Jesse Garcia

26 March 2019

N-Queen Problem

N-queen problem broken down is having a board of n*n size and placing the queen in the spots that they will not be able to attack each other. In short, they need to be place like knights move. There are a lot of ways to come to a solution and the bigger the board the more solutions that are available. We were asked to solve the problem with Breadth first search, Breadth first search with pruning, Hill climbing, and Simulated annealing. Breadth first search is getting the n*n board and starting at the top breach down from every possible move and its possible moves. So, for a board of 4*4, there are 4^1 possible move. For those 4 possible moves there are 4^2 moves all the way to 4^4 (^ means power). So, for a board of 4*4 there are 4^4 moves which equate to 256 possibilities. For BFS w/ pruning, I decided to prune but eliminating boards with illegal moves. Once the move is made it gets sent to a check function. If it's a valid move return the board if not, eliminate it. Hill climbing on the other hand generates a random board and based on that board find its possible neighbors (all the possible next moves from that position). Once those are found pick the best move and do the same as the start. Simulated annealing is quite different. Simulated annealing you generate a new random board, based on that you randomly select a queen to move to a random open spot. Assess that move, if it produces less attacking make the move, if not base on the probability that you're accepting a bad move make the move anyway.

For BFS the results are interesting as the board gets bigger it takes a lot longer to find a solution. I was only able to run for n=8. At n=9 I would run out of memory having 15Gb. My swap memory is at 24Gb but I'm not sure of how much of that was being used or not. After looking at the results of memory taken up, I believe that if I made the algorithm more efficient using a single array like I did for Hill climbing and SA then it might be able to perform the calculations better. Here are the results of BFS:

	Best Time	Worst Time	# of Solutions
n=1	.0001s	.0002s	1
n=2	.0004s	.0008s	0

n=3	.0011s	.0018s	0
n=4	.0041s	.0154s	2
n=5	.0533s	.0626s	10
n=6	.6831s	.6994s	4
n=7	12.3655s	12.445s	41
n=8	4m 24s	4m 27s	93
n=9	-	-	-

By these numbers I believe that to find n=9 it would take about 24 minutes. To predict how long it would take to find the answer to n=30 given that for every n=17 the solutions go up 10^7+1 one can assume that for n=30 there are over a Quintillion solution. That is 1*10^18 so it will take that many years if not longer to calculate that many solutions using BFS.

BRF with pruning produces a lot of better results naturally. Unlike BFS there is no need to travel down a branch that is known not to produce a possible solution. By making this elimination it saves both time and memory. So, in the algorithm once a move is made it is evaluated if it's a valid move then it will save it for future possible move, if not, it will not return that board into the deque/queue. Here are the results:

	Best Time	Worst Time	# of Solutions
n=1	.0003s	.0003s	1
n=2	.0004s	.0004s	0
n=3	.0004s	.0009s	0
n=4	.0044s	.0045s	2
n=5	.0132s	.0335s	10
n=6	.0323s	.0352s	4
n=7	.1288s	.1383	41
n=8	.7052s	.7116s	93
n=9	3.786s	3.8315s	353
n=10	21.298s	21.384s	725
n=11	2m 22.37s	3m 35s	2681
n=12	-	-	-

With BFS with pruning n=8 got significantly better which would leave me to believe that finding n=30 would take a little less time but not by much.

Hill climbing becomes an interest results as well. Although, I'm not sure if I did this part 100% correctly at n=11 it doesn't produce any results for me. Not sure if it gets stuck into local maximum or if it just takes a lot more time at that level. The results are:

	Best Time	Worst Time	# of Solutions
n=1	.0003s	.0003s	1
n=2	-	-	0
n=3	-	-	0
n=4	.0019s	.0208s	2
n=5	.0132s	.0309s	10
n=6	.0844s	.343s	4
n=7	.1748s	1.9156s	41
n=8	9.1121s	17.173s	93
n=9	1.514s	27s	353
n=10	23s	-	725
n=11	-	-	2681

As you can see hill climbing produced some spiritic results. For n=9 one time it took 1.514 seconds and another 27 seconds. I believe the difference is because depending on which board the algorithm is going on it will get stuck at local maximum. Depending on how long it takes it to break out will determine the time being better than others.

The last algorithm that is programmed is Simulated annealing. SA works by starting off with initializing a random board. Once that random board is initialized it is sent to the SA function where it picks a random queen to move to a random spot. Then it checks the fitness of that and compares it to the previous state. If it's a better fitness, then it will choose that one. If not base on the probability that it's going to pick a worst one. It keeps doing this until the temperature cools. Once it cools down it starts again with a higher temperature. These are the results for SA:

	Best Time	Worst Time	# of Solutions
n=1	.0003	.0003	1
n=2	.0002s	.0002s	0
n=3	.0001s	.0001s	0

n=4	2.8538s	27.3299s	2
n=5	5.0067s	7.5864s	10
n=6	1.5614s	2.0435s	4

Not sure if I made simulated annealing algorithm correctly but getting n=7 has not produced any results while the others have. I am aware that it is supposed to be the best algorithm for big solutions, I'm just not getting those results.

In conclusion BFS with pruning is better than BFS since you do not have to search through branches that are known not to produce valid solutions. Hill climbing does better than BFS with pruning. Instead of searching through solutions, starting off with a random board give you a better start point to a possible solution. Sometimes you might end up with a solution off the first round and others may take a few seconds more. I believe Simulated annealing would be best, but I may not have programmed it correctly because it does not perform as expected. As pointed out before, the amount of time to find a solution depends on the board size. For a board size of n=11 thru n=13 it is 1x10^3 solutions. For n=15, its 1x10^6. For n=16 and 17, 1x10^7 and after that the exponent increments by one for every board increased.