Analysis of N-Queen Problem

Upasana Ghosh
UE143110

U.I.E.T, Panjab University
Email: ghoshupasana05@gmail.com

AIM - This paper provides a comprehensive analysis of N-queen Problem using backtracking algorithm.

INTRODUCTION

The N–Queen is the problem of placing N chess queens on an n×n chessboard so that no two queens attack each other. A queen can attack other queen if they share the same row, column, or diagonal. Chess composer Max Bezzel published the eight queens puzzle in 1848. Franz Nauck published the first solutions in 1850. Nauck also extended the puzzle to the n-queens problem, with n queens on a chessboard of n × n squares. In 1972 Edsger Dijkstra used this problem to illustrate the power of what he called structured programming. He published a highly detailed description of a depth-first backtracking algorithm.

The N-Queen Problem is one of the most popular NP-hard problems. Researchers are in an agreement on the issue that a minimum requirement of any efficient algorithm is that, it runs in polynomial time that is O (n^c), for some constant c. To solve these problems in a reasonable amount of time, heuristic methods must be used. N-Queen problem is NP-hard problem.

APPROACH

The problem of N-Queen is solved using backtracking algorithm. In backtracking algorithms solution is build step by step i.e., one step at a time. If at some step it become clear that the current path cannot lead to a solution due to some boundary condition or the path has been fully explore, it goes back to the previous step i.e., backtrack and choose a different path. It is a refinement of the brute force approach as it systematically searches for a solution to a problem among all available options. It does so by assuming that the solutions are epresented by vectors (*v*1, ..., *vm*) of values and by traversing, in a depth first manner, the domains of the vectors until the solutions are found. General backtracking methods could solve N-queens problems for n up to 100. It is a traditional approach to solve the computational problems. For n-queen problem algorithms finds the solution.

Generalized Algorithm:

```
 Pick a starting point.
while (Problem is not solved)
For each path from the starting point.
check if selected path is safe, if yes select it  and make
recursive call to rest of the problem
If recursive calls returns true, then return true.
else undo the current move and return false.
End For
If none of the move works out, return false, NO SOLUTON.
```

Explicit constraints are the rules that restricts xi to take values from particular set *S*. Collection of all those tuples which satisfy explicit constraints are called as solution space of that set S. Implicit constraints are the rules that determine which of the tuples in the solution space satisfies the criteria function. Answer state are problems which satisfies implicit constraints. Backtracking algorithm determines problem solution by systematically searching the solution space for the given problem instance. The constraint used to solve N-Queen problem in this paper is i^{th} queen is placed at i^{th} row.

EXPERIMENT

In this experiment, three functions are created, that is, Nqueen(), place() and printing() functions. place() function checks whether the given position is suitable for the current queen. It return true if the selected queen can be placed in that position else return false. printing() function prints the result of N-Queen on the output screen. Nqueen() function calls place() to check whether the proposed position is safe to place the selected queen. If it is safe, then it places the queen at that location and recursively calls itself for next queen position, else it tries to place the queen in other available locations. If no suitable position is there to place the selected queen, then it backtrack and try to place the previous queen in some other safe location. This process continues until all the queens are placed in the chessboard. If none of the move works out it will return false indicating solution is not possible..

Platform used for compilation and execution:

Operating system: Window 10
RAM: 4GB
IDE: Dev-C++ version 4.9.9.2
Compiler: GCC 3.4.2

ALGORITHM

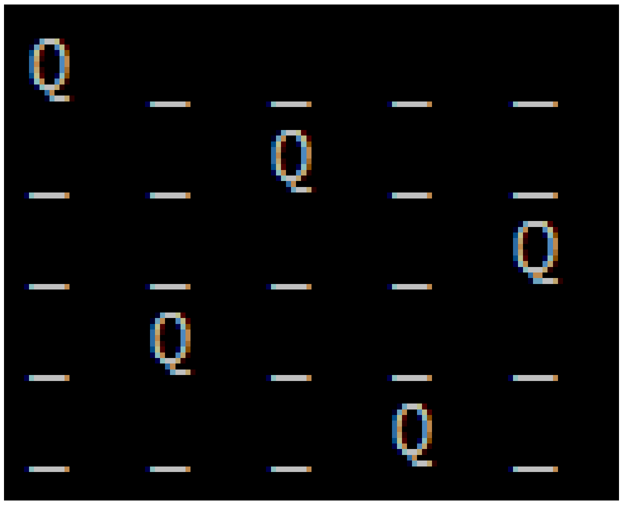
```
Algo NQueen(k,n)
{  for i = 1 to n do
      { if Place (k,i) then
            { x[k] = i;
               if(k = n) then
```

```
                write x
        else    NQueen(k + 1, n)
      }
  }}


Algo Place (k, i)
 { for j = 1 to k-1 do
    {if (x[j] = i ) or Abs(x[j]-i) = Abs(j-k))
         then return false;
      else
         return true;
   }
 }
```

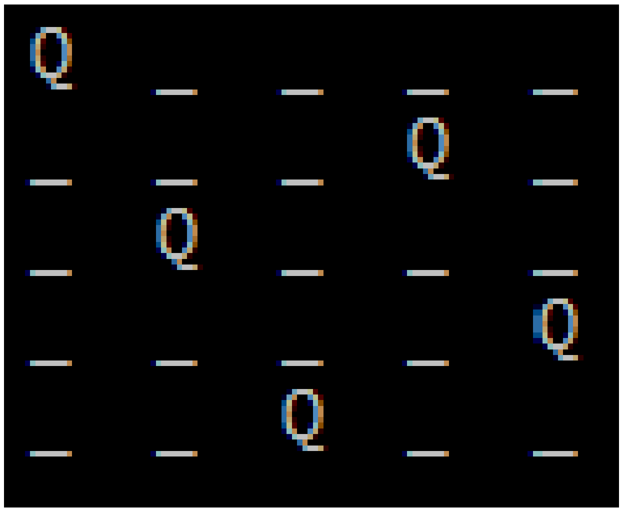
RESULT

Figure 1 to Figure 10 represents the position of 5 queens in a 5x5 chessboard. There are ten different ways to place a queen in a 5x5 chessboard.



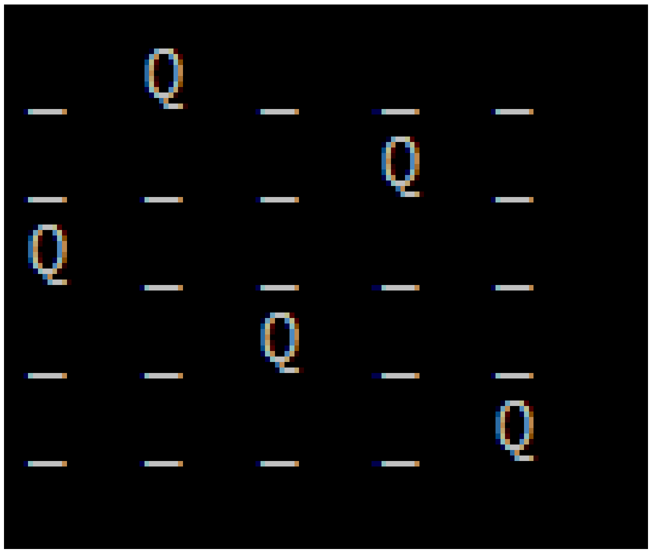
```
Q _ _ _ _
_ _ Q _ _
_ _ _ _ Q
_ Q _ _ _
_ _ _ Q _
```

Fig 1



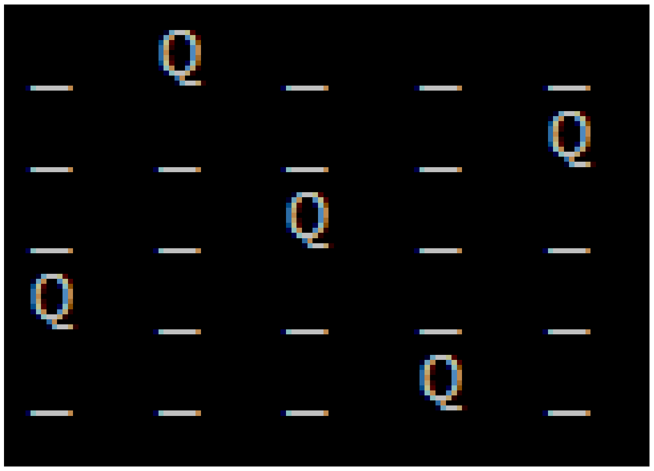
```
Q _ _ _ _
_ _ _ Q _
_ Q _ _ _
_ _ _ _ Q
_ _ Q _ _
```

Fig 2



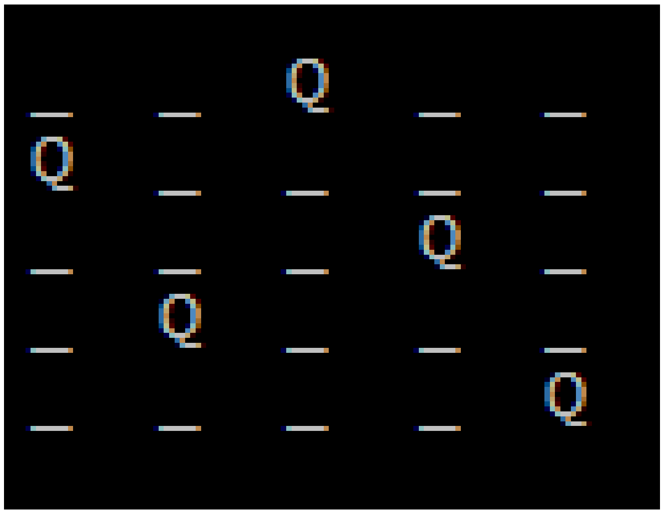
```
_ Q _ _ _
_ _ _ Q _
Q _ _ _ _
_ _ Q _ _
_ _ _ _ Q
```

Fig 3



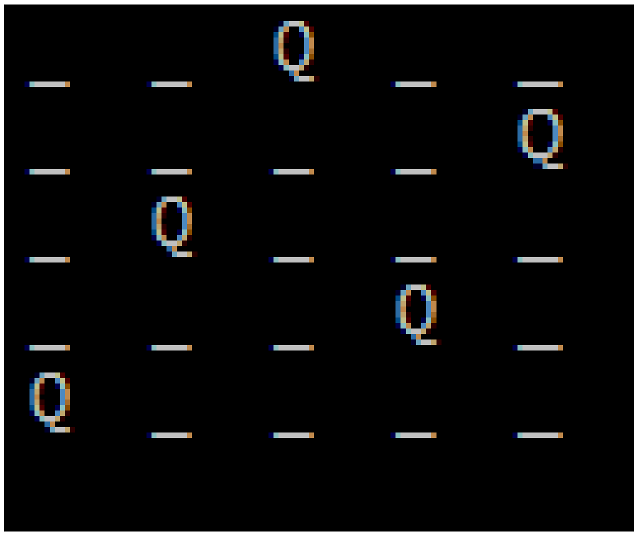
```
_ Q _ _ _
_ _ _ _ Q
_ _ Q _ _
Q _ _ _ _
_ _ _ Q _
```

Fig 4



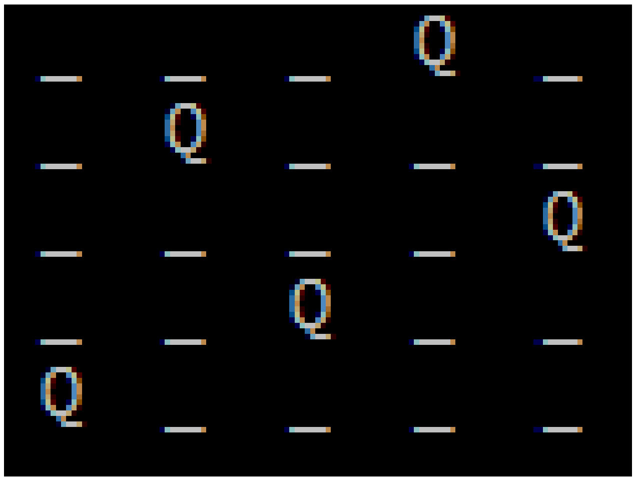
```
_ _ Q _ _
Q _ _ _ _
_ _ _ Q _
_ Q _ _ _
_ _ _ _ Q
```

Fig 5



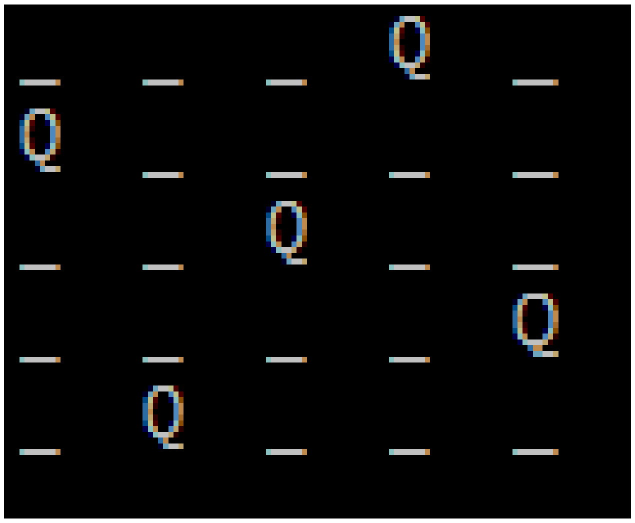
```
_ _ Q _ _
_ _ _ _ Q
_ Q _ _ _
_ _ _ Q _
Q _ _ _ _
```

Fig 6



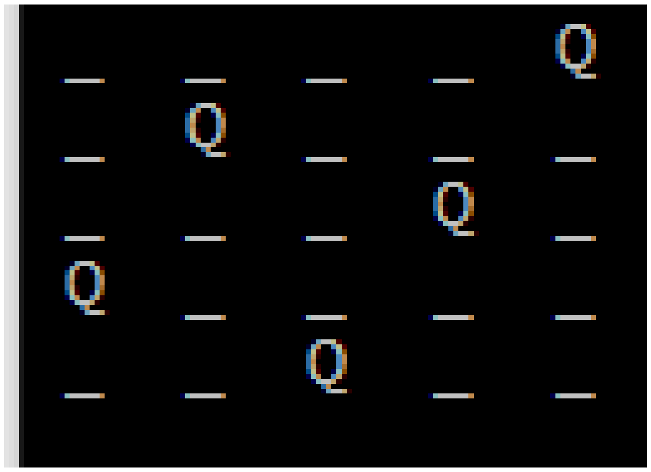
```
_ _ _ Q _
_ Q _ _ _
_ _ _ _ Q
_ _ Q _ _
Q _ _ _ _
```

Fig 7



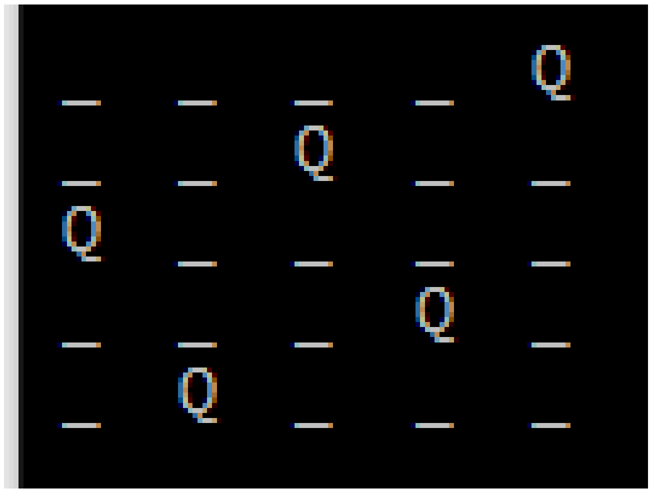
```
_ _ _ Q _
Q _ _ _ _
_ _ Q _ _
_ _ _ _ Q
_ Q _ _ _
```

Fig 8



```
_ _ _ _ Q
_ Q _ _ _
_ _ _ Q _
Q _ _ _ _
_ _ Q _ _
```

Fig 9



```
_ _ _ _ Q
_ _ Q _ _
Q _ _ _ _
_ _ _ Q _
_ Q _ _ _
```

Fig 10

DISCUSSION

Backtracking algorithm to solve N-Queens problem is one of the traditional method to solve this problem. Backtracking approach is a modification of the brute force approach, which systematically searches for a solution to a problem among all available options and by traversing, in a depth first manner until the solutions are found. The time complexity of N-Queen problem is $O(N^N)$ where N is the number of queen. But as the bounding conditions are not static, the problem size reduces from $O(N^N)$ to N! with the application of implicit and explicit constraints. This means that it would require less execution time for small dimension problems. Hence for small dimensions problem, backtracking approach is quite efficient and reliable. But its performance decreases as the complexity increases i.e., if the number of queens increases, the time complexity of backtracking algorithm also increases. This is because, as the number of queens increases, size of chess board also increases and hence the size of state space tree increases which results in larger computations. An interesting fact is that there is no guarantee that with the increase in number of queens, the number of solutions would also increase. For example, the number of solutions for 6-Queen puzzle is four which is lesser than the number of solutions for 5-Queen problem that is 10.

H. S. Stone and J. M. Stone in the paper “An empirical study of the n-queens problem” concluded that General backtracking methods is capable of solving N-queens problems for up to 100 queens. L. V. Kal e. in the paper “An almost perfect heuristic for the n non-attacking queens problem” introduced a specialized backtracking heuristic solutions that is capable of solving N-queens problems for up to 1000 queens. Matthew L. Ginsberg and David Mcallester in the paper "Gsat And Dynamic Backtracking” introduced dynamic backtracking algorithm but that does not solve constraint satisfaction problem dynamically. Gerald and Thomas in the paper "Dynamic Backtracking for Dynamic Constraint Satisfaction Problems." proposed some alteration to support a dynamic constraint satisfaction but this method suffers due to heavy time complexities.

Instead of using backtracking algorithm, dynamic programming can also be used to solve n-queen problem. I. Rivin and R. Zabih in the paper “A dynamic programming algorithm for the n-queens problem” introduced a dynamic programming for computing n-queen problem in O(f(n)8n), where f(n) is a low-order polynomial. The paper discussed about the combinatorial problems and search algorithm. The paper proves that this dynamic solution is better than backtracking method. Dynamic Programming gives better results for less number of queens. But it is inefficient for large number of queens. Some other algorithms have been introduced that are quite efficient for solving larger dimension problem. Ant Colony Optimization, DNA Sticker Algorithm, Genetic Algorithm, Gravitation Search Algorithm and Particle Swarm Optimization are some algorithms that are capable of solving this problem with even large number of queens efficiently. These Heuristic algorithms are modeled from physical and biological processes in nature. Theses algorithm operate randomly and search along with the space. These algorithms just use fitness function for guiding the search, but because of having collective intelligence, they are capable of finding solutions.

CONCLUSION

Backtracking algorithm is an effective method to find solutions for a problem which have more optional path to consider. But for larger number of queens, this approach is computationally inefficient. Hence, some other heuristic algorithm should be used for computing large dimensional n-queen problem.

REFERENCES

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stin *Introduction to Algorithm*, 3rd ed. The MIT Press, Cambridge, Massachusetts, London, England.
[2] Lijo V. P. and Jasmin T. Jose, “*Solving N-Queen Problem by Prediction* “, (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 6 (4) , 2015, 3844-3848.
[3] Amarbir Singh and Sandeep Singh Dhillon, “*A Comparative Study of Algorithms for N-Queen Problem”,* International Journal of Advance Foundation And Research In Science & Engineering (IJAFRSE) Volume 1, Special Issue, ICCICT 2015.
[4] H. S. Stone and J. M. Stone, “*An empirical study of the n-queens problem*, IBM, J.Res. Develop. July 1987.
[5] L. V. Kal e. in the paper “*An almost perfect heuristic for the n non-attacking queens problem”*, Information Processing Letters, Vol. 34:173{178}, April 1990.
[6] Gerald and Thomas in the paper "*Dynamic Backtracking for Dynamic Constraint Satisfaction Problems*”, In Proceedings Of The Ecai-94 Workshop On Constraint Satisfaction Issues Raised By Practical Applications, (1994).
[7] I. Rivin and R. Zabih, “A *dynamic programming algorithm for the n-queens problem*, Information processing letters 41(1992) 253-256.
[8] Ginsberg, Matthew L., And David Mcallester. "*Gsat And Dynamic Backtracking*.", *Principles And Practice Of Constraint Programming*, Springer Berlin Heidelberg, (1994).
[9] N-Queen Problem in *Wikipedia*. Retrieved April 20, 2016 from https://en.wikipedia.org/wiki/Eight_queens_puzzle