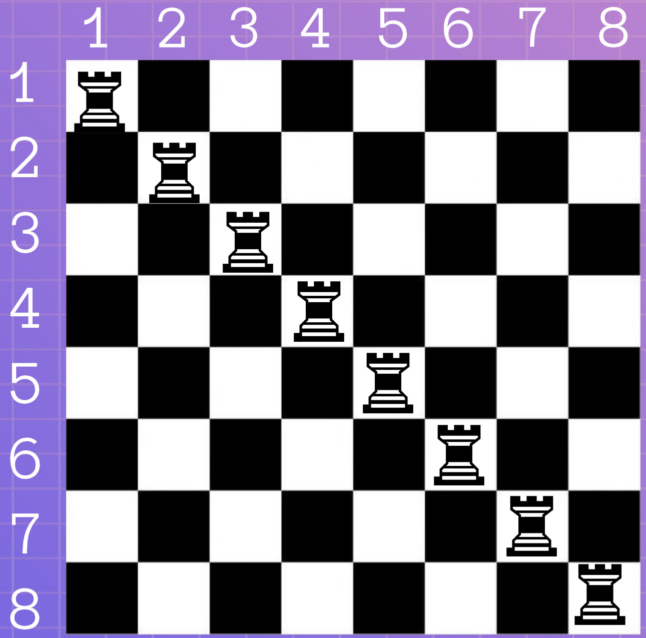
PROBLEMA DE LAS N-TORRES CON OBSTÁCULOS

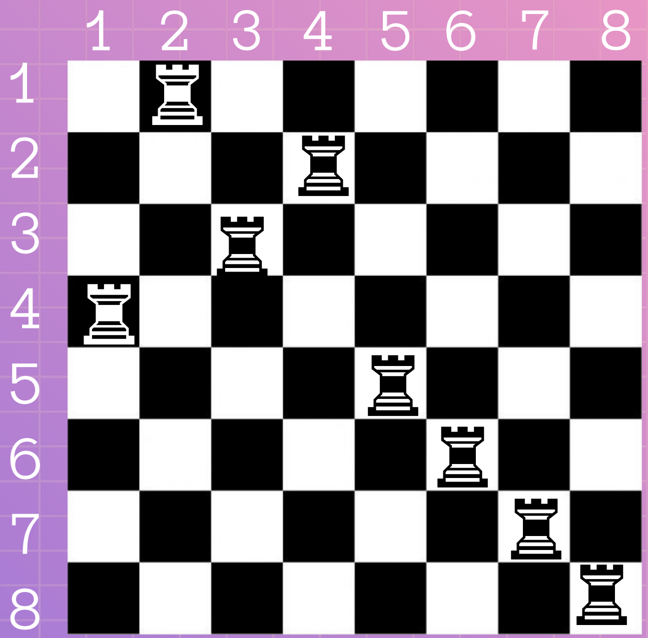
PAULA CUADRA RUIZ

SAMUEL HIDALGO BERLANGA

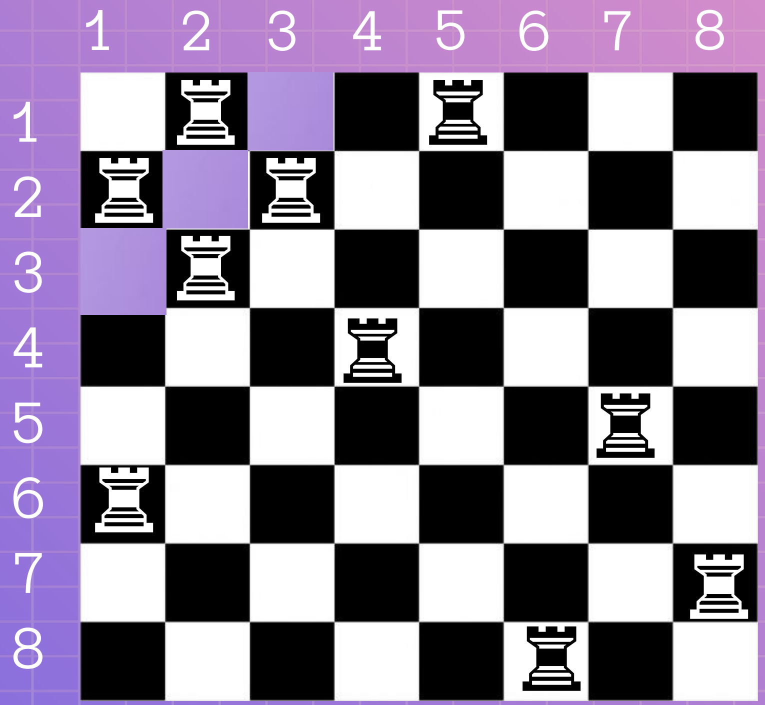
SERGIO CAMACHO MARÍN

♜ PROBLEMA DE LAS N-TORRES ♜





♜ PROBLEMA DE LAS N-TORRES CON ♜ OBSTÁCULOS



PUNTO DE PARTIDA

- PROBLEMA DE LAS N-REINAS

```
#const n = 8.

% domain number
(1..n).

% alldifferent
1 { q(X,Y) : number(Y) } 1 :- number(X).
1 { q(X,Y) : number(X) } 1 :- number(Y).

% remove conflicting answers
:- q(X1,Y1), q(X2,Y2), X1 < X2, Y1 == Y2.
:- q(X1,Y1), q(X2,Y2), X1 == X2, Y1 < Y2.
:- q(X1,Y1), q(X2,Y2), X1 < X2, Y1 + X1 == Y2 + X2.
:- q(X1,Y1), q(X2,Y2), X1 < X2, Y1 - X1 == Y2 - X2.
#show q/2.
```

DESARROLLO Y DIFICULTADES

```
1 { t(X,Y) : number(Y) } 1 :- number(X).
1 { t(X,Y) : number(X) } 1 :- number(Y).
```

Enunciado inicial que solo permite 1 torre por fila/columna.
Debido a los obstáculos esto no es cierto

```
{t(X,Y) : number(X),number(Y) } == k.
```

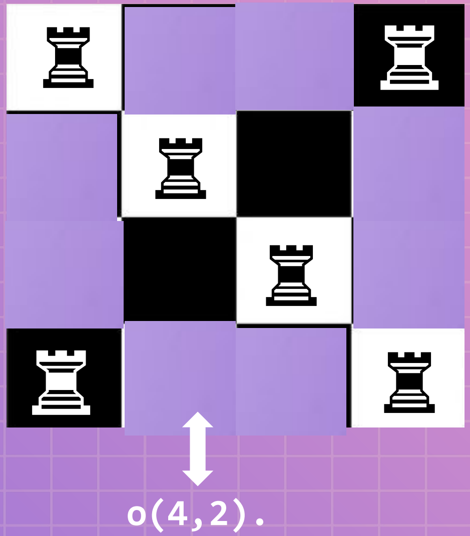
K es el número de torres que queremos colocar.
De esta forma no presenta la limitación del caso base

DESARROLLO Y DIFICULTADES

```
:- t(X1,Y1), t(X2,Y2), X1<X2, Y1=Y2, not o(X3,Y3), X1=X3, Y1<Y3, Y3<Y2, Y3=Y1..Y2.
:- t(X1,Y1), t(X2,Y2), X1=X2, Y1<Y2 , not o(X3,Y3) , Y1=Y3 , X1<X3 , X3<X2, X3=X1..X2.
:- t(X1,Y1), t(X1,Y2), Y2= Y1+1.
:- t(X1,Y1), t(X2,Y1), X2 = X1+1.
```

Primer planteamiento de los obstáculos. Solamente da satisfacible en casos en los que entre una torre y otra, **todas las casillas son obstáculos**

Declaramos los obstáculos con o(X,Y) donde X,Y son las coordenadas del tablero

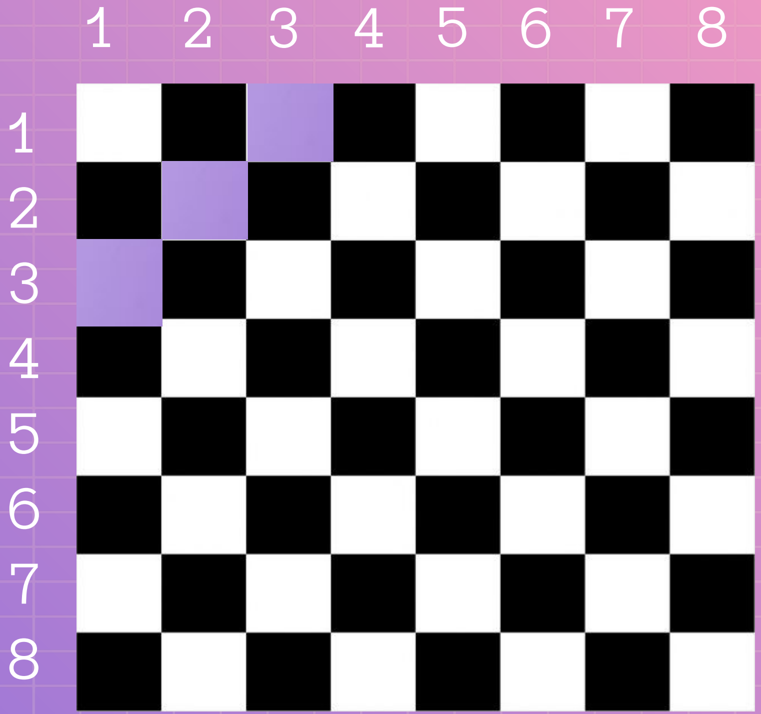


PRIMERA SOLUCIÓN VÁLIDA

```
#const n = 8.
```

```
number(1..n).
```

El número de torres que queremos posicionar en el tablero

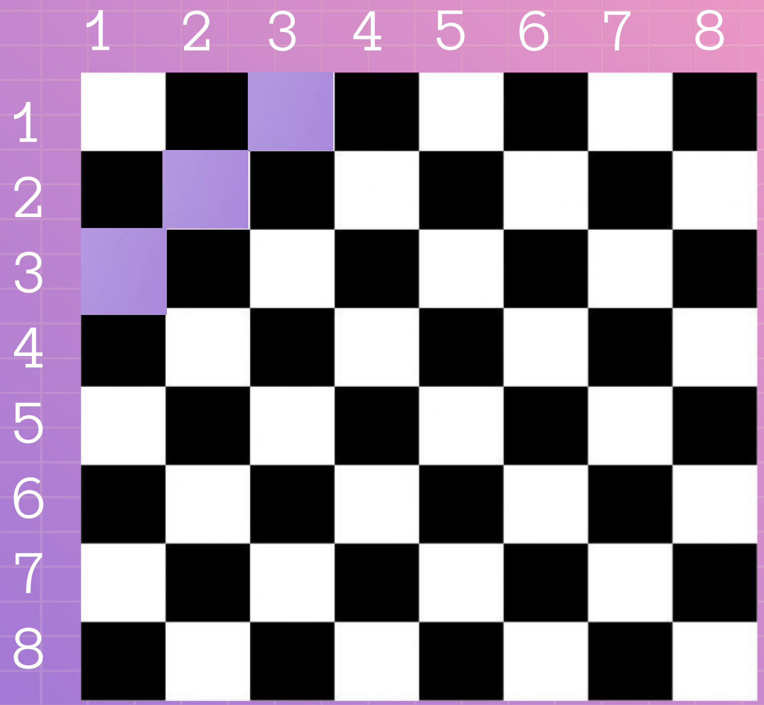



```
{t(X,Y) : number(X),number(Y) } == 10.
```

o(2,2).
o(3,1).
o(1,3).

Colocamos un número determinado de obstáculos

Siendo o(X,Y) obstáculos con coordenadas:
X = 1..8.
Y = 1..8.



Definimos un predicado “libre” para saber que casillas están libres de obstáculos

```
libre(X1,Y1) :- not o(X1,Y1), X1=1..n , Y1=1..n.
```

Inferencia que nos indica que no puede haber una torre encima de un obstáculo

```
:- t(X1,Y1), o(X2,Y2), X1==X2 , Y1==Y2.
```

Calculamos el número de casillas libres que podría haber entre ellas si consideramos que no hay obstáculos

Las dos torres están en la misma fila

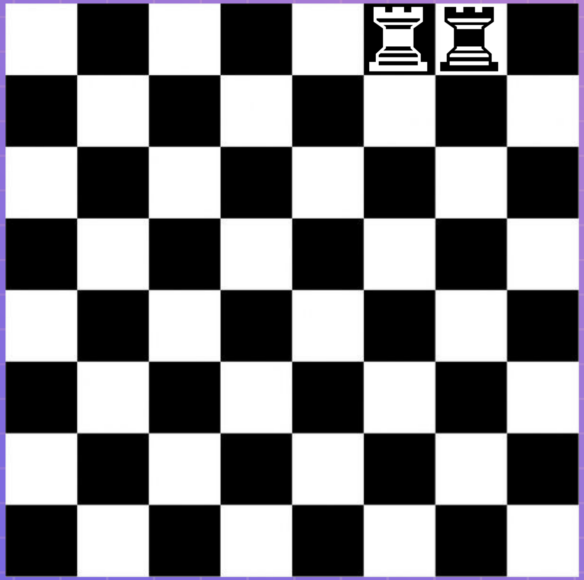
```
:- t(X1,Y1), t(X2,Y1), X1<X2, #count{X,Y1 : libre(X,Y1), X=X1+1..X2-1 }==(n-X1-(n-X2+1)).
```

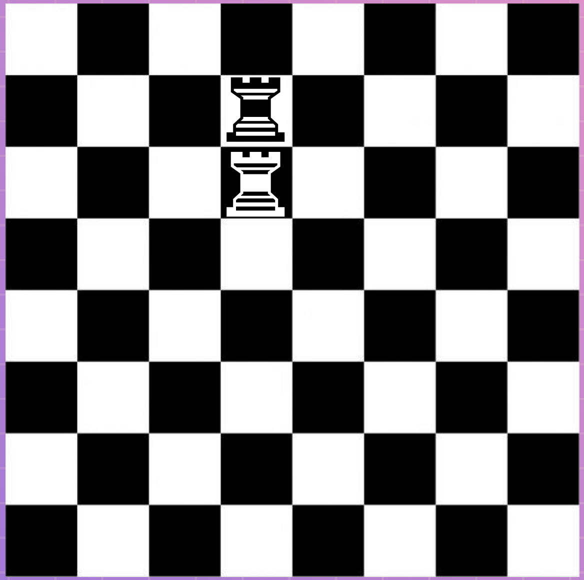
Comparamos ese número con el número de casillas libres reales que hay en la fila.
Si el número es el mismo, no hay obstáculos.

Las dos torres están en la misma columna

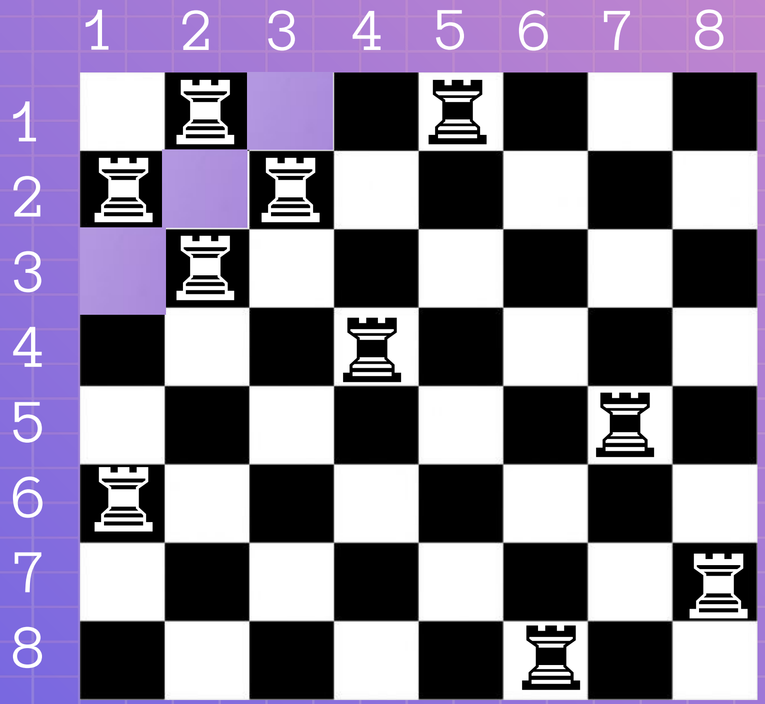
```
:- t(X1,Y1), t(X1,Y2), Y1<Y2, #count{X1,Y : libre(X1,Y), Y=Y1+1..Y2-1 }==(n-Y1-(n-Y2+1)).
```

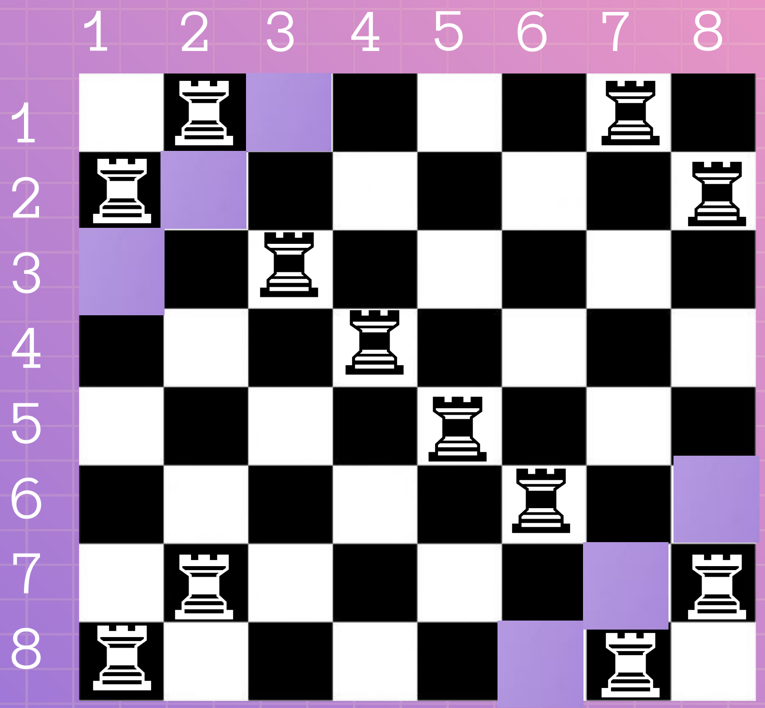
```
:- t(X1,Y1), t(X1,Y2), Y2= Y1+1.
:- t(X1,Y1), t(X2,Y1), X2 = X1+1.
```





TRES OBSTÁCULOS Y 10 TORRES
SEIS OBSTÁCULOS Y 12 TORRES





number(1) number(2) number(3) number(4) number(5) number(6) number(7) number(8) o(2,2) o(3,1) o(1,3) libre(1,1) libre(1,2) libre(1,4) libre(1,5) libre(1,6) libre(1,7) libre(1,8) libre(2,1) libre(2,3) libre(2,4) libre(2,5) libre(2,6) libre(2,7) libre(2,8) libre(3,2) libre(3,3) libre(3,4) libre(3,5) libre(3,6) libre(3,7) libre(3,8) libre(4,1) libre(4,2) libre(4,3) libre(4,4) libre(4,5) libre(4,6) libre(4,7) libre(4,8) libre(5,1) libre(5,2) libre(5,3) libre(5,4) libre(5,5) libre(5,6) libre(5,7) libre(5,8) libre(6,1) libre(6,2) libre(6,3) libre(6,4) libre(6,5) libre(6,6) libre(6,7) libre(6,8) libre(7,1) libre(7,2) libre(7,3) libre(7,4) libre(7,5) libre(7,6) libre(7,7) libre(7,8) libre(8,1) libre(8,2) libre(8,3) libre(8,4) libre(8,5) libre(8,6) libre(8,7) libre(8,8)

t(2,1) t(6,1) t(1,2) t(3,2) t(2,3) t(4,4) t(1,5) t(8,6) t(5,7) t(7,8)

SATISFIABLE

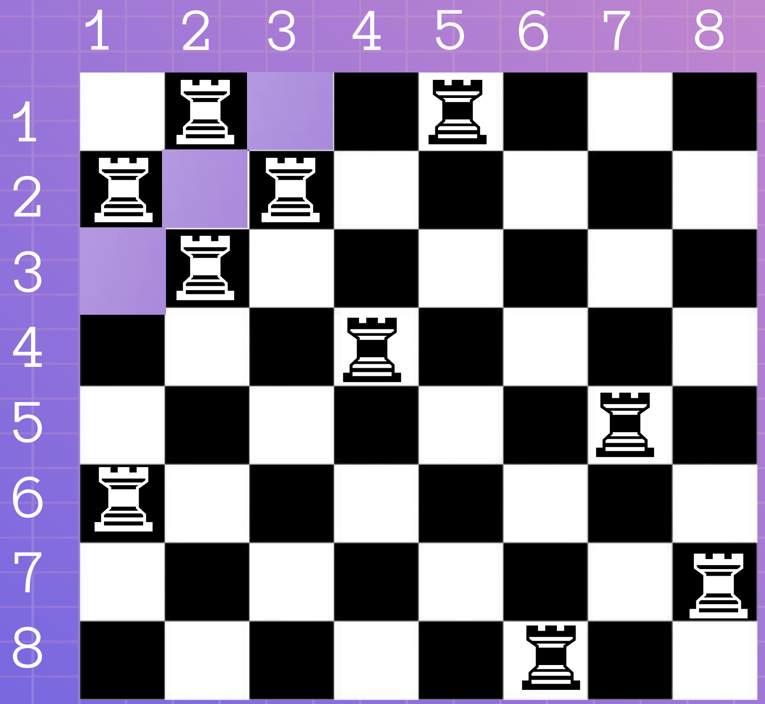
CÓDIGO FINAL

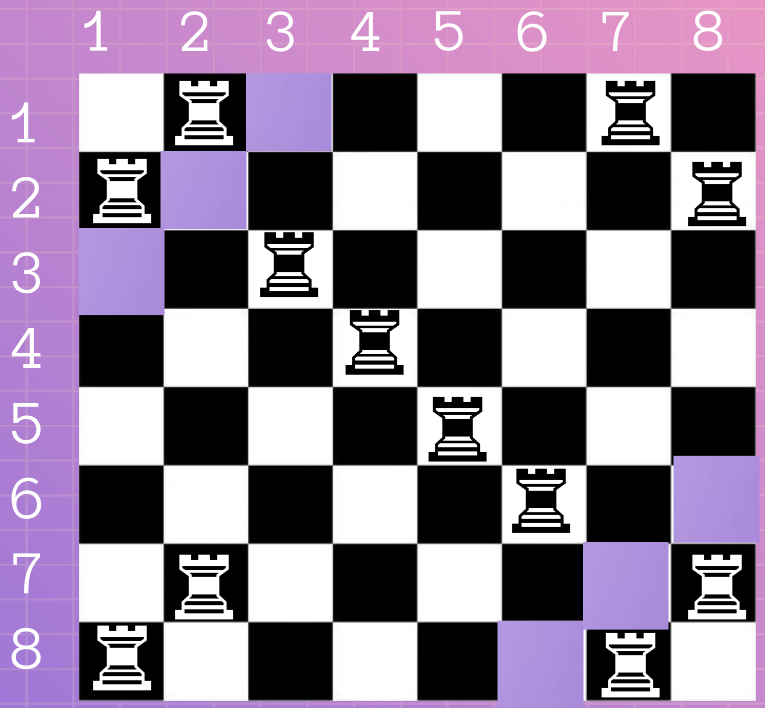
```
#const n = 8.

number(1..n).

%El número de torres que queremos posicionar en el tablero.

{t(X,Y) : number(X),number(Y) } == 12.

o(2,2). o(3,1). o(1,3). o(8,6). o(7,7). o(6,8).


:- t(X1,Y1), o(X2,Y2), X1==X2 , Y1==Y2.

:- t(X1,Y1), t(X2,Y1), X1 < X2, #count{X,Y1 : o(X,Y1), X=X1+1..X2-1 } < 1.

:- t(X1,Y1), t(X1,Y2), Y1 < Y2, #count{X1,Y : o(X1,Y) ,Y=Y1+1..Y2-1 } < 1.


#show t/2.
```

Cuenta los obstáculos, y si el número de obstáculos es menor a 1, no se puede dar la situación de dos torres en la misma fila/columna

[Running clingo (potassco.org)](#)

TRES OBSTÁCULOS Y 10 TORRES
SEIS OBSTÁCULOS Y 12 TORRES





AMPLIACIONES: CALCULAR MÁXIMO N-TORRES

- Utilizamos iclingo, una ampliación de clingo.

- Valores acumulativos. #cumulative k.

- Podemos indicar que una variable k aumenta a medida que avanzan las iteraciones.

- Con la acción --istop=UNSAT el programa acaba con la última entrada satisfacible.

Gracias por su atención