, I just dequeue the list and get the last element in the list which is B. So I will have a queue of linked list of nodes. For space complexity it may seem that lots of nodes are recreated when it branches out to two or more children, and copy the list to another list for new child. In term of time complexity, it might be slower when comparing to A* because I have to deal with node as the last element of a list. Many for loops are required to do so.

1.2 Memory Limited A* Search

For memory limited a* search, I choose to use a priority queue of node whereas beam search I use queue of linked list of nodes. The reason I use priority queue is that I will always get the lowest f cost every time. For multiple parents, I will create a new node and set their parent to previous node. So there is no need to worry about multiple parents. For example, Node C has parents of A and B. So I will create a new node of C and set its parent to A, and then set the original C's parent to B, so in this case there are two Cs with different f cost even though they are the same node. So if I want to display solution path or alternate path, I will recursively call the node's parent and put them into a list to display. For space complexity, it definitely saves more space than my beam search because I don't need to create a new list whenever a node branches out to more than one children. It is also an memory limited to only 15 nodes, so when it reaches 15 nodes, I will start removing highest f cost node that is not part of the path to add new node. In term of time complexity, it is probably faster than beam because it does not require many lists, but for the memory part, I have to loop through the search tree to remove the appropriate node, that is time consuming.

Student ID: 18249833
Name: Xhien Yi Tan (Xavier)

2. Bugs/Problems

2.1 Beam Search

The Beam Search is fully working for all my test files. This search is not a complete nor optimal search because it depends on the k value. Alternative solutions can be found only for k = 2 or k = 3, because there is no alternate path for k = 1. The only problem that I currently have is that I can't display every partial path whenever a solution is found. I can print some of them but not all. The other problem I have is that I can't factor out repeated code from the way I implement my algorithm, so therefore I have lot of lines, but it works perfectly fine even though some of the code is repeated.

2.2 Memory Limited A* Search

The A* search is working, it can find all alternate paths as well as solution. If the heuristic is admissible and consistent, my program should be able to find the optimal solution. The problem for this search is that I can't implement SMA*, but my memory limited method works similarly like SMA*. For the memory limit part, I have limited my search tree (a list of nodes that have been expanded) to have only 15 nodes at most (as described in the spec). For example, when the search tree is full, I will remove the highest f cost that is not part of the current path to make room for new node. It works similarly like SMA* where when the search tree is full, it removes the highest f cost from the tree and make its parent to remember the f cost, but for me, I didn't make the parent to remember its child f cost.

3. Testing

I have provided various graphs for testing even though it is not necessary to submit. My searches work well for these graphs. The graphs contains linear graph, a looping graph and as well as the four graphs provided in the tutorial.