A REPORT ON THE Nqueens Visualizer

Submitted by

Vishal Kumar

12222786

Submitted to

Rahul Rajput

"School of computer science and Engineering"

Lovely Professional University

Phagwara, Punjab

ABSTRACT

Graphical simulation is an attempt of predicting the aspects of behavior of some system by developing an approximate model of it. Simulations have great impact on education and in training. Simulation based learning is a practical way of learning of practices that involves building connections: connections among what is being learned and what is important to the actor and the situations in which it is applied.

N-Queens problem refers to the problem in which one has to place N-Queens on an n*n chess board such that no queen is attacking the other, i.e. no two queens occupy the same row, column or diagonal. Here we use graphical simulation to view various solutions to N-Queens problem. The n-queens problem is implemented by using C++ and OpenGL.

The graphical simulation is used because n-queens problem is more complicated. It is easy for any system user or normal user to understand this problem by observing simulations. With the help of simulation the problem is explained very clearly and effectively. This will help in generating interest among perceivers because it has real time applications and will make learning better.

Key words: graphical simulation, n-queens problem,

Table of Contents

1 Introduction

1.1 Graphical Simulation

Simulation-based learning is budding demand for learning dynamic aspects. Learning about dynamic phenomena is essential in many domains. That approach allows the learner to imitate real processes using models and to acquire knowledge and experience during the process of learning.

The proposed graphical modelling method is based on systematic point of view, experiential learning and expertbased modeling requirements. The contextual graph of action specification is used as a basis to set the requirements for service software specification and attributes of learning objects (LO).

1.2 Description of the Problem

The n-queens problem, originally introduced in 1850 by Carl Gauss, can be stated as follows: find the placement of n queens on an n×n chessboard, such that no one queen can be taken by any other. N solutions were proposed but later numerous solutions were published relying on specific formula for placing queens.

The eight queens puzzle is an example of the more general n-queens problem of placing n queens on an n×n chessboard, where solutions exist only for n = 1 or n ≥ 4.The eight queens puzzle is the problem of placing eight chess queens on 8x8 chessboard so that none of them can capture any other using the standard chess queen's moves. The queens must be placed in such a way that no two queens attack each other. Thus, a solution requires that no two queens share the same row, column, or diagonal.

1.3 Motivation

This paper is to simulate the N-Queens problem that is used in the context of placing n-queens on an n*n chess board. It is good to animate the n-queens problem because this topic will illustrate how we can get the more graphically simulated placement of queens on a chess board such that no two queens lie along the same row, column or diagonal.

The graphical simulation is used to demonstrate more practical problems whose solutions are permutations, of which there are many. One such problem is the travelling salesperson problem: Find the shortest route on a map that visits each city exactly once and returns to the starting city. Here the permutation is the list of cities that are visited, except for the first. It can be done by making the animation in step by step which can be easily understood by any system user or normal user. With the help of visualization the problem is explained very clearly.

1.4 Objectives

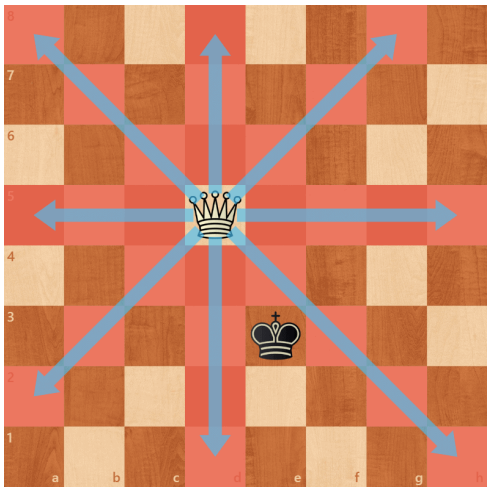
Backtracking is a prime algorithmic technique that uses depth first search to explore the solution space which is most naturally formulated in terms of recursive programs.
Suppose that we are given an n × n chessboard, where n is a positive integer.A queen can attack in horizontal, vertical, and the two diagonal directions. The nqueens problem asks to find a configuration of n queens on an n × n chessboard such that no two attack one another. There are no solutions to this problem for n = 2 or 3, but it can be shown that a solution exists for each n 4. The main idea is to proceed row by row. Suppose that the values of row[k] for all 0 k < I, are selected then the value for row[i] that is consistent with the requirements is selected that have outlined above. If that is not possible, then backtracking is done and different value for row[i-1] is chosen.

2 . N-QUEENS PROBLEM

The puzzle was originally proposed in 1848 by the chess player Max Bezzel, and over the years, many mathematicians including Gauss have worked on this puzzle and its generalized n-queens problem. The first solutions were provided by Franz Nauck in 1850. Nauck also extended the puzzle to n-queens problem (on an n×n board—a chessboard of arbitrary size). In 1874, S. Günther proposed a method of finding solutions by using determinants, and J.W.L.Glaisher refined this approach. Edsger Dijkstra used this problem in 1972 to illustrate the power of what he called stuctured programming. A highly detailed description of the development of a depth-first backtracking algorithm is published by him.

2.1 Problem Statement

This problem is not an interesting part of computer science, but it exemplifies the general pattern for the recursive search and backtrack programs. The n-queens problem consists of n*n chess board, the queen is the most powerful of all chess pieces. n queens are to be placed on the chessboard such that no queen can attack the other. This is viewed as follows:



n of n-queens problem varies from 1 to infinity .There exist no solution for some n queens problem like n=2 and for larger n there arises memory,speed and computational problems for solving.

Generally, the puzzle has a unique solution. There are certain techniques to solve the puzzle by hand and these rules can be implemented into a computer program. These techniques are presented in more detail in 2.3.

2.2 Examples

2.2.1 One Queen problem

One queen Solution

2.2.2 Four Queens Problem

There exists only two distinct solutions.

2.3 Why backtracking?

Most often,n queens problem is used as an example of a problem which can be solved with a recursive algorithm, by phrasing the n queens problem inductively in terms of adding a single queen to any solution to the problem of placing n−1 queens on an n-by-n chessboard. The induction bottoms out with the solution to the 'problem' of placing 0 queens on an n-by-n chessboard, which is the empty chessboard.

This technique is more efficient than the brute force search algorithm, which considers all 648 = 248 = 281,474,976,710,656 possible blind placements of eight queens, and then eliminates to remove all placements that place two queens either on the same square (leaving only 64!/56! = 178,462,987,637,760 possible placements) or in mutually attacking positions. This very poor algorithm producing the same results over and over again in all the different permutations of the placing eight queens, as well as repeating the same computations for the different sub-sets of each solution. A better brute-force algorithm places a single queen on each row, leading to only 88 = 224 = 16,777,216 blind placements.

One algorithm solves the eight queens puzzle by procreating the permutations of the numbers 1 through 8 (of which there are 8! = 40,320), and uses the elements of each permutation as indices to place a queen on each row. Then itpermutation as indices to place a queen on each row. Then it eliminates those boards with diagonal attacking positions.

The backtracking depth-first search program,an improvement over the permutation method, constructs the search tree by considering one row of the board at a time, eliminating the most non solution board positions at a very early stage in their construction. Because it avoids rook

and diagonal attacks even on incomplete boards, it examines only 15,720 possible queen placements.

A further improvement which observes only 5,508 possible queen placements is to combine the permutation based method with the early pruning method: the permutations are generated depth-first, and the search space is pruned if the partial permutation produces a diagonal attack. Constraint programming can also be very effective on this problem.

An alternative to exhaustive search is an 'iterative repair' algorithm, which typically begins with all queens on the board, for example with one queen per column. It then counts the number of attacks, and uses a heuristic to determine how to improve the placement of the queens. The 'minimum-conflicts' heuristic — moving the piece with the largest number of conflicts to the square in the same column where the number of conflicts is the smallest — is particularly effective: it finds a solution to the 1,000,000 queen problem in less than 50 steps on average. This assumes that the initial configuration is 'reasonably good' — if a million queens all start in the same row, it will obviously take at least 999,999 steps to fix it. A 'reasonably good' starting point can for instance be found by placing each queen in its own row and column so that it conflicts with the smallest number of queens already on the board.

Note that 'iterative repair', unlike the 'backtracking' search outlined above, does not guarantee a solution: like all hill climbing (i.e., greedy) procedures, it may get stuck on a local optimum (in which case the algorithm may be restarted with a different initial configuration). On the other hand, it can solve problem sizes that are several orders of magnitude beyond the scope of a depth-first search.

3. BACKTRACKING

Backtracking can be viewed as programmer's swiss army knife. We use trail and error to solve a problem using backtraking however, it is a well organized trial and error but an assurance that same thing cannot be tried twice must be made. Another assurance of trying all possibilities for finite problem must be made (assuming there is enough computing power for trying all possibilities).
Recursion combined with back tracking can be used to solve many problems. When a dead end is reached then backtrack to the last place where there was a choice in paths to take, and then take the other path (assuming there were only two possibilities). If even this turns out to be a dead end, backtracking to the previous point is made. The recursive solution can be achieved this ways. In case of large problems the problem is split into smaller parts by taking one step in the space - by making one move. The base case where we attain solution occurs when the property has been fully satisfied and we arrive at the exit.

3.2 Algorithm steps

The algorithm for planning n queens on an nxn chessboard using backtracking with recursion begins by placing a queen to the upper line of the table and continues by moving to the bottom. The steps of the algorithm are listed as follows:
1. initially place the first queen in the left upper corner of the
table.
2. save all the possible attack positions.
3. Move to the placement of the next queen (which can only be placed to the next line).
4. Search for a valid position. If there is one go to step 8.
5. else i.e if there is no valid position for the queen then it is deleted (the x coordinate is 0).
6. Move to the previous queen i.e backtrack.
7. Go to step 4.
8. place the queen in the first valid position .
9. Attack positions are saved.
10. If the queen processed is the last stop otherwise go to step 3.

This algorithm presented can be turned into a structured program by the addition of an infinite loop that includes steps 3 to 10 and can only be stopped on reaching step 10.

3.3 Details

The most important detail of the backtracking algorithm is the function that saves the attacked positions (marks the invalid locations for the rest of the queens). This part of the
algorithm mostly determines the speed and efficiency.
One simple approach is to mark queens position on a 2 dimensional matrix(array programming).Zero is used to represent no threat (valid spot) whereas
every other number of that matrix means valid location. Whenever a queen is to be placed the diagonals,rows/columns and lines it threatens should take the number of the line of that

queen(unless they already are not zero because another queen also
threatens them). Whenever a queen must be removed from line k all the numbers of the matrix that are equal to k should become zero (numbers equal to k are those that are threatened just by k queen so when this queen is removed they should
become zero(no threat)).

Another method that is better than the one mentioned above is to save the row /column and the diagonals that each queen occupies.The rows/columns are saved as boolean one dimentional array/matrix with true implying occupied row/column and false implying free row/column. The diagonals is saved on two boolean one dimentional array/matrix and access ed with x - y and x + y numbers (where x is the number of the row / column of the queen and y is the number of the line). The value assigned for x and y for each queen position:upper left queen has x = 1 and y = 1 and the lower right queen has x = N and y = N (for a table size of N). The diagonals with x – y accessed the positive slope whereas x + y accessed the negative slope diagonals. This method along with other minor optimizations yields a 20x times speed up over the
first method.

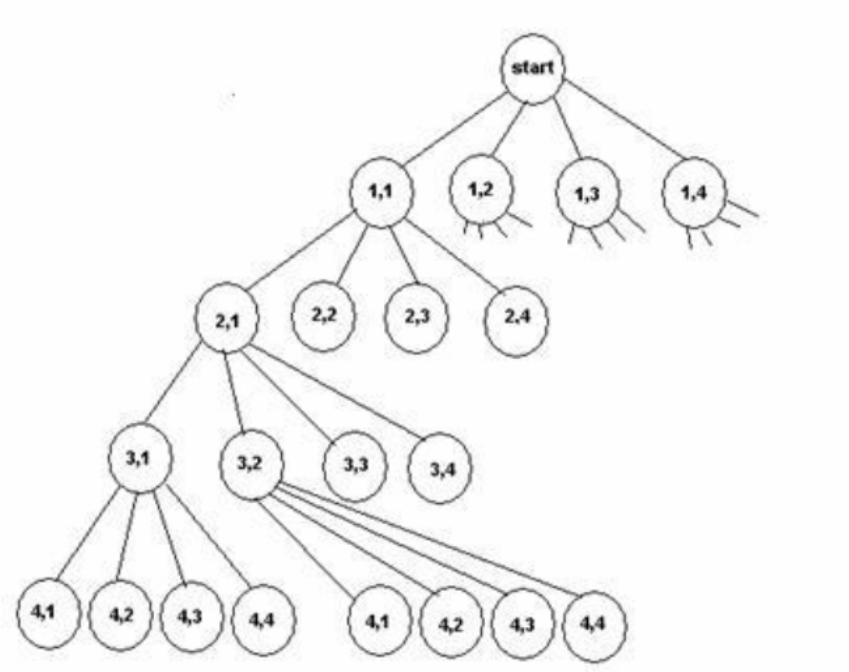
The last and probably the best way of marking queens positions is by the use of bitfields . The speed penalty of using the second method over the third is less than the penalty of using the first over the second method.

The time complexity of backtracking algorithm is exponential.

Backtracking is nothing but a modified depth-first search of a tree which consists of all of the possible sequences that can be constructed from the problem set. Perform a depth-first traversal of the tree until we reach a node that is non-viable or non-promising, at which point we "prune the subtree" rooted at this node, and continue the depth-first traversal of the tree.

The diagram below partially illustrates the state space tree for the instance of the N-Queens problem when N = 4. The ordered pair (i, j) within each node indicates the row,
column placement of a queen. Each queen can be placed in one of the N columns for each of the N rows, for a total of 4 x 4 x 4 x 4 = 256 candidate solutions are possible .

Therefore, there are NN candidate solutions in general.



State space for 4 queens

When there appears no child node that leads to a promising solution, the algorithm backtracks and removes the last queen places.

Placement of queens for N=8 can be as follows

Q . . . . . .
. . . Q . . . .
. Q . . . . . .
. . . . Q . . .
. . Q . . . . .
. . . . . .

Queens are placed safely without resulting in any attack in column 1-5 but there is now no safe square in column 6. Solution is begun by placing the first queen in column 1, row 1 followed by second queen column 2. obviously it cannot be placed in row 1, but, because of a diagonal threat from the queen in column 1, it cannot be placed in row 2 also. So it is placed in row 3. And continued attempts are made to place the next queen in column 3. The row there that is not threatened is row 5. Moving on to column 4, a queen in row 2 is placed and in column 5 a queen in row 4 is placed. But now in column 6 there is no square that is not threatened (see diagram)and it is known that the solution, must have a queen in each column.

So now backtrack to column 5 and find an alternative safe location, if possible. The only safe location possible is row 8 and trial to place a queen in row 6 with this change is made and failed again. So now backtrack to row 4 as row 5 has more options and so on. To view the recursion procedure used to solve the problem, start by placing a queen in the first column and call the recursive method to place a queen which, in turn, calls the same method to place the next queen, and so on until we reach the degenerate case where there is no columns in which a queen can be placed.

3.4 Efficiency

When place Queen(i) is called, i queens have been placed in rows 0 .. i -1. There are n(n - 1) … (n - i + 1) ways to place the first i queens when i > 0. When i = 0, there is 1 call to promising( ). Thus for each iteration of the for-loop in place N Queens( ), the number of "promising" nodes that are visited is less than or equal to:
{1 + n + n(n-1) + … + n(n-1) ….2}
and since the for-loop in place N Queens executes n times,

$$\#\text{ of promising nodes} \leq n\{1 + n + n(n-1) + \ldots. + n(n-1) \ldots 2\} = n\,(n!)\,\{1/n! + 1/(n-1)! + \ldots + 1/1!\}$$

$$= n\,(n!) \sum_{i=1}^{n} 1/i! \leq n\,(n!)\,(e-1) \quad \text{since} \sum_{i=0}^{\infty} 1/i! = e$$

This bound must be multiplied by O(n) -- the run-time of promising( ) in our algorithm,

4.CONCLUSION

Various techniques like brute force, permutations, greedy methods which can be used to solve n queens problem but among all backtracking with recursion is very efficient.
Since backtracking with recursion is useful to solve the problem where the iteration is more prominent in finding the solutions, it is used to solve n queens problem.

Graphical simulation of n-queens problem using backtracking with recursion is successful to the extent of finding and displaying all the distinct solutions to the single digit values of n,while a constraint(whether to read single digit or double digit from the solution list to place queen) occurs for double digit values.
The graphical simulation assists learners in understanding the solutions possible for n queens problem using backtracking. The graphical interface is very efficient and enables the users to visualize every solution of n queens problem.

5.REFERENCES

[1].Bitner, J.R. and E.M. Reingold (1975), "Backtracking programming techniques," Communications of the ACM, Vol. 18, No. 11, pp. 651-56. [2].Gauss and the Eight Queens Problem: A Study in Miniature of the Propagation of Historical Error, Campbell, Nov. 1977, Historia Mathematica, Vol. 4 No. 4. [3].Purdom, P.W. and C.A. Brown (1983), "An analysis of backtracking with search rearrangement," SIAM Journal of Computing, Vol. 12, No. 4, pp. 717-33. [4].Bernhardsson, B. (1991), "Explicit solutions to the nqueens problems for all n," ACM SIGART Bulletin, Vol. 2, No. 7. [5]. Ellis Horowitz, Sartaj Sahni, Sanguthevar Rajasekaran "Fundamentals of Computer Algorithms,2nd edition", Universities Press, 2008. [6]. An Introduction To Programming And OO Design Using Java , J.Nino And F.A.Hosch, John Wiley & Sons [7]. Herbert Schildt "The Complete Reference Java, J2SE 5th edition", Tata McGraw Hill Edition,2007. [8]. Barry J.Holmes, Daniel T.Joyce "Object Oriented Programming with Java,2nd edition", Jones and Bartlett Publishers,2001. [9].Y.Daniel Liang "Introduction to Java Programming, 7th edition",Prentice Hall,2008. [10]. Big Java ,2nd edition, Cay Horstman, Wiley Student Edition , Wiely India Private Limited.