Report LINGI2261: Assignment 2

Group N°151

Student1: Sacha Defrère

Student2: Guillaume Pasture

March 10, 2022

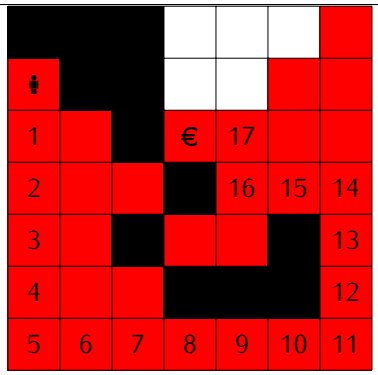
1 Search Algorithms and their relations (3 pts)

Consider the maze problems given on Figure 1. The goal is to find a path from 🚹 to € moving up, down, left or right. The black cells represent walls. This question must be answered by hand and doesn't require any programming.

1. Give a consistent heuristic for this problem. Prove that it is consistent. Also prove that it is admissible. **(1 pt)**

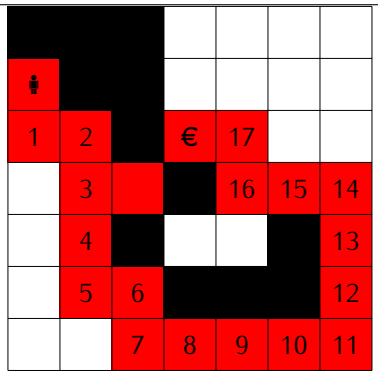
The distance of manhattan between the actual position of 🚹 and € is a consistent and admissible heuristic for this problem.
It is admissible because the manhattan distance is in this case the minimum distance to travel from a point A to a point B while ignoring the walls, which will always be less than or equal to the optimal distance with walls. $h(n) \leq optimal\ cost$ is always true.
It is also consistent because for each neighbour cell, either we follow the path of the manhattan distance and the cost is equal to the diminution of the heuristic, either we move away from the path and both the cost and the heuristic increase. $h(n) \leq cost(n, p) + h(p)$ is always true.

2. Show on the left maze the states (board positions) that are visited when performing a uniform-cost graph search, by writing the order numbers in the relevant cells. We assume that when different states in the fringe have the smallest value, the algorithm chooses the state with the smallest coordinate (i, j) ($(0, 0)$ being the bottom left position, i being the horizontal index and j the vertical one) using a lexicographical order. **(1 pt)**



Legend : Red cell = visited cell

3. Show on the right maze the board positions visited by A^* graph search with a manhattan distance heuristic (ignoring walls), by writing the order numbers in the relevant cells. A state is visited when it is selected in the fringe and expanded. When several states have the smallest path cost, they are visited in the same lexicographical order as the one used for uniform-cost graph search. **(1 pt)**



Legend : Red cell = visited cell

2 PageCollect problem (17 pts)

1. Model the PageCollect problem as a search problem; describe: **(2 pts)**

   - States
   - Initial state
   - Actions / Transition model
   - Goal test
   - Path cost function

States: The states are the representation of the grid at a given step (including p, #, @ and X), with some additional information such as the total cost accumulated so far or the heuristic value for this state.
Initial state: It is the state described in the file with initial student position, pages positions, walls positions.
Actions / Transition model: The four possible actions are moving the student to an adjacent cell in each of the cardinal directions (NSWE). Each action is only possible if there is no wall on the arrival cell.
Goal test: When we initialize the problem, we create a final state where the pages have been removed and the student is in the examiner's cell. To see if we have arrived at the right solution, we compare the current state with the final one.
Path cost function: The path cost function is f(n) = g(n) + h(n) where g(n) is the number of steps done by the student and h(n) is the heuristic. For h(n) we choose to compute the sum of the distance between the position of the player and the different pages. Once all pages are collected, the heuristic becomes the distance between the player and the examiner.

2. Give an upper bound on the number of different states for a PageCollect problem with a map of size $n \times m$, with k pages to collect. Justify your answer precisely. **(1 pt)**

For each and every problem, there is a total of $m \times n$ possible positions for the student (assuming there are no walls), to which we have to multiply the number of different possible subsets of pages that have already been collected, noted $Card(\mathcal{P}(pages)) = 2^k$.
The upper bound on the number of different states is $m \times n \times 2^k$.

3. Give an admissible heuristic for a PageCollect instance with k pages. Prove that it is admissible. What is its complexity ? **(2 pts)**

An admissible heuristic that would guarantee to find the optimal path is the following :
For every permutation of the pages, we compute the sum of the manhattan distances from the student to the first page, then from the first page to the second page, ..., then from the last page to the examiner. We then assign heuristic values to the neighbour cells on the basis of the smallest sum, which corresponds to the optimal order to collect the pages (ignoring walls). This heuristic is indeed admissible because it is the optimal concatenation of admissible segments (see question 1).
Unfortunately, its complexity is $\mathcal{O}(k!)$, which is simply unusable even with small values of k. This is why this heuristic was not used in our code.

4. **Implement** your solver. Extend the *Problem* class and implement the necessary methods and other class(es) if necessary. **(1 pt)**

5. **Experiment**, compare and analyze informed (*astar_search*) and uninformed (*breadth_first_graph_search*) graph search of aima-python3 on the 10 instances of PageCollect provided. Report in a table the time, the number of explored nodes and the number of steps to reach the solution. Are the number of explored nodes always smaller with *astar_search*? What about the computation time? Why? When no solution can be found by a strategy in a reasonable time (say **3 min**), indicate the reason (time-out and/or exceeded the memory). **(4 pts for the whole question)**

As can be seen in the table below, the number of nodes explored by the A^* search is always smaller than the one of the BFS, proving that using informed search is way more efficient in terms of decision making.
As for the computation time, which goes hand in hand with the number of explored nodes, we can clearly see that the A^* search is way more efficient than the BFS, which demonstrates once again the advantage of using heuristics and informed search. The rare cases where the A^* search took more time is either due to the tests being done in different context, e.g. with more programs running on the machine, either to those cases being small and fast to solve, not letting the time to the A^* search to make intelligent decisions.
The only three cases that remained unsolved, one by the A^* search and two by the BFS, are due to a time-out (the one by the A^* search still has a filled entry, but we can see that $T(i08) = 286.46 > 180$).

	A^* Graph			BFS Graph		
Inst.	NS	T(s)	EN	NS	T(s)	EN
i01	19	0.065	133	19	0.083	278
i02	11	0.0097	33	11	0.014	64
i03	37	0.24	303	37	0.28	584
i04	26	0.038	94	26	0.044	173
i05	90	12.53	2533	-1	-1	-1
i06	41	0.57	515	41	7.32	4111
i07	21	0.032	64	21	0.039	106
i08	82	286.46	10236	-1	-1	-1
i09	21	0.058	86	21	0.04	135
i10	11	0.018	35	11	0.014	63

NS: Number of steps — **T**: Time — **EN**: Explored nodes

6. **Submit** your program on INGInious, using the A^* algorithm with your best heuristic(s). Your file must be named *pagecollect.py*. You program must print to the standard output a solution to the PageCollect instance given in argument, satisfying the described output format. **(5 pts)**