Report LINFO1361: Assignment 3

Group N°052

Student1: Bette Jonas

Student2: Huet Anatole

April 24, 2024

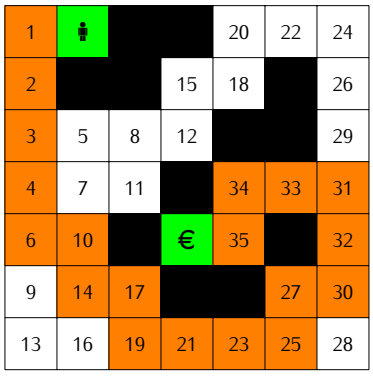
1 Search Algorithms and their relations (3 pts)

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.

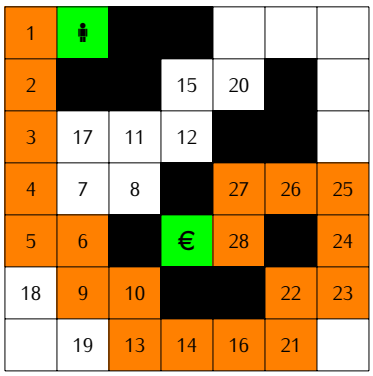
1. Give a consistent heuristic for this problem. Prove that it is consistent. Also prove that it is admissible. **(1 pt)**

A consistent heuristic could be to take every legal neighboor (not a wall) of the current cell that we didn't visited yet and calculate the manhattan distance to the goal. We then take the minimum of these distances when we arrive to a cell already visited.
This heuristic is consistent because the manhattan distance is always positive and the minimum of positive numbers is also positive. Even more, in the best case, the calculated score would be the same as the cost of the shortest path.
It is also admissible because the manhattan distance is the shortest distance between two points in a grid and we are not overestimating the distance to the goal.

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)**



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)**



2 N-Amazons problem (8 pts)

1. Model the N problem as a search problem; describe: **(2 pts)**
   - States
   - Initial state
   - Actions / Transition model
   - Goal test
   - Path cost function

States: A state is represented as an N-element array, where a value of r in the c-th entry means there is an empress at column c, row r, and a value of -1 means that the c-th column has not been filled in yet.

Initial State: an array filled with -1 meaning no pieces has been placed on the board yet.

Actions: A piece is being placed on the c-th column of row r.

Goal test: Every row has a piece and no pieces can be eaten by another.

Path Cost Reduction: The number of conflicts (the bigger it is, the worst it is)

2. Give an upper bound on the number of different states for an N-Amazons problem with N=n. Justify your answer precisely. **(0.5 pt)**

The upper bound is N!

For the first queen, there are N choices for its row and N choices for its column. Once the position of the first queen is fixed, the second queen cannot be placed in the same row, column, or diagonal as the first queen. This leaves (N-1) choices for the row and (N-1) choices for the column for the second queen. Similarly, for each subsequent queen, the number of available choices reduces by 1 in each direction (row, column, and diagonals) due to the constraints.

3. Give an admissible heuristic for a N=n. Prove that it is admissible. What is its complexity ? **(1 pts)**

An admissible heuristic is counting the number of pawn that is attacked. The goal is therefore with a heuristic which value is zero.
Since each pair of attacking queens represents at least one move that needs to be made to resolve the conflict (either by moving one of the queens or by adding blocking queens), the total number of conflicting pairs serves as a lower bound on the number of moves needed to reach a goal state. Therefore, it is admissible.

This heuristic is efficient to compute and provides a reasonable estimate of the minimum number of moves required to reach a goal state, making it suitable for guiding search algorithms like A* search in the N-Queens problem. It is in O(N*N*N) as for each pieces we need to check if there are any conflicts in the matrix.

5. **Implement** your solver. Extend the *Problem* class and implement the necessary methods and other class(es) if necessary. **(0.5 pt)**

6. **Experiment**, compare and analyze informed (*astar_graph_search*), uninformed (*breadth_first_graph_search* and *depth_first_graph_search*) graph search of aima-python3 on N = [10, 11, 12, 13, 20, 25, 30]. **(3 pts for the whole question)**

   Report in a table the time and the number of explored nodes and the number of steps to reach the solution.

   Are the number of explored nodes always smaller with *astar_graph_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).

We decided to run for N on instance 1 With NS = number of paths expanded & EN = number in frontier.

BFS is really bad for this case as there are a lot of different states to explore. DFS works well for small board size but times out after because it extends its solution based on every action it did. A* is an optimized version of DFS by using an heuristic wich gives a priority of the next state to explore.

	$A^\star$ Graph			BFS Graph			DFS Graph		
N	NS	T(s)	EN	NS	T(s)	EN	NS	T(s)	EN
10	313	0.02169	14	2298	0.1533	29	325	0.0075	17
11	22	0.003	22	7149	1.075	181	24	0.00079	26
12	28	0.02188	88	23923	21.9423	526	173	0.006	26
13	502	0.06	30	87922	361.88	1308	107	0.0046	38
20	28	0.01889	88	TO	TO	TO	51861	3.9636	80
25	47915	20.65	125	TO	TO	TO	319795	37.077	147
30	214187	128.36	202	TO	TO	TO	TO	TO	TO

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

6. **Submit** your program on INGInious, using the A^* algorithm with your best heuristic(s). Your file must be named *namazon.py*. Your program should be able to, given an integer as argument, return the correct output. Your program must print to the standard output a solution to the N's given in argument for the N-Amazons problem, satisfying the described output format. **(2 pts)**

3 Local Search: Sudoku Problem (8 pts)

1. Formulate the Sudoku problem as a Local Search problem (problem, cost function, feasible solutions, optimal solutions). **(2 pts)**

Problem: Find a solution to a Sudoku puzzle.
Cost Function: The cost function is the number of conflicts in the board.
Feasible Solutions: A feasible solution is a board where no conflicts are present.
Optimal Solutions: An optimal solution is a board where no conflicts are present and all the numbers are placed correctly.

1. You are given a template on Moodle: `sudoku.py`. Implement your own simulated annealing algorithm and your own `objective_score` function. Your program will be evaluated in on 15 instances (during 3 minutes each) of which 5 are hidden. We expect you to solve (get the optimal solution) at least 12 out of the 15. **(6 pt)**