|  |
| --- |
| 2802ICT Intelligent systems |
| N-Queens Problem |
|  |
|  |
| **Tomas Sheiles-Hutton s5132012** |
| **Trimester 1 2019** |

|  |
| --- |
|  |

# Introduction

Originally developed in the 1940’s by Max Bezzel, the 8 queens problem has since become a staple problem in computer science. The original problem involved determining the number of permutations of a chess board populated with eight individual queens such that no queen could take another in a single move. While a relatively simple concept, due to the large number of possible arrangements of such a board combined with the large freedom of movement allowed by the queen piece; determining the total number of solutions for a given size of chess board can quickly become too time and memory expensive for computers to produce a solution using only brute force methods. In order to effectively compute the problem, an algorithm must be used to reduce the state space of the problem and effectively traverse the possible solutions. It is for this reason that the problem can be used to evaluate the effectiveness of a particular search algorithm and quantitatively measure its performance when solving problems with very large search spaces. For this report four separate algorithms were implemented using python for the purpose of analysing how each distinct approach performs when solving the queens problem.

# Part A: Uninformed searches

Uninformed search strategies are characterized by the fact that they generate and traverse a problem without any additional or specific information relating to the current state of any particular graph. The order in which uninformed algorithms traverse a problem is not altered by any heuristic specific to the problem being solved. Because of this, in order to ensure all possible goal states are reached, every node in the graph must eventually be explored to produce the optimal solution dramatically increasing the computational cost needed to complete the search.

## Breadth First Search

Breadth first search is an uninformed method of traversing a tree graph. The method used in this algorithm comprehensively explores the problem set by expanding the children of an origin node and then exploring each in order of its addition to the queue before exploring the children of the second level in the graph (First in First out).

In terms of time and space complexity, this algorithm increases exponentially in both cases ( where b is the width of the graph and d is the depth of each branch) because of this exponential growth, the algorithm either quickly reaches the limits of the computer’s memory or completion times far exceeding reasonable levels. It is however complete and will explore every available option given a finite graph.

For this specific implementation of the algorithm, an array of equal length to that of the board was used to represent its current state; where the vertical columns of the board are represented by the array’s value and the horizontal rows represented by the position of each value in the array; this was used to both reduce memory requirements of other implementations and to eliminate the need to verify the position of the queens on the horizontal axis. The algorithm then iteratively generates all child nodes of the origin node and adds them to a FIFO queue. Once the nodes are removed from the queue, the validity of each one is individually verified by first ensuring that no two nodes share the value (and thus the same vertical axis) or the same diagonal by calculating the gradient between each pair of queens on the board () and comparing to 1 or -1 to determine if a straight diagonal line could pass between them. All solutions are then appended to a separate array of goal states and the program continues until all nodes are explored.

### Solutions using BFS:

|  |  |  |
| --- | --- | --- |
| **N** | **Solutions** | **Time to complete in seconds** |
| 1 | N/a | N/a |
| 2 | 0 | 0 |
| 3 | 0 | 0 |
| 4 | 2 | 0.0009 |
| 5 | 10 | 0.007 |
| 6 | 4 | 0.10 |
| 7 | 40 | 2.18 |
| 8 | 92 | 50.12 |
| 9 | N/a | N/a |

N = 1: while technically N=1 has a single solution of a single piece on the only available space, as no other queens can be placed, it is not counted as a valid solution.

N = 2 & 3: For both of these sizes, no valid solution exists where all pieces can be placed on the board.

N = 9: Once the algorithm reaches this size, the exponential amount of memory required to hold the graph exceeds the capabilities of the computer’s hardware.

## Depth First Search

Similar to Breadth first search, depth first search is also an uninformed method of searching a graph. However it has a number of key distinctions that separates it from Breadth First Search. The main differentiation between the two is the order in which the nodes in the tree are explored. Depth first algorithms use a Last in First Out stack to store and traverse the problem, meaning the entire extent of the first added nodes children is explored before the second original child is.

Because of this differentiation, the complexity of the algorithm is different. While it maintains an exponential time complexity the space complexity becomes linear where b is the width and d is the depth of the graph. The implications of this are that depth first algorithms have the potential to be run for longer periods of time as the amount of memory used does not increase significantly overtime. This however forgoes the completeness of the breadth first solution as it has the possibility to become stuck in infinite loops.

This implementation of the depth first search uses similar methods to that of the breadth first. The main difference is the order in which nodes are added to the stack. The nodes are removed from the beginning of the stack and they are, each child node is appended to replace its position.

### Solutions using DFS:

|  |  |  |
| --- | --- | --- |
| **N** | **Solutions** | **Time to complete in seconds** |
| 1 | N/a | N/a |
| 2 | 0 | 0 |
| 3 | 0 | 0 |
| 4 | 2 | 0.0009 |
| 5 | 10 | 0.007 |
| 6 | 4 | 0.10 |
| 7 | 40 | 1.857 |
| 8 | 92 | 39.07 |
| 9 | 352 | 936.73 |
| 10 | 724 | 25473.99 |

N = 9 & 10: Due to the fact depth first solutions are linearly space complex, these two solutions could be determined but the exponential time constraints limited higher number solutions.

## Predicting Larger board sizes

Using the results obtained from both the breadth and depth first algorithms, educated predictions can be used to estimate the number of solutions as well as the time it would take to compute the problem for significantly larger sized boards. This estimate can be used to overcome the limitations placed on the algorithm due to time and memory constraints and obtain a relatively accurate answer for these larger problems. Because both of these algorithms exponentially expand both in time to complete and number of valid solutions an exponential curve can be added to the data and then used to calculate the estimated answers to larger problems.

### BFS:

Use exponential trend line

* Predict the time to complete N=30
* Suggest way to prune the results