Practical 1 A - 5 sept 2020

Aim : Tower of hanoi

Solution :

var hanoi = function(disc,src,aux,dest){

if(disc > 0){

hanoi(disc -1,src,dest,aux);

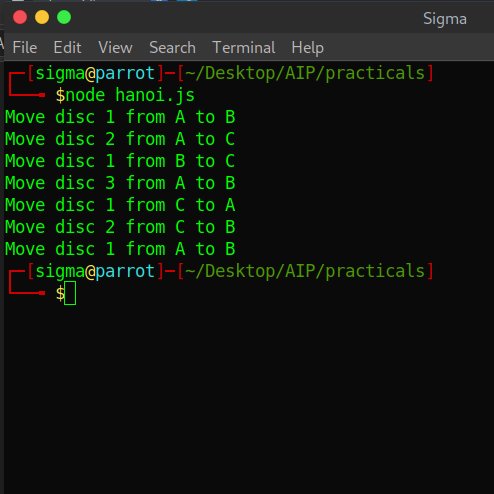
console.log('Move disc '+disc+' from '+src+' to '+dest);

hanoi(disc -1,aux,src,dest);

}

};

hanoi(3,'A','C','B');



Practical : 1 B - 5 sept 2020

Aim : NQueen Problem

Solution :

var n = 4;

solveNQ();

function printSolution(board){

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

for(var j=0; j<n; j++) {

var m = board[i][j]+"";

m = m.replace("1","Q");

m = m.replace("0","#");

process.stdout.write(" "+m+" ");

}

process.stdout.write("\n");

}

process.stdout.write("\n");

}

function isSafe(board, row, col){

*// Checks the ← direction*

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

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

return false;

}

}

*// Checks the ↖ direction*

for(var i=row, j=col; i>=0 && j>=0; i--, j--){

if (board[i][j] === 1) {

return false;

}

}

*// Checks the ↙ direction*

for(var i=row, j=col; j>=0 && i<n; i++, j--){

if (board[i][j] === 1){

return false;

}

}

return true;

}

function recurseNQ(board, col){

if(col===n){

printSolution(board); *// <-- print another solution when n==8*

return;

}

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

if(isSafe(board, i, col)){

board[i][col]=1;

recurseNQ(board, col+1);

*//if(recurseNQ(board, col+1)===true) //<-- you don't need this*

*// return true;*

board[i][col]=0;

}

}

return false;

}

function solveNQ(){

var board = generateBoard(n);

recurseNQ(board, 0);

}

function generateBoard(n){

var board=[];

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

board[i]=[];

for(var j=0; j<n; j++){

board[i][j]=0;

}

}

return board;

}

Output :

