

Assignment 1: N-Queens

2802ICT



March 23, 2019

S5048240

Samuel Grumley

Contents

[Part A: 2](#_Toc4366605)

[Part B: 3](#_Toc4366606)

[Hill climb: 3](#_Toc4366607)

[Simulated Annealing: 3](#_Toc4366608)

[Comparison 3](#_Toc4366609)

[Results Part 1: 4](#_Toc4366610)

[Comparative results 4](#_Toc4366611)

[Visual Results 4](#_Toc4366612)

[BFS (no pruning) 4](#_Toc4366613)

[BFS (with pruning) 5](#_Toc4366614)

[Results Part B 7](#_Toc4366615)

[Simulated Annealing - Cooling rate comparison 7](#_Toc4366616)

[Simulated Annealing - Temperature comparison 7](#_Toc4366617)

[Comparative results 8](#_Toc4366618)

[Visual Results 9](#_Toc4366619)

[Simulated Annealing 9](#_Toc4366620)

[Highest search 12](#_Toc4366621)

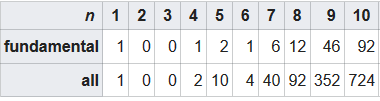
# Part A:

## BFS:

The BFS search solution to the n-queens problem works by iteratively adding a queen in the next available spot and then adding into a queue to be evaluated when it is at the front of the queue. Each time the node is evaluated, if there is an available spot the node will spawn new nodes (child nodes) into the queue to be evaluated at the next depth. This method alone without pruning or limitations will run every combination (brute force) the solutions and takes impractical amounts of time. To reduce time the first condition set was that if the number of queens on the current node was above n, stop searching and return the results. Since no state after that will even be valid within the problem. This at least allowed the problem to be computed and results to be shown although only for n=4, anything higher took too long to compute. This was still a very inefficient algorithm and required pruning. This was achieved by evaluation the state for how many collisions were made, if there was more than one collision, no permutations of that state would ever result in 0 collisions and therefore was abandoned. This increased the time by huge amounts.

## Prediction

(Note: my program could not find all solutions for anything above n = 8 within a decent amount of time. Therefore, my number of solutions for anything above is from an online source <https://en.m.wikipedia.org/wiki/Eight_queens_puzzle> )



Using the data above we can estimate the amount of solutions for n = 30

Polynomial regression best fits the nonlinear relationship between the number of solutions and n, and therefore used to determine the amount of solutions for n=30

Polynomial regression:

y = 18.10606061 x2 - 141.3545455 x + 202.8666667

y=18.106061(30^2) − (141.354546)(30)+202.866667

y=12257.684851

# Part B:

## Hill climb:

Since this part of the discussion is based on comparisons of algorithms The Stochastic hill climbing algorithm has been adopted in place of the standard hill climb search. The Stochastic algorithm will in most cases reach the goal state giving the time ran data equal value to other algorithms. The algorithm works as a standard hill climb until it hits a local maximum and the cost function cannot be improved, from here it takes a single queen and randomises its position within its row. Then continues its climb until the goal state is reached.

The function ‘countControl’ is responsible for evaluating if the search is stuck at local maximum it determines this based upon the heuristic plateauing over two iterations, it will then replace the queen and continue to monitor the progression of the heuristic.

## Simulated Annealing:

The simulated annealing expands on the hill climb implementation by implementing it based on the probability that the next node will be a good move. This algorithm makes bigger moves earlier but smaller ones as it starts to get close to the goal state. It tries to stay at global maximum but reducing big moves. The annealing function will check for the best move and compare it with the current board (delta = eval (next Node) – eval(current node)) this is then put into the formula e^(delta/Temperature). Temperature and cooling factor are chosen values that work with delta to create give the probability of accepting the move. If there are no best moves, like the random restart the queen will be temporarily given a random spot in its column and the same probability process will happen to see if the move should happen. After each move the cooling factor is multiplied by the temperature to slow down the probability of big moves as time goes on.

## Comparison

Simulated annealing was significantly faster in each search but really started to show the difference as n got higher. There was quite a bit of variance between each search due to the board starting in a random state. To combat this, tests were done in sets of 10 for each n value. Anything above n=24 took too long for the Hillclimb to compute, hence, this is where the tests stopped for testing in sets of 10. There was a 131 second difference between the averages of both results for n = 24, demonstrating that for this problem Simulated Annealing was the superior algorithm. Due to the nature of Simulated Annealing there was a lot more tests involved in fine tuning the parameters to achieve this result. Under the headings: Cooling rate comparison and temperature comparison you can find the data comparing the SA parameters. Although faster speeds were achieved they were often with failures and to achieve the maximum n value the parameters used were the most efficient without any record of failure. This led to n=45 being the highest value calculated successfully. Both the start and goal states can be found in the .txt files attached to the submission. Comparatively, the Hillclimb only achieved n=30 as its highest successful computation (within time constraints). The states of this can be found at the bottom of this document under the heading Highest Search.

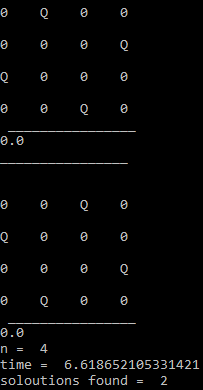
# Results Part 1:

## Comparative results

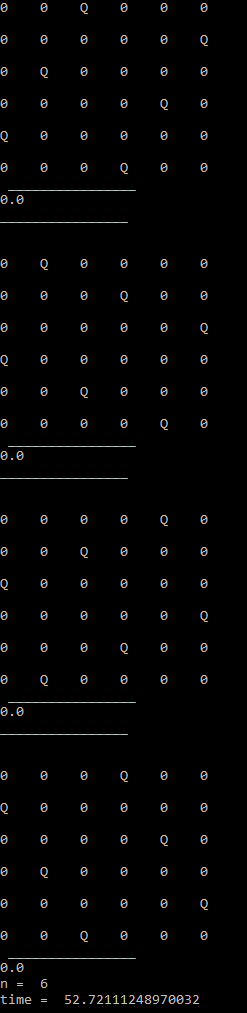
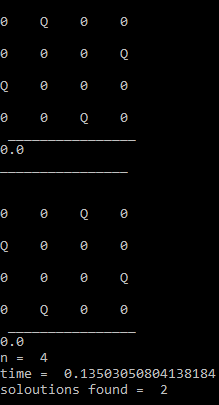
|  |  |  |
| --- | --- | --- |
| n | pruning | No pruning |
| 4 | 0.135 | 6.619 |

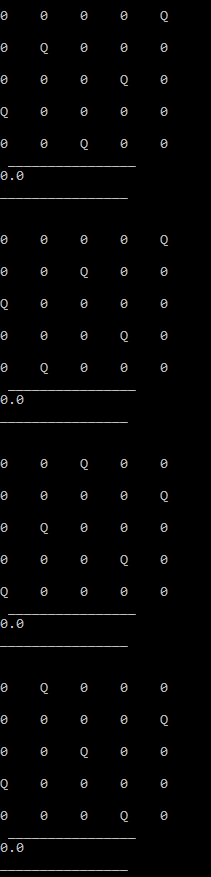
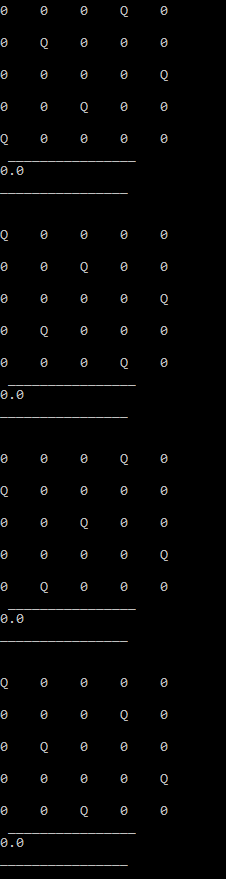
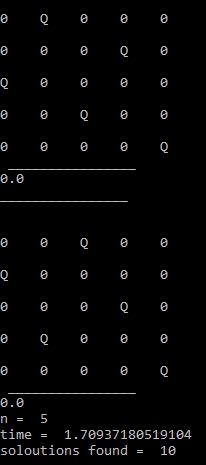
## Visual Results

### BFS (no pruning)



### BFS (with pruning)



# Results Part B

## Simulated Annealing - Cooling rate comparison

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N=25 | Iterations = 10 | | Temperature = 5 | | Cooling rate = 0.95 |
| Total time | | Average time | | fails | |
| 742.378 | | 74.238 | | 0 | |

Min = 17.565 max = 178.990

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N=25 | Iterations = 10 | | Temperature = 5 | | Cooling rate = 0.99 |
| Total time | | Average time | | fails | |
| 779.018 | | 77.902 | | 0 | |

Min = 13.297 max = 161.961

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N=25 | Iterations = 10 | | Temperature = 5 | | Cooling rate = 0.90 |
| Total time | | Average time | | fails | |
| 524.147 | | 52.415 | | 0 | |

Min = 29.932 max = 106.361

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N=25 | Iterations = 10 | | Temperature = 5 | | Cooling rate = 0.85 |
| Total time | | Average time | | fails | |
| 506.075 | | 50.608 | | 1 | |

Min = 20.328 max = 115.532

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N=25 | Iterations = 10 | | Temperature = 5 | | Cooling rate = 0.80 |
| Total time | | Average time | | fails | |
| 434.131 | | 43.413 | | 3 | |

Min = 13.107 max = 111.876

## Simulated Annealing - Temperature comparison

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N=25 | Iterations = 10 | | Temperature = 2 | | Cooling rate = 0.90 |
| Total time | | Average time | | fails | |
| 433.527 | | 43.353 | | 0 | |
| Min = 10.305 | | | Max = 94.038 | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N=25 | Iterations = 10 | | Temperature = 5 | | Cooling rate = 0.90 |
| Total time | | Average time | | fails | |
| 524.147 | | 52.415 | | 0 | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N=25 | Iterations = 10 | | Temperature = 10 | | Cooling rate = 0.90 |
| Total time | | Average time | | fails | |
| 558.106 | | 55.811 | | 1 | |
| Min = 9.634 | | | Max = 152.385 | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N=25 | Iterations = 10 | | Temperature = 15 | | Cooling rate = 0.90 |
| Total time | | Average time | | fails | |
| 513.07 | | 51.307 | | 1 | |
| Min = 18.130 | | | Max = 84.917 | | |

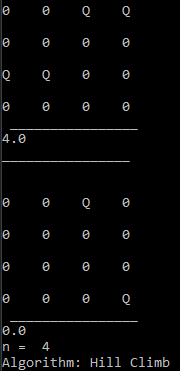
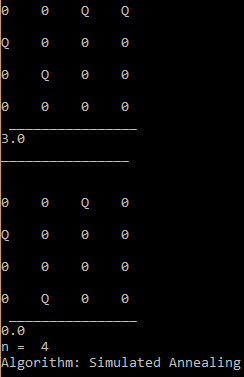
## Comparative results

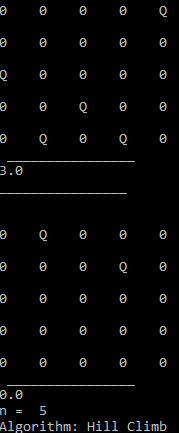
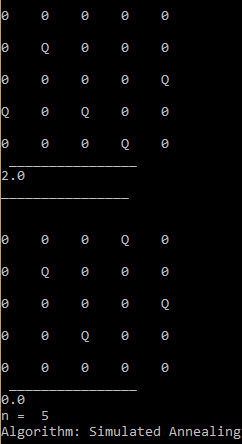
The average is of 10 runs

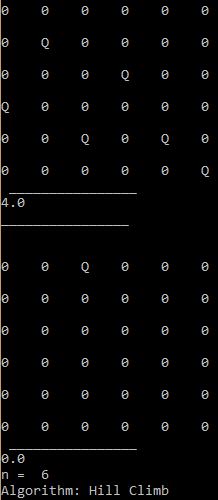
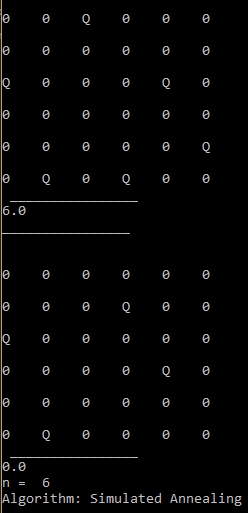
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| n | Simulated annealing | | | Hillclimb | | |
| Average | Max | Min | Average | Max | Min |
| 4 | 0.043010 | 0.01000142 | 0.0019922 | 0.006100 | 0.014002 | 0.00195 |
| 6 | 0.374089 | 0.12004637 | 0.0059890 | 0.059013 | 0.096034 | 0.01600 |
| 8 | 1.526343 | 0.37408399 | 0.0510118 | 0.223115 | 0.321084 | 0.13301 |
| 10 | 5.444585 | 0.94321179 | 0.2080514 | 1.018630 | 1.294271 | 0.64916 |
| 14 | 35.49992 | 6.98328518 | 0.6341431 | 8.254107 | 11.31355 | 5.69027 |
| 18 | 114.3264 | 29.2724370 | 3.2916908 | 28.78707 | 40.01099 | 20.8766 |
| 24 | 429.5307 | 100.822025 | 12.833143 | 173.9199 | 244.0224 | 126.701 |

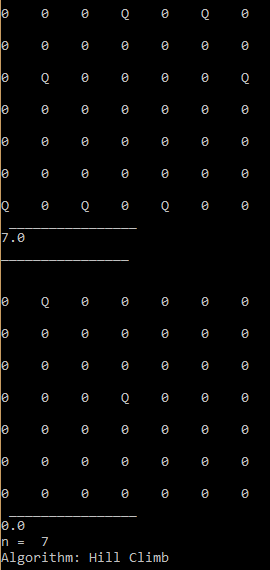
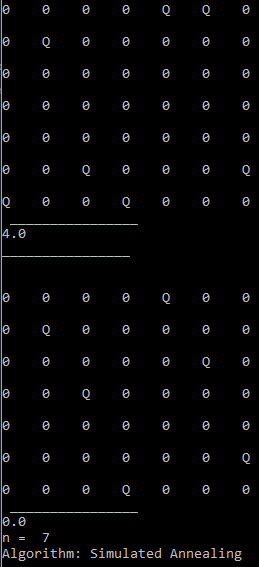
## Visual Results

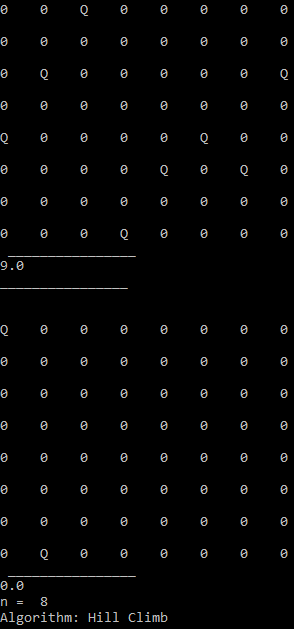
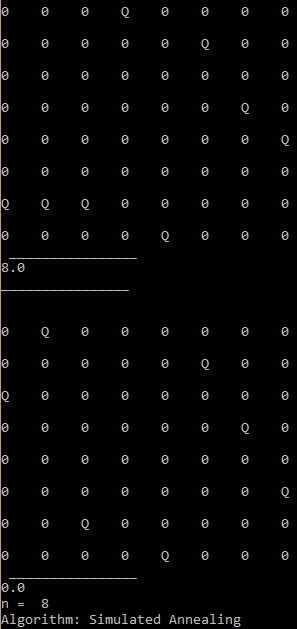
### Simulated Annealing

## Highest search

Start State (Hillclimb):

Goal State:

