**J - Chessboard and Queens**

**1. Reducción del Problema**

* **Descripción en palabras:**

Se nos presenta un tablero de ajedrez de 8×8 en el que algunas casillas están libres (.) y otras están ocupadas o reservadas (\*). El objetivo es ubicar ocho reinas en el tablero de forma que ninguna pueda atacar a otra, y solo es posible colocarlas en las casillas libres. Cabe destacar que las casillas reservadas no bloquean los ataques entre reinas. Lo que se busca es determinar cuántas formas diferentes existen para colocar las ocho reinas cumpliendo estas condiciones.

* **Descripción en lenguaje matemático:**

Tenemos un tablero de ajedrez de 8×8. Cada casilla (i, j) puede estar libre ('.') o reservada ('\*'). El objetivo es contar el número de formas de colocar 8 reinas Q1, Q2, ..., Q8 en el tablero cumpliendo las siguientes condiciones:

1. Cada reina debe colocarse en una casilla libre ('.').
2. No puede haber dos reinas en la misma fila, en la misma columna ni en la misma diagonal. Es decir, para cualquier par de reinas Qx y Qy, se debe cumplir que ix ≠ iy, jx ≠ jy y |ix − iy| ≠ |jx − jy|.
3. Deben colocarse exactamente 8 reinas.

El problema consiste en determinar cuántas formas distintas existen de ubicar las 8 reinas respetando estas restricciones.

**2. Reducción de la Solución**

**Descripción en palabras:**Se asigna exactamente una reina a cada una de las 8 filas del tablero. Para cada fila, se elige una columna libre tal que no conflija con ninguna reina ya colocada en la misma columna ni en ninguna de las dos diagonales. Este proceso se puede realizar mediante búsqueda exhaustiva (backtracking) con poda inmediata en el caso de detectar conflicto de columna o diagonal.

**Descripción en lenguaje matemático:**  
 Sea R = {1, 2, …, 8}. Buscamos todas las funciones biyectivas π: R → R (permutaciones de columnas) que satisfagan:

1. Para todo i en R, la casilla (i, π(i)) está libre (es ‘.’).
2. Para todo par distinto i, j en R se cumple:  
    a) π(i) ≠ π(j)  
    b) |π(i) − π(j)| ≠ |i − j|

El número de soluciones es el cardinal del conjunto  
 { π ∈ S₈ : (i, π(i)) libre para todo i y |π(i)−π(j)| ≠ |i−j| para todo i≠j }.

**3. Código Realizado y Análisis**

* Enlace(s) Código: <https://vjudge.net/solution/61083810/o9SpIowZdGSPTaJh9HeT>

**4. Casos de Prueba**

**Caso 1**  
 Entrada:  
 8  
 ........  
 ........  
 ........  
 ........  
 ........  
 ........  
 ........  
 ........  
 Salida: 92

Caso 2  
 Entrada:  
 8  
 \*.......  
 ........  
 ........  
 ........  
 ........  
 ........  
 ........  
 ........  
 Salida: 0

Caso 3  
 Entrada:  
 8  
 ........  
 ..*.....  
 .....*..  
 .*......  
 ......*.  
 .....\*..  
 ........  
 ........  
 Salida: 4

Caso 4  
 Entrada:  
 8  
 *.*.*.*.  
 .*.*.*.* *.*.*.*.  
 .*.*.*.* *.*.*.*.  
 .*.*.*.* *.*.*.*.  
 .*.*.*.* Salida esperada: 0

**Justificación**:  
 – El Caso 1 corresponde al problema clásico sin casillas reservadas, con 92 soluciones conocidas.  
 – El Caso 2 bloquea toda una fila fijas en columna 1, imposibilitando cualquier colocación de reinas en cada fila distinta.  
 – El Caso 3 introduce algunas casillas reservadas pero mantiene varias configuraciones válidas, demostrando que el algoritmo detecta y explora las ramas factibles.  
 – El Caso 4 convierte alternadamente cada casilla en reservada, dejando huecos insuficientes para ubicar 8 reinas sin conflicto.

**5. Iteración en Caso de Solución Incorrecta (o explicación Solución Correcta)**

En este caso lo hice a la primera así que no es necesario responder a esto e iré directamente a la estrategia.

* Descripción del error:
* Proceso de depuración:
* Solución corregida:
* **Estrategia (Si la solución fue correcta en el primer intento):**

La estrategia empleada es la de **backtracking con poda**, aplicada de la siguiente manera:

1. **Asignación por filas** Se recorre el tablero fila por fila (de la 0 a la 7), intentando colocar exactamente una reina en cada una.
2. **Elección de columna válida** Para cada fila, se prueba cada columna de la 0 a la 7. Antes de colocar la reina, se comprueba si la casilla está libre (‘.’ y no ‘\*’) y si la posición no entra en conflicto con ninguna de las reinas ya situadas en filas anteriores:

* No comparte columna con otra reina.
* No está en la misma diagonal (se usa la condición |fila – fila\_previa| ≠ |columna – columna\_previa|).

1. **Poda inmediata** Siempre que se detecta un conflicto (columna ocupada, diagonal en ataque o casilla reservada), se descarta esa rama y se retrocede (“backtrack”) sin seguir explorando columnas siguientes en esa fila.
2. **Recursión y conteo** Si se logra colocar una reina en la fila actual sin conflictos, se avanza recursivamente a la siguiente fila. Cuando se colocan reinas en las 8 filas (caso base de la recursión), se incrementa el contador de soluciones.
3. **Restauración del estado** Al volver de la llamada recursiva (tras explorar todas las configuraciones derivadas de una colocación), se “deshace” la última asignación (se vuelve a marcar la fila como sin reina) y se sigue probando la siguiente columna disponible.

Gracias a esta búsqueda exhaustiva con poda en cada paso de validación, el algoritmo recorre todas las permutaciones posibles de columnas (en el peor caso 8! = 40 320) pero evita generar subárboles de búsqueda inmediatamente cuando detecta un conflicto, reduciendo de forma significativa el número de configuraciones exploradas realmente.

**6. Preguntas de Aprendizaje**

* **Temas aplicados**:
* backtracking con poda: (lo aprendi muy bien gracias a tantos intentos del ejercicio i)
* Dificultad de la implementación: Fácil tirando a media, diría que lo backtracking pero ya sabía implementarlo por ejercicio anterior.
* Recursos utilizados: Ninguno

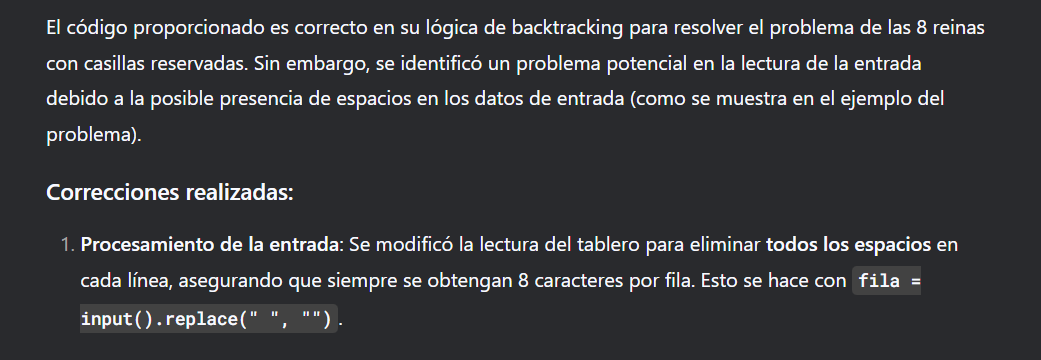
**7. Feedback LLM**

* Envío código LLM: <https://vjudge.net/solution/61083866/alCnCvQKSR51E0VOCVXa>
* Comparación de su códigos:

La versión del LLM sacrifica algo de simplicidad en favor de detección de conflictos en tiempo constante mediante set, reduciendo el coste de cada poda de O(fila) a O(1).

La mía es más directa y fácil de seguir para quien no esté familiarizado con la técnica de transformar diagonales en índices de conjuntos, aunque cada validación recorre todas las filas anteriores. Ambas alcanzan un rendimiento más que suficiente para un tablero de 8×8 así que por eso mi código es mejor.

* Feedback de LLM: Pida al LLM que evalúe y corrija su código:



Pues sí estoy de acuerdo pero modificar la lectura es innecesario para este este ejercicio