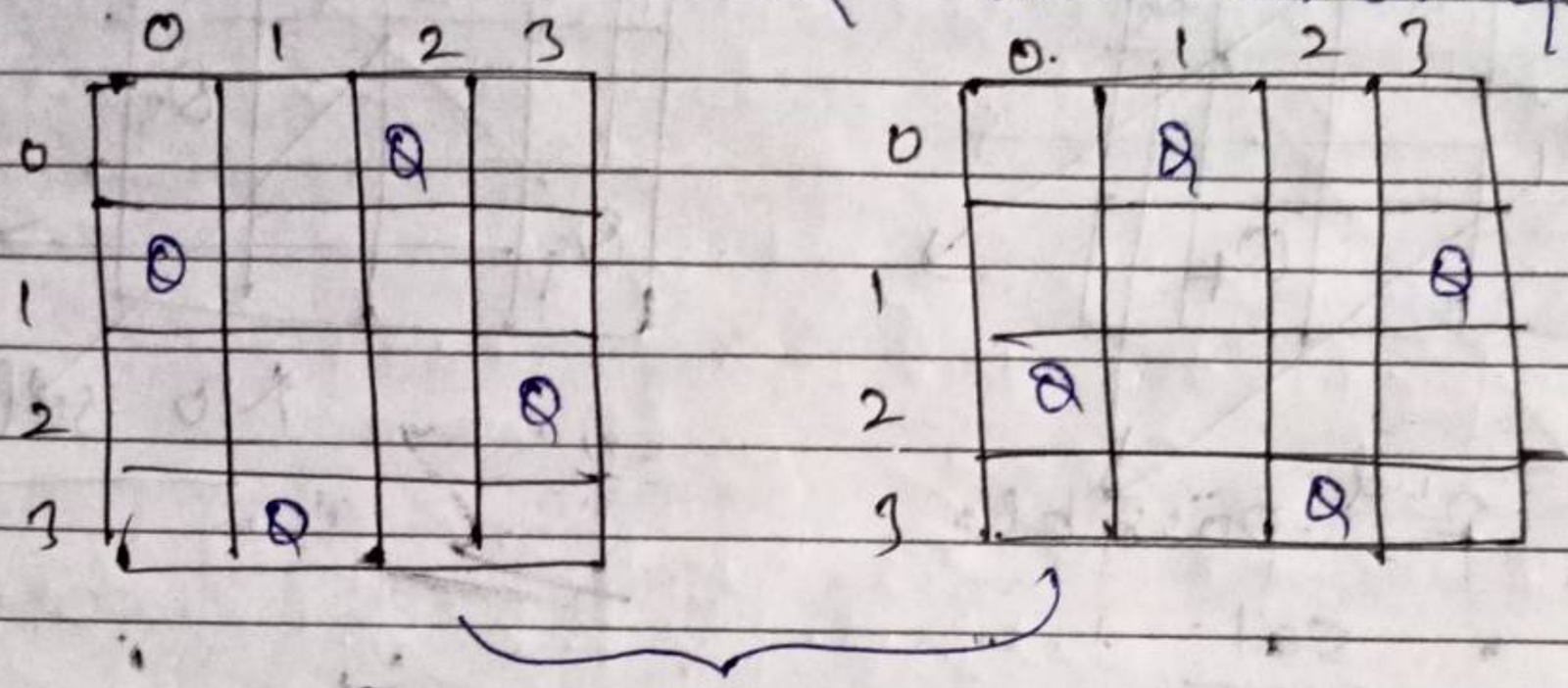
DMC (Level 4)

PAGE No
DATE

N- Queen Problem

Problem statement

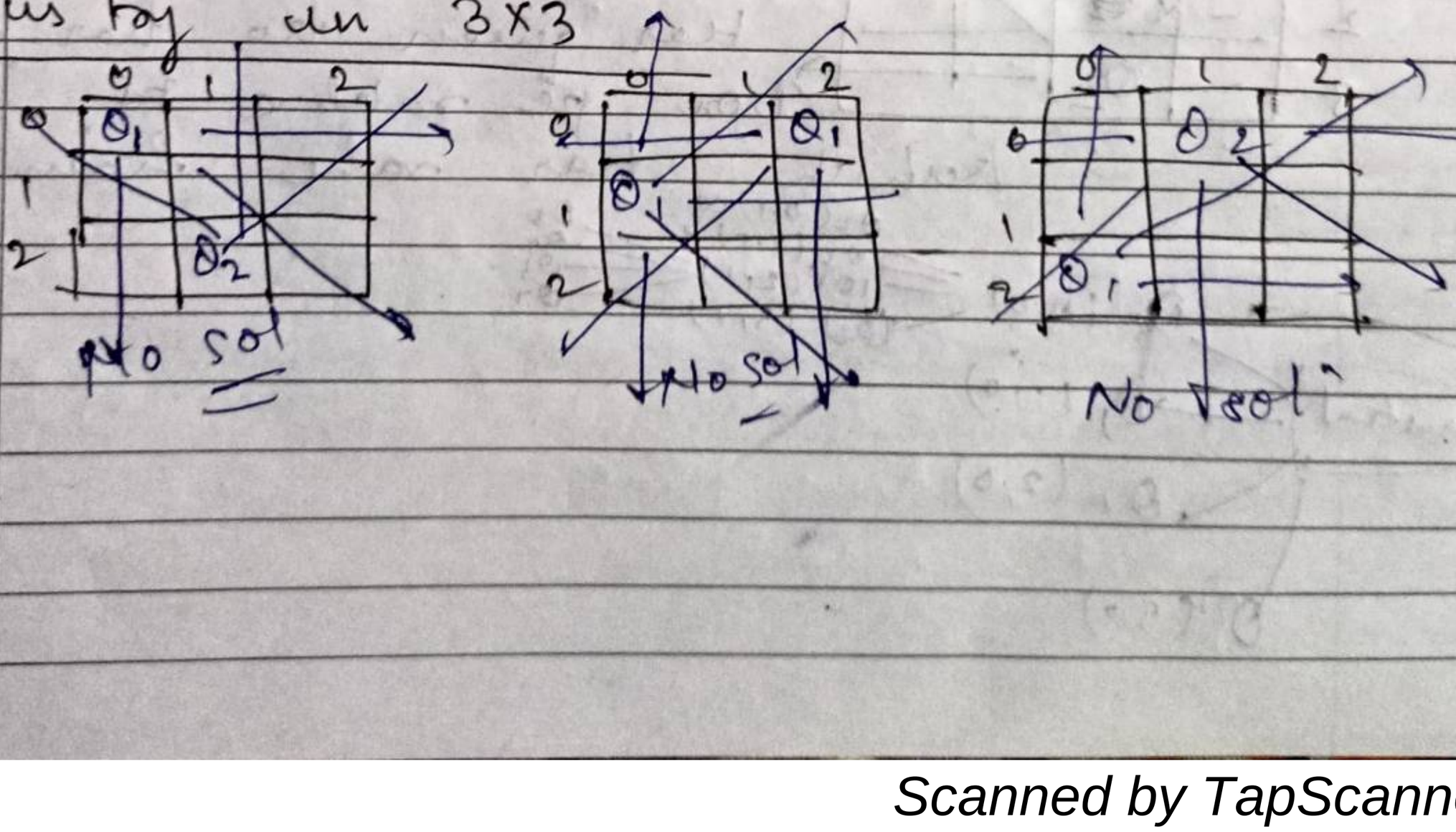
There is NXM chess board and we have to place Queen and we know Queen can move in all 8 directions. So we have to place N Queens such that no Queen can attack the other queen.



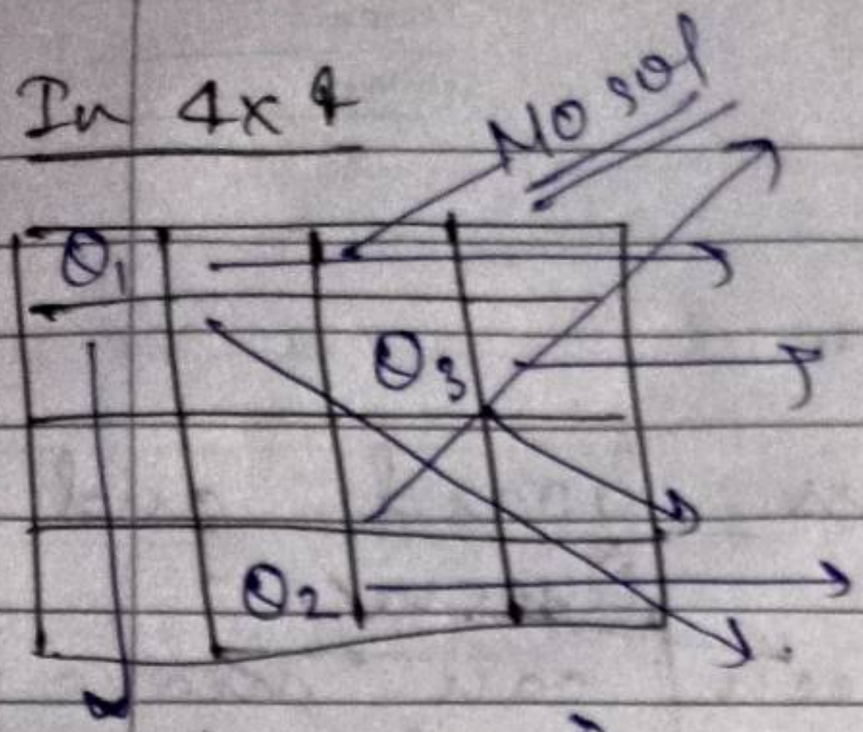
2 ways.

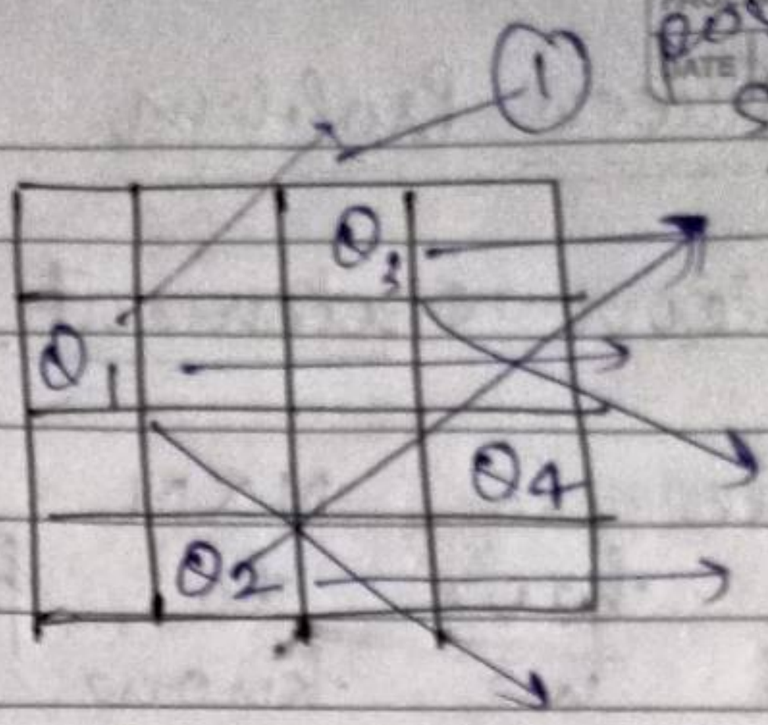
We can't place 2 Queens in ia 2x2 board. there are no ways of it.

Let us try in 3x3



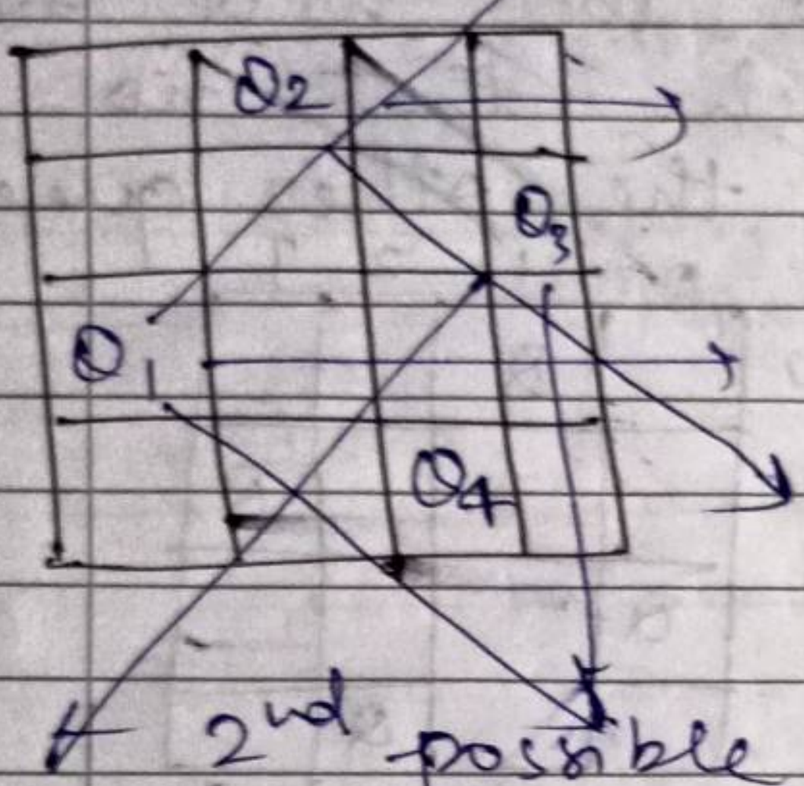
In 4x4



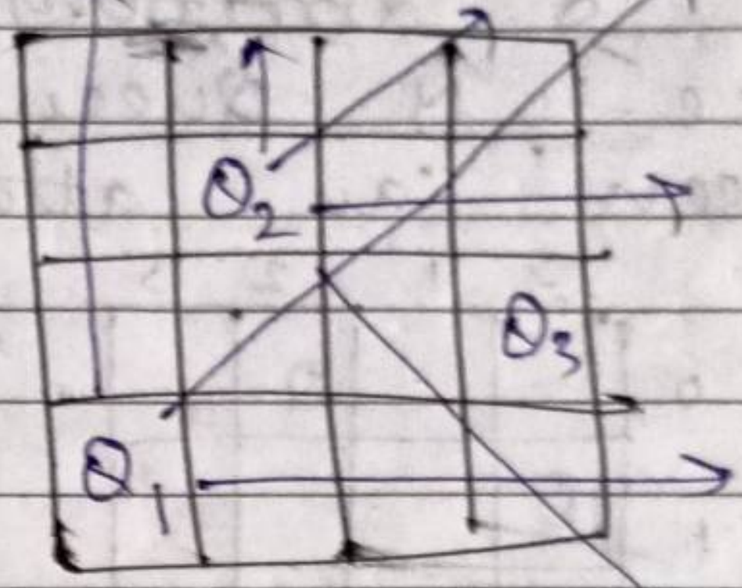


PAGE NO. DATE

possible soln

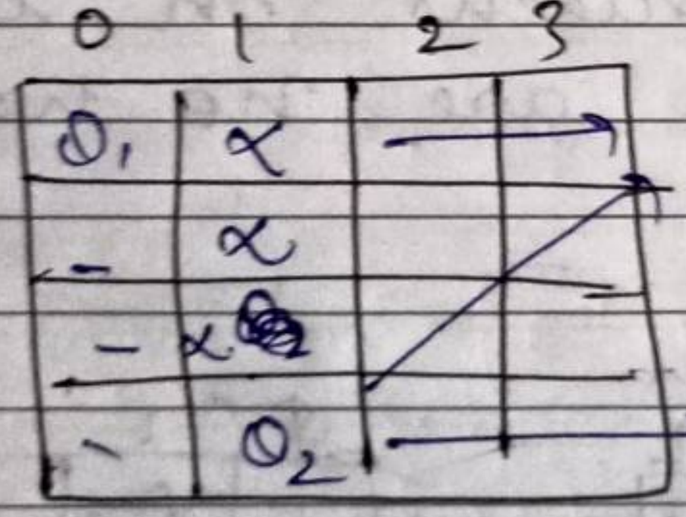


2nd possible sol.



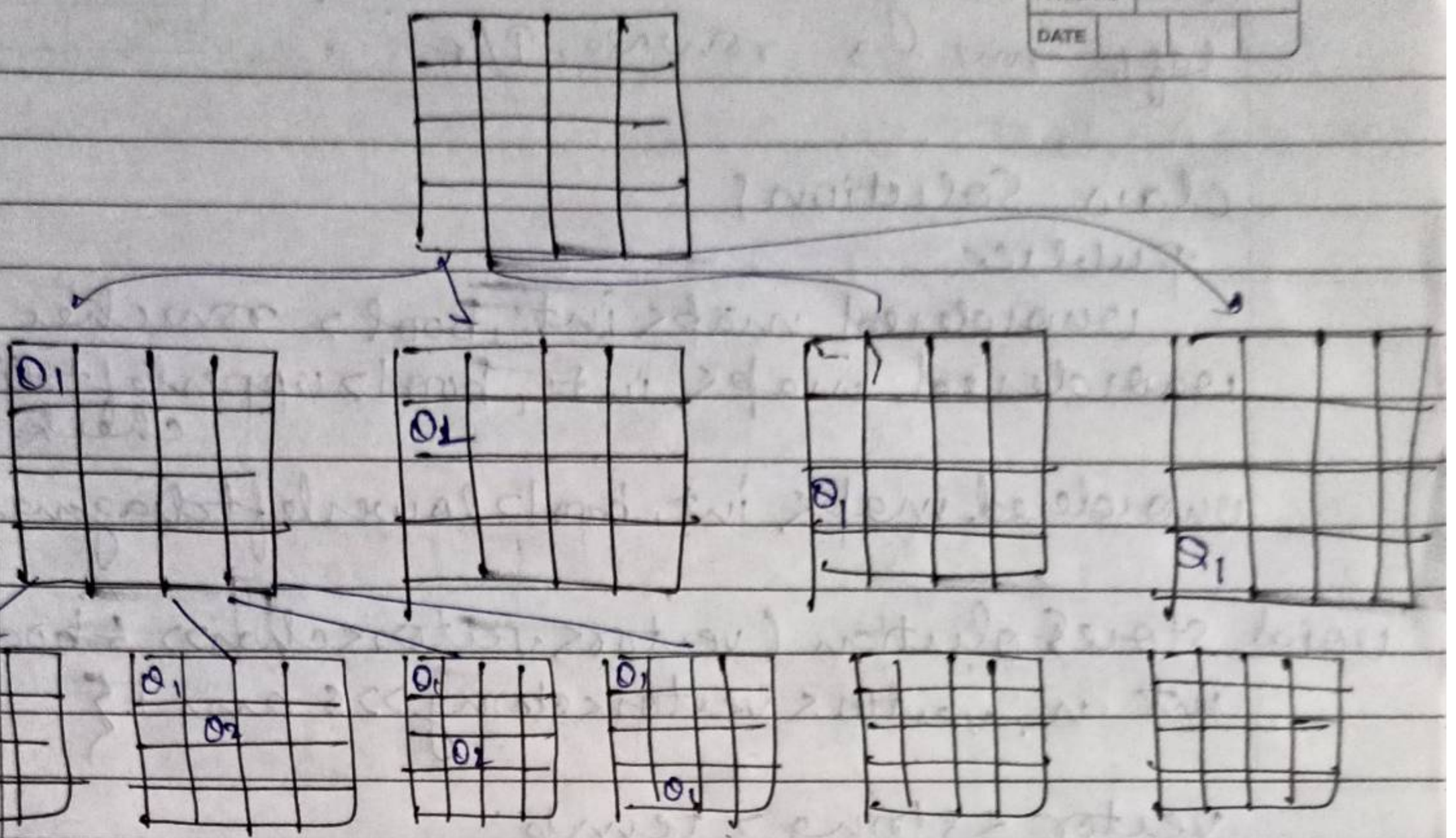
No sol.

Approach



Har ek position check karega

Har queen ko har ek box pe rakhne ki koshish kar rahe hain



Har combination try kar sake hain.
to bs yahi dekho ki ek case solve
karo or Recursion sambhal lega.

~~Partial~~

Code

```
int main () {
    int n = 4;
    vector<vector<char>> board (n, vector<char>(n,'.'));
    int col = 0;
    solve (board, col, n);
```

upper diagonal → (n-1) + col - row T/F
lower diagonal → row + col T/F
left row → rowNo. T/F

```
class Solution {
  public:
    unordered_map<int,bool> rowcheck;
unordered_map<int,bool> upperleftdiagonal
                                  check;
unordered_map<int,bool> lowerleftdiagonalcheck;

void storeSolution(vector<vector<char>>& board,
int n, vector<vector<string>>& ans) {

vector<string> temp;
for(int i=0; i<n; i++) {
        string output = "";
    for(int j=0; j<n; j++) {
        output.push_back(board[i][j]);
    }
    temp.push_back(output);
}
ans.pushback(temp);
}
bool isSafe(int row, int col, vector<vector
<char>>& board, int n) {
if (rowcheck[row]==true)
    return false;
if(upperdiagonalcheck[n-1+col-row]==true);
    return false;
if (bottomleftdiagonalchek[row+col]==true)
    return false;
return true;
}
```

```
void solve (vector<vector<char>> & board,
int col, int n, vector<vector<string> & ans){

if( col >=n)      // base case
   storeSolution (board, n, ans);
     return;
}

for (int row =0; row < n ; row++) {
      if (isSafe (row, col, board, n)) {
   board [row] [col] = 'Q' ;
   rowcheck[row] = true;
   upperleftDiagonalcheck [n-1 + col-row] = true;
   bottomleftDiagonal check [row + col] = true;

solve (board, col+1, n, ans) ;
   board [row] [col] = '-';
upperLeftDiagonal Check [ n-1 + col-row ] = false;
bottomleftDiagonal check [ row + col] = false;
   }

}

   vector< vector <string>> solveNQueens (int n)
vector<vector<char>> board (n, vector<char> (n, '-'));
vector<vector<string>> ans;
      int col = 0 ;
      solve (board, col, n, ans);
      return ans;
   }

};
```

PAGE No
DATE

MAP

A data structure.

↳ Key → Value

Love → 9
Babbas → D
Rahul → 21
Varun → 19.

map<string, int> mp;
↑ ↑
Key value.

Value insert. mp["love"] = 98
mp["Rahul"] = 36

Access cout << m["love"];
Op → 98