Report LINFO1361: Assignment 1

Group N°1

Student1: Victor Carballes Cordoba (NOMA : 34472100)

Student2: Krystian Targonski (NOMA : 42942000)

February 29, 2024

1 Python AIMA (3 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* are making a call to the same function. How is their fundamental difference implemented (be explicit)? **(0.5 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? **(0.5 pt)**

4. What kind of structure is used to implement the *reached nodes minus the frontier list*? What properties must thus have the elements that you can put inside the reached nodes minus the frontier list? **(0.5 pt)**

5. How technically can you use the implementation of the reached nodes minus the frontier 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) **(0.5 pt)**

2 The PacMan Problem (17 pts)

(a) **Describe** the set of possible actions your agent will consider at each state. Evaluate the branching factor**(1 pt)**

(b) How would you build the action to avoid the walls? **(1 pt)**

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? **(2 pts)**

   (b) What are the advantages and disadvantages of using the tree and graph search **for this problem**. Which approach would you choose? **(2 pts)**

3. **Implement** a PacMan solver 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 1 minute. 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	0,007	50	334	0,001	6	43	/	/	/	/	/	/
i_02	0,010	109	681	0,001	13	95	0,00	4	12	0,00	4	12
i_03	0,793	14379	39702	0,210	3795	10583	/	/	/	/	/	/
i_04	4,617	45803	234229	0,753	7158	38644	/	/	/	/	/	/
i_05	1,128	9553	56748	0,158	1409	8143	/	/	/	/	/	/
i_06	0,001	22	27	0,001	8	13	/	/	/	/	/	/
i_07	0,072	1017	4392	0,014	180	836	/	/	/	/	/	/
i_08	0,001	13	7	0,001	6	6	0,00	5	3	0,00	5	3
i_09	0,021	195	1133	0,003	32	162	/	/	/	/	/	/
i_10	0,010	104	681	0,002	13	95	0,00	4	12	0,00	4	12

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 inputs the four numbers previously described separated by space character, and print to the standard output a solution to the problem satisfying the format described in Figure 3. Under INGInious (only 1 minute timeout per instance!), we expect you to solve at least 12 out of the 15 ones. **(6 pts)**

5. **Conclusion.**

   (a) How would you handle the case of some fruit that is poisonous and makes you lose? **(0.5 pt)**

   (e) Do you see any improvement directions for the best algorithm you chose? (Note that since we're still in uninformed search, *we're not talking about informed heuristics*). **(0.5 pt)**