n-Queens Heuristic Analysis

Justin Cai

Artificial Intelligence 1 TJHSST 2016 Fall

I solved the nQueens problem as a CSP by implementing a DFS algorithm with forward propagation. The variables of the CSP are the n columns of a chessboard, and the values of each variable are the n rows.

I couldn’t figure out why my DFS was running so fast, so my heuristics have minimal improvement. Code was implemented in Python 3.5 and tests were performed on a MacBook Pro.

Plain DFS

|  |  |  |
| --- | --- | --- |
| n | Nodes | Time |
| 4 | 10 | .09 |
| 8 | 36 | .08 |
| 15 | 106 | .06 |
| 27 | 354 | .087 |
| 50 | 1226 | .247 |
| 77 | 2929 | .866 |

Heuristic 1

I shuffled the choices (columns) after picking the first unassigned row.

|  |  |  |
| --- | --- | --- |
| N | Nodes | Time |
| 4 | 10 | .066 |
| 8 | 29 | .068 |
| 15 | 108 | .03 |
| 27 | 352 | .076 |
| 50 | 1226 | .193 |
| 77 | 2927 | .871 |

Heuristic 2

I chose the row with the most constraints instead of the first unassigned row, and returned the choices (cols) for that row.

|  |  |  |
| --- | --- | --- |
| N | Nodes | Time |
| 4 | 10 | .066 |
| 8 | 36 | .067 |
| 15 | 106.03 | .08 |
| 27 | 354 | .086 |
| 50 | 1226 | .194 |
| 77 | 2929 | .874 |

Heuristic 3

I shuffled the choices (columns) after choosing the row with the most constraints.

|  |  |  |
| --- | --- | --- |
| N | Nodes | Time |
| 4 | 10 | .06 |
| 8 | 29 | .05 |
| 15 | 106 | .07 |
| 27 | 352 | .068 |
| 50 | 1226 | .214 |
| 77 | 2927 | .848 |

All the heuristics had basically the same amount of nodes created, no matter what size N was, so I decided to only use one graph.