CS-301
N Queens problem

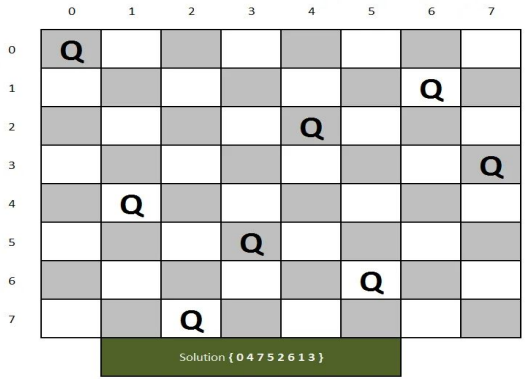
Assigned By -
Prof. Bhaskar Chaudhury

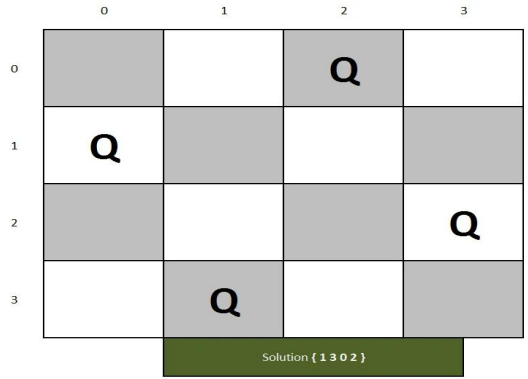
Performed By -
Deep Godhani (201701418)
Sairam Venkatachalam (201701433)

INTRODUCTION

In this project, our main objective was to find the number of solutions to the n queens problem and find ways to parallelize the algorithms used.

The N queens problem requires us to place N queens on an N x N board such that no two queens attack each other.



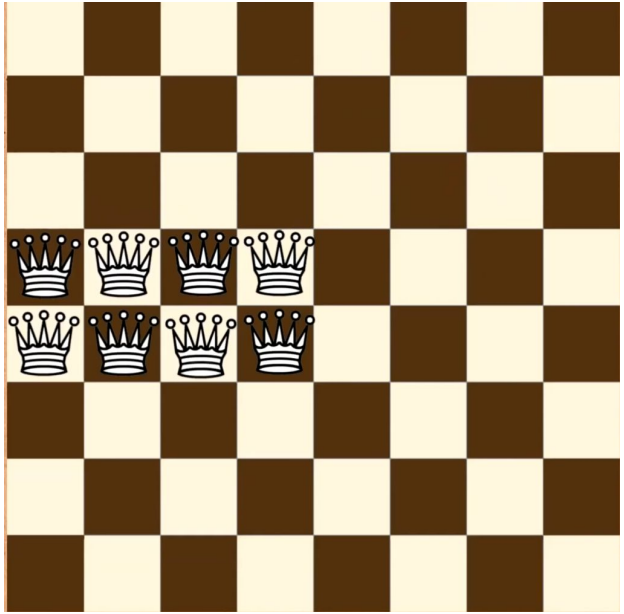


ALGORITHM’s

1. **Brute Force method:**

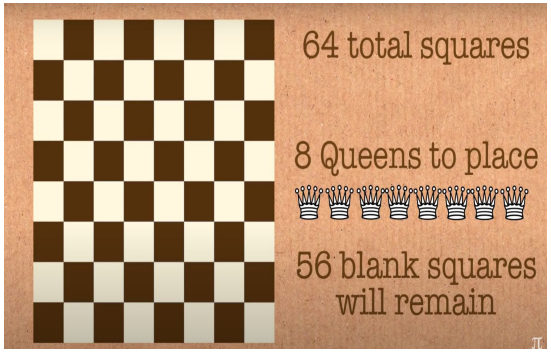
   In this Naive approach we would check all the possible placements for N queens on the board and find which of them are valid solutions.

   This approach has the highest time complexity due to the fact that we have to check each and every position even redundant positions like this .



Drawbacks with the brute force method

We will have to check a total of 64C8 different positions for an 8 x 8 board which is 4,426,165,368 different positions.



Pseudo Code -

While (possible configuration)

Generate configuration

If(queens don't attack)

count ++

Time-Complexity -

$O(n^2Cn)$

2. Backtracking Method:

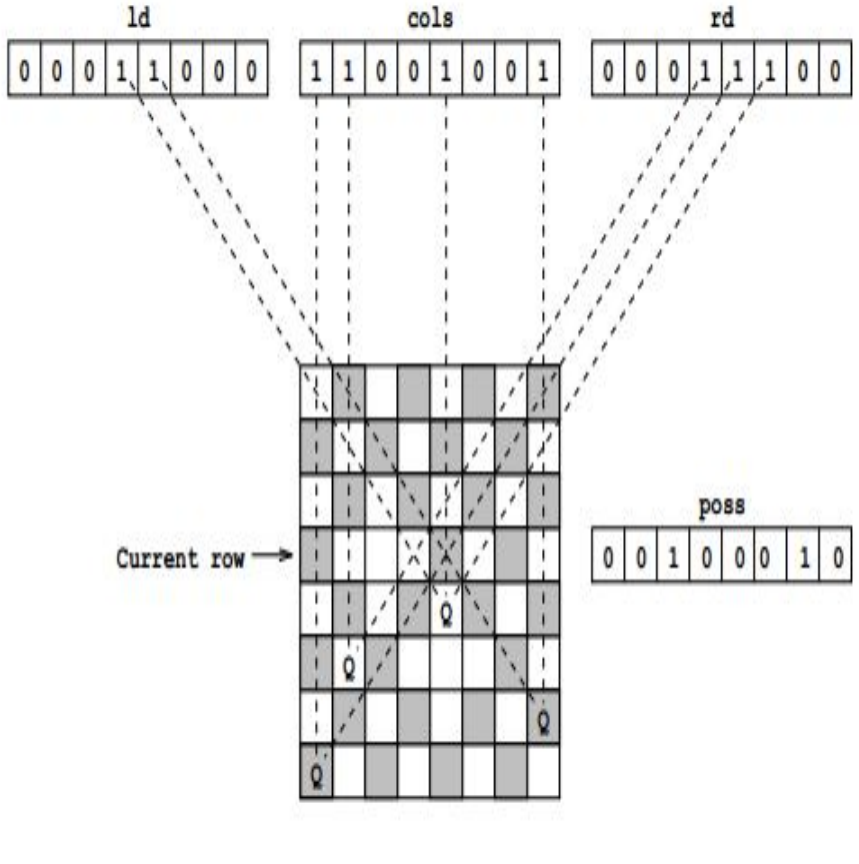
The idea is to place queens one by one in different rows, starting from the topmost row. When we place a queen in a row, we check for clashes with already placed queens. In the current row, if we find a column for which there is no clash, we mark this row and column as part of the solution. If we do not find such a column due to clashes then we backtrack and return false.

Pseudo Code -

```
Nqueens(queens,row,col) {

    If(attack(queens,row,col)==true)

          return ;

   queens[col] = row

    If(row==size-1)

            count++;

    else

             for(all columns)

                    Nqueens(queens,row+1,col)

}
```

3. Bitmasking Approach:

The idea behind this method is to maintain arrays to keep track of which squares are attacked.This way in order to check whether the placement of a queen in a certain column j is possible we can refer to the bitmap of the current row and check its jth bit. After the placement of each queen, the bitmap is updated. The bitmask for each row is generated using the left diagonal,right diagonal and the column arrays for the respective cells. For (r,c) we need col[c] , left[r-c+N-1] and right[r+c] for queen's check.



Scope For Parallelization

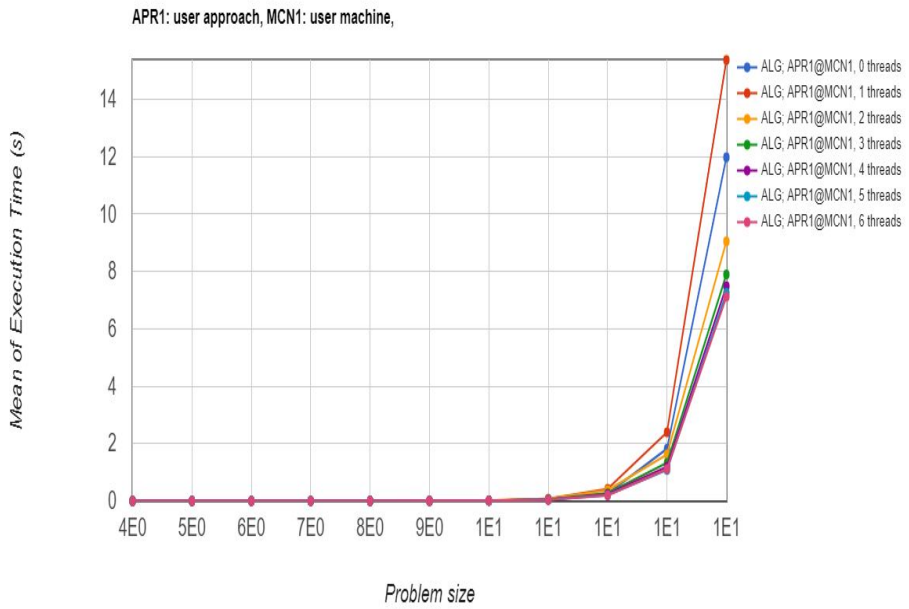
In the case of a serial backtracking algorithm, we saw that the placement of the queen in the current row depends only on the previous row's queens position. For example, the placement of the queen in the second row depends on the position of the first row's queen. Similarly, the position of the third-row queen will depend on the position of the first and the second row's queens and so on .. continues for other rows. Moreover, we saw that there exists many independent paths which can be executed simultaneously.

Hence we are able to parallelize these processes and run them simultaneously to lower our execution time.

Parallelization Strategies

Method 1 : Backtracking Algorithm

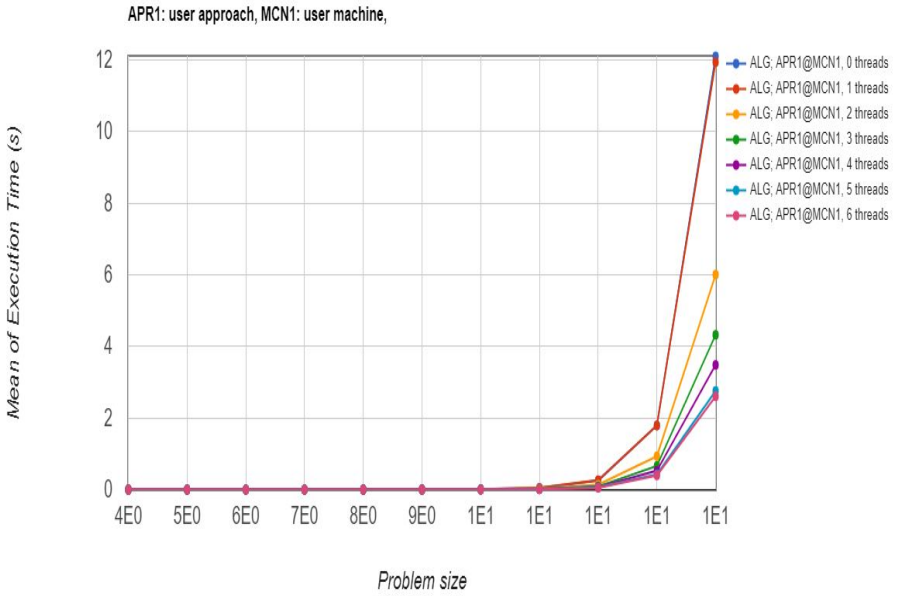
In this method we divide the task between the processors by generating small tasks at each row of the N x N Board by using the backtracking algorithm.When a certain processor is idle it can execute any one of these tasks immediately. Due to which the processors are not left idle in this method, but still the execution time of this is large because the task are partially dependent on each other.



Parallelization Strategies

Method 2 : Backtracking Algorithm

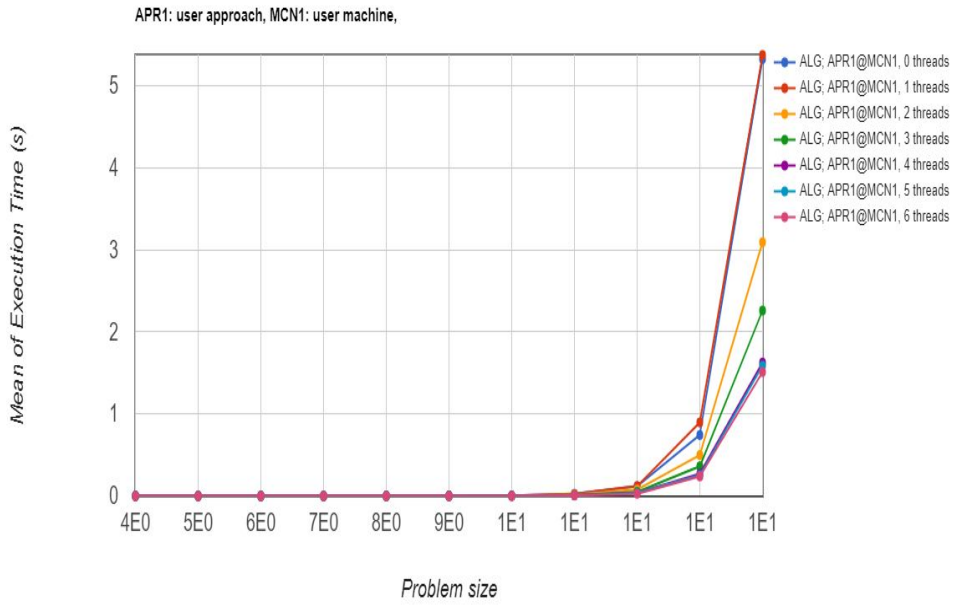
In this method we assign each column of the first-row to each processor to execute and find all the solutions with a certain placement of queen. In this method the work is not distributed equally because it is possible that a particular column has more solutions while some columns don't have even a single solution. But the advantage of this method is that the task generated are independent of each other so can be executed simultaneously.



Parallelization Strategies

Method 3 : Backtracking Algorithm

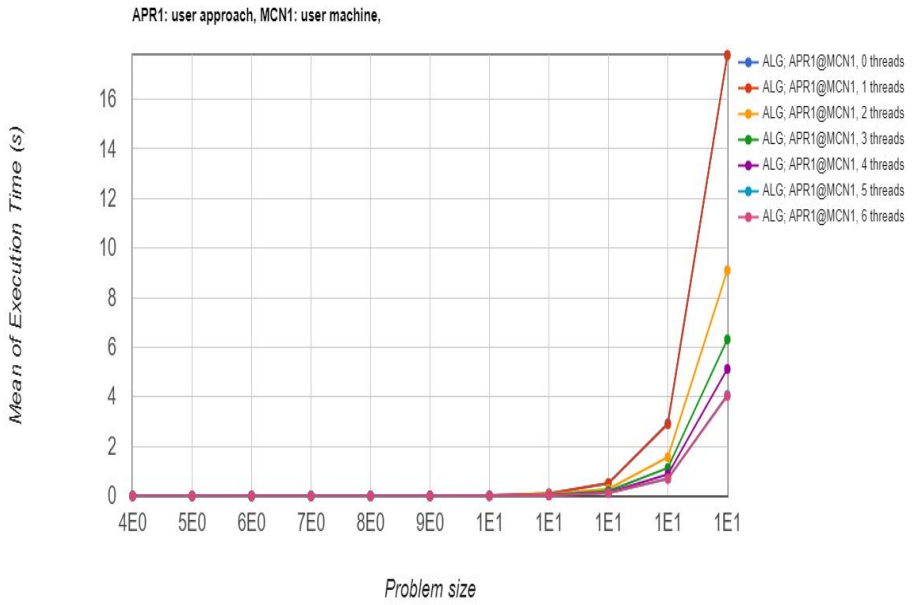
In this method we select a certain depth (rows) till which we perform the placement of queens level-wise will after which we search each solution linearly. This method leads to the optimization to the middle ground between the first-two methods of parallelization. We have used the depth of N/2 while performing this algorithm.



Parallelization Strategies

Method 4 : Backtracking + Bitmasking

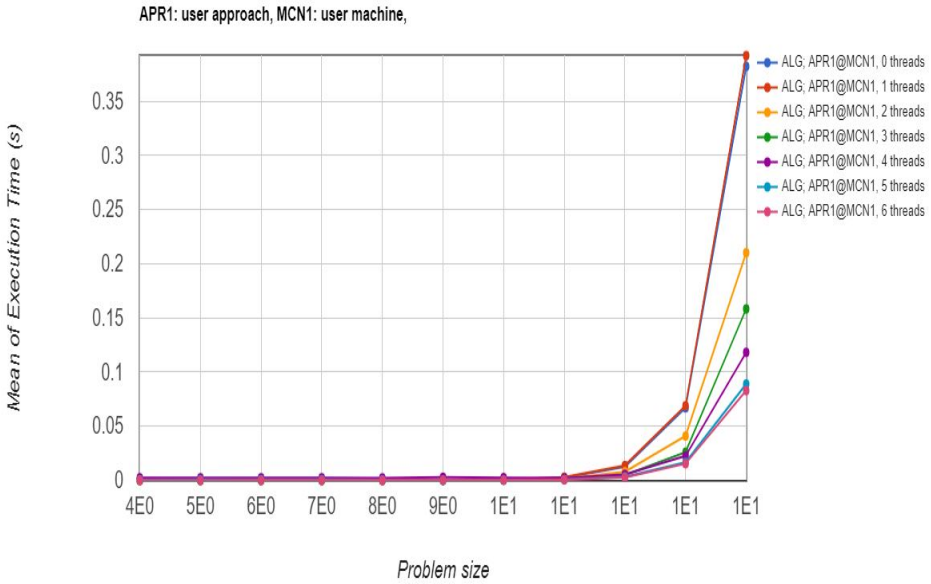
In this method we had tried to decrease the time of calculating function of whether the provided place is valid placement or not. We had used the backtracking + bitmasking technique to reduce total execution time. We have used three bit arrays of column, left and right diagonal for calculating the above function in O(1) time, but increases the space complexity by O(3n-2).



Parallelization Strategies

Method 5 : Backtracking + Bitmasking

In this method we had also used backtracking + bitmasking but tried to even further optimize the method-4 by using the integer in place of three bit arrays which helps to reduce the space complexity this can ultimately helps us to decrease cache misses this increase size of N. Resulting in O(1) space complexity.



Conclusion

In conclusion we find by analyzing the speedup curves, that the method of parallelizing backtracking using bit masking (space optimized) is the most effective method among those we have used as it produces the most significant speedup and the lowest mean execution time.