Report. Satisfiability test of clauses and its application

1 Team Members :

- Venu Madhav Pendurthi, R11783168, vpendurt@ttu.edu
- Ayesha Anjum Shaik, R11800013, ayshaik@ttu.edu
- Sai Pranitha Kambham, R11793937, skambham@ttu.edu

2 Introduction to the Problem Statement

- "**MiniSat** is a minimalistic, open-source SAT solver, developed to help researchers and developers alike to get started on SAT". We use this due to its key features,
  - Easy to install, run, modify in all OS
  - Highly efficient and provides optimal output
- MiniSat accepts the input in **DIMACS CNF(Conjunctive Normal Form)** format, which is simple line-oriented text clauses format built from term, clause and expression
- **Problem Statement :** To place the 'n' queens in an nxn chessboard so that no two queens can attack each other either horizontally, vertically or diagonally
- **Input :** positive integer n
- **Output :** Optimal way how n queens can be placed on n*n chessboard so that no two queens can kill each other in an array format 1 to n with the positive numbers representing the queen positions and negative digits representing the empty locations.

- Algorithm used : **Backtracking Algorithm**
  1. Start with the first column
  2. If queens placed in every column, then we found a solution
  3. Else try placing a queen in each row in the column and check if the placed position is safe.
  4. If the placed position is safe, place the queen and proceed to the next column.
  5. If the placed position does not lead to solution, empty the placed position and backtrack a position.
  6. Else If the entire row is not safe for placing queens, try in the next row. Do this recursively till we find a solution.
  7. If all of the rows of the column doesn't seem to be safe, then backtrack to the next column and search for another position

3 Pseudo-Code

- Import all the files needed to execute the code

- function **main(arguments)**{

```
        Using exception to print logs,
        If argument input is given through command line,
        else, take user input(Scanner) in IDE terminal
        writer object to run PrintWriter() package to generate queens.cnf file
        clauses object to initiate ArrayList of strings
        run clausegenerator() method
        run generateCNFFile() method
end
}
```

- function **createClauses()** {

```
        call addrows() method
        call addcolumns() method
        call RightLeftdiagonal() method
        call LeftRightdiagonal() method
end
}
```

- function **addrows()** {

```
        for i = 1 to n
        initiate an integer array RowValues
            for j = 1 to n
                RowValues[j] = (j + 1) + (i * n)
                call FinalClause() method
end
}
```

```
• function addcolumns() {
        for i = 1 to n
        initiate an integer array ColumnValues
                for j = 1 to n
                        ColumnValues[j] = (j + 1) + (i * n)
                call FinalClause() method
end
}
```

```
• function RightLeftdiagonal() {
        initiate start=0
        for i= n to 1
                initiate an integer array RightdiagonolValues
                initiate y=0
                for j=1 to n * (start + 1)
                        RightdiagonolValues[y] = j
                        y++
                start++
                call FinalClause(RightdiagonolValues, bool) method
        initiate y = n - 1
        for i= n+1 to n*n
                initiate integer array LeftdiagonolValues
                initiate start = 0
                for j = i to n*n
                        LeftdiagonolValues[start] = j
                        start++
                y–
                call FinalClause(LeftdiagonolValues, bool) method
end
}
```

```
• function LeftRightdiagonal() {
        for i = n to 1
                initiate an integer array LeftdiagnolValue
                initiate y=0
                for j=i to x¡i
                        LeftdiagnolValue[x] = j
                        x++
                call FinalClause(LeftdiagnolValue, bool) method
```

```
    initiate y = n - 1
    for i= 2*n to n*n
        initiate integer array RightdiagnolValue
        initiate x = 0
        for j = i to n*n
            RightdiagnolValue[x] = j
            x++
        y–
        call FinalClause(RightdiagnolValue, bool) method
end
}
```

- function **FinalClause(arguments)** {

```
    if array.length is not 1
        if not diagnol
            initiate String s
            for i = 0 to array length
                s += array[i] + ” ”
            s += ”0”
            add s to clauses
        for k = 1 to array length
        add + array[k] + ” -” + array[l] + ” 0” to clauses
end
}
```

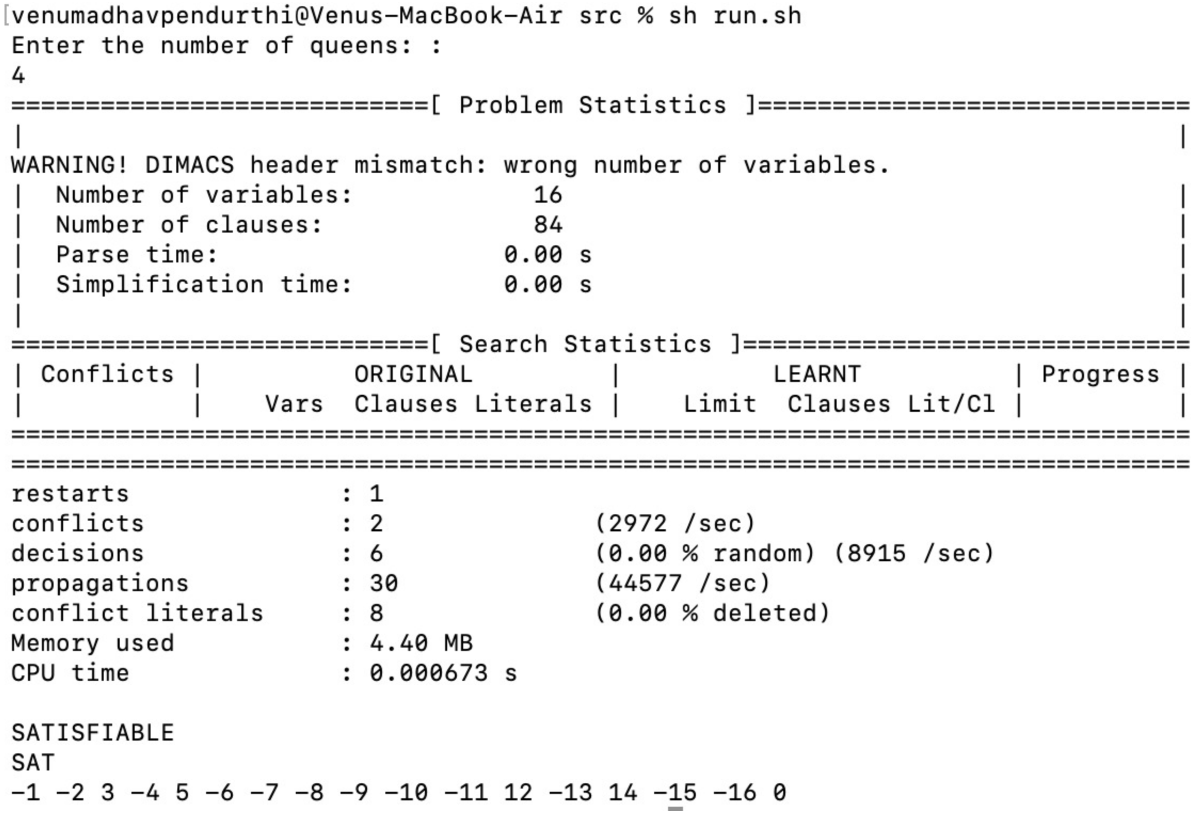
- function **generateCNFFile()** {

```
    print ”p cnf ” + n + ” ” + clauses.size()
    for i = 1 to clauses.size()
        print clauses.get(i)
end
}
```

4 Execution

- **In mac :**
  1. Open the terminal
  2. Install minisat solver % **sudo port install minisat**
  3. Execute the run.sh file % **sh run.sh**
  4. In run.sh,

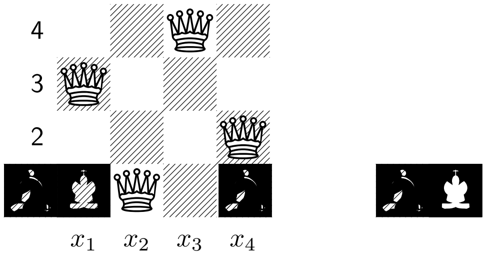
     Take user input for the number of queens
     read user input
     Execute % **java nqueens.java**
     Execute % **minisat queens.cnf output**
     Execute % **cat output**



```
[venumadhavpendurthi@Venus-MacBook-Air src % sh run.sh
Enter the number of queens: :
4
============================[ Problem Statistics ]=============================
|                                                                             |
WARNING! DIMACS header mismatch: wrong number of variables.
|  Number of variables:            16                                         |
|  Number of clauses:              84                                         |
|  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             : 2              (2972 /sec)
decisions             : 6              (0.00 % random) (8915 /sec)
propagations          : 30             (44577 /sec)
conflict literals     : 8              (0.00 % deleted)
Memory used           : 4.40 MB
CPU time              : 0.000673 s

SATISFIABLE
SAT
-1 -2 3 -4 5 -6 -7 -8 -9 -10 -11 12 -13 14 -15 -16 0
```

The output in chessboard is represented as in the following figure :


4
3
2
x_1
x_2
x_3
x_4