DEV.F::

Clase 6

Recursión y Backtracking

Comenzamos en 10 min

6:40pm (hora CDMX)

Presentación creada por contribución de Diego Lechuga

Temas de la clase (180 min)

Recursión

Concepto

Caso base

Llamada recursiva

Ejemplo

Backtracking

Concepto

Combinaciones

Retroceso

Ejemplo

Recomendaciones

Cuando usar recursión y backtracking

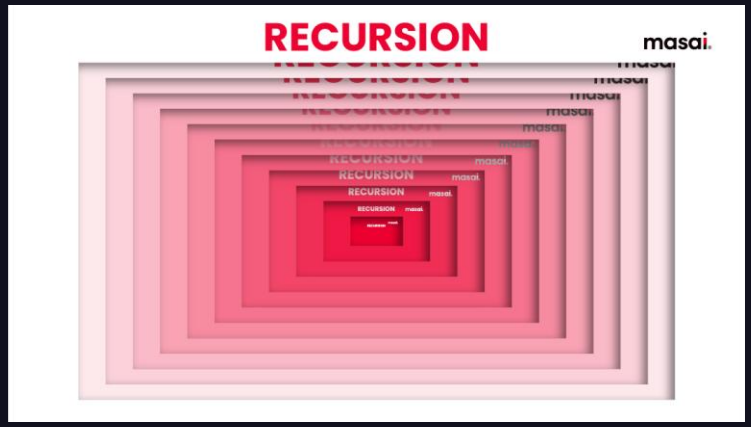
Por qué aprender técnicas de algoritmos

Reto de la semana

DEV.F::

Recursión

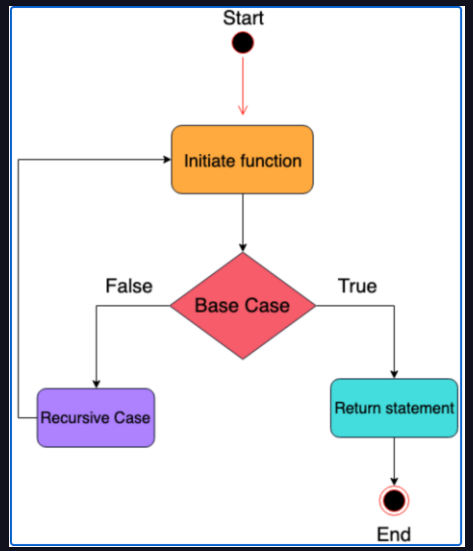
La recursión es un concepto de programación que consiste en que una función se llama a sí misma. Se utiliza para resolver problemas complejos dividiendo la solución en problemas más pequeños.



DEV.F::

Caso base

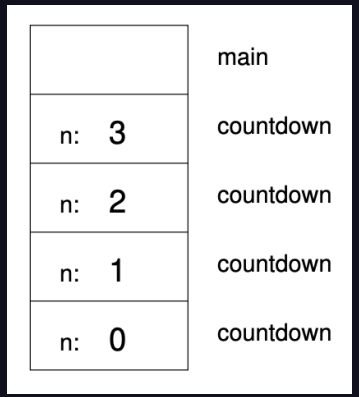
El caso base es un escenario o entrada que se considera el más probable o simple para una función o modelo. Se utiliza para analizar y tomar decisiones en diferentes contextos, como en la recursión, en el análisis de escenarios y en la inteligencia artificial.



DEV.F::

Llamada recursiva

Una llamada recursiva es cuando una función se llama a sí misma dentro de su propia definición para resolver un problema, dividiéndolo en subproblemas más pequeños. Cada llamada recursiva debe acercar la solución al caso base, que es la condición donde la función deja de llamarse a sí misma.



DEV.F::

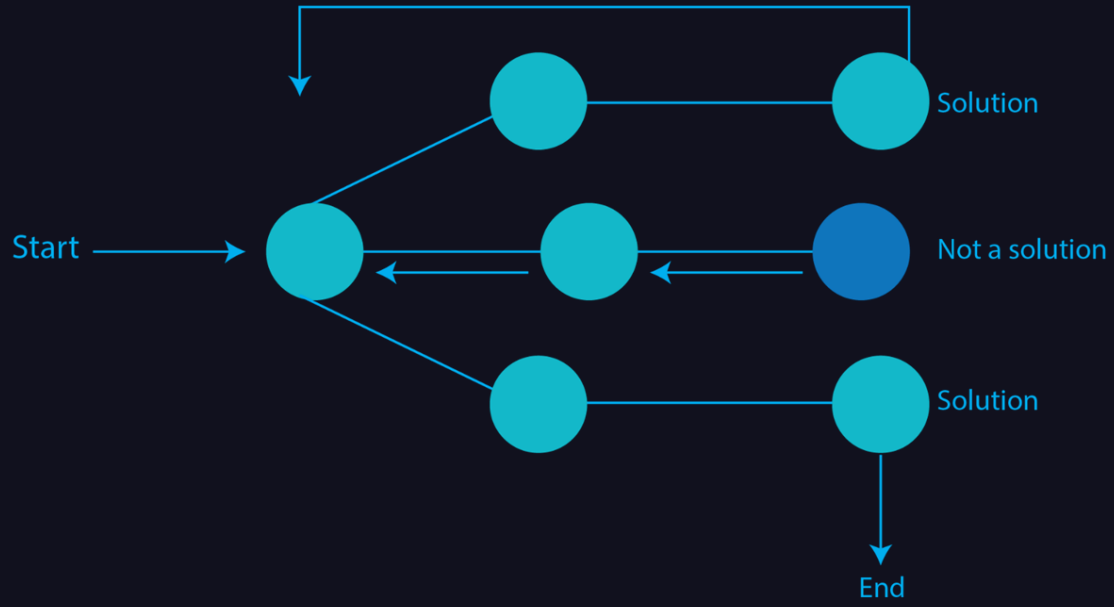
Factorial Recursivo

```
1  function factorial(x) {
2      debugger;
3          if (x === 0) {
4              return 1;
5          }
6          return x * factorial(x - 1);
7      }
8          let total = factorial(3);
9          console.log(total);
10
```

DEV.F::

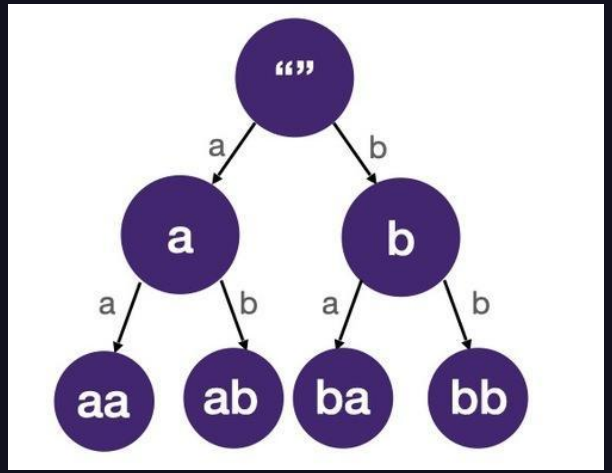
Backtracking

El backtracking es una estrategia para encontrar soluciones a problemas que satisfacen restricciones. Se trata de un método recursivo que explora todas las opciones posibles para encontrar una solución.



Combinatoria

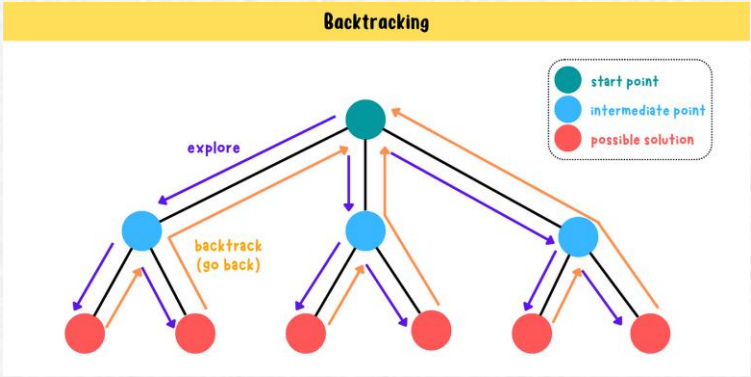
La combinatoria en backtracking es una técnica para generar o encontrar todas las combinaciones posibles de elementos en un conjunto, explorando todas las opciones y retrocediendo (backtracking) cuando una solución parcial no es válida o no cumple con las condiciones necesarias.



Retroceso

El retroceso en **backtracking** es el proceso de deshacer una decisión parcial para regresar a un estado anterior y explorar otras posibilidades. Es fundamental para probar diferentes **combinaciones** o soluciones en problemas de búsqueda, evitando así caminos que no conducen a la solución final.

Imagina que estás resolviendo un laberinto. Quieres encontrar la salida, pero hay muchos caminos posibles. **Backtracking** es una forma de explorar el laberinto probando un camino, y si no funciona, retrocedes para probar otro.



Backtracking con Strings

```
function permute(str, path = "", result = []) {
    if (str.length === 0) {
        result.push(path);
    } else {
        for (let i = 0; i < str.length; i++) {
            const remaining = str.slice(0, i) + str.slice(i + 1);
            console.log(remaining);
            permute(remaining, path + str[i], result);
        }
    }
    return result;
}

const result = permute("abc");
console.log(result);

```

Recursión

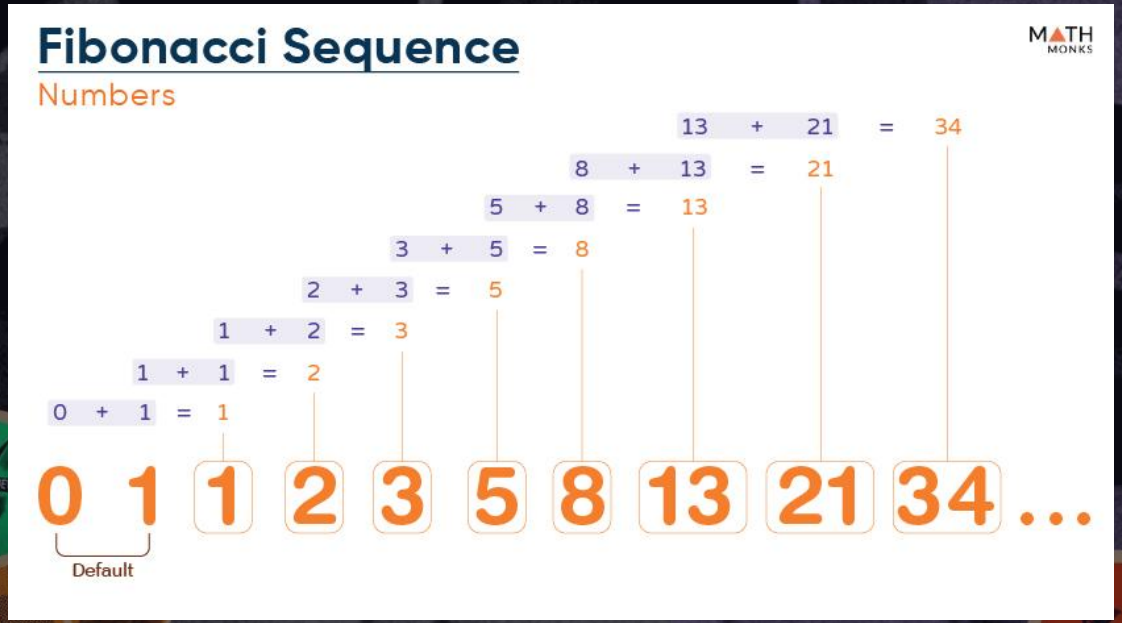
- No siempre necesita backtracking
- Resuelve problemas complejos a partir de dividirlos en problemas más pequeños
- La recursión es parte del propio backtracking y es más simple de escribir
- Problemas comunes que usan recursividad pueden ser: Recorrer Arboles Grafos, Las Torres de Hanoi, Divide y venceras, Merge sort, Quick sort y Busqueda Binaria

Backtracking

- El backtracking siempre utiliza recursividad
- El backtracking siempre eliminará las opciones que no contengan una solución óptima al problema
- Requiere de una implementación mucho más compleja
- Ejemplos comunes de uso de backtracking pueden ser la solución de laberintos o un algoritmo que resuelve sudokus

Reto de la semana: Fibonacci

- Crea una función que imprima las series de fibonacci utilizando recursividad.



No olviden

Las lecturas de edu.devf.la