Srinidhi Krishnan

n-Queens Heuristic Analysis

Artificial Intelligence 1

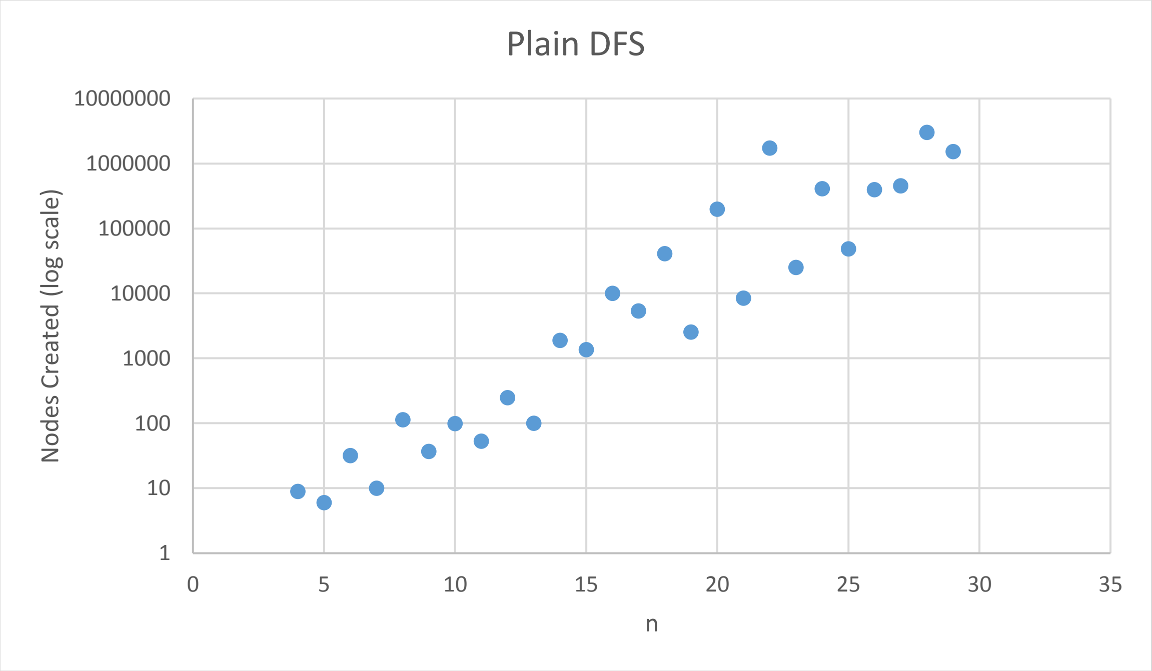
TJHSST 2016 Fall

I solved the nQueens problem as a constraint satisfaction problem using depth first search. The variables of the CSP are the n columns of a chessboard, and the values of each variable are the n rows. Initially, my code could solve n=30 queens in under 2 minutes. But my best implementation can solve n=99 queens in under 1 minute.

CASE 1: Plain DFS

The plain DFS chooses the next column based on a simple for loop. There was no row heuristic.

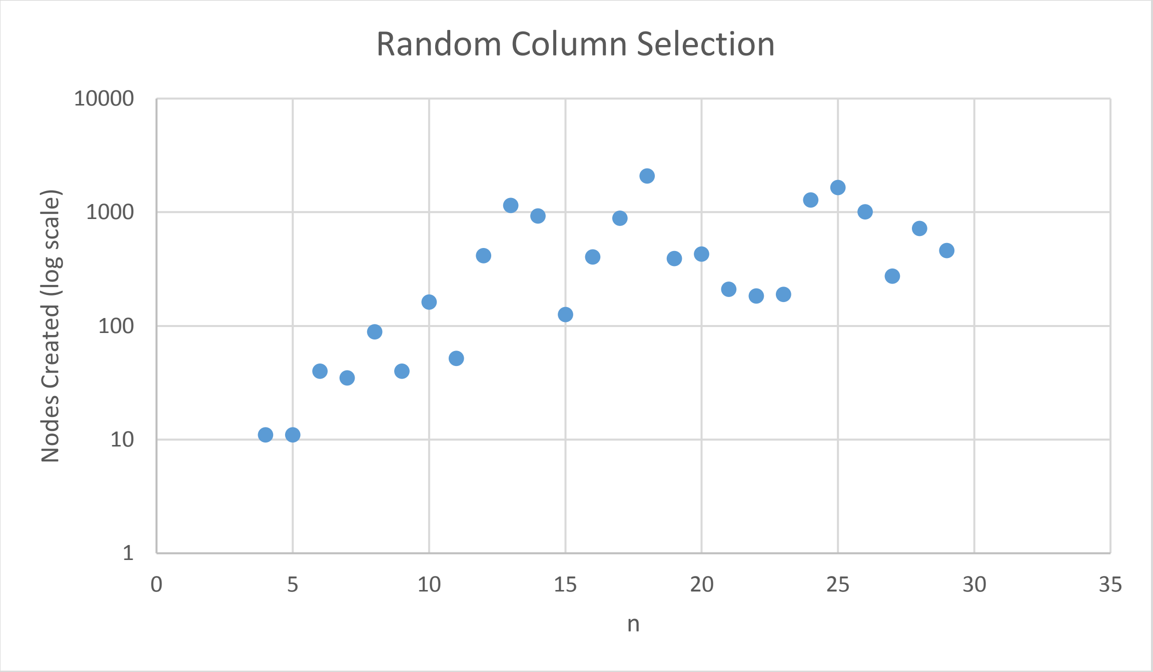
N	Goal Tests	Nodes Created	Time (s)	Nodes/sec
4	9	10	0.001001119613647461	9988.816385
7	10	22	0.0	infinity
10	99	121	0.004004240036010742	30217.96868
13	100	144	0.006004810333251953	23980.77408
16	10047	10106	0.6244425773620605	16184.03416
19	2546	2642	0.17612481117248535	15000.72581
22	1736276	1736403	120.77117919921875	14377.62727
25	48434	48610	3.873748779296875	12548.56801
28	3005957	3006170	263.47441935539246	11409.72246



CASE 2: Random Column Selection

The next available column was chosen using randint() from Python’s random library.

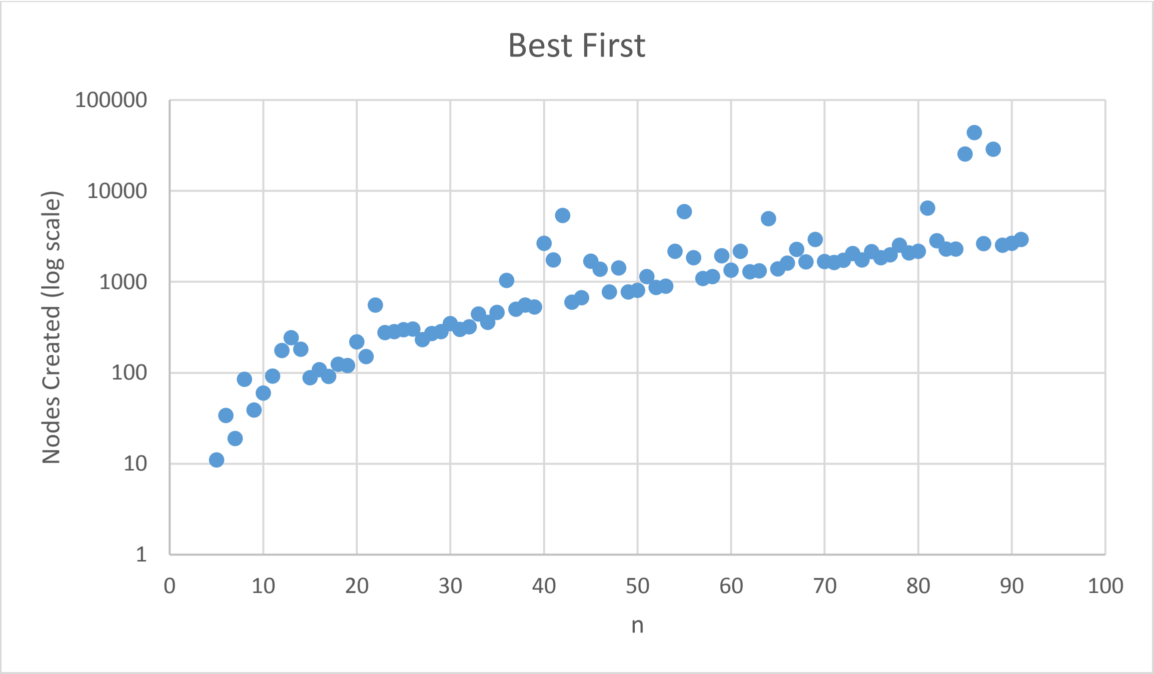
N	Goal Tests	Nodes Created	Time (s)	Nodes/sec
4	10	11	0.0019979476928710938	5505.649642
7	24	35	0.006005525588989258	5827.966176
10	139	162	0.03002023696899414	5396.359801
13	1105	1148	0.23616766929626465	4860.953252
16	332	404	0.14910554885864258	2709.490043
19	291	393	0.07104349136352539	5531.822725
22	38	183	0.05101490020751953	3587.187258
25	1476	1652	0.3572821617126465	4623.796475
28	481	720	0.22014522552490234	3270.568318



CASE 3: Best first

I chose the best column to pursue first in each step of the DFS algorithm. Best is defined as the column with the fewest available rows. If there were any ties, the column with the higher index was chosen.

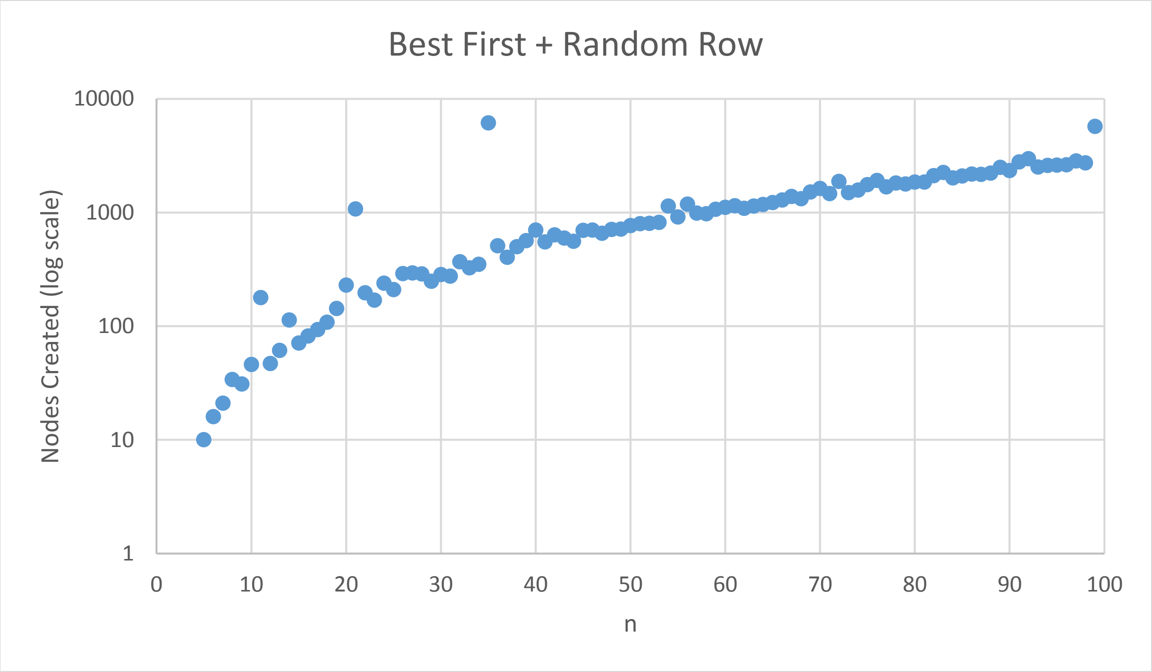
N	Goal Tests	Nodes Created	Time (s)	Nodes/sec
4	9	10	0.0010082721710205078	9917.956964
7	6	11	0.0010004043579101562	10995.55386
10	28	34	0.0020029544830322266	16974.92394
13	8	19	0.0009987354278564453	19024.05729
16	76	85	0.00702667236328125	12096.76439
19	21	39	0.0029957294464111133	13018.53211
22	36	60	0.00503230094909668	11922.97532
25	68	92	0.008005142211914062	11492.61282
28	147	176	0.017012596130371094	10345.27586



CASE 4: Best first with row heuristic

The best column with the least number of available moves was once again chosen, but implemented with a random row heuristic. The variable list was shuffled before being outputted to the for-loop in DFS. Based on the graphs of CASE 3 and CASE 4, this heuristic affected higher n's more than it did lower n's.

N	Goal Tests	Nodes Created	Time (s)	Nodes/sec
4	5	7	0.0	infinity
7	8	10	0.002000570297241211	8497.576928
10	25	16	0.0040018558502197266	11494.6669
13	264	18	0.03502535820007324	8650.875125
16	129	60	0.028020143508911133	6745.147466
19	57	64	0.06904983520507812	2172.344069
22	24	109	0.03404688835144043	4346.946437
25	35	183	0.09906983375549316	1907.745202
28	114	56	0.1250898838043213	2486.212238



Note: Code was implemented in Python 3.5 and tests were performed on an HP Pavilion with an intel core i5-4210u 1.70 ghz.