**Data Structure and Algorithm Practicals**

15. Practical based on backtracking- N Queen’s problems

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<body>

<script>

function getnqueens (order) {

if (order<4) {

console.log ("n queens problem apply for order bigger than 3");

return;

}

var nqueens=[];

var backtracking=false;

rowloop:

for (var row=0;row<order;row ++) {

if (nqueens [row] === undefined) {

nqueens [row]=[];

}

for (var col=0;col<order;col ++) {

if (nqueens [row] [col] === 0) {

continue;

} else if (backtracking &&nqueens [row] [col] == 1) {

if (col === order-1) {

resetrow (nqueens, order, row);

row=row-2;

continue rowloop;

}

nqueens [row] [col]=0;

backtracking=false;

continue;

}

nqueens [row] [col]=1;

if (isqueenvalid (nqueens, row, col)) {

continue rowloop;

} else if (col == order-1) {

backtracking=true;

resetrow (nqueens, order, row);

row=row-2;

continue rowloop;

} else {

nqueens [row] [col]=0;

continue;

};

}

}

return nqueens;

}

function resetrow (nqueens, order, row) {

for (var col=0;col<order;col ++) {

nqueens [row] [col]=undefined;

}

}

function isqueenvalid (nqueens, row, col) {

for (var i=0;i<col;i ++) {

if (nqueens [row] [i] == 1) {

return false;

}

}

for (var j=1;j<row + 1;j ++) {

if (nqueens [row-j] [col] == 1 || (nqueens [row-j] [col-j]!=undefined &&nqueens [row-j] [col-j] == 1) || ( nqueens [row-j] [col + j]!=undefined &&nqueens [row-j] [col + j] == 1)) {

return false;

}

}

return true;

}

function printqueens (queens) {

for (var row=0;row<queens.length;row ++) {

var rowtext="";

for (var col=0;col<queens.length;col ++) {

if (queens [row] [col] === undefined) {

queens [row] [col]=0;

}

rowtext=rowtext + queens [row] [col] + "";

}

console.log (rowtext);

}

}

var queens=getnqueens (4);

printqueens (queens);

</script>

</body>

</html>