Project. Satisfiability test of clauses and its application

Team

1) Lohith Bhargav Doppalapudi, R11786637

2) Thulasi Priya Nallapothula, R11797226

3) Sri Ram Koppaku, R11842335

Intro:
The N-queens problem is about placing n-chess queens on an n*n chessboard so that no two queens are positioned in same vertical, horizantal, and diagonal. we represent the n*n chess board as matrix. Using Back Tracking the problem is solved.

Possibilities:
The two possibilities in solving NQueens problem are HillClimbing, Backtrack Algorithim

HillClimbing Algorithm: It is an iterative algorithm that starts with an arbitary solution to a problem, then attempts to find a better solution by incrementally changing a single element of the solution. This is a local search algorithm. The algorithm does not maintain a search tree, so the data structure for the current node need only record the state and the value of the objective function.

Backtrack Algorithm: If a queen is under attack at all the positions in a row, coloum, and diagonal we backtrack and change the position of the queen placed prior to the current position. We repeat this process of placing a queen and backtracking until all the N queens are placed successfully.

Pseudocode:

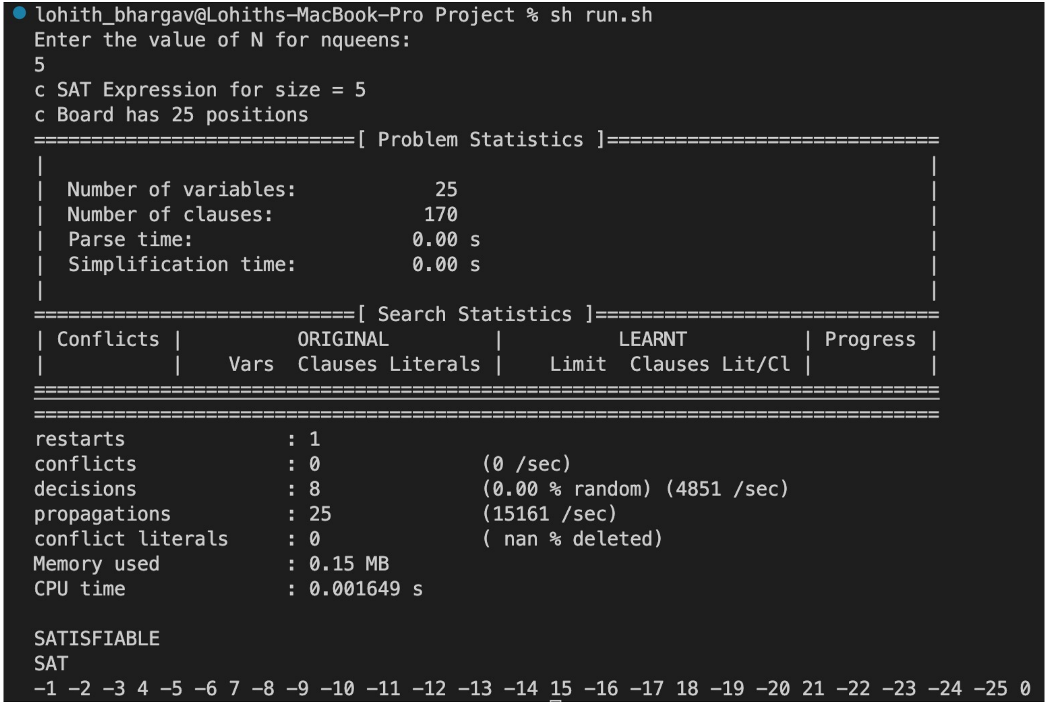
```
def minimumOne(List):
    initialize a variable
    loop over the list
        checking for minimum one variable in list for true
    appending ‘0’ for each row

def maximumOne(List) :
    initialize a variable
    loop over the list
        loop over the list’s list
        checking for maximum one variable in list for true

def varmap(row,column,size):
    to return the grid

def preciselyOne(List):
    initialize a variable
    variable is appended with minimumOne(list) return value
    variable is appended with maximumOne(list) return value

loop over row in (0, N):
        initialize a list
        loop over col in (0, N):
    appending position to check to have precisely 1 queen per row

loop over col in (0, N):
        initialize a list
        loop over row in (0, N):
    appending position to check to have precisely 1 queen per column

loop over col in (0, N):
        initialize a list
        loop over x in (0, N):
    appending position to check to have precisely 1 queen per column

loop over row in (N-1, -1, -1):
        initialize a list
        loop over x in (0, N-row):
    appending position to list for maximum of 1 queen per -ve diagonal from left

loop over col in (1, N):
        initialize a list
        loop over x in (0, N-col):
    appending position to list for maximum of 1 queen per -ve diagonal from top
```

```
loop over row in (N-1, -1, -1):
        initialize a list
        loop over x in row(0, N-row):
    appending position to list for maximum of 1 queen per +ve diagonal from right

loop over col in (N-2, -1, -1):
        initialize a list
        loop over x in col(0, col+1):
    appending position to list for maximum of 1 queen per +ve diagonal from top

creating a cnf file to store in dimacs CNF format

appending the p cnf no.of positions, no.of variables and clauses
```

Files in RAR:
Nqueens.py which generates the CNF file with DIMACS format
Run.sh is a scripting language commands file that contains computer program to be run by Unix shell

How to Execute:
Run the script code (run.sh) by sh run.sh command
It will prompt for the nqueens input of matrix

Output:



```
● lohith_bhargav@Lohiths-MacBook-Pro Project % sh run.sh
  Enter the value of N for nqueens:
  5
  c SAT Expression for size = 5
  c Board has 25 positions
  ============================[ Problem Statistics ]=============================
  |                                                                             |
  |  Number of variables:            25                                         |
  |  Number of clauses:             170                                         |
  |  Parse time:                   0.00 s                                       |
  |  Simplification time:          0.00 s                                       |
  |                                                                             |
  ============================[ Search Statistics ]==============================
  | Conflicts |          ORIGINAL         |          LEARNT          | Progress |
  |           |    Vars  Clauses Literals |    Limit  Clauses Lit/Cl |          |
  ===============================================================================
  ===============================================================================
  restarts              : 1
  conflicts             : 0              (0 /sec)
  decisions             : 8              (0.00 % random) (4851 /sec)
  propagations          : 25             (15161 /sec)
  conflict literals     : 0              ( nan % deleted)
  Memory used           : 0.15 MB
  CPU time              : 0.001649 s

  SATISFIABLE
  SAT
  -1 -2 -3 4 -5 -6 7 -8 -9 -10 -11 -12 -13 -14 15 -16 -17 18 -19 -20 21 -22 -23 -24 -25 0
```