CSE 3111 / CSE 3213

ARTIFICIAL INTELLIGENCE

FALL 2023

Programming Assignments Report

Burak Göksu – 1903015029

Zeynep Naz Ceyhan – 210315003

Submission Date: 24 December 2023

1 Development Environment

I developed and tested the NQueens problem implementation in Python 3.11 on a Windows operating system. The code was written using Visual Studio Code as the integrated development environment (IDE), taking advantage of its features for Python development. The machine used for development is equipped with an Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz, providing a powerful processing capability with a base clock speed of 2.60 GHz and multiple cores.

2 Problem Formulation:

PA1:

The code defines an NQueens class to tackle the N-Queens problem. It initializes with a user-specified or randomly generated state, ensuring its validity. The goal is to arrange queens on a chessboard such that they do not attack each other.

PA2 and PA3:

State Specification: *Explain how you represented the state in your NQueens problem. Mention the variables and their meanings.*

Initial State: *Describe how you set up the initial state for your NQueens problem. Did you generate it randomly or let the user input it?*

Possible Actions: *Outline the possible actions in your NQueens problem. For example, "Move Queen i to row j."*

Transition Model: *Explain how the transition model works in your problem. How does the state change when an action is applied?*

Goal Test: *Describe the conditions under which you consider a state as a goal state. In this case, when the number of attaacking pairs is 0.*

Path Cost: *Explain how you defined the cost for each action. In this case, it seems each move has a cost of 1.*

3 Results

PA1:

The code creates an instance for a 7x7 chessboard, showcasing the problem representation and calculating attacking queen pairs in the initial state.

PA2 and PA3:

Run your implementation with different parameters and inputs. For each run, include Algorithm Used: Specify which search algorithm you used (BFS, UCS, DFS, DLS, IDS, Greedy, A, Hill Climbing, Hill Climbing Random Restart, Genetic). Parameters: Include the values of parameters such as N, initial state, and depth limit for DLS. Output Results: Display the output results, including the resulting state, path taken, total cost, viewer statistics, and the time taken for the algorithm.*

4 Discussion

PA1:

Random State Generation: *Users can explore various configurations by generating random initial states.*

Validity Check*: The code rejects invalid states, ensuring adherence to N-Queens rules.*

Attacking Pairs Calculation: *The metric of attacking pairs provides insights into the conflict level of the initial state.*

User Interaction: *Manual input allows users to experiment with specific scenarios.*

Algorithmic Extension: *The code sets the stage for implementing solving algorithms, like simulated annealing or genetic algorithms, for optimal solutions.*

PA2 and PA3:

Discuss the results of different search algorithms: Completeness: Discuss whether each algorithm was able to find a solution (if one exists). Optimality: Evaluate the optimality of the solutions found by each algorithm. Time and Space Complexity: Compare the time and space complexity of the algorithms. Discuss which algorithms performed well in terms of runtime and memory usage. Graph Search vs. Tree Search: Compare the graph search and tree search versions of the algorithms. Effect of Depth Limit in DLS: Discuss the impact of the depth limit in the depth-limited search algorithm.

5 Output

PA1:

Part 2: Testing

Do not change this part. This is the test code.

```
problem = NQueens(7)
print(problem)
print(problem._count_attacking_pairs(problem.state))
```

```
Enter the state manually or press 'r' to generate a random state: r
Representation of NQueens Problem:
N = 7
State = 1732546
Count Attacking Pairs : 6
```

PA2 and PA3:

```
**************************************************
breadth_first
Resulting path:
[(None, '2323'), (('Move Queen 2 to row 4', 2, 4), '2423'), (('Move Queen 3 to row 1', 3, 1), '2413')]
Resulting state: 2413
Total costs: 2
Value: 6
Correct solution?: True
**************************************************
**************************************************
astar
Resulting path:
[(None, '2323'), (('Move Queen 2 to row 4', 2, 4), '2423'), (('Move Queen 3 to row 1', 3, 1), '2413')]
Resulting state: 2413
Total costs: 2
Value: 6
Correct solution?: True
**************************************************
**************************************************
hill_climbing
Resulting path:
[(('Move Queen 4 to row 3', 4, 3), '2413')]
Resulting state: 2413
Total costs: 3
Value: 6
Correct solution?: True
**************************************************
**************************************************
```

```
hill_climbing_random_restarts
Resulting path:
[(('Move Queen 4 to row 3', 4, 3), '2413')]
Resulting state: 2413
Total costs: 2
Value: 6
Correct solution?: True
************************************************
************************************************
genetic
Resulting path:
[('crossover', '2231')]
Resulting state: 2231
Total costs: 0
Value: 4
Correct solution?: False
************************************************
************************************************
uniform_cost
Resulting path:
[(None, '2323'), (('Move Queen 2 to row 4', 2, 4), '2423'), (('Move Queen 3 to row 1', 3, 1), '2413')]
Resulting state: 2413
Total costs: 2
Value: 6
Correct solution?: True
************************************************
************************************************
```

```
limited_depth_first
Resulting path:
[(None, '2323'), (('Move Queen 4 to row 4', 4, 4), '2324'), (('Move Queen 4 to row 4', 4, 4), '2324'), (('Move
Resulting state: 2413
Total costs: 5
Value: 6
Correct solution?: True
************************************************
************************************************
iterative_limited_depth_first
Resulting path:
[(None, '2323'), (('Move Queen 3 to row 1', 3, 1), '2313'), (('Move Queen 2 to row 4', 2, 4), '2413')]
Resulting state: 2413
Total costs: 2
Value: 6
Correct solution?: True
************************************************
************************************************
greedy
Resulting path:
[(None, '2323'), (('Move Queen 2 to row 4', 2, 4), '2423'), (('Move Queen 3 to row 1', 3, 1), '2413')]
Resulting state: 2413
Total costs: 2
Value: 6
Correct solution?: True
************************************************
************************************************
```