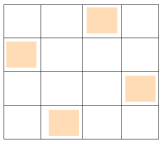
Lab10: Reducción de N-reinas a SAT

1. **Explicación de la reducción**

   El problema de colocar n reinas en un tablero de ajedrez de $n \times n$, de forma que no se coman unas a otras, es un problema difícil. Por ejemplo, una solución para 4-reinas es la que se indica en el tablero de abajo.



   Podemos reducir este problema a encontrar una asignación que haga cierta una determinada fórmula. A continuación explicamos cómo se transforma la instancia n-reinas en una instancia del problema SAT. Haremos todo el desarrollo para el caso 4-reinas.

   La fórmula booleana φ_n en forma normal conjuntiva está formada por n^2 variables. En nuestro ejemplo son 4^2 variables.

$x_{1,1}$	$x_{1,2}$	$x_{1,3}$	$x_{1,4}$
$x_{2,1}$	$x_{2,2}$	$x_{2,3}$	$x_{2,4}$
$x_{3,1}$	$x_{3,2}$	$x_{3,3}$	$x_{3,4}$
$x_{4,1}$	$x_{4,2}$	$x_{4,3}$	$x_{4,4}$

   En general, una variable $x_{i,j}$ con $1 \leq i, j \leq n$ se evalúa a 1 si, y sólo si, la posición (i, j) en el tablero está ocupada por una reina.

   La fórmula φ_n está formada por las siguientes cláusulas:

   (a) Un conjunto de cláusulas que expresan que en cada fila i debe haber una posición j ocupada por una reina y sólo una posición.

   En una fila hay una reina:

$$(x_{i,1} \vee x_{i,2} \vee \cdots \vee x_{i,n-1} \vee x_{i,n}) \text{ para cada } i \text{ tal que } 1 \leq i \leq n$$

   En nuestro ejemplo:

$$(x_{1,1} \vee x_{1,2} \vee x_{1,3} \vee x_{1,4}) \wedge (x_{2,1} \vee x_{2,2} \vee x_{2,3} \vee x_{2,4}) \wedge$$
$$(x_{3,1} \vee x_{3,2} \vee x_{3,3} \vee x_{3,4}) \wedge (x_{4,1} \vee x_{4,2} \vee x_{4,3} \vee x_{4,4})$$

   Una fila no puede albergar dos reinas:

$$\neg(x_{i,j} \wedge x_{i,k}) \text{ para cada } 1 \leq i, j, k \leq n \text{ con } j < k$$

   Para que la codificación sea una CNF hay que transformar las fórmulas anteriores a sus correspondientes CNFs:

$$(\neg x_{i,j} \vee \neg x_{i,k}) \text{ para cada } 1 \leq i, j, k \leq n \text{ con } j < k$$

   En nuestro ejemplo, para la fila 1:

$$(\neg x_{1,1} \vee \neg x_{1,2}) \wedge (\neg x_{1,1} \vee \neg x_{1,3}) \wedge (\neg x_{1,1} \vee \neg x_{1,4}) \wedge$$
$$(\neg x_{1,2} \vee \neg x_{1,3}) \wedge (\neg x_{1,2} \vee \neg x_{1,4}) \wedge (\neg x_{1,3} \vee \neg x_{1,4})$$
$$\dots$$

Para la fila 4:

$$(\neg x_{4,1} \vee \neg x_{4,2}) \wedge (\neg x_{4,1} \vee \neg x_{4,3}) \wedge (\neg x_{4,1} \vee \neg x_{4,4}) \wedge$$
$$(\neg x_{4,2} \vee \neg x_{4,3}) \wedge (\neg x_{4,2} \vee \neg x_{4,4}) \wedge (\neg x_{4,3} \vee \neg x_{4,4})$$

(b) Un conjunto de cláusulas que expresan que en cada columna j debe haber una posición i y sólo una ocupada por una reina.

En una columna hay una reina:

$$(x_{1,j} \vee x_{2,j} \vee \dots \vee x_{n-1,j} \vee x_{n,j}) \text{ para cada } j \text{ tal que } 1 \leq j \leq n$$

En nuestro ejemplo:

$$(x_{1,1} \vee x_{2,1} \vee x_{3,1} \vee x_{4,1}) \wedge (x_{1,2} \vee x_{2,2} \vee x_{3,2} \vee x_{4,2}) \wedge$$
$$(x_{1,3} \vee x_{2,3} \vee x_{3,3} \vee x_{4,3}) \wedge (x_{1,4} \vee x_{2,4} \vee x_{3,4} \vee x_{4,4})$$

Una columna no puede albergar dos reinas:

$$\neg(x_{i,j} \wedge x_{k,j}) \text{ para cada } 1 \leq j, i, k \leq n \text{ con } i < k$$

Transformando las fórmulas anteriores a sus correspondientes CNFs:

$$(\neg x_{i,j} \vee \neg x_{k,j}) \text{ para cada } 1 \leq j, i, k \leq n \text{ con } i < k$$

En nuestro ejemplo, para cada columna j:

$$(\neg x_{1,j} \vee \neg x_{2,j}) \wedge (\neg x_{1,j} \vee \neg x_{3,j}) \wedge (\neg x_{1,j} \vee \neg x_{4,j}) \wedge$$
$$(\neg x_{2,j} \vee \neg x_{3,j}) \wedge (\neg x_{2,j} \vee \neg x_{4,j}) \wedge (\neg x_{3,j} \vee \neg x_{4,j})$$

(c) En las diagonales no puede haber dos reinas.

Hay dos tipos de diagonales: Las diagonales descendentes son las que van de una casilla a la casilla de la siguiente columna y siguiente fila. Las diagonales ascendentes son las que van de una casilla a la siguiente columna pero en una fila anterior.

Dos posiciones (i, j) e (i', j') están en una diagonal descendente siempre y cuando $i - j = i' - j'$. Dos posiciones (i, j) e (i', j') están en una diagonal ascendente siempre y cuando $i + j = i' + j'$.

no puede haber dos reinas en las diagonales descendentes:

$$(\neg x_{i,j} \vee \neg x_{i',j'}) \text{ para cada } 1 \leq i, j, i', j' \leq n \text{ con } i < i' \text{ y además } i - j = i' - j'$$

En nuestro ejemplo las diagonales descendentes son:

$$[x_{1,3}, x_{2,4}],\ [x_{1,2}, x_{2,3}, x_{3,4}],\ [x_{1,1}, x_{2,2}, x_{3,3}, x_{4,4}],\ [x_{2,1}, x_{3,2}, x_{4,3}] \text{ y } [x_{3,1}, x_{4,2}]$$

La cláusulas correspondientes son:

$$(\neg x_{1,3} \vee \neg x_{2,4}) \wedge (\neg x_{1,2} \vee \neg x_{2,3}) \wedge (\neg x_{1,2} \vee \neg x_{3,4}) \wedge (\neg x_{2,3} \vee \neg x_{3,4}) \wedge$$
$$(\neg x_{1,1} \vee \neg x_{2,2}) \wedge (\neg x_{1,1} \vee \vee \neg x_{3,3}) \wedge (\neg x_{1,1} \vee \vee \neg x_{4,4}) \wedge$$
$$(\neg x_{2,2} \vee \vee \neg x_{3,3}) \wedge (\neg x_{2,2} \vee \vee \neg x_{4,4}) \wedge (\neg x_{3,3} \vee \vee \neg x_{4,4}) \wedge$$
$$(\neg x_{2,1} \vee \neg x_{3,2}) \wedge (\neg x_{2,1} \vee \vee \neg x_{4,3}) \wedge (\neg x_{3,2} \vee \neg x_{4,3}) \wedge (\neg x_{3,1} \vee \neg x_{4,2})$$

no puede haber dos reinas en las diagonales ascendentes:

$(\neg x_{i,j} \vee \neg x_{i',j'})$ para cada $1 \leq i, j, i', j' \leq n$ con $i > i'$ y además $i + j = i' + j'$

Las diagonales ascendentes son:

$[x_{2,1}, x_{1,2}]$, $[x_{3,1}, x_{2,2}, x_{1,3}]$, $[x_{4,1}, x_{3,2}, x_{2,3}, x_{1,4}]$, $[x_{4,2}, x_{3,3}, x_{2,4}]$ y $[x_{4,3}, x_{3,4}]$

La cláusulas correspondientes son:

$$(\neg x_{2,1} \vee \neg x_{1,2}) \wedge (\neg x_{3,1} \vee \neg x_{2,2}) \wedge (\neg x_{3,1} \vee \neg x_{1,3}) \wedge (\neg x_{2,2} \vee \neg x_{1,3}) \wedge$$
$$(\neg x_{4,1} \vee \neg x_{3,2}) \wedge (\neg x_{4,1} \vee \vee \neg x_{2,3}) \wedge (\neg x_{4,1} \vee \vee \neg x_{1,4}) \wedge$$
$$(\neg x_{3,2} \vee \vee \neg x_{2,3}) \wedge (\neg x_{3,2} \vee \vee \neg x_{1,4}) \wedge (\neg x_{2,3} \vee \vee \neg x_{1,4}) \wedge$$
$$(\neg x_{4,2} \vee \neg x_{3,3}) \wedge (\neg x_{4,2} \vee \vee \neg x_{2,4}) \wedge (\neg x_{3,3} \vee \neg x_{2,4}) \wedge (\neg x_{4,3} \vee \neg x_{3,4})$$

Esta es la construcción de φ_n. Observa que a partir de una interpretación que satisface la fórmula, se pueden colocar la n reinas en el tablero. Si la variable $x_{i,j}$ tiene asignado el valor 1, en la posición (i, j) habrá una reina.

2. **Primera tarea a realizar**

   Debes implementar la función `reduce_nqueens_to_SAT` que dado un n devuelve la lista de cláusulas que codifica el problema de las n reinas. Esto es, φ_n, pero en formato DIMACS (para minisat). Para ello abre el fichero `nqueensToSat.py`

   ATENCIÓN:

   La lista que devuelve tu función debe estar en formato DIMACS. Por tanto la primera sublista debe ser: `["p", "cnf", num_vars, num_clauses]`

   donde `num_vars` es el número de variables de tu fórmula y `num_clauses` el número de cláusulas. Además las sublistas siguientes tienen que acabar con un 0 y estar formadas por literales que son números positivos o negativos.

   Por tanto es necesario asociar a cada variable $x_{i,j}$ un único número $cod(x_{i,j})$. Una codificación sencilla puede ser:

   $$\boxed{cod(x_{i,j}) \equiv (i-1)n + j}$$

   donde n es el número de reinas a colocar.

   En nuestro ejemplo:

$x_{1,1}$	1	$x_{1,2}$	2	$x_{1,3}$	3	$x_{1,4}$	4
$x_{2,1}$	5	$x_{2,2}$	6	$x_{2,3}$	7	$x_{2,4}$	8
$x_{3,1}$	9	$x_{3,2}$	10	$x_{3,3}$	11	$x_{3,4}$	12
$x_{4,1}$	13	$x_{4,2}$	14	$x_{4,3}$	15	$x_{4,4}$	16

3. **Segunda tarea a realizar**

   Hay preparados unos juegos de prueba para comparar tiempos de ejecución de varios programas que solucionan el problema de la n reinas:

   - Una solución hecha con bactracking `nqueens_backtracking.py`.
   - La solución dada por `minisat` a partir de tu codificación.
   - La solución dada por tu `sat_solver` a partir de tu codificación. Recuerda que has hecho un `3sat_solver` para solucionar 3SAT. Las instancias para el problema de las n reinas contienen cláusulas con más de tres literares. Por lo tanto, tienes que actualizar tu `3sat_solver` para que sea un `nsat_solver`.

     Cambia tu función `solve_3SAT` por otra que admita cláusulas con cualquier número de literales. Llama a esta nueva función `solve_nSAT`. Nombra el fichero que contiene `solve_nSAT` como nsat_solver.py y cópialo en la carpeta `Code for Students` del laboratorio 10.

   Ahora ya puedes ejecutar los juegos de prueba. Ejecútalos uno a uno. Cada juego genera un fórmula aleatoria con un determinado número de variables y llama a la función `reduce_nqueens_to_SAT` que devuelve tu codificación φ_n. A continuación llama a `minisat` con φ_n. Después llama a la función `nqueens_backtracking.py` y por último llama a tu `solve_nSAT`. Para cada llamada muestra la solución encontrada en la ventana plots.

   Mira los tiempos de cada llamada en la carpeta `myresults` donde se habrá creado un fichero `n_queens_times.csv`