**Program 1:**

const perf = require('execution-time')();

var x = []

var count = 0

n = 0

unique = false

function place(k,i){

var tp = 0

for(var j = 1; j <= k-1; j++){

tp = j

if( ( x[j] == i ) || ( Math.abs( x[j]-i) == Math.abs(j-k) ) ){

return false;

}

}

console.log("Next position of queen ",tp+1,i)

return i;

}

let nqueens = (k,n) => {

for(var i = 1; i <= n; ++i)

{

if(place(k,i))

{

x[k]=i;

if(k == n)

write();

else

nqueens(k+1,n);+0

}

}

};

function write(){

console.log("----------------------")

var i = 0;

for( var x\_tmp of x ){

var line = ""

if(i != 0){

for( j = 1 ; j <= n ; ++j){

if( x[i] == j ){

line += "Q";

}

else{

line += "."

}

}

console.log(line)

}

i += 1

}

console.log("\n")

}

////////////////////Question 1////////////////////

q1 = 4

n = q1

///////////////////Question 2.1/////////////////////

// q21 = [4,5,6,7]

// for( var t of q21 ){

// n = t

// perf.start();

// nqueens(1,n)

// const results = perf.stop();

// console.log(results.time);

// }

// ///////////////////Question 2.2/////////////////////

// q22 = [4,5,6,7,8,9,10]

console.log("Predicted positions are given below For n = ",n)

nqueens(1,q1)

**Output:**

Predicted positions are given below For n = 4

Next position of queen 1 1

Next position of queen 2 3

Next position of queen 2 4

Next position of queen 3 2

Next position of queen 1 2

Next position of queen 2 4

Next position of queen 3 1

Next position of queen 4 3

----------------------

.Q..

...Q

Q...

..Q.

Next position of queen 1 3

Next position of queen 2 1

Next position of queen 3 4

Next position of queen 4 2

----------------------

..Q.

Q...

...Q

.Q..

Next position of queen 1 4

Next position of queen 2 1

Next position of queen 3 3

Next position of queen 2 2

**Program 2:**

const perf = require('execution-time')();

var x = []

var count = 0

n = 8

unique = false

matrix = []

function place(k,i){

for(var j = 1; j <= k-1; j++){

if( ( x[j] == i ) || ( Math.abs( x[j]-i) == Math.abs(j-k) ) ){

return false;

}

}

return true;

}

let nqueens = (k,n) => {

for(var i = 1; i <= n; ++i)

{

if(place(k,i))

{

x[k]=i;

if(k == n){

// write()

// console.log(matrix, x.slice(1))

if( !validate( matrix, x.slice(1) ) ){

matrix.push(x.slice(1))

}

}

else

nqueens(k+1,n);+0

}

}

};

function write(){

var counter = 0

for( var temp\_x of matrix ){

console.log("----------------------")

console.log("Solutions ",counter += 1)

var i = 0;

for( var x\_tmp of x ){

var line = ""

if(i != 0){

for( j = 1 ; j <= n ; ++j){

if( x[i] == j ){

line += "Q";

}

else{

line += "."

}

}

console.log(line)

}

i += 1

}

console.log("\n")

}

}

function validate(matrix,x){

var a1 = x

var n = a1.length

var a2 = []

for( var t of a1 ){

a2.push((n-t)+1)

}

for( var temp\_x of matrix ){

var t0\_q = false

var t1\_q = false

var t2\_q = false

var t3\_q = false

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

if(temp\_x[i] != a1.reverse()[i]){

break

}

t1\_q = true

}

if (t1\_q ){

break

}

}

return t1\_q//(t0\_q || t1\_q || t2\_q || t3\_q)

}

ns = [4,5,6,7,8]

for( var temp\_n of ns ){

x = []

count = 0

n = temp\_n

unique = false

matrix = []

q1 = n

nqueens(1,q1)

console.log("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

console.log("Total Unique Solutions are for n = ",n)

console.log(matrix.length)

write()

}

**Output:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Total Unique Solutions are for n = 4

1

----------------------

Solutions 1

...Q

.Q..

..Q.

.Q..

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Total Unique Solutions are for n = 5

5

----------------------

Solutions 1

....Q

..Q..

Q....

...Q.

.Q...

----------------------

Solutions 2

....Q

..Q..

Q....

...Q.

.Q...

----------------------

Solutions 3

....Q

..Q..

Q....

...Q.

.Q...

----------------------

Solutions 4

....Q

..Q..

Q....

...Q.

.Q...

----------------------

Solutions 5

....Q

..Q..

Q....

...Q.

.Q...

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Total Unique Solutions are for n = 6

3

----------------------

Solutions 1

.....Q

...Q..

.Q....

....Q.

..Q...

.Q....

----------------------

Solutions 2

.....Q

...Q..

.Q....

....Q.

..Q...

.Q....

----------------------

Solutions 3

.....Q

...Q..

.Q....

....Q.

..Q...

.Q....

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Total Unique Solutions are for n = 7

9

----------------------

Solutions 1

......Q

....Q..

..Q....

.....Q.

...Q...

...Q...

.Q.....

----------------------

Solutions 2

......Q

....Q..

..Q....

.....Q.

...Q...

...Q...

.Q.....

----------------------

Solutions 3

......Q

....Q..

..Q....

.....Q.

...Q...

...Q...

.Q.....

----------------------

Solutions 4

......Q

....Q..

..Q....

.....Q.

...Q...

...Q...

.Q.....

----------------------

Solutions 5

......Q

....Q..

..Q....

.....Q.

...Q...

...Q...

.Q.....

----------------------

Solutions 6

......Q

....Q..

..Q....

.....Q.

...Q...

...Q...

.Q.....

----------------------

Solutions 7

......Q

....Q..

..Q....

.....Q.

...Q...

...Q...

.Q.....

----------------------

Solutions 8

......Q

....Q..

..Q....

.....Q.

...Q...

...Q...

.Q.....

----------------------

Solutions 9

......Q

....Q..

..Q....

.....Q.

...Q...

...Q...

.Q.....

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Total Unique Solutions are for n = 8

12

----------------------

Solutions 1

.......Q

.....Q..

...Q....

......Q.

....Q...

....Q...

..Q.....

....Q...

----------------------

Solutions 2

.......Q

.....Q..

...Q....

......Q.

....Q...

....Q...

..Q.....

....Q...

----------------------

Solutions 3

.......Q

.....Q..

...Q....

......Q.

....Q...

....Q...

..Q.....

....Q...

----------------------

Solutions 4

.......Q

.....Q..

...Q....

......Q.

....Q...

....Q...

..Q.....

....Q...

----------------------

Solutions 5

.......Q

.....Q..

...Q....

......Q.

....Q...

....Q...

..Q.....

....Q...

----------------------

Solutions 6

.......Q

.....Q..

...Q....

......Q.

....Q...

....Q...

..Q.....

....Q...

----------------------

Solutions 7

.......Q

.....Q..

...Q....

......Q.

....Q...

....Q...

..Q.....

....Q...

----------------------

Solutions 8

.......Q

.....Q..

...Q....

......Q.

....Q...

....Q...

..Q.....

....Q...

----------------------

Solutions 9

.......Q

.....Q..

...Q....

......Q.

....Q...

....Q...

..Q.....

....Q...

----------------------

Solutions 10

.......Q

.....Q..

...Q....

......Q.

....Q...

....Q...

..Q.....

....Q...

----------------------

Solutions 11

.......Q

.....Q..

...Q....

......Q.

....Q...

....Q...

..Q.....

....Q...

----------------------

Solutions 12

.......Q

.....Q..

...Q....

......Q.

....Q...

....Q...

..Q.....

....Q...

**Program 3: (timing & extra credit)**

const perf = require('execution-time')();

var x = []

var count = 0

n = 0

unique = false

nodes = 1

function place(k,i){

var tp = 0

for(var j = 1; j <= k-1; j++){

tp = j

if( ( x[j] == i ) || ( Math.abs( x[j]-i) == Math.abs(j-k) ) ){

return false;

}

}

return i;

}

let nqueens = (k,n) => {

nodes += 1

for(var i = 1; i <= n; ++i)

{

if(place(k,i))

{

x[k]=i;

if(k == n){

}

else

nqueens(k+1,n);+0

}

}

};

///////////////////Question 2.2/////////////////////

function main(){

ns = [4,5,6,7,8,9,10]

time\_graph\_data = []

node\_estimation\_data = []

for( var temp\_n of ns ){

x = []

count = 0

n = temp\_n

unique = false

q1 = n

perf.start();

nqueens(1,n)

const results = perf.stop();

time\_graph\_data.push([n,results.time] );

total\_nodes = 1

for ( var i = 0; i <= (n-1) ; i++ ) {

temp = 1

for ( var j = 0; j < (i+1) ; j++ ) {

temp = temp \* (n-j)

}

total\_nodes += temp

}

node\_estimation\_data.push([nodes,total\_nodes])

}

console.log(time\_graph\_data)

console.log(node\_estimation\_data)

}

main()

Output:

node Nqueens-sampath3.js

N val vs Time

[ [ 4, 0.64123 ],

[ 5, 0.18798199999999998 ],

[ 6, 0.9123859999999999 ],

[ 7, 1.302951 ],

[ 8, 2.3110269999999997 ],

[ 9, 3.193562 ],

[ 10, 15.474864 ] ]

Counted nodes vs total nodes

[ [ 16, 65 ],

[ 60, 326 ],

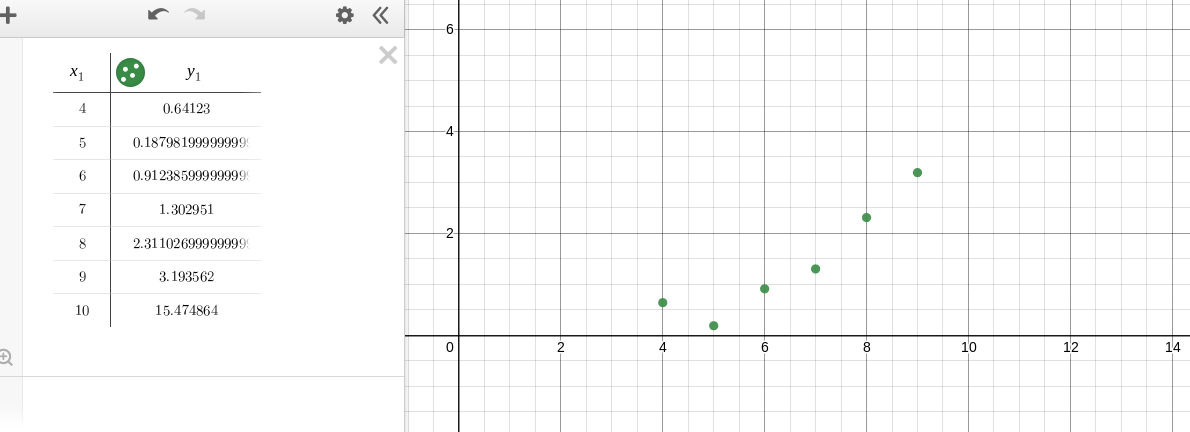
[ 209, 1957 ],

[ 721, 13700 ],

[ 2686, 109601 ],

[ 10728, 986410 ],

[ 45543, 9864101 ] ]

****