ASSIGNMENT AIR-4

Roll No: 41205

Problem Statement:

Implement crypt-arithmetic problem or n-queens or graph coloring problem (Branch and Bound and Backtracking)

Objective:

1. Understand backtracking and branch and bound algorithms
2. Apply the algorithms for popular problems such as n-queens

Outcome: One will be able to apply different techniques to solve the n-queens problem

Pre-requisites:

1. 64-bit Linux OS
2. Programming Languages: Python

Hardware Specification:

1. x86_64 bit
2. 2/4 GB DDR RAM
3. 80 - 500 GB SATA HD
4. 1GB NIDIA TITAN X Graphics Card

Software Specification:

1. Ubuntu 14.04

Theory:

- The N Queen is the problem of placing N chess queens on an N×N chessboard so that no two queens attack each other.
- The expected output is a binary matrix which has 1s for the blocks where queens are placed. For example, following is the output matrix for above 4 queen solution.
  {0, 1, 0, 0}
  {0, 0, 0, 1}
  {1, 0, 0, 0}
  {0, 0, 1, 0}
- Backtracking Algorithm
- The idea is to place queens one by one in different columns, starting from the leftmost column. When we place a queen in a column, we check for clashes with already placed queens. In the current column, if we find a row for which there is no clash, we mark this row and column as part of the solution. If we do not find such a row due to clashes, then we backtrack and return false.

1. Start in the leftmost column
2. If all queens are placed
   return true
3. Try all rows in the current column.
4. Do following for every tried row.
   a) If the queen can be placed safely in this row then mark this [row, column] as part of the solution and recursively check if placing queen here leads to a solution.
   b) If placing the queen in [row, column] leads to a solution then return true.
   c) If placing queen doesn't lead to a solution then unmark this [row, column] (Backtrack) and go to step (a) to try other rows.
5. If all rows have been tried and nothing worked,
6. Return false to trigger backtracking.

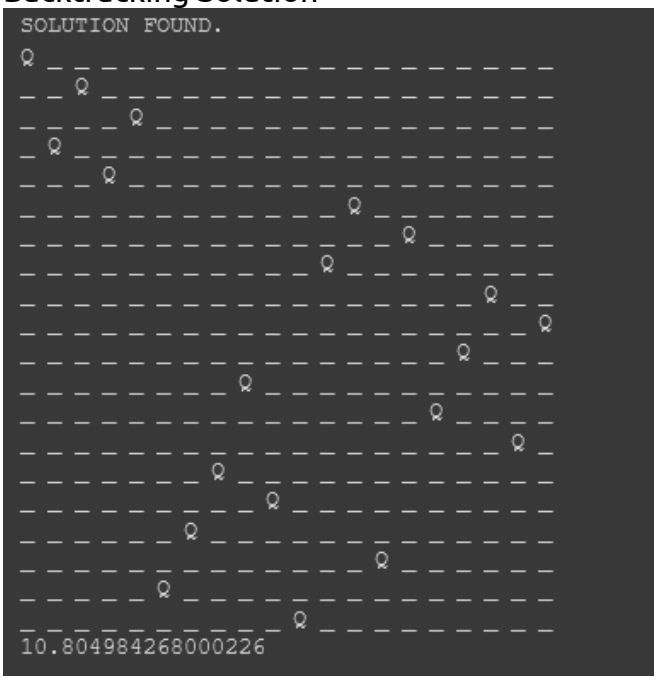
- For branch and bound, we maintain two additional vectors that indicate the diagonals that have been blocked.
- The lookup of any diagonal happens in O(1) time.
- This makes it much quicker for us to check if the position is valid or invalid.
- Thus, the branch and bound solution runs much faster than the backtracking one.

Test Cases:

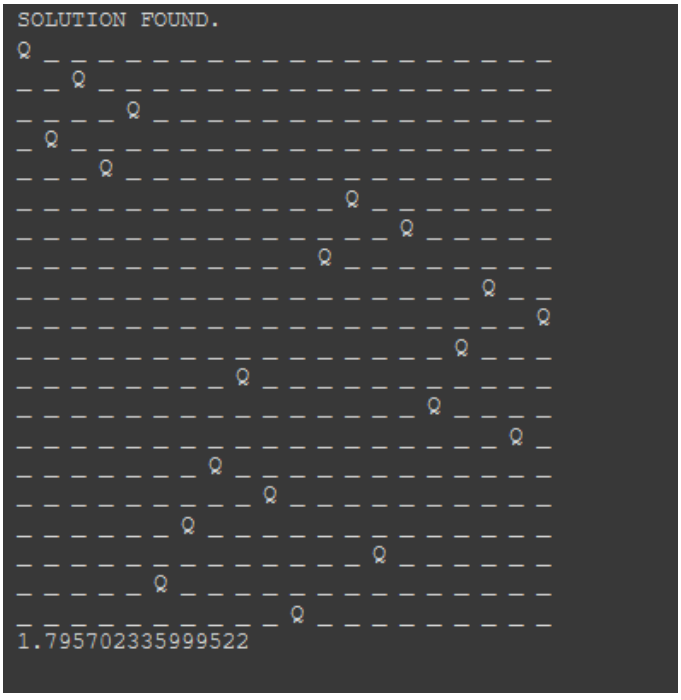
#	Input	Expected Output	Actual Output	Result
1	N = 8	Solved successfully Branch and bound faster than backtracking	Solved successfully Backtracking: 0.007 secs Branch and Bound: 0.002 secs	Success
2	N = 20	Solved successfully Branch and bound faster than backtracking	Solved successfully Backtracking: 10.8 secs Branch and Bound: 1.8 secs	Success

Output:

Backtracking Solution



Branch and Bound solution



Conclusion: Thus, we were able to solve the n-queens problem using both backtracking and branch and bound algorithms.