Report LINFO1361: Assignment 1

Group N°085

Student1: Foret Félix 87352100

Student2: Lernoux Lore 77562000

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)**

Nous devons implémenter la classe Pacman, qui implémente la classe Problem avec ses méthodes actions(), result() et goal_test(). La méthode action() permet de trouver toutes les directions que Pacman peut envisager, ainsi que le nombre de pas qu'il peut y faire. La méthode result() applique l'action à l'état dans lequel Pacman se trouve, et retourne le nouvel état après application de l'action. Enfin, goal_test() vérifie que tous les fruits ont bien été mangés par Pacman.

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)**

breadth_first_graph_search utilise une FIFO queue et *popleft* pour prendre les noeuds ajoutés en premiers (donc les noeuds visités en premiers), tandis que *depth_first_graph_search* utilise une stack pour prendre les noeuds les plus récents d'abord, donc visités en derniers. Cela implique que *breadth_first_graph_search* va effectuer une recherche en largeur et *depth_first_graph_search* une recherche en profondeur.

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)**

La recherche par graphe, *graph_search*, garde en mémoire les états visités afin de ne pas ajouter deux fois un même état à la frontière. Cela permet d'éviter de parcourir plusieurs fois un chemin démarrant d'un même noeud comme le fait *tree_search*.

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)**

On utilise un set et un dictionnaire, mais le dictionnaire est simplement utilisé comme un set (toutes les valeurs sont les mêmes (*True*), on vérifie simplement la présence d'une clé). Ces deux structures de données utilisent des tables de hashage pour vérifier l'appartenance ou non d'un élément. Les éléments que l'on place dedans doivent donc être hashables.

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)**

On peut implémenter une fonction de hashage pour la classe *Node* qui ne va hasher que les variables correspondant à l'état actuel du noeud et pas celles correspondantes au chemin qui a mené à cet état. Grâce à cela, les hashs de deux états symétriques seront égaux car la seule chose qui les différencie sera le chemin utilisé.

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)**

Dans le cas où il n'y a aucun mur dans la grid, Pacman va considérer les déplacements verticaux et horizontaux, de toutes les longueurs possibles. Il va donc considérer (i-1) * (j-1) possibilités. Le facteur de branchement est donc (i-1) * (j-1). Dans le cas où des murs se trouvent dans la grid, ce facteur sera plus petit car Pacman ne considère plus les déplacements au delà du mur (puisqu'il ne peut pas le traverser).

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

Pour chaque direction envisagée, on laisse la possibilité au Pacman d'avancer dans cette direction tant qu'il ne rencontre pas la frontière de la grid ou un mur ; si c'est le cas, on arrête la possibilité de Pacman d'avancer dans cette direction.

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)**

Depth Avantage: consomme peu de mémoire. Inconvénients: peut parcourir beaucoup de noeuds lointains même si la solution optimale est dans un noeud proche, trouve une solution qui n'est pas forcément une solution optimale.
Breadth Avantage: trouve une solution optimale. Inconvénients: utilise beaucoup de mémoire car la frontière est grande.
Nous avons opté pour la recherche en largeur, car elle garantit de trouver une solution optimale et la mémoire utilisée, bien que plus grande, n'est pas excessive étant donné la petite taille des instances.

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

Arbre Avantage: consomme moins de mémoire. Inconvénient: peut parcourir plusieurs fois les mêmes noeuds et tomber dans des boucles infinies donc ne pas trouver de solution.
Graphe Avantage: ne parcours jamais deux fois le même noeud, donc garanti de trouver une solution (si elle existe et que le nombre de noeuds est fini). Inconvénient: consomme plus de mémoire pour stocker tous les noeuds visités.
Nous choisirions la recherche par graphe afin de garantir un résultat, de plus la mémoire nécessaire pour retenir tous les états visités est petite étant donné la taille des instances.

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	0.0074	74	739	0.0013	7	51	/	/	/	0.0083	54	26
i_02	0.0063	140	1169	0.0012	15	67	0.0001	3	26	0.0003	3	18
i_03	/	/	/	0.0099	152	163	/	/	/	0.0037	33	64
i_04	33.109	309142	3429250	0.0186	134	402	/	/	/	0.0243	141	93
i_05	0.3428	5582	53849	0.0063	49	214	/	/	/	0.0076	38	97
i_06	0.0098	254	2051	0.0009	14	64	/	/	/	0.0034	42	23
i_07	0.0565	1547	11674	0.0022	34	71	/	/	/	0.0043	55	20
i_08	0.0018	53	270	0.0003	8	25	/	/	/	0.0010	19	13
i_09	0.0030	61	476	0.0006	8	31	/	/	/	0.0026	22	38
i_10	0.0069	140	1169	0.0013	15	67	0.0002	3	26	0.0003	3	18

T: Time — **EN**: Explored nodes — **RNQ**: Remaining nodes in the queue — **/**: Test timed out

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)**

Dans *Pacman.actions()*, avant d'ajouter une action dans la liste (avant les lignes 33, 38, 43 et 48), il faudrait ajouter la ligne suivante:
if state.grid[x][y-k] == "poison": continue
pour simplement ne pas considérer une action qui mènerait le Pacman sur une case empoisonnée.

   (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)**

On pourrait faire un *early goal test*, c'est-à-dire vérifier directement en générant les nouveaux noeuds dans la frontière si ceux-ci ne mènent pas à la réponse finale, donc que tous les fruits seraient mangés. On ne devrait donc pas attendre le *late goal test*.