¿Cómo funciona el backtracking en este problema?

En el caso del problema de las N reinas, el algoritmo utiliza backtracking para colocar reinas en un tablero de tamaño NxN de manera que no se ataquen entre sí. El backtracking sigue estos pasos:

1. Colocar una reina en una fila. Comienza en la primera fila y prueba colocar una reina en cada columna de esa fila.
2. Antes de colocar la reina, se asegura de que no haya ninguna otra reina en la misma columna, en la diagonal superior izquierda o en la diagonal superior derecha. Esto se verifica con el método isSafe.
3. Si la reina puede colocarse de manera segura, el algoritmo pasa a la siguiente fila y repite el proceso, probando cada columna de esa fila.
4. Retroceso (backtracking): Si el algoritmo no puede colocar una reina en ninguna columna de una fila, retrocede y mueve la reina en la fila anterior a la siguiente columna posible. Deshace su última acción y prueba una opción diferente.

¿Qué pasa cuando el algoritmo encuentra una solución?

Cuando el algoritmo encuentra una solución el programa devuelve true y termina. El tablero resultante muestra las posiciones de las reinas.

¿Qué ocurre cuando no puede colocar más reinas?

Si el algoritmo no puede colocar una reina en ninguna columna de una fila, retrocede a la fila anterior y mueve la reina en esa fila a la siguiente columna disponible. Este proceso se repite hasta que todas las filas sean exploradas.

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

Cuando el algoritmo retrocede, se deshace de la colocación de una reina en una posición y la vuelve a poner en 0 en el tablero.

En Python Tutor se puede ver cómo el estado del tablero cambia, mostrando cómo se colocan y se eliminan las reinas a medida que el algoritmo explora diferentes configuraciones y retrocede.

¿Qué modificaciones harías para aumentar N a 8?

Para cambiar el valor de N a 8, simplemente se tendría que modificar la constante N en el código a 8:

private static final int N = 8;

¿Cómo crees que cambiaría el tiempo de ejecución?

El tiempo de ejecución crecería exponencialmente con el aumento de N debido a que el algoritmo está basado en backtracking, que tiene una complejidad de tiempo de O(N!).

A medida que N crece el número de posibles configuraciones para colocar las reinas aumenta muy rápido, haciendo que el algoritmo sea mucho más lento a medida que N se incrementa.

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

El método isSafe es crucial porque garantiza que las reinas colocadas en el tablero no se ataquen entre sí. Sin esta verificación, el algoritmo podría colocar reinas en posiciones inseguras (en la misma columna o en las diagonales). Este método es lo que permite que el backtracking funcione correctamente al asegurarse de que cada movimiento sea válido antes de continuar con la recursión.