Report LINFO1361: Assignment 1

Group N°...

Student1: .....

Student2: .....

February 26, 2023

1 Python AIMA (5 pts)

1. In order to perform a search, what are the classes that you must define or extend? Explain precisely why and where they are used inside a *tree_search*. Be concise! (e.g. do not discuss unchanged classes). **(1 pt)**

2. Both *breadth_first_graph_search* and *depth_first_graph_search* have almost the same behaviour. How is their fundamental difference implemented (be explicit)? **(1 pt)**

3. What is the difference between the implementation of the *..._graph_search* and the *..._tree_search* methods and how does it impact the search methods? **(1 pt)**

4. What kind of structure is used to implement the *closed list*? What properties must thus have the elements that you can put inside the closed list? **(1 pt)**

5. How technically can you use the implementation of the closed list to deal with symmetrical states? (hint: if two symmetrical states are considered by the algorithm to be the same, they will not be visited twice) **(1 pt)**

2 The Tower sorting problem (15 pts)

1. **Describe** the set of possible actions your agent will consider at each state. Evaluate the branching factor considering n tower with a maximal size $m$and c colors (the factor is not necessarily impacted by all variables) **(1.5 pts)**

2. **Problem analysis.**

   (a) Explain the advantages and weaknesses of the following search strategies **on this problem** (not in general): depth first, breadth first. Which approach would you choose? **(1.5 pts)**

(b) What are the advantages and disadvantages of using the tree and graph search **for this problem**. Which approach would you choose? **(1 pts)**

3. **Implement** a solver for the Tower sorting problem in Python 3. You shall extend the *Problem* class and implement the necessary methods -and other class(es) if necessary- allowing you to test the following four different approaches:

   - *depth-first tree-search (DFSt)*;
   - *breadth-first tree-search (BFSt)*;
   - *depth-first graph-search (DFSg)*;
   - *breadth-first graph-search (BFSg)*.

   Experiments must be realized (*not yet on INGInious!* use your own computer or one from the computer rooms) with the provided 10 instances. Report in a table the results on the 10 instances for depth-first and breadth-first strategies on both tree and graph search (4 settings above). Run each experiment for a maximum of 3 minutes. You must report the time, the number of explored nodes as well as the number of remaining nodes in the queue to get a solution. **(4 pts)**

Inst.	BFS						DFS					
	Tree			Graph			Tree			Graph		
	T(s)	EN	RNQ	T(s)	EN	RNQ	T(s)	EN	RNQ	T(s)	EN	RNQ
i_01												
i_02												
i_03												
i_04												
i_05												
i_06												
i_07												
i_08												
i_09												
i_10												

T: Time — **EN**: Explored nodes — **RNQ**: Remaining nodes in the queue

4. **Submit** your program (encoded in **utf-8**) on INGInious. According to your experimentations, it must use the algorithm that leads to the **best results**. Your program must take as only input the path to the instance file of the problem to solve, and print to the standard output a solution to the problem satisfying the format described earlier. Under INGInious (only 45s timeout per instance!),

we expect you to solve at least 10 out of the 15 ones. Solving at least 10 of them will give you all the points for the implementation part of the evaluation. **(6 pts)**

5. **Conclusion.**

   (a) Are your experimental results consistent with the conclusions you drew based on your problem analysis (Q2)? **(0.5 pt)**

   (b) Which algorithm seems to be the more promising? Do you see any improvement directions for this algorithm? Note that since we're still in uninformed search, *we're not talking about informed heuristics*). **(0.5 pt)**