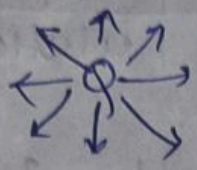
* n-Queen [Important Ques]

Input n → n×n board i.e 2D matrix
→ n-Queen need to be placed in it.

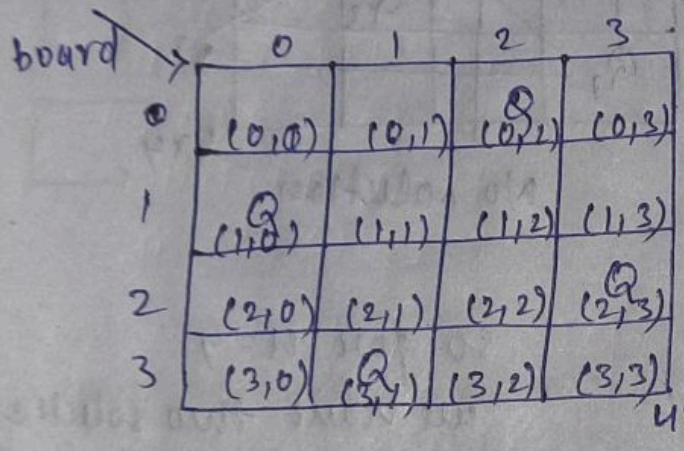
N-Queen such that no Queen can attack the other Queen.

A single Queen can make 8 different movements.

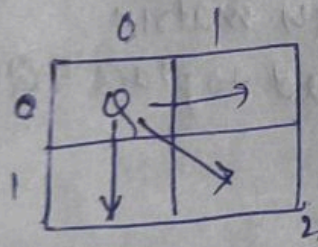
i.e 

[up, down, left, right, left up Diagonal, left Down Diagonal, Right up Diagonal, Right Down Diagonal

Single Q can make 1 movement at a time.



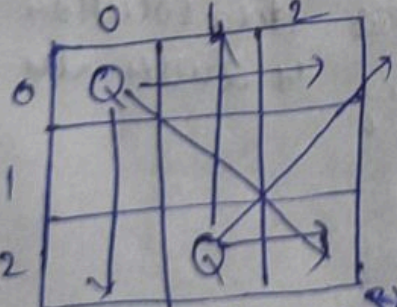
Input → 4×4 board & we need to place 4 Queen such that no Queen can attack the other Queen.



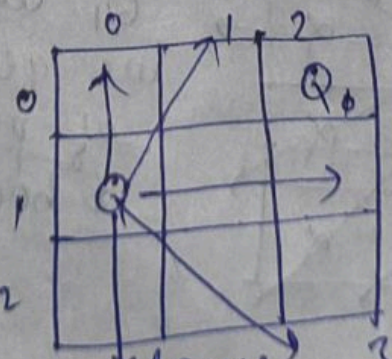
n=2 → 2×2 board
→ 2 Queens.

No solution

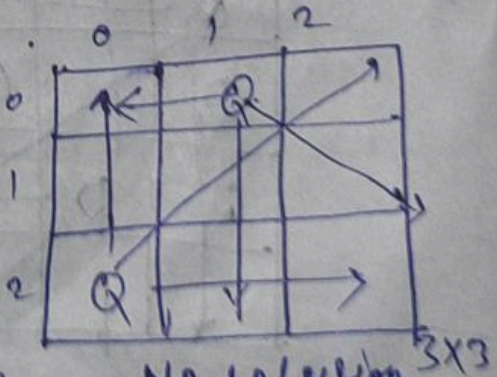
2 Queen can't be placed at 2×2 board.



we can't place 3 Queen in 3×3 board

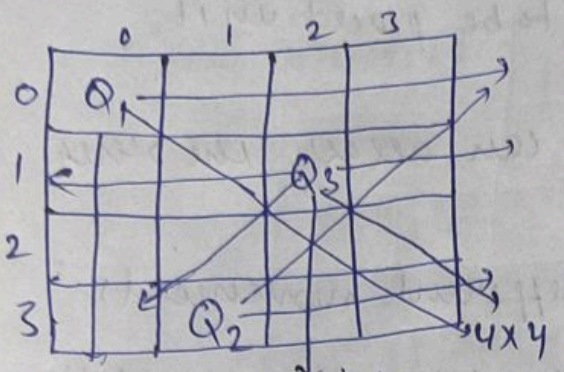


Same. No solution

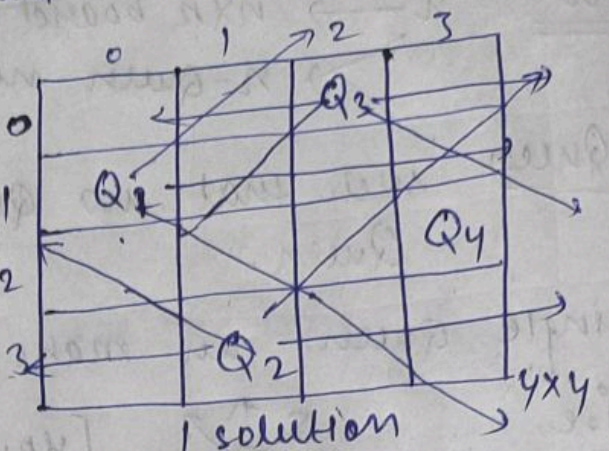


No solution

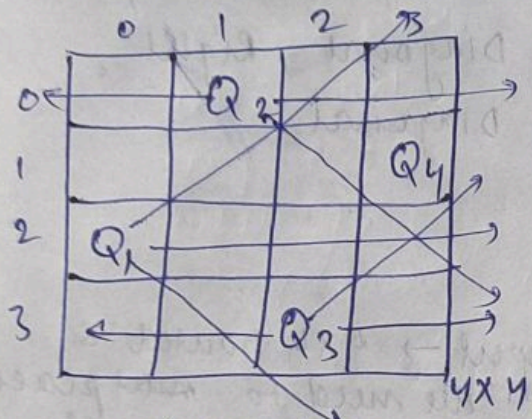
Input = n=4 ⟶ 4×4 board
⟶ 4 Queen need to be placed in it



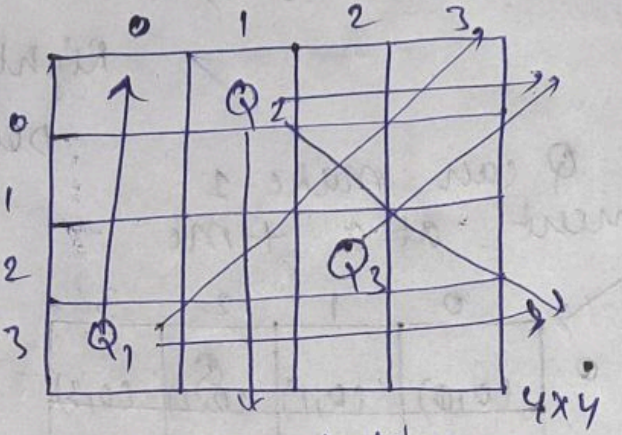
No possible soln



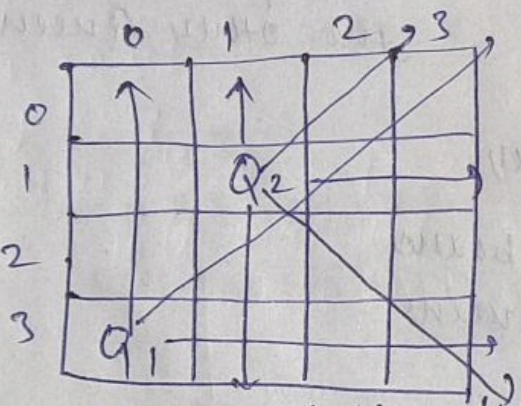
1 solution



1 solution



No solution

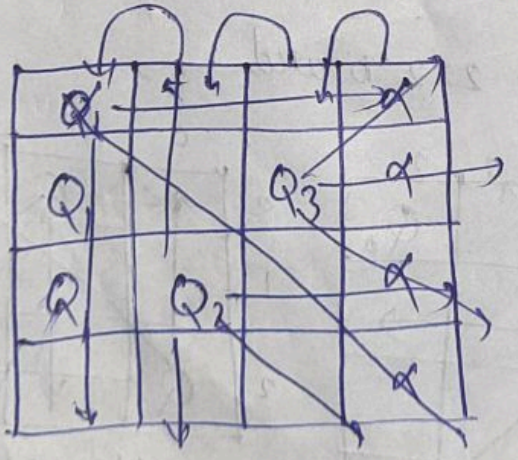


No solution.

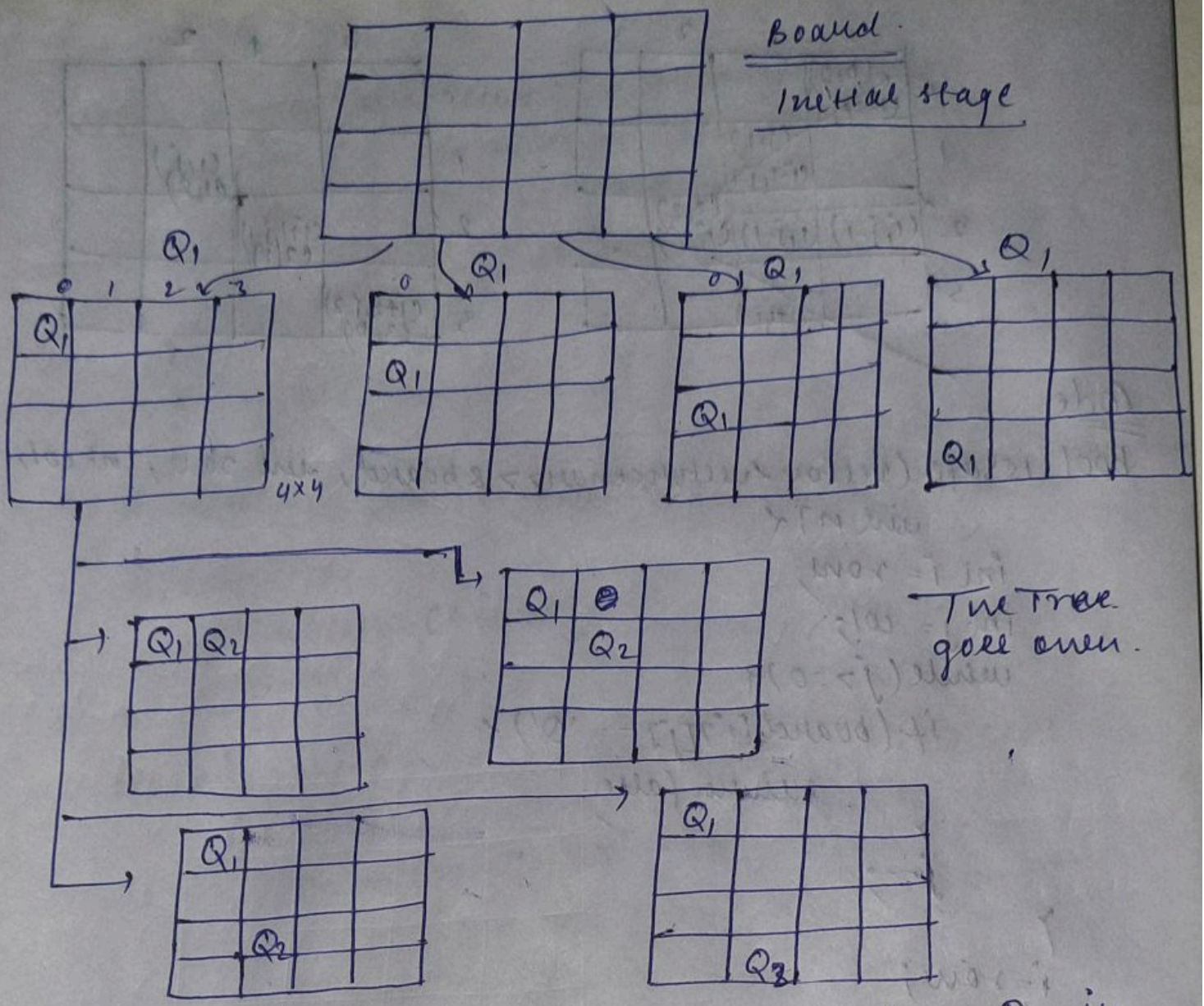
so for n=4
we have two solution

n=4 ⟶ 4×4 matrix
⟶ need to place 4 Queens

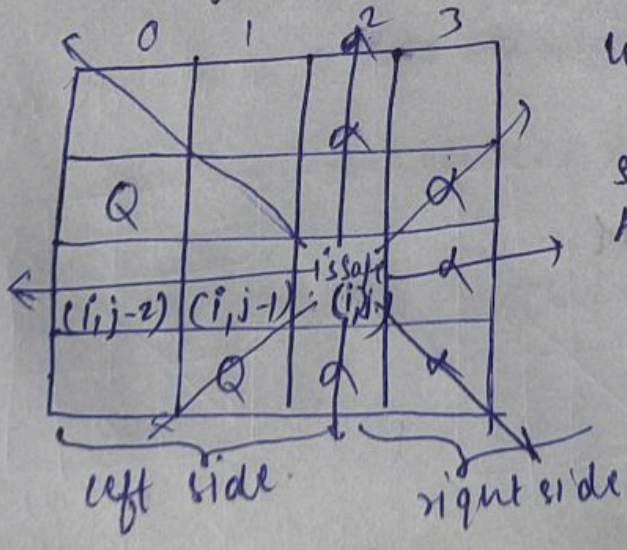
→



we are trying to place Q_1 in every row until we get a solution of placing 4 Queen in a board.

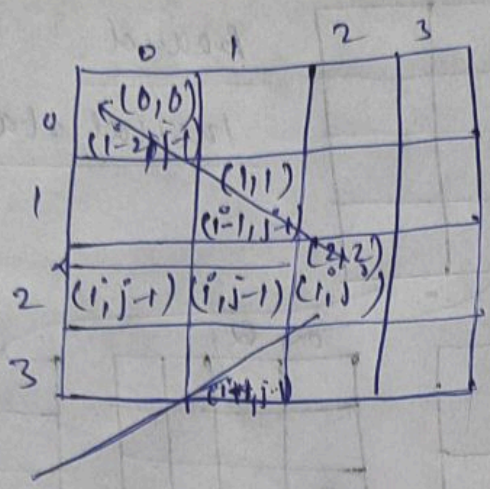


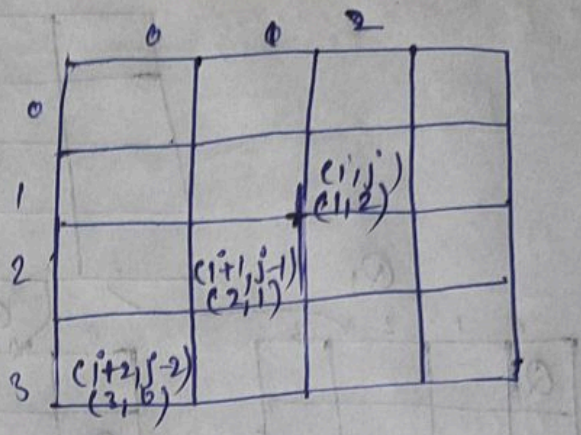
We can observe that we are trying to place Q_1 in 1st column of every row & Q_2 in 2nd column of every row & goes on. until we get a solution. When we can't place a next Queen at any place i.e any row then we ask to change the current position of Queen.



We will not check in right side whether it is safe to place Queen bcz we haven't placed the Queen in the right side.

So we will check in the left side.





Code

```
bool isSafe(vector<vector<char>>& board, int row, int col,
            int n){
    int i = row;
    int j = col;
    while(j>=0){
        if(board[i][j] == 'Q'){
            return false;
        }
        j--;
    }
    i=row;
    j=col;
    while(i>=0 && j>=0){
        if(board[i][j] == 'Q'){
            return false;
        }
        i--;
        j--;
    }
    i=row;
    j=col;
    while(i<n && j>=0){
        if(board[i][j] == 'Q'){
            return false;
        }
        i++;
        j--;
    }
}
```

```
    return true;
}
void printBoard (vector<vector<char>> &board){
    for(int i=0; i< board.size(); i++){
        for(int j=0; j< board[i].size(); j++){
            cout<< board[i][j] <<" ";
        }cout<<endl;
    }
    cout<<endl;
}
void nQueen (vector<vector<char>> &board, int col, int n
{
    if (col>=n){
        printBoard(board);
        return;
    }
    for(int row=0; row< board.size(); row++){
        if(isSafe(board, row, col, n){
            board[row][col] ='Q';
            nQueen( board, col+1, n);
            board[row][col] ='-';
        }
    }
}
int main(){
    int ~~main~~ n;
    ~~int a~~ cin>>n;
    if(n==1||n==2||n==3){
        cout<<"No solution";
    }
    vector<vector<char>> board(n, vector<char>(n,'-'));
    int col=0;
    nQueen(board, col, n);
}
```