

Instituto Politécnico Nacional
La Técnica al Servicio de la Patria

Unidad Profesional Interdisciplinaria de
Ingeniería Campus Tlaxcala UPIIT

Algoritmos y Estructuras de Datos

Esaú Eliezer Escobar Juárez

Backtracking o
vuelta atras.

Backtracking

- "En la antigüedad, antes de que se inventaran las computadoras, los alquimistas estudiaban las propiedades místicas de los números. Al carecer de computadoras, tenían que depender de los dragones para hacer su trabajo por ellos. Los dragones eran bestias inteligentes, pero también perezosas y de mal genio. Lo peor algunos a veces quemaban a su guardián con un solo eructo de fuego. Pero la mayoría de los dragones simplemente no cooperaban, ya que la violencia requería demasiada energía. Esta es la historia de cómo Martin, un aprendiz de alquimista, descubrió la recursividad burlándose de un dragón perezoso".
- - David S. Touretzky, *Common Lisp: A Gentle Introduction to Symbolic Computation*

Backtracking o vuelta atrás

- Suponga que tiene que tomar una serie de *decisiones*, entre varias *opciones*, donde
 - No tienes suficiente información para saber qué elegir
 - Cada decisión conduce a un nuevo conjunto de opciones.
 - Alguna secuencia de opciones (posiblemente más de una) puede ser una solución a su problema
- **Retroceder** es una forma metódica de probar varias secuencias de decisiones, hasta encontrar una que "funcione".

Resolver un laberinto

- Dado un laberinto, encuentra un camino de principio a fin
- En cada intersección, debe decidir entre tres o menos opciones:
 - Ir directamente
 - Ve a la izquierda
 - Ve a la derecha
- No tienes suficiente información para elegir correctamente
- Cada elección conduce a otro conjunto de opciones.
- Una o más secuencias de elecciones pueden (o no) llevar a una solución.
- Muchos tipos de problemas de laberintos se pueden resolver retrocediendo

Un ejemplo más concreto

- Sudoku
- Matriz de 9 por 9 con algunos números rellenos
- Todos los números deben estar entre 1 y 9
- Objetivo: cada fila, cada columna y cada mini matriz deben contener los números entre 1 y 9 una vez cada uno
 - sin duplicados en filas, columnas o mini matrices

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | | | 7 | | | | |
| 6 | | | 1 | 9 | 5 | | | |
| | 9 | 8 | | | | | 6 | |
| 8 | | | | 6 | | | | 3 |
| 4 | | | 8 | | 3 | | | 1 |
| 7 | | | | 2 | | | | 6 |
| | 6 | | | | | 2 | 8 | |
| | | | 4 | 1 | 9 | | | 5 |
| | | | | 8 | | | 7 | 9 |

Resolver un Sudoku – Fuerza Bruta

- Un algoritmo de fuerza bruta es un enfoque simple pero general
- Pruebe todas las combinaciones hasta que encuentre una que funcione
- Este enfoque no es inteligente, pero las computadoras son rápidas
- Luego intenta mejorar los resultados de fuerza bruta.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | | | 7 | | | | |
| 6 | | | 1 | 9 | 5 | | | |
| | 9 | 8 | | | | | 6 | |
| 8 | | | | 6 | | | | 3 |
| 4 | | | 8 | | 3 | | | 1 |
| 7 | | | | 2 | | | | 6 |
| | 6 | | | | | 2 | 8 | |
| | | | 4 | 1 | 9 | | | 5 |
| | | | | 8 | | | 7 | 9 |

Resolviendo el Sudoku

- Solución Sudoku de fuerza bruta
 - Si no hay celdas abiertas, entonces resuelto.
 - Escanear celdas de izquierda a derecha, de arriba a abajo para la primera celda abierta
 - Cuando se encuentra una celda abierta, comience a recorrer los dígitos del 1 al 9.
 - Cuando se coloca un dígito, verifique que la configuración sea legal
 - Ahora resuelve el tablero

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | 1 | | 7 | | | | |
| 6 | | | 1 | 9 | 5 | | | |
| | 9 | 8 | | | | | 6 | |
| 8 | | | | 6 | | | | 3 |
| 4 | | | 8 | | 3 | | | 1 |
| 7 | | | | 2 | | | | 6 |
| | 6 | | | | | 2 | 8 | |
| | | | 4 | 1 | 9 | | | 5 |
| | | | | 8 | | | 7 | 9 |

Pregunta

- Después de colocar un número en una celda, ¿el problema restante es muy similar al problema original?

A. Sí

B. No

Resolver Sudoku – Pasos Sig.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | 1 | | 7 | | | | |
| 6 | | | 1 | 9 | 5 | | | |
| | 9 | 8 | | | | | 6 | |
| 8 | | | | 6 | | | | 3 |
| 4 | | | 8 | | 3 | | | 1 |
| 7 | | | | 2 | | | | 6 |
| | 6 | | | | | 2 | 8 | |
| | | | 4 | 1 | 9 | | | 5 |
| | | | | 8 | | | 7 | 9 |



| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | 1 | 2 | 7 | | | | |
| 6 | | | 1 | 9 | 5 | | | |
| | 9 | 8 | | | | | 6 | |
| 8 | | | | 6 | | | | 3 |
| 4 | | | 8 | | 3 | | | 1 |
| 7 | | | | 2 | | | | 6 |
| | 6 | | | | | 2 | 8 | |
| | | | 4 | 1 | 9 | | | 5 |
| | | | | 8 | | | 7 | 9 |



| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | 1 | 2 | 7 | 4 | | | |
| 6 | | | 1 | 9 | 5 | | | |
| | 9 | 8 | | | | | 6 | |
| 8 | | | | 6 | | | | 3 |
| 4 | | | 8 | | 3 | | | 1 |
| 7 | | | | 2 | | | | 6 |
| | 6 | | | | | 2 | 8 | |
| | | | 4 | 1 | 9 | | | 5 |
| | | | | 8 | | | 7 | 9 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | 1 | 2 | 7 | 4 | 8 | | |
| 6 | | | 1 | 9 | 5 | | | |
| | 9 | 8 | | | | | 6 | |
| 8 | | | | 6 | | | | 3 |
| 4 | | | 8 | | 3 | | | 1 |
| 7 | | | | 2 | | | | 6 |
| | 6 | | | | | 2 | 8 | |
| | | | 4 | 1 | 9 | | | 5 |
| | | | | 8 | | | 7 | 9 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | 1 | 2 | 7 | 4 | 8 | 9 | |
| 6 | | | 1 | 9 | 5 | | | |
| | 9 | 8 | | | | | 6 | |
| 8 | | | | 6 | | | | 3 |
| 4 | | | 8 | | 3 | | | 1 |
| 7 | | | | 2 | | | | 6 |
| | 6 | | | | | 2 | 8 | |
| | | | 4 | 1 | 9 | | | 5 |
| | | | | 8 | | | 7 | 9 |

uh oh!

Sudoku – Un camino cerrado

- Hemos llegado a un callejón sin salida en nuestra búsqueda

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | 1 | 2 | 7 | 4 | 8 | 9 | |
| 6 | | | 1 | 9 | 5 | | | |
| | 9 | 8 | | | | | 6 | |
| 8 | | | | 6 | | | | 3 |
| 4 | | | 8 | | 3 | | | 1 |
| 7 | | | | 2 | | | | 6 |
| | 6 | | | | | 2 | 8 | |
| | | | 4 | 1 | 9 | | | 5 |
| | | | | 8 | | | 7 | 9 |

- Con la configuración actual, ninguno de los nueve dígitos funciona en la esquina superior derecha

Retrocediendo

- Cuando la búsqueda llega a un callejón sin salida, **retrocede** a la celda anterior que estaba tratando de llenar y pasa al siguiente dígito
- Regresaríamos a la celda con un 9 y eso también resulta ser un callejón sin salida, así que retrocedemos de nuevo.
 - por lo que el algoritmo debe recordar qué dígito probar a continuación
- Ahora en la celda con el 8. Intentamos el 9 y avanzamos nuevamente.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | 1 | 2 | 7 | 4 | 8 | 9 | |
| 6 | | | 1 | 9 | 5 | | | |
| | 9 | 8 | | | | | 6 | |
| 8 | | | | 6 | | | | 3 |
| 4 | | | 8 | | 3 | | | 1 |
| 7 | | | | 2 | | | | 6 |
| | 6 | | | | | 2 | 8 | |
| | | | 4 | 1 | 9 | | | 5 |
| | | | | 8 | | | 7 | 9 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | 1 | 2 | 7 | 4 | 9 | | |
| 6 | | | 1 | 9 | 5 | | | |
| | 9 | 8 | | | | | 6 | |
| 8 | | | | 6 | | | | 3 |
| 4 | | | 8 | | 3 | | | 1 |
| 7 | | | | 2 | | | | 6 |
| | 6 | | | | | 2 | 8 | |
| | | | 4 | 1 | 9 | | | 5 |
| | | | | 8 | | | 7 | 9 |

Características de la fuerza bruta y el Backtracking

- Los algoritmos de fuerza bruta son lentos
- No emplean mucha lógica
 - Por ejemplo, sabemos que un 6 no puede ir en las últimas 3 columnas de la primera fila, pero el algoritmo de fuerza bruta seguirá adelante de cualquier manera.
- Pero, los algoritmos de fuerza bruta son bastante fáciles de implementar como solución de primer paso.
 - El retroceso es una forma de algoritmo de fuerza bruta

Ideas clave

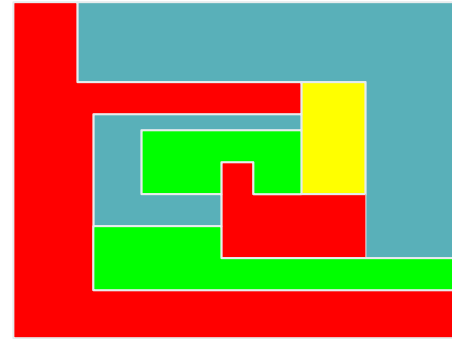
- Después de intentar colocar un dígito en una celda queremos resolver el nuevo tablero de sudoku
 - ¿No es esa una versión más pequeña (o más simple) del mismo problema con el que comenzamos?!?!?!?
- Después de colocar un número en una celda, debemos recordar el siguiente número para intentarlo en caso de que las cosas no funcionen.
- Necesitamos saber si las cosas funcionaron (encontraron una solución) o no, y si no probaron el siguiente número.
- Si probamos todos los números y ninguno de ellos funciona en nuestra celda, debemos informar que las cosas no funcionaron.

Retroceso (Backtracking) Recursivo

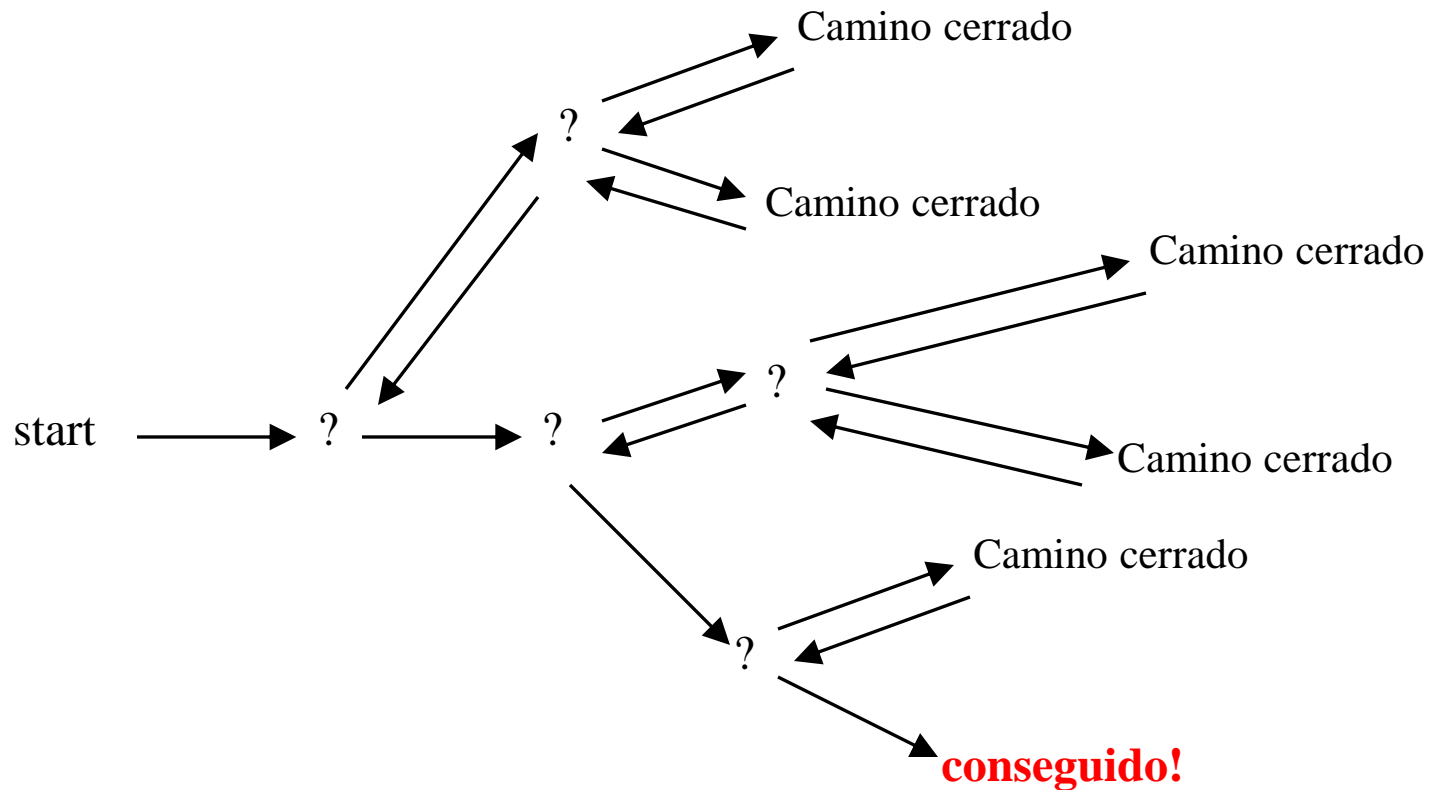
- Los problemas como Sudoku se pueden resolver mediante el retroceso recursivo
- Recursivo porque las versiones posteriores del problema son solo versiones ligeramente más simples del original
- Retroceder porque es posible que tengamos que probar diferentes alternativas

Colorear un mapa

- Deseas colorear un mapa con no más de cuatro colores.
 - Rojo, amarillo, verde y azul
- Los países adyacentes deben estar en diferentes colores.
- No tienes suficiente información para elegir colores.
- Cada elección conduce a otro conjunto de opciones.
- Una o más secuencias de elecciones pueden (o no) llevar a una solución.
- Muchos problemas de coloración se pueden resolver retrocediendo

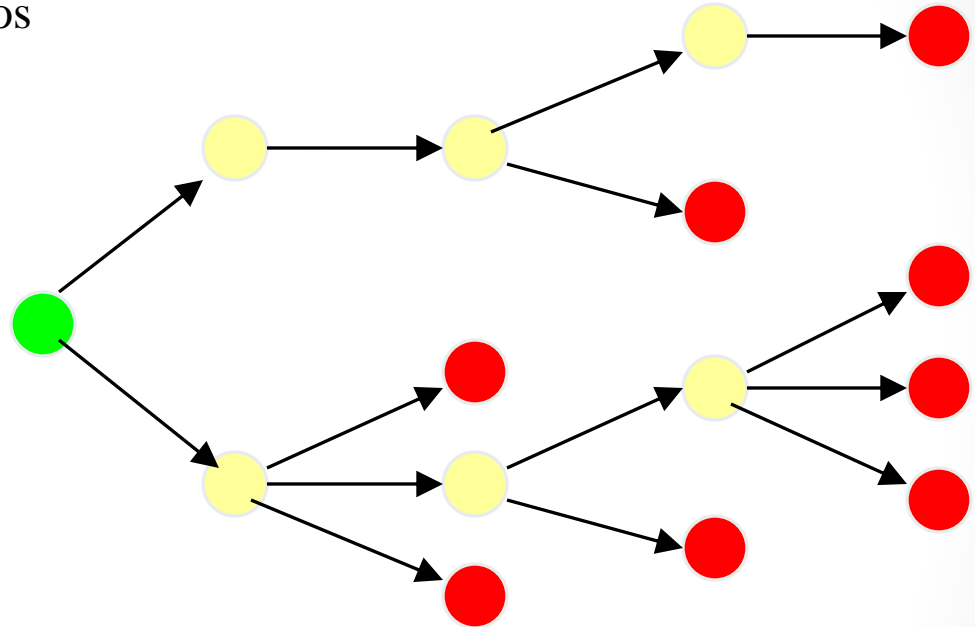


Backtracking (animación)






Terminología I

Un árbol se compone de nodos



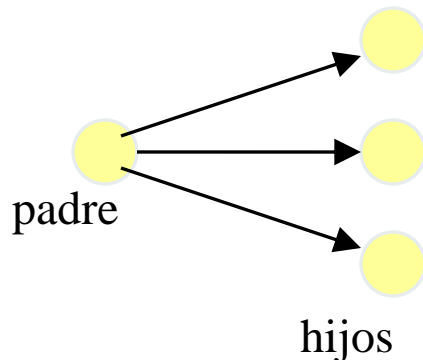
Hay tres tipos de nodos:

-  El (único) nodo raíz
-  Nodos internos
-  Nodos hoja

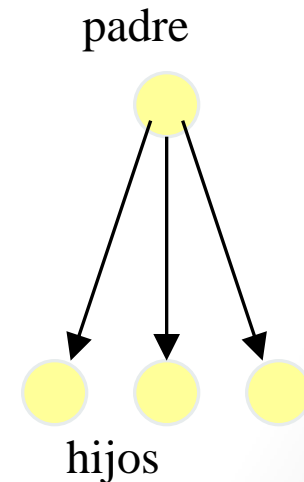
Backtracking se puede considerar como buscar en un árbol un nodo hoja "objetivo" en particular

Terminología II

- Cada nodo que no es hoja en un árbol es un **padre** de uno o más nodos (sus **hijos**)
- Cada nodo en el árbol, aparte de la raíz, tiene exactamente un **padre**.



Por lo general, sin embargo, dibujamos nuestros árboles *hacia abajo*, con la raíz en la parte superior.



Arboles reales y virtuales

- Existe un tipo de estructura de datos llamada árbol.
 - Pero **no** lo estamos usando aquí
- Si hacemos un diagrama de la secuencia de elecciones que hacemos, el diagrama se ve como un árbol
 - De hecho, lo hicimos hace un par de diapositivas.
 - Nuestro algoritmo de retroceso "barre un árbol" en el "espacio de problemas"

El algoritmo backtracking

- Backtracking es realmente simple -- “explora” cada nodo, como sigue:
- Para “explorar” el nodo N:
 1. Si N es un nodo objetivo, regresar “éxito”
 2. Si N es un nodo hoja, regresar “fallo”
 3. Para cada hijo C de N,
 - 3.1. Explorar C
 - 3.1.1. Si C fue exitoso, regresar “éxito”
 4. Regresar “fallo”

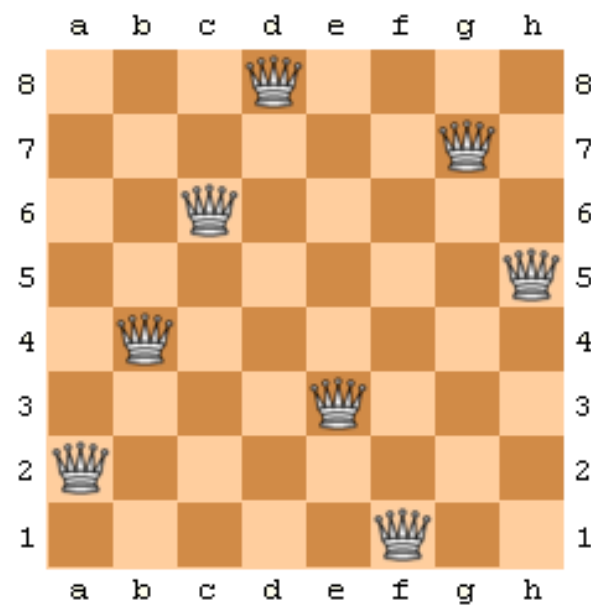
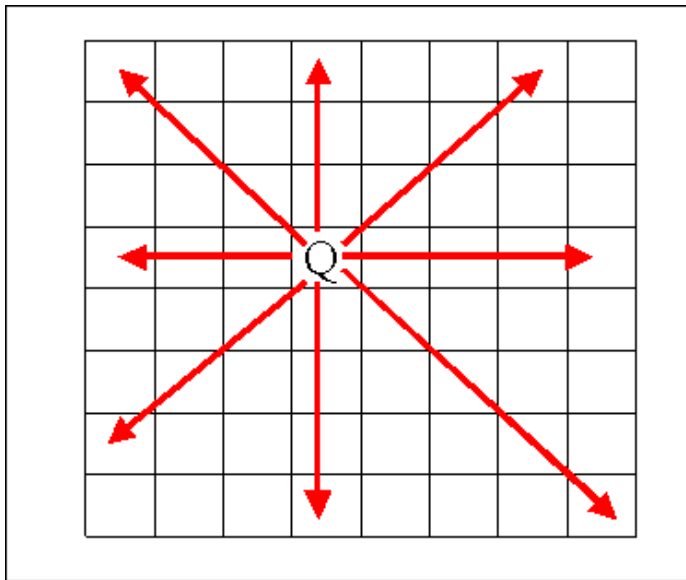


El problema de las 8 reinas



Problema de las 8 reinas

- Un rompecabezas clasico de ajedrez
 - Coloca 8 piezas de reina en un tablero de ajedrez para que ninguna de ellas pueda atacarse entre sí.



El problema de N Reinas

- Coloque N Reinas en un tablero de ajedrez N por N para que ninguna de ellas pueda atacarse entre sí
- ¿Número de posibles ubicaciones?
- En 8 x 8
 - $64 \cdot 63 \cdot 62 \cdot 61 \cdot 60 \cdot 59 \cdot 58 \cdot 57 = 178,462,987,637,760/8!$
 - $= 4.426.165.368$
 - ¿De cuántas formas puedes elegir k cosas de un conjunto de n elementos?

$$\binom{n}{k} = \frac{n \cdot (n-1) \cdots (n-k+1)}{k \cdot (k-1) \cdots 1} = \frac{n!}{k!(n-k)!} \quad \text{if } 0 \leq k \leq n \quad (1)$$

- En este caso hay 64 casillas y queremos elegir 8 de ellas para poner reinas

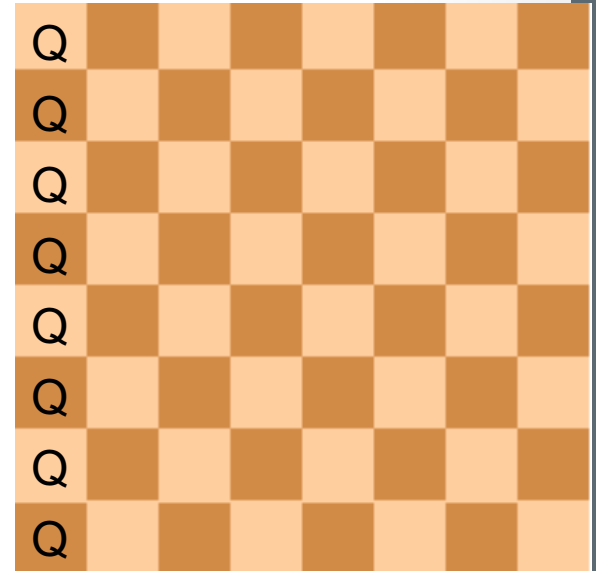
Pregunta 2

Para obtener soluciones válidas, ¿cuántas reinas se pueden colocar en una columna determinada?

- 0
- 1
- 2
- 3
- 4
- Cualquier número

Reducir el espacio de búsqueda

- El cálculo anterior incluye configuraciones como esta
- Incluye muchas configuraciones con varias reinas en la misma columna
- ¿Cuántas reinas puede haber en una columna?
- Número de configuraciones $8 * 8 * 8 * 8 * 8 * 8 * 8 * 8 = 16.777.216$
- Hemos reducido el espacio de búsqueda en dos órdenes de magnitud aplicando algo de lógica



Una solución a 8 reinas

- Si el número de reinas es fijo y me doy cuenta de que no puede haber más de una reina por columna, puedo recorrer las filas de cada columna.

```
for(int c0 = 0; c0 < 8; c0++){
    board[c0][0] = 'q';
    for(int c1 = 0; c1 < 8; c1++){
        board[c1][1] = 'q';
        for(int c2 = 0; c2 < 8; c2++){
            board[c2][2] = 'q';
            // a little later
            for(int c7 = 0; c7 < 8; c7++){
                board[c7][7] = 'q';
                if( queensAreSafe(board) )
                    printSolution(board);
                board[c7][7] = ' '; //pick up queen
            }
            board[c6][6] = ' '; // pick up queen
```

N Reinas

- El *problema* con N reinas es que no sabes cuántos bucles for escribir.
- Haz el problema de forma recursiva
- Escribe código recursivo con clase y demostración.
 - mostrar retroceso con punto de interrupción y opción de depuración

Recursive Backtracking

- ¡¡¡Debes practicar !!!
- Aprenda a reconocer los problemas que se ajustan al patrón
- ¿Todas las soluciones o una solución?
- Informar resultados y actuar sobre los resultados