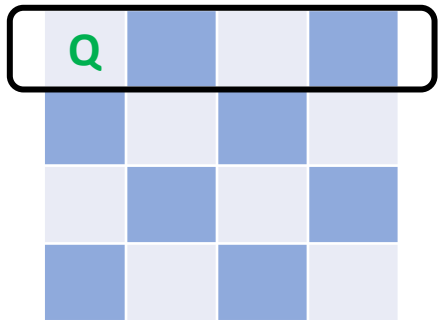
ECE 220: Computer Systems & Programming

N-Queen Problem Run-time Stack

Thomas Moon

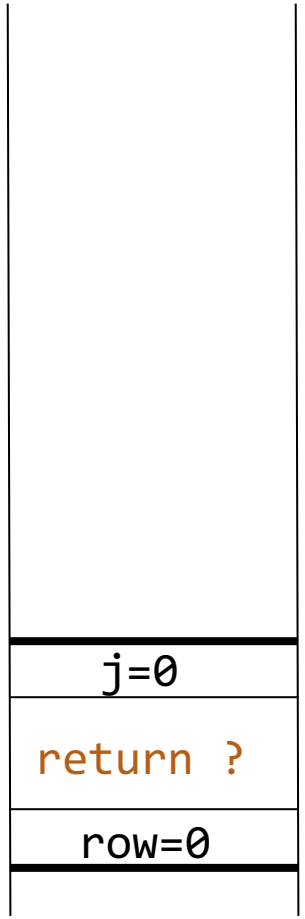
```
int Solve(int board[N][N], int row)
{
    if(row==N)
        return true;
    else{
        for(j=0;j<N;j++){
            if(isSafe(board, row, j) == true){
                board[row][j]=1;
                if(Solve(board, row+1) == true)
                    return true;
                else
                    board[row][j]=0;
            }
        }
        return false;
    }
}
```

row 0



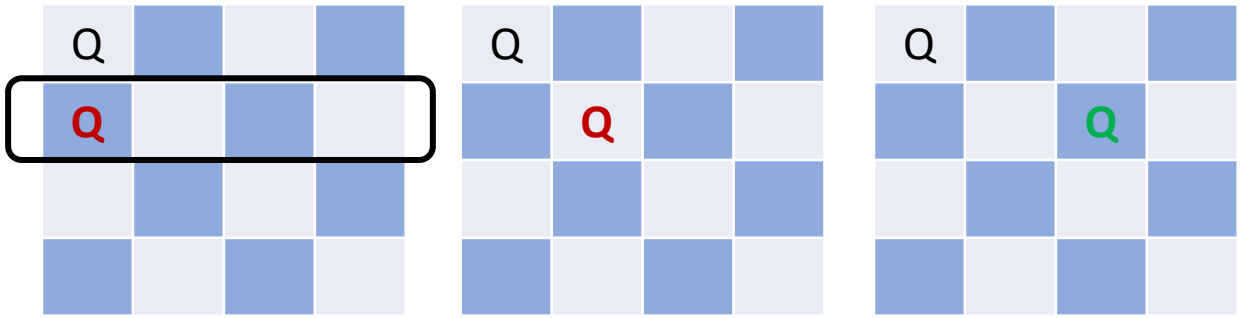
Solve(board, 0)

board[0][0]=1;



```
int Solve(int board[N][N], int row)
{
    if(row==N)
        return true;
    else{
        for(j=0;j<N;j++){
            if(isSafe(board, row, j) == true){
                board[row][j]=1;
                if(Solve(board, row+1) == true)
                    return true;
                else
                    board[row][j]=0;
            }
        }
        return false;
    }
}
```

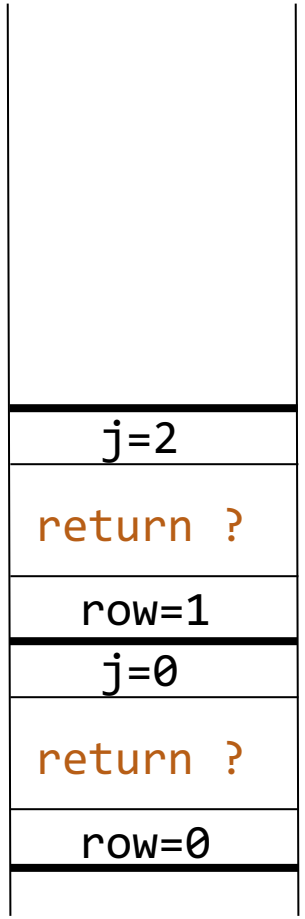
row 1



Solve(board, 1)

board[1][2]=1;

board[0][0]=1;

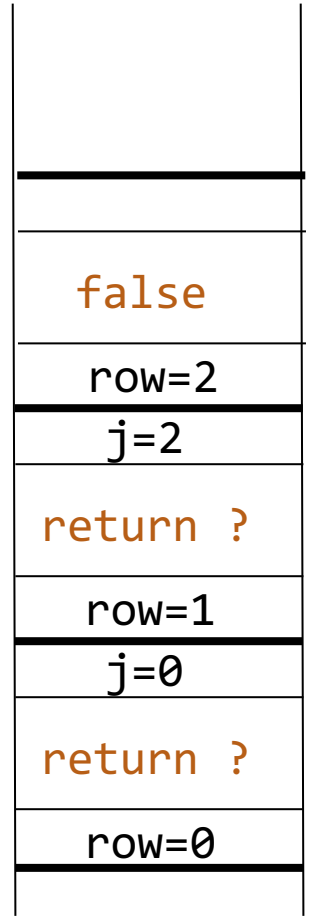


```
int Solve(int board[N][N], int row)
{
    if(row==N)
        return true;
    else{
        for(j=0;j<N;j++){
            if(isSafe(board, row, j) == true){
                board[row][j]=1;
                if(Solve(board, row+1) == true)
                    return true;
                else
                    board[row][j]=0;
            }
        }
        return false;
    }
}
```

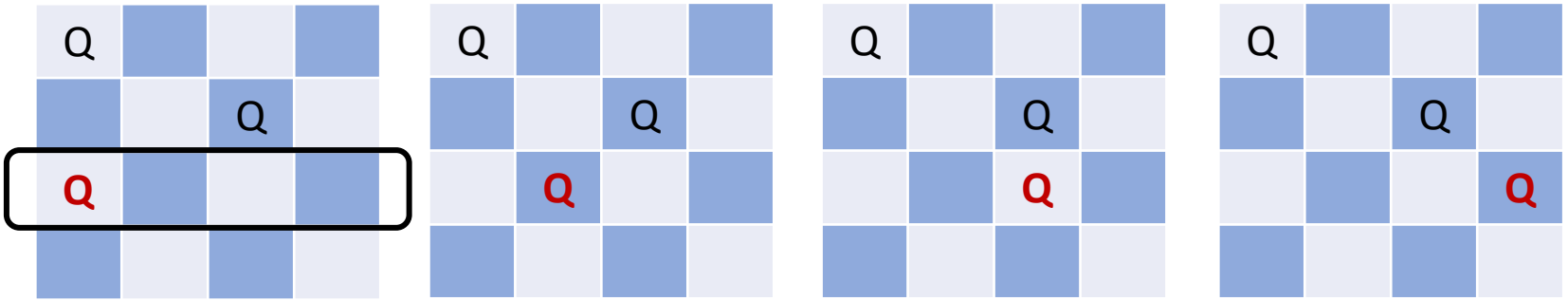
Solve(board, 2)

board[1][2]=1;

board[0][0]=1;



row 2



```
int Solve(int board[N][N], int row)
{
    if(row==N)
        return true;
    else{
        for(j=0;j<N;j++){
            if(isSafe(board, row, j) == true){
                board[row][j]=1;
                if(Solve(board, row+1) == true)
                    return true;
                else
                    board[row][j]=0;
            }
        }
        return false;
    }
}
```

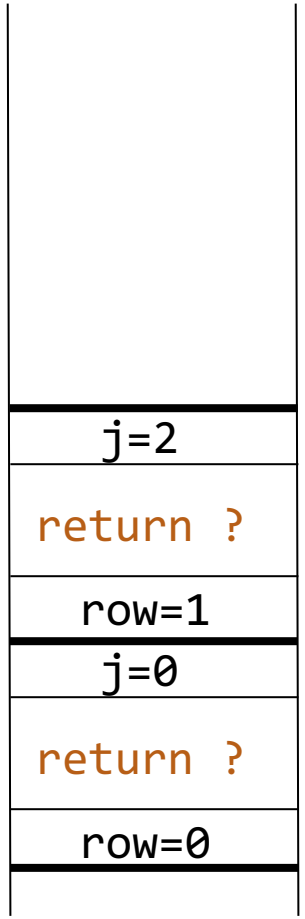
Solve(board, 2) = false

backtrack

Solve(board, 1)

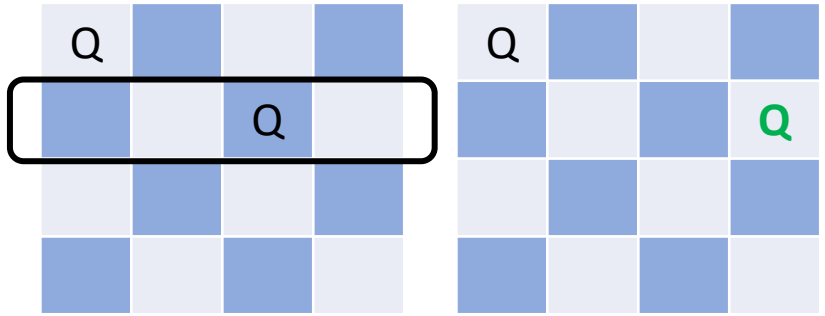
board[1][2]=0;

board[0][0]=1;



```
int Solve(int board[N][N], int row)
{
    if(row==N)
        return true;
    else{
        for(j=0;j<N;j++){
            if(isSafe(board, row, j) == true){
                board[row][j]=1;
                if(Solve(board, row+1) == true)
                    return true;
                else
                    board[row][j]=0;
            }
        }
        return false;
    }
}
```

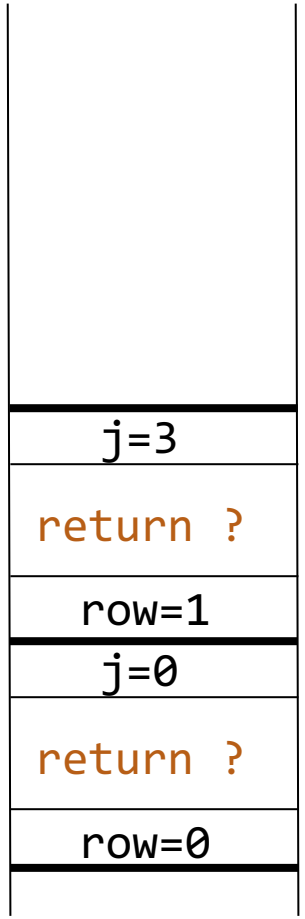
Backtrack to row1
and make
a new choice



Solve(board, 1)

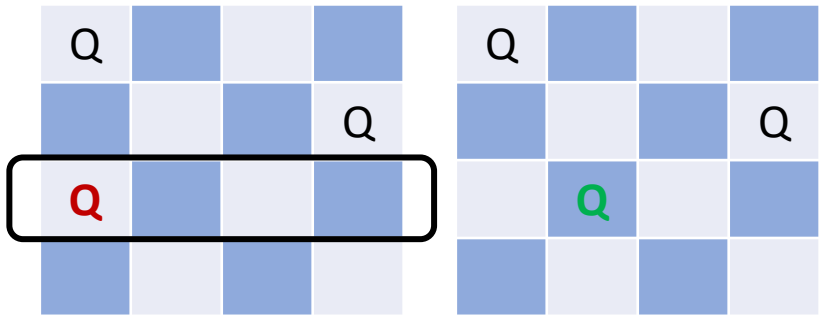
board[1][3]=1;

board[0][0]=1;



```
int Solve(int board[N][N], int row)
{
    if(row==N)
        return true;
    else{
        for(j=0;j<N;j++){
            if(isSafe(board, row, j) == true){
                board[row][j]=1;
                if(Solve(board, row+1) == true)
                    return true;
                else
                    board[row][j]=0;
            }
        }
        return false;
    }
}
```

row 2

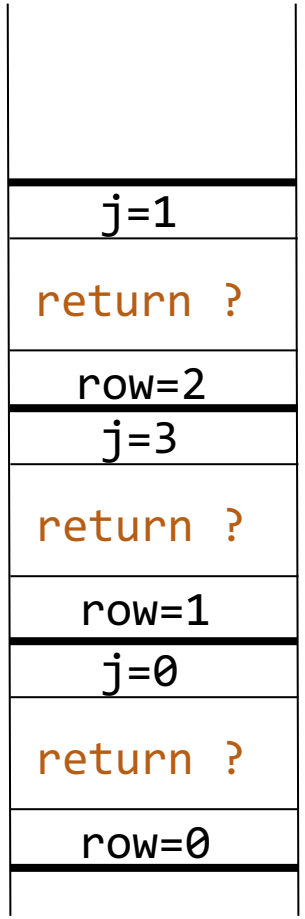


Solve(board, 2)

board[2][1]=1;

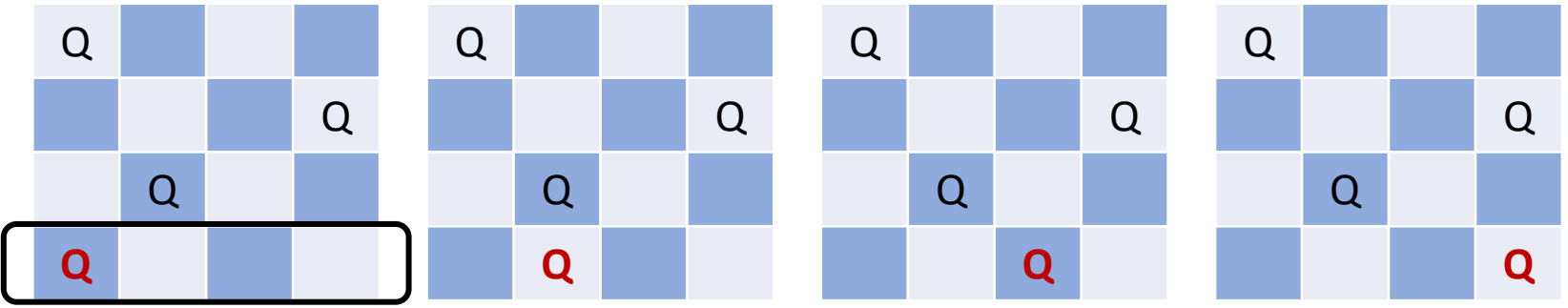
board[1][3]=1;

board[0][0]=1;



```
int Solve(int board[N][N], int row)
{
    if(row==N)
        return true;
    else{
        for(j=0;j<N;j++){
            if(isSafe(board, row, j) == true){
                board[row][j]=1;
                if(Solve(board, row+1) == true)
                    return true;
                else
                    board[row][j]=0;
            }
        }
        return false;
    }
}
```

row 3

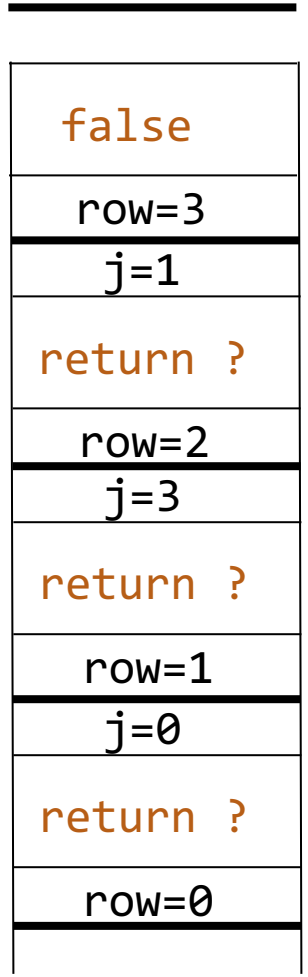


Solve(board, 3)

board[2][1]=1;

board[1][3]=1;

board[0][0]=1;



```
int Solve(int board[N][N], int row)
{
    if(row==N)
        return true;
    else{
        for(j=0;j<N;j++){
            if(isSafe(board, row, j) == true){
                board[row][j]=1;
                if(Solve(board, row+1) == true)
                    return true;
                else
                    board[row][j]=0;
            }
        }
        return false;
    }
}
```

Solve(board, 3) = false

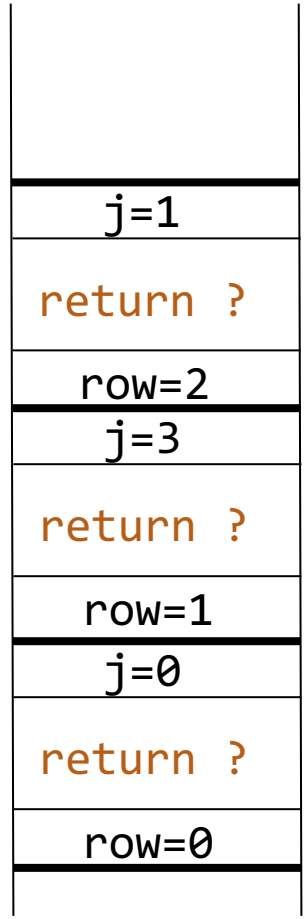
backtrack

Solve(board, 2)

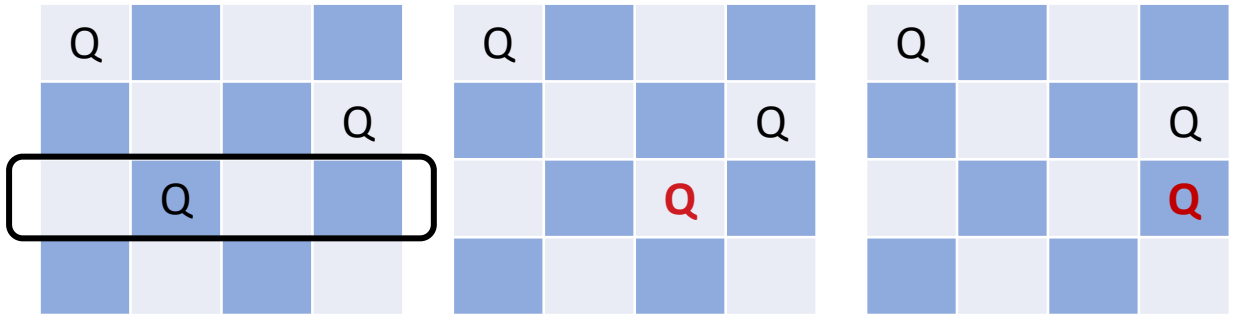
board[2][1]=0;

board[1][3]=1;

board[0][0]=1;



Backtrack to row2 and make a new choice



```
int Solve(int board[N][N], int row)
{
    if(row==N)
        return true;
    else{
        for(j=0;j<N;j++){
            if(isSafe(board, row, j) == true){
                board[row][j]=1;
                if(Solve(board, row+1) == true)
                    return true;
                else
                    board[row][j]=0;
            }
        }
        return false;
    }
}
```

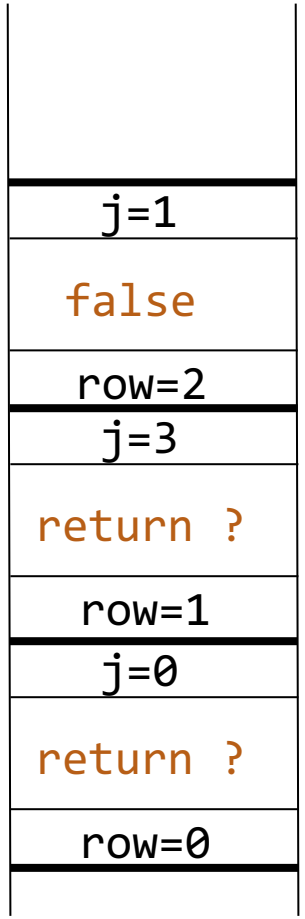
backtrack

Solve(board, 2)

board[2][1]=0;

board[1][3]=1;

board[0][0]=1;



```
int Solve(int board[N][N], int row)
{
    if(row==N)
        return true;
    else{
        for(j=0;j<N;j++){
            if(isSafe(board, row, j) == true){
                board[row][j]=1;
                if(Solve(board, row+1) == true)
                    return true;
                else
                    board[row][j]=0;
            }
        }
        return false;
    }
}
```

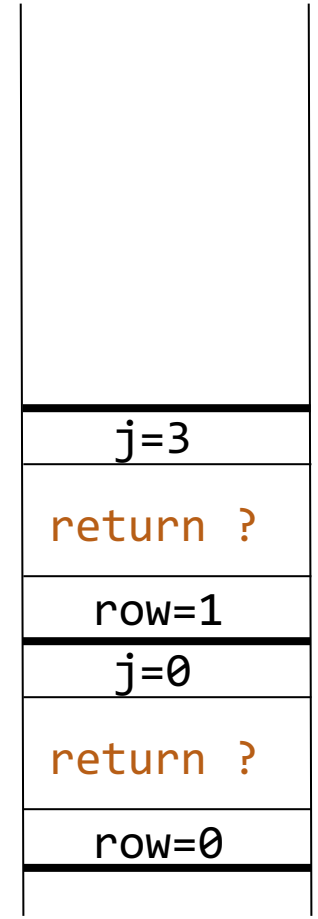
Solve(board, 2) = false

backtrack

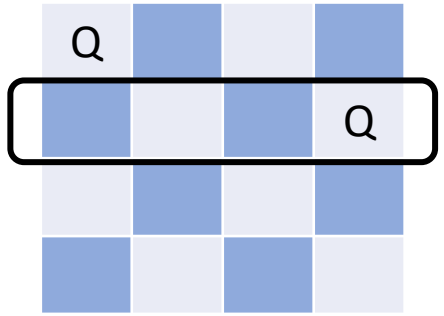
Solve(board, 1)

board[1][3]=0;

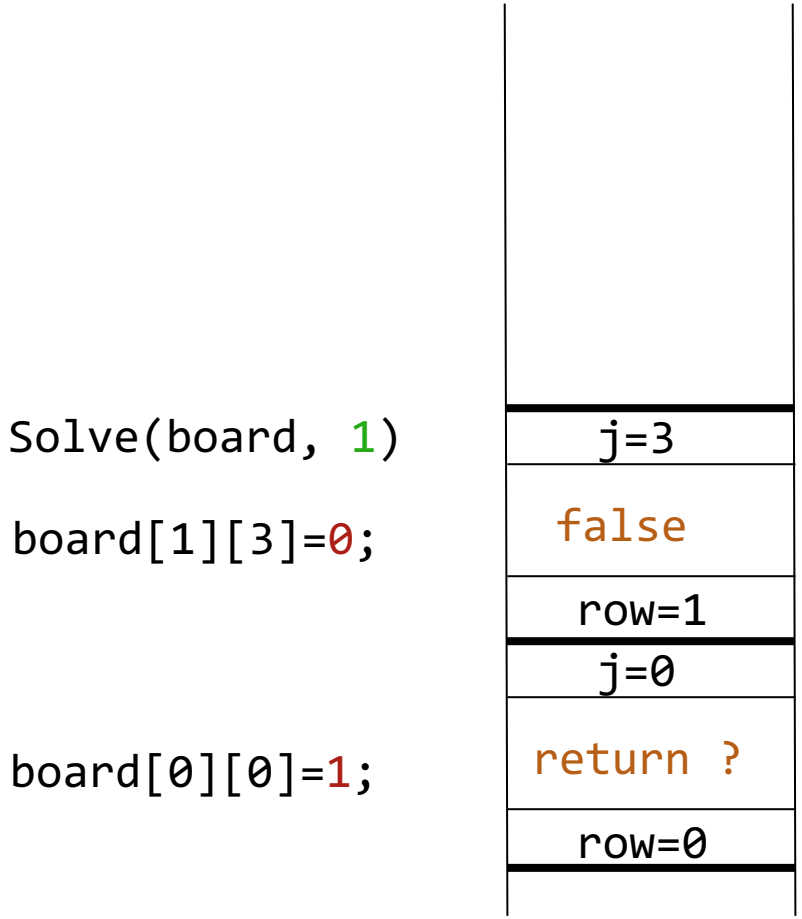
board[0][0]=1;



Backtrack to row1
and make
a new choice



```
int Solve(int board[N][N], int row)
{
    if(row==N)
        return true;
    else{
        for(j=0;j<N;j++){
            if(isSafe(board, row, j) == true){
                board[row][j]=1;
                if(Solve(board, row+1) == true)
                    return true;
                else
                    board[row][j]=0;
            }
        }
        return false;
    }
}
```



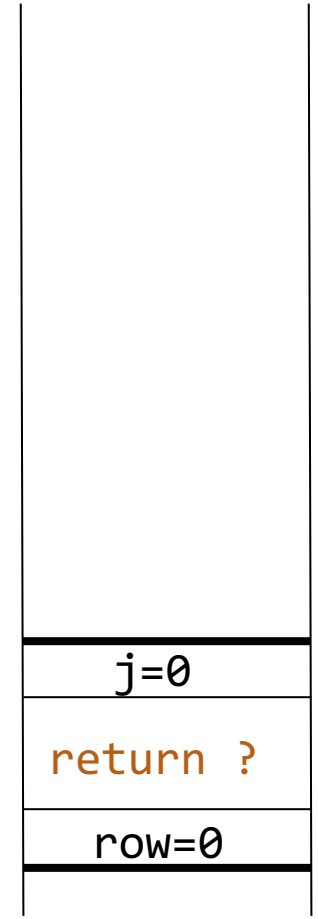
```
int Solve(int board[N][N], int row)
{
    if(row==N)
        return true;
    else{
        for(j=0;j<N;j++){
            if(isSafe(board, row, j) == true){
                board[row][j]=1;
                if(Solve(board, row+1) == true)
                    return true;
                else
                    board[row][j]=0;
            }
        }
        return false;
    }
}
```

Solve(board, 1) = false

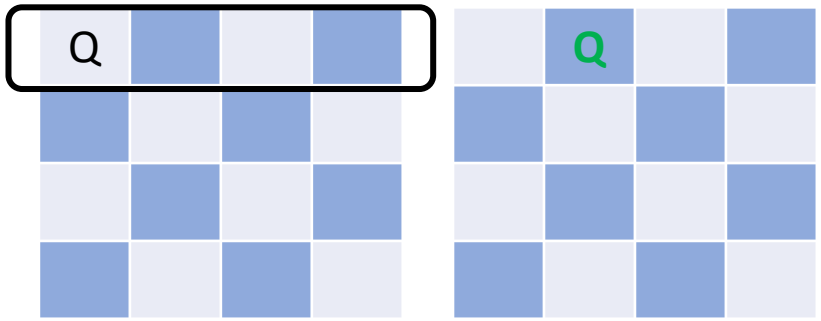
backtrack

Solve(board, 0)

board[0][0]=0;

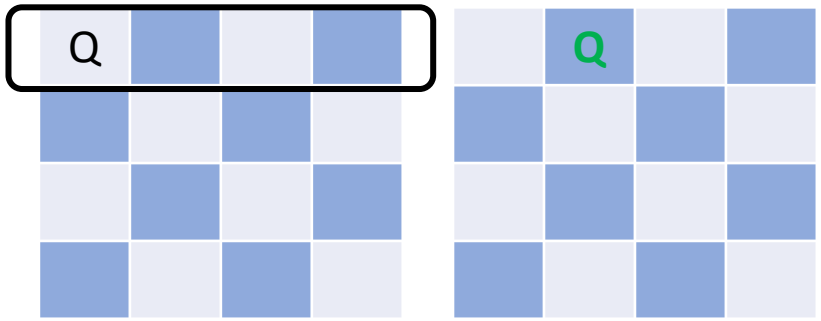


Backtrack to row0 and make a new choice



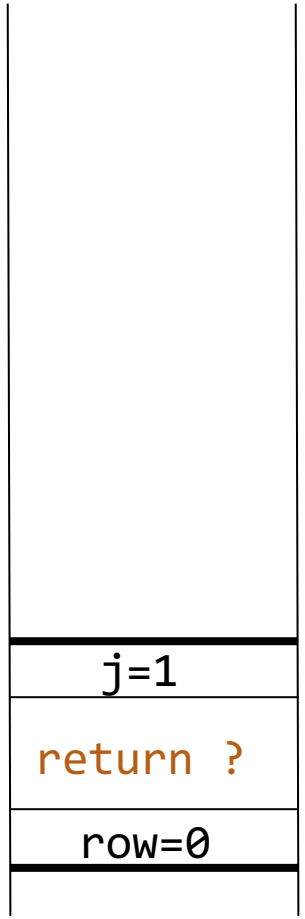
```
int Solve(int board[N][N], int row)
{
    if(row==N)
        return true;
    else{
        for(j=0;j<N;j++){
            if(isSafe(board, row, j) == true){
                board[row][j]=1;
                if(Solve(board, row+1) == true)
                    return true;
                else
                    board[row][j]=0;
            }
        }
        return false;
    }
}
```

Backtrack to row0
and make
a new choice




```
Solve(board, 0)

board[0][1]=1;
```

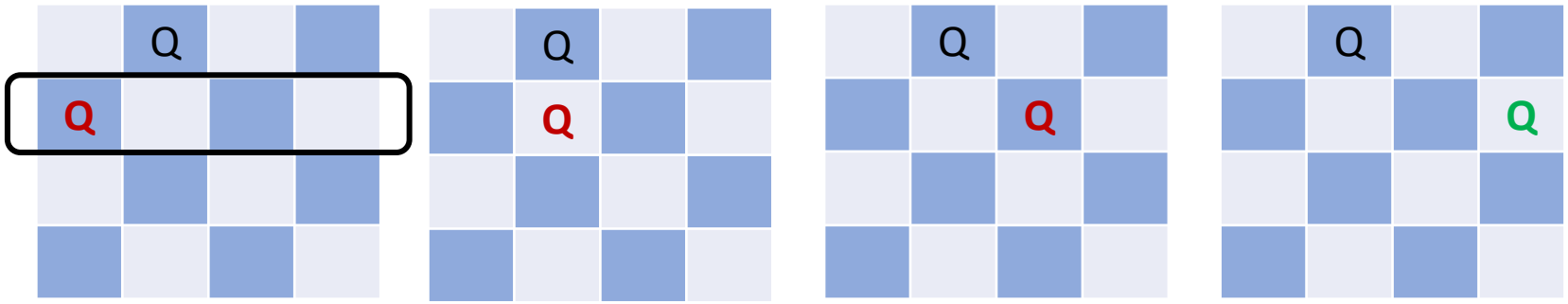


```
int Solve(int board[N][N], int row)
{
    if(row==N)
        return true;
    else{
        for(j=0;j<N;j++){
            if(isSafe(board, row, j) == true){
                board[row][j]=1;
                if(Solve(board, row+1) == true)
                    return true;
                else
                    board[row][j]=0;
            }
        }
        return false;
    }
}
```

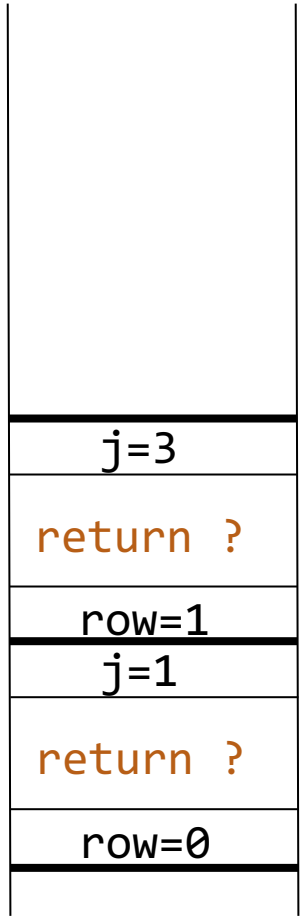
Solve(board, 1)

board[1][3]=1;

row 1

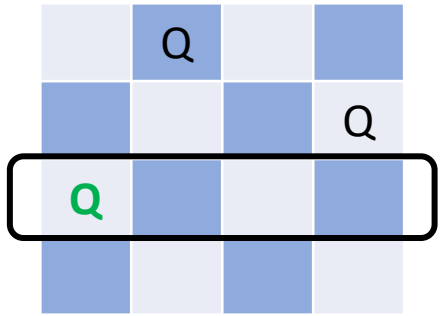


board[0][1]=1;



```
int Solve(int board[N][N], int row)
{
    if(row==N)
        return true;
    else{
        for(j=0;j<N;j++){
            if(isSafe(board, row, j) == true){
                board[row][j]=1;
                if(Solve(board, row+1) == true)
                    return true;
                else
                    board[row][j]=0;
            }
        }
        return false;
    }
}
```

row 2

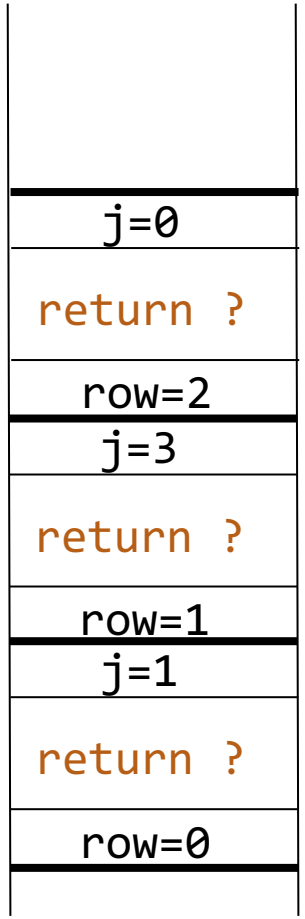


Solve(board, 2)

board[2][0]=1;

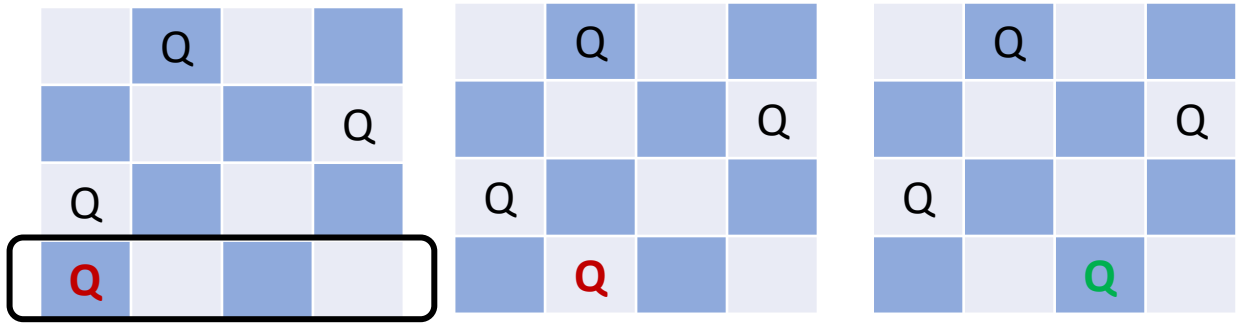
board[1][3]=1;

board[0][1]=1;



```
int Solve(int board[N][N], int row)
{
    if(row==N)
        return true;
    else{
        for(j=0;j<N;j++){
            if(isSafe(board, row, j) == true){
                board[row][j]=1;
                if(Solve(board, row+1) == true)
                    return true;
                else
                    board[row][j]=0;
            }
        }
        return false;
    }
}
```

row 3



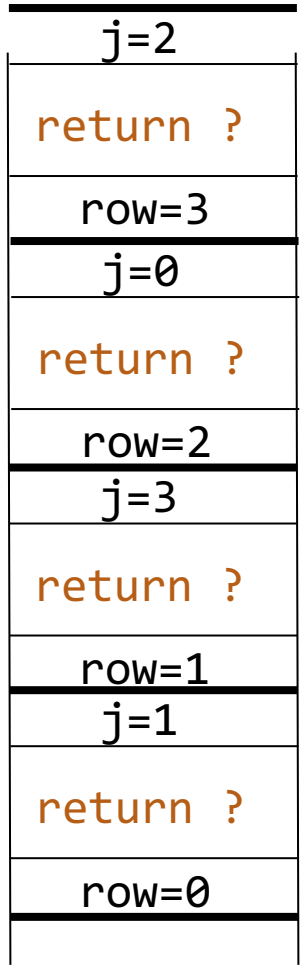
Solve(board, 3)

board[3][2]=1;

board[2][0]=1;

board[1][3]=1;

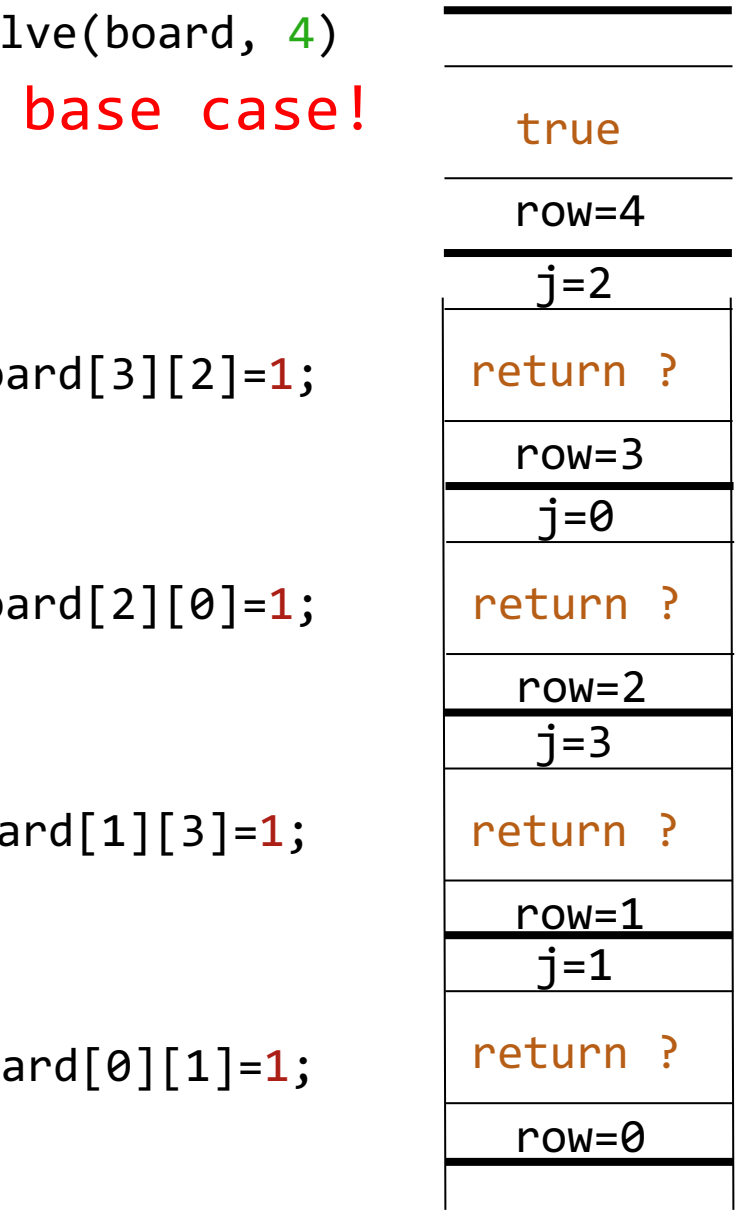
board[0][1]=1;



```
int Solve(int board[N][N], int row)
{
    if(row==N)
        return true;
    else{
        for(j=0;j<N;j++){
            if(isSafe(board, row, j) == true){
                board[row][j]=1;
                if(Solve(board, row+1) == true)
                    return true;
                else
                    board[row][j]=0;
            }
        }
        return false;
    }
}
```

Solve(board, 4)

base case!



```
int Solve(int board[N][N], int row)
{
    if(row==N)
        return true;
    else{
        for(j=0;j<N;j++){
            if(isSafe(board, row, j) == true){
                board[row][j]=1;
                if(Solve(board, row+1) == true)
                    return true;
                else
                    board[row][j]=0;
            }
        }
        return false;
    }
}
```

Solve(board, 4)= true

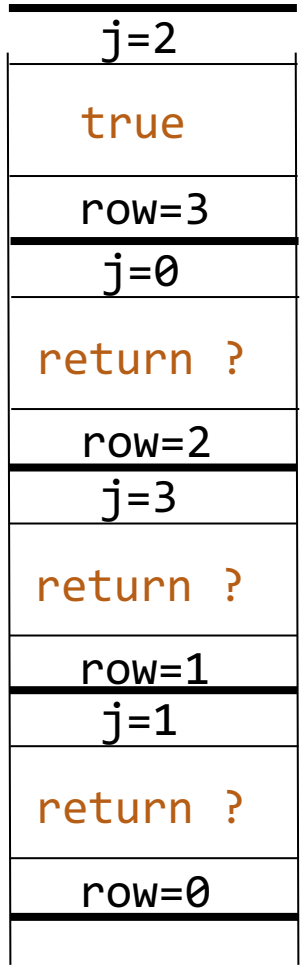
Solve(board, 3)

board[3][2]=1;

board[2][0]=1;

board[1][3]=1;

board[0][1]=1;



```
int Solve(int board[N][N], int row)
{
    if(row==N)
        return true;
    else{
        for(j=0;j<N;j++){
            if(isSafe(board, row, j) == true){
                board[row][j]=1;
                if(Solve(board, row+1) == true)
                    return true;
                else
                    board[row][j]=0;
            }
        }
        return false;
    }
}
```

Solve(board, 3)= true

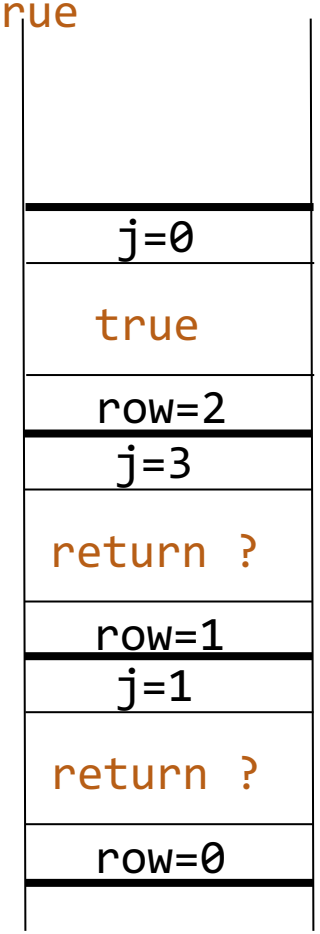
board[3][2]=1;

Solve(board, 2)

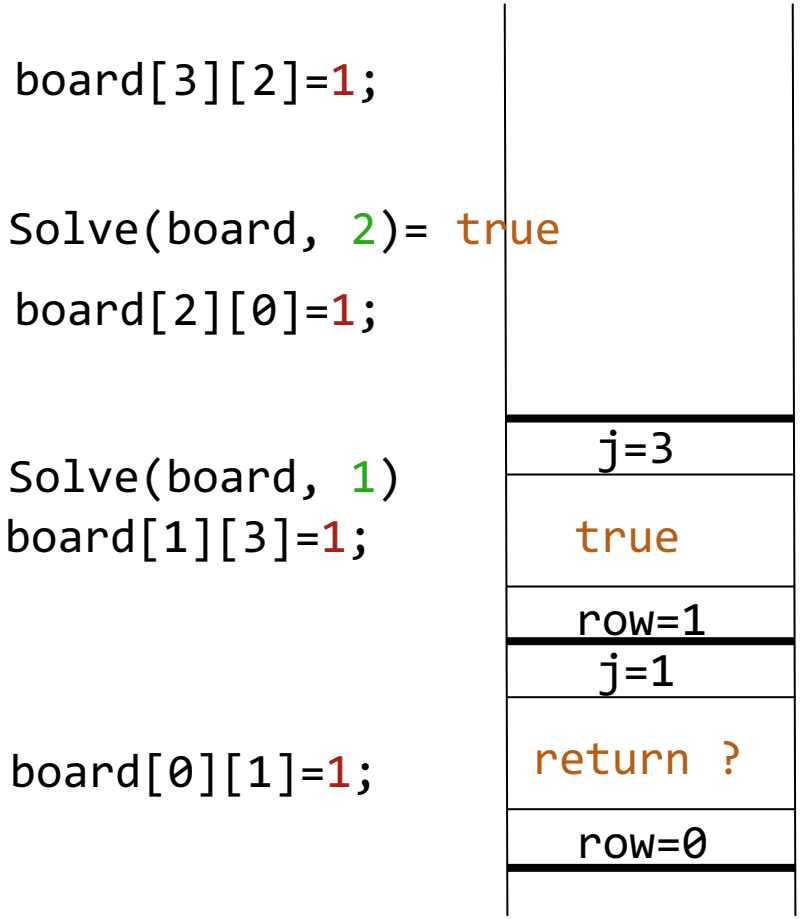
board[2][0]=1;

board[1][3]=1;

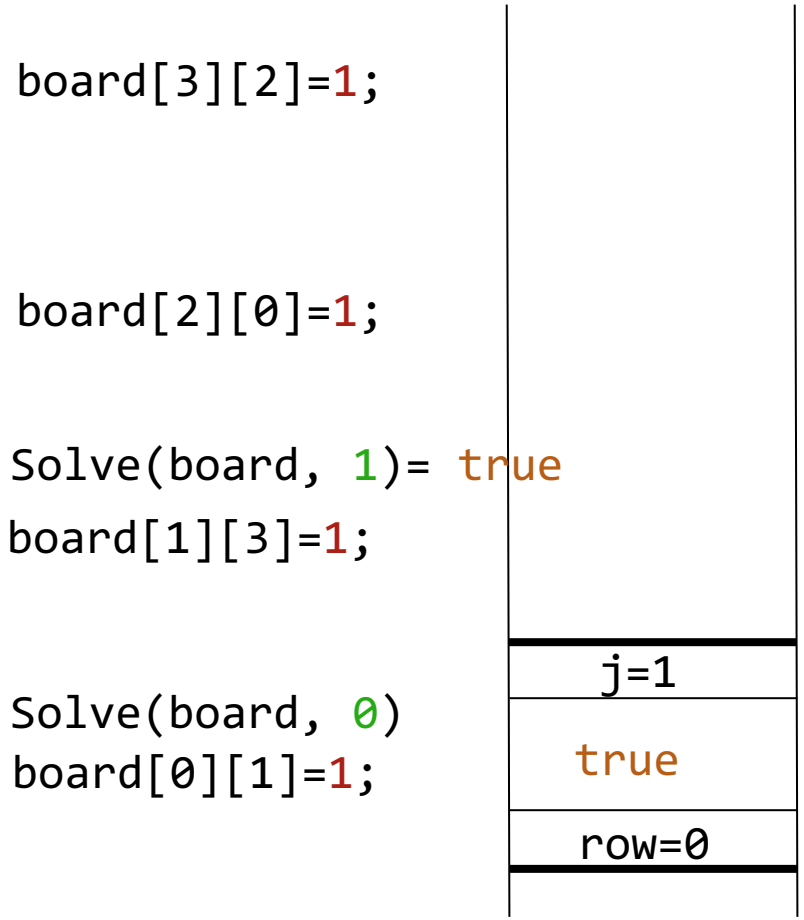
board[0][1]=1;



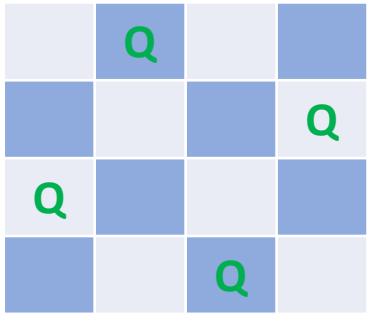
```
int Solve(int board[N][N], int row)
{
    if(row==N)
        return true;
    else{
        for(j=0;j<N;j++){
            if(isSafe(board, row, j) == true){
                board[row][j]=1;
                if(Solve(board, row+1) == true)
                    return true;
                else
                    board[row][j]=0;
            }
        }
        return false;
    }
}
```



```
int Solve(int board[N][N], int row)
{
    if(row==N)
        return true;
    else{
        for(j=0;j<N;j++){
            if(isSafe(board, row, j) == true){
                board[row][j]=1;
                if(Solve(board, row+1) == true)
                    return true;
                else
                    board[row][j]=0;
            }
        }
        return false;
    }
}
```



```
int Solve(int board[N][N], int row)
{
    if(row==N)
        return true;
    else{
        for(j=0;j<N;j++){
            if(isSafe(board, row, j) == true){
                board[row][j]=1;
                if(Solve(board, row+1) == true)
                    return true;
                else
                    board[row][j]=0;
            }
        }
        return false;
    }
}
```




```
board[3][2]=1;

board[2][0]=1;

board[1][3]=1;

Solve(board, 0)= true
board[0][1]=1;
```