**1. ¿Cómo funciona el backtracking en este problema?**

El backtracking es una técnica de búsqueda que prueba todas las posibles soluciones de manera sistemática. En este caso:

* Se intenta colocar una reina en cada fila del tablero.
* Si es seguro colocar una reina en una posición (usando el método isSafe), se coloca y se pasa a la siguiente fila.
* Si no es posible colocar una reina en ninguna columna de la fila actual, el algoritmo retrocede (**backtrack**) a la fila anterior, descoloca la reina y prueba otra columna.
* Este proceso se repite hasta encontrar una solución o determinar que no hay solución.

**2. ¿Qué pasa cuando el algoritmo encuentra una solución?**

* Cuando el algoritmo encuentra una solución (es decir, cuando row == N), devuelve true y termina la recursión.
* El tablero se imprime con las posiciones de las reinas marcadas con 1.

**3. ¿Qué ocurre cuando no puede colocar más reinas?**

* Si no es posible colocar una reina en ninguna columna de la fila actual, el método solve devuelve false.
* Esto hace que el algoritmo retroceda a la fila anterior, descoloque la reina (board[row][col] = 0) y pruebe otra columna.

**4. ¿Qué sucede en el código cuando el algoritmo "retrocede"? ¿Cómo se visualiza en Python Tutor?**

* El retroceso ocurre cuando el método solve devuelve false.
* En el código, esto se refleja en la línea board[row][col] = 0, donde se descoloca la reina.
* En Python Tutor, puedes ver cómo el algoritmo retrocede a la fila anterior y prueba nuevas columnas. Cada vez que retrocede, el estado del tablero cambia, y las reinas se recolocan en nuevas posiciones.

**5. ¿Qué modificaciones harías para aumentar N a 8? ¿Cómo crees que cambiaría el tiempo de ejecución?**

* Para aumentar N a 8, simplemente cambia el valor de la constante N a 8:

java

Copy

private static final int N = 8;

* El tiempo de ejecución aumentará exponencialmente con N, ya que el problema de las N-Reinas tiene una complejidad de tiempo de **O(N!)**. Para N = 8, el tiempo de ejecución será manejable, pero para valores más grandes (como N = 30), el tiempo será significativamente mayor.

**6. ¿Por qué el método isSafe es crucial en este algoritmo?**

* El método isSafe verifica si es seguro colocar una reina en una posición específica del tablero.
* Comprueba:
  1. **Columna**: Si hay otra reina en la misma columna.
  2. **Diagonal superior izquierda**: Si hay otra reina en la diagonal hacia arriba a la izquierda.
  3. **Diagonal superior derecha**: Si hay otra reina en la diagonal hacia arriba a la derecha.
* Sin este método, el algoritmo no podría garantizar que las reinas no se ataquen entre sí.