Backtracking

Bach. Rodolfo Mercado Gonzales
Universidad Nacional de Ingeniería

Backtracking

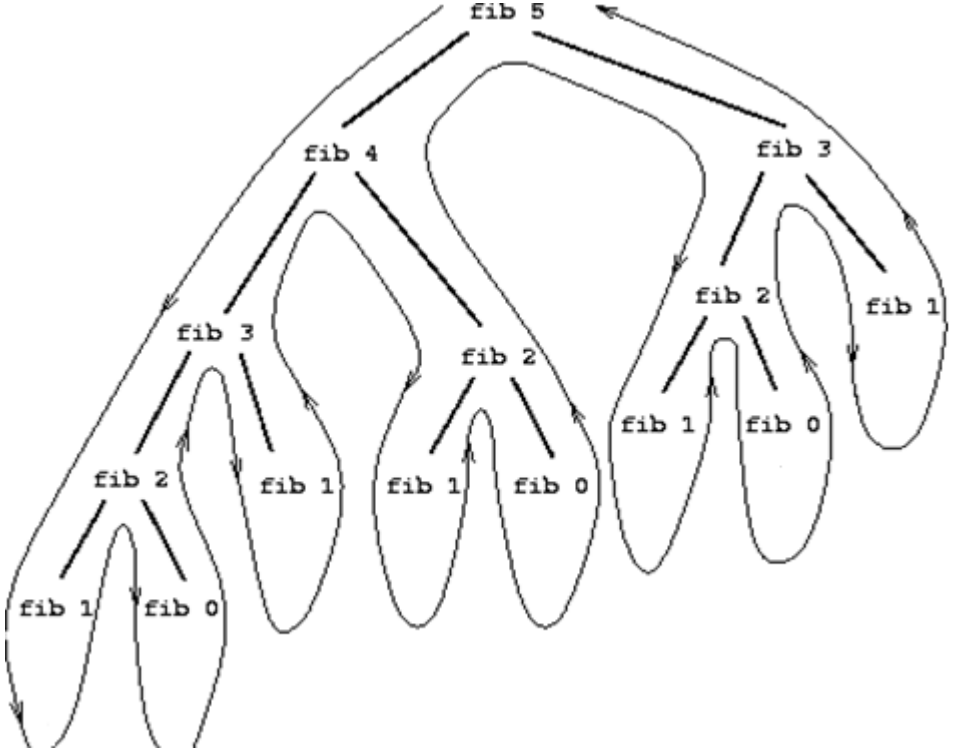
- Técnica de fuerza bruta basada en la recursividad.
- Permite iterar por todas las posibles configuraciones de un espacio de búsqueda.
- Se usa cuando es necesario obtener una configuración que cumpla ciertas reglas y los constraints dados son pequeños.

Objetivos comunes del backtracking

- Encontrar 1 configuración válida (o determinar si no existe ninguna)
- Encontrar todas las configuraciones válidas.
- Contar cuántas configuraciones válidas hay
- Encontrar la mejor de las soluciones válidas (bajo determinado criterio)

Backtracking

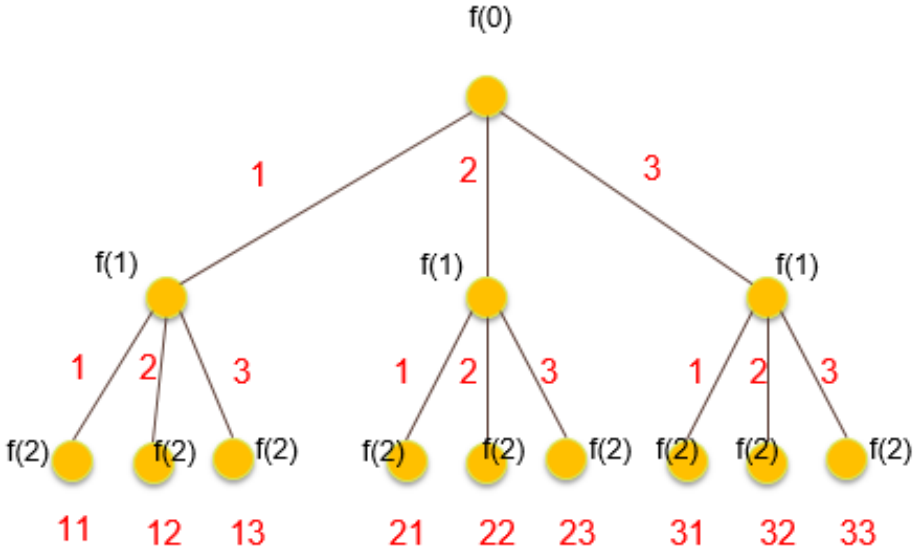
- Aprovecha el recorrido en profundidad que tienen las funciones recursivas
- Todo las aristas pertenecientes al camino desde la raíz hasta una hoja forman una configuración.



Configuraciones

Generar todas las configuraciones posibles de n elementos usando los números del 1 al m.

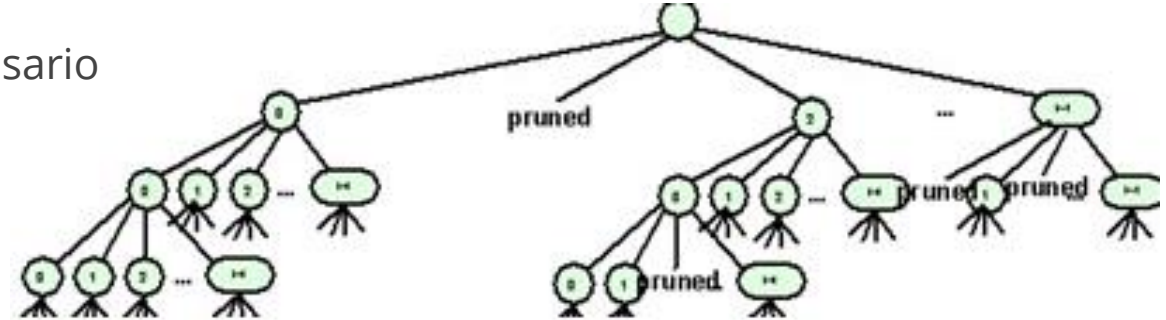
Tenemos que completar un arreglo de n elementos, donde cada elemento puede ser un número del 1 al m.



$n = 2 \ y \ m = 3$

Pruning

- Es una especie de “validación temprana” que se realiza para optimizar el backtracking.
- Esta validación nos permite saber si una configuración está siendo formada correctamente en lugar de esperar a que se haya completado toda la configuración y recién ahí validar.
- Con el pruning ya no es necesario explorar todas las alternativas.



Estructura general del backtracking

1. Testear si la configuración ya ha sido encontrada
2. Sino, para cada opción válida en la posición actual (pruning)
   2.1. Escoger la opción
   2.2. Marcar arreglo de usados en caso de ser necesario
   2.3. Recursivamente completar las siguientes posiciones.
   2.4. Desmarcar arreglo de usados.

Estructura general del backtracking

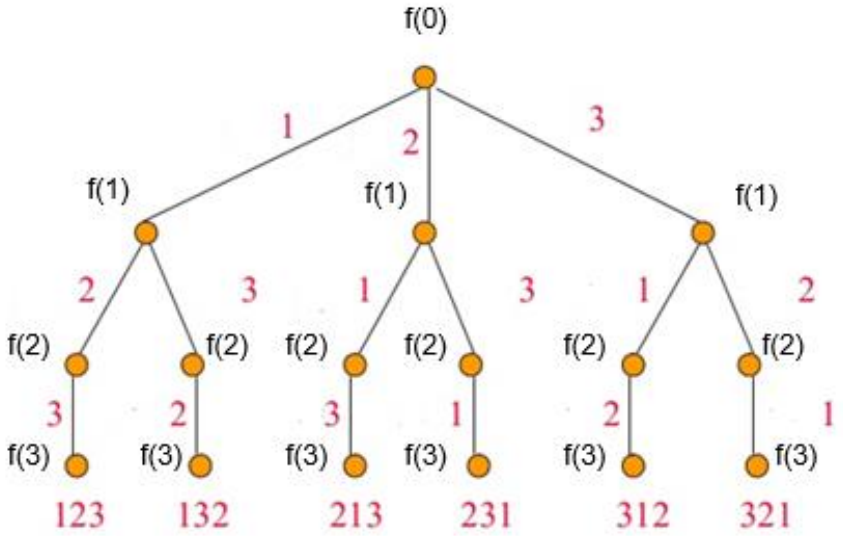
Pseudocódigo general (para determinar si existe una solución) :

```
bool solve(Configuracion conf) {
    if(no hay mas opciones) { //caso base
        if(conf es valido) {
            return true;
        } else {
            return false;
        }
    }

    for(todas las opciones posibles) {
        Probar la opcion c;
        //resolver el problema recursivamente luego de elegir "c"
        bool ok = solve(conf con la opcion c);

        if(ok) {
            return true;
        } else {
            deshacer la opcion c;
        }
    }

    return false; //ya probaste todas las opciones y no encontraste solucion
}
```

Permutaciones

Generar todas las permutaciones de los números del 1 al n.

Tenemos que completar un arreglo de n elementos, donde cada elemento puede ser un número del 1 al n y todos deben ser distintos.

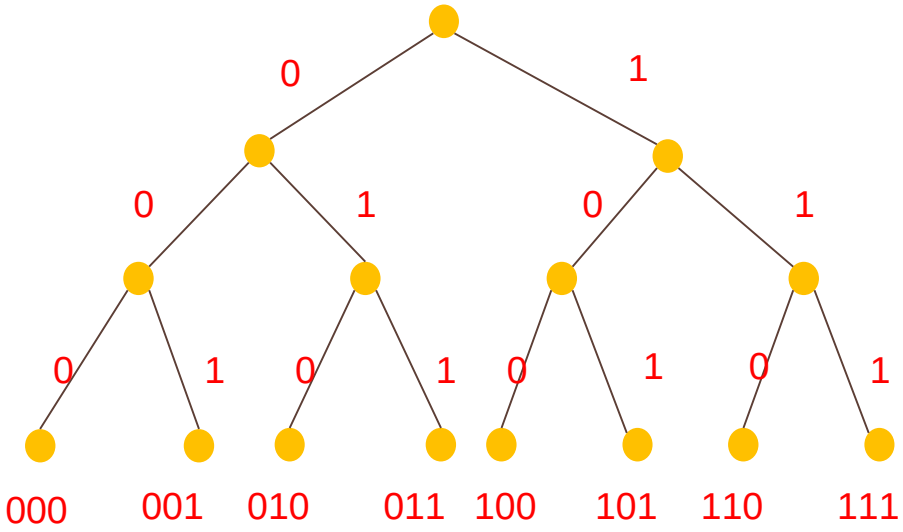


$n = 3$

Subconjuntos

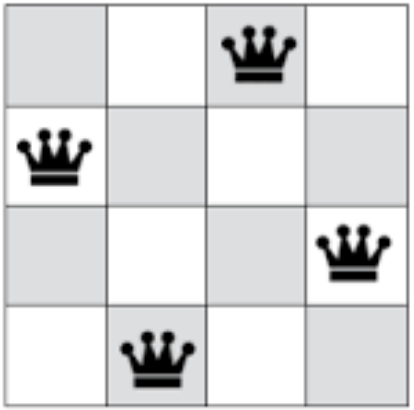
Generar todos los subconjunto de un arreglo de n elementos.

Tenemos que completar un arreglo de n elementos, donde cada elemento puede ser un 0 (no está en el subconjunto) o 1 (está en el subconjunto)

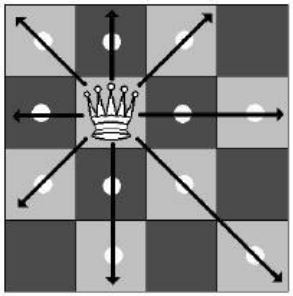


N Reinas

Dado un tablero de N x N se desea colocar N reinas de tal forma que ninguna de ellas pueda atacar a las otras.



Una reina puede moverse a lo largo de una fila, columna o diagonal de la posición en que se encuentra.



N Reinas

N = 3

Se debe validar que la fila, columna y diagonal en que se colocará cada reina sean posiciones válidas.

En las diagonales podemos encontrar propiedades con sus coordenadas.

D1	0	1	2
0	0	-1	-2
1	1	0	-1
2	2	1	0

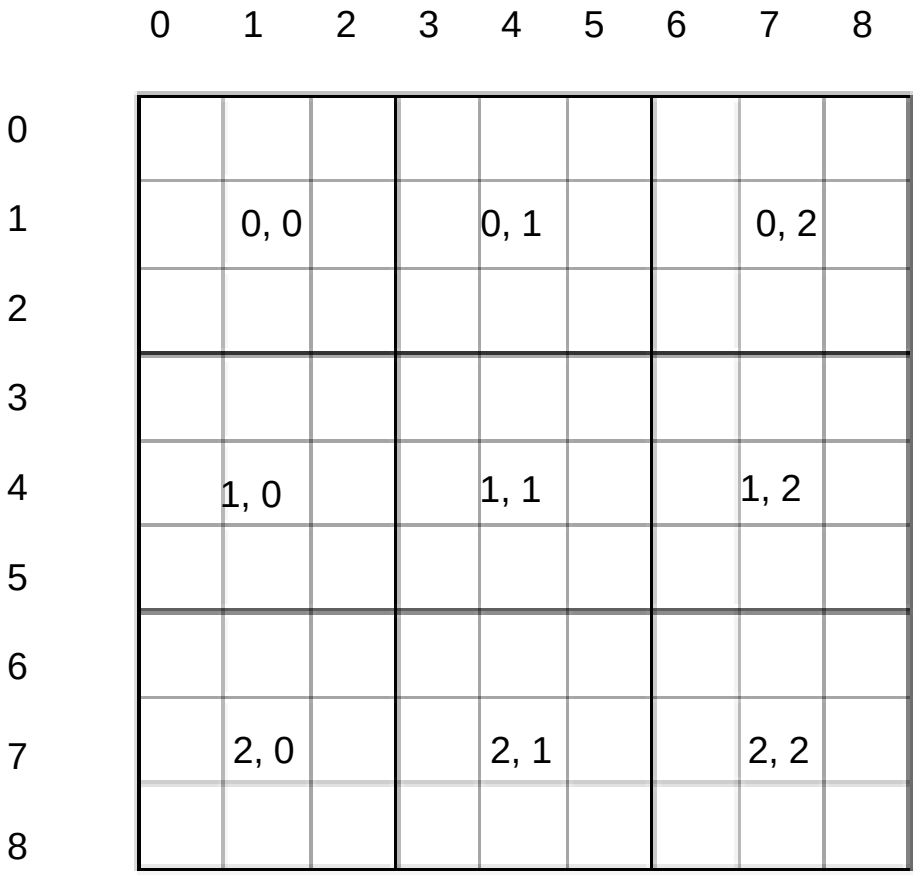
D2	0	1	2
0	0	1	2
1	1	2	3
2	2	3	4

Sudoku

- Juego matemático que se presenta como un tablero de 9 x 9, compuesto por subtableros de 3 x 3 denominados regiones.
- El tablero presenta algunas celdas vacías y otras con números del 1 al 9.
- El objetivo es llenar con números las celdas vacías, tal que los números del 1 al 9 aparezcan exactamente una vez en cada fila, columna y región.

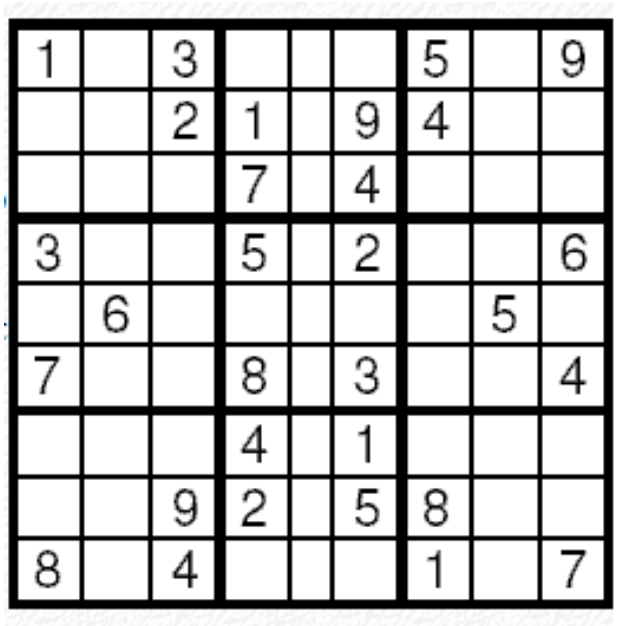
Sudoku

Filas, columnas
y regiones



Sudoku

Completar el siguiente sudoku



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

Problemas

UVA 291 – The House of Santa Claus

Codeforces 244B – Undoubtely Lucky Numbers

Referencias

- Skiena, Algorithm Design
- Stanford : https://see.stanford.edu/materials/icspacs106b/H19-RecBacktrackExamples.pdf

¡ Good luck and have fun !