DISEÑO DE ALGORITMOS

Jonathan García

PASOS DISEÑO ALGORITMOS

Definición del problema

Especificación datos de entrada

Especificación datos de salida

1. Formular el problema detalladamente.
2. Pensar en algoritmos para resolver el problema
3. Diseñar un algoritmo para el problema
4. Analizar el algoritmo
    1. Si es correcto, usarlo
    2. Si no es correcto volver al paso 2

FUERZA BRUTA

Estrategia para:

- Probar todos las posibles combinaciones
- En n grandes es menos eficiente
- Hace lazos anidados

```
c = primero(P)
mientras c != null
    si valido(P,c) entonces mostrar(P,c)
    c = siguiente(P,c)
```

PROBLEMAS

Maximum Subarray – LeetCode

Longest Common Prefix - LeetCode

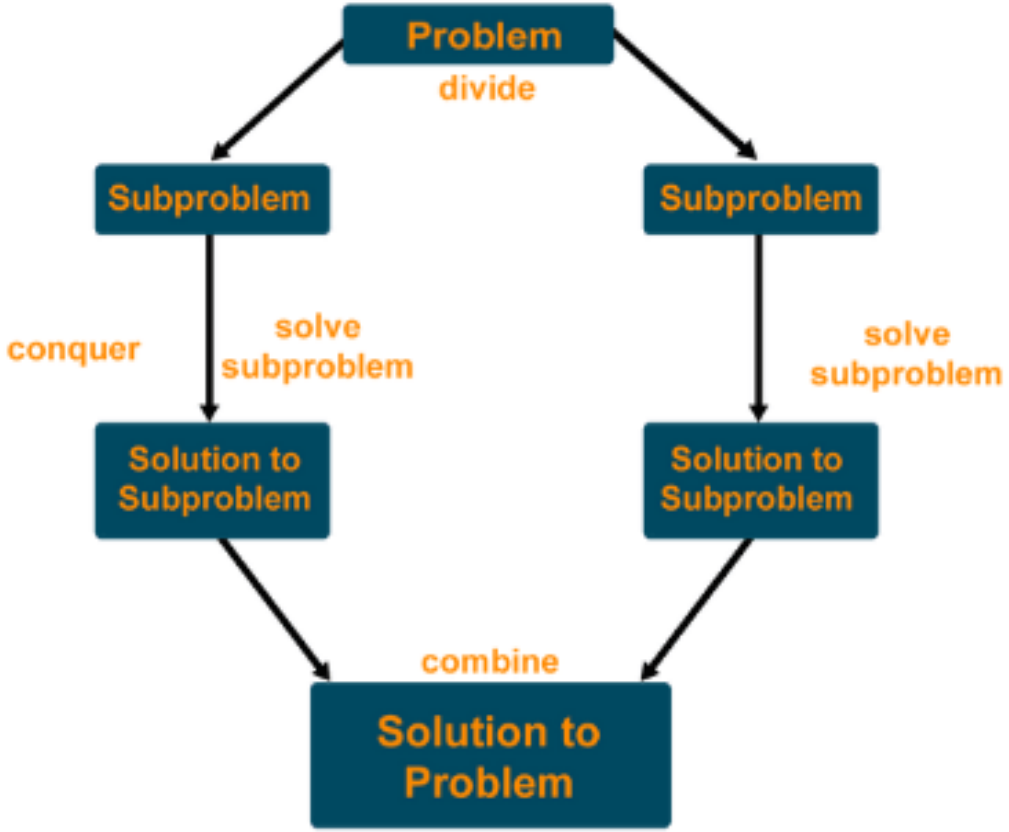
Available Captures for Rook – LeetCode

Longest Substring Without Repeating Characters – LeetCode

DIVIDIR Y CONQUISTAR

Se divide el problema en instancias más pequeñas, se resuelve de forma independiente cada una de las instancias y se combina las soluciones pequeñas para dar una solución al problema original.

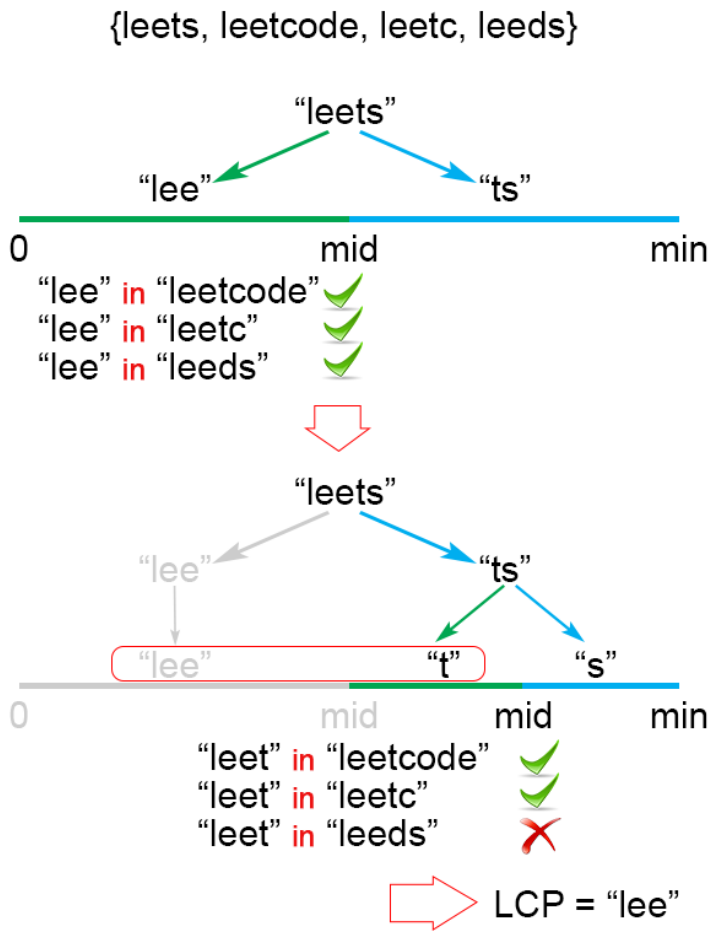
Divide and Conquer Algorithm



PROBLEMAS

Longest Common Prefix – LeetCode

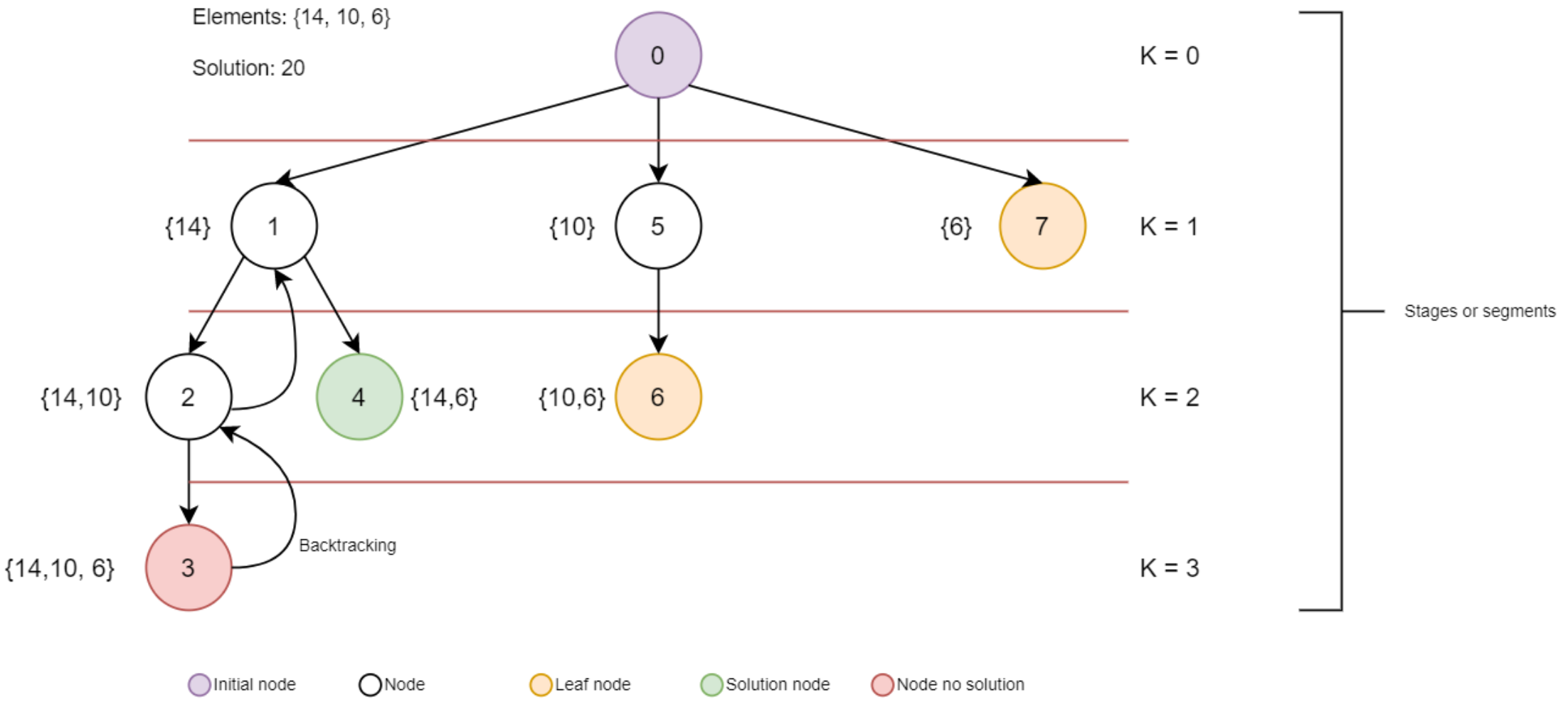
Majority Element – LeetCode



Searching for the longest common prefix (LCP)
in dataset {leets, leetcode, leetc, leeds}

BACKTRACKING

Escoge un camino buscando una solución, si por el camino no se llega a la solución se retrocede hasta hallar un nuevo camino.

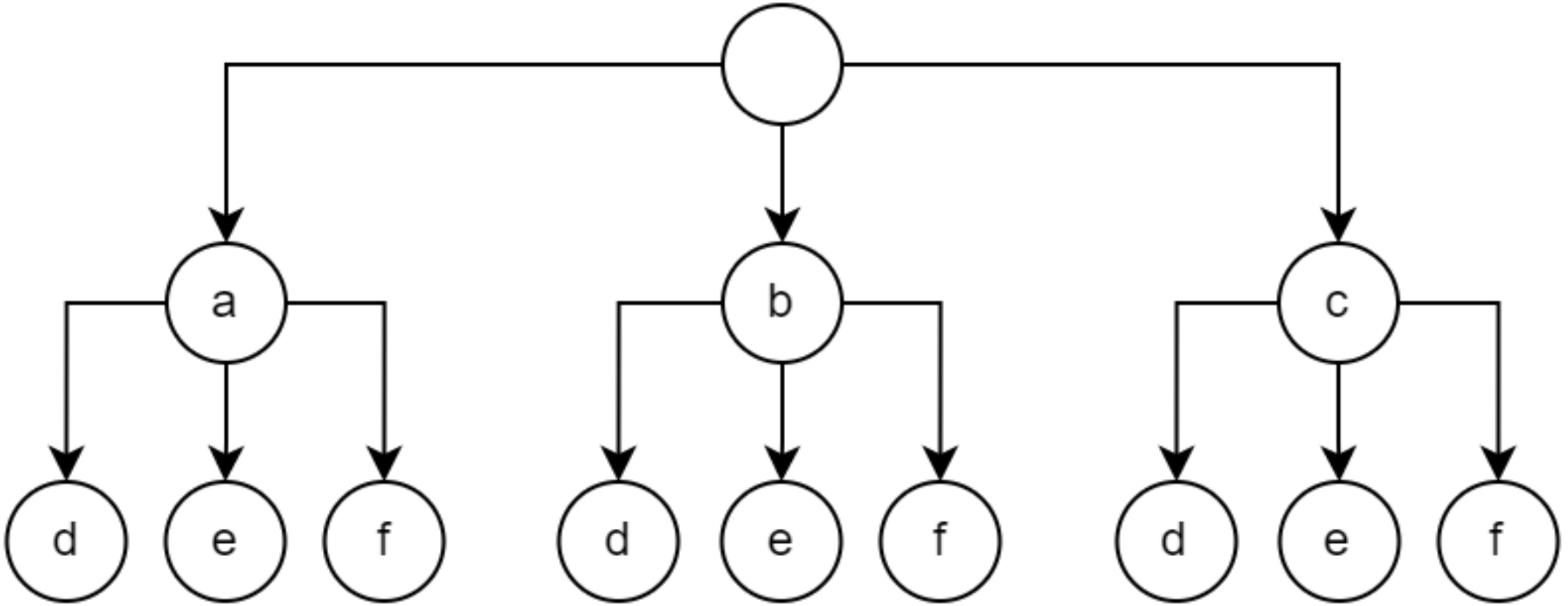

Elements: {14, 10, 6}
Solution: 20
0
K = 0
{14}
1
{10}
5
{6}
7
K = 1
{14,10}
2
4
{14,6}
{10,6}
6
K = 2
{14,10, 6}
3
Backtracking
K = 3
Stages or segments
Initial node
Node
Leaf node
Solution node
Node no solution

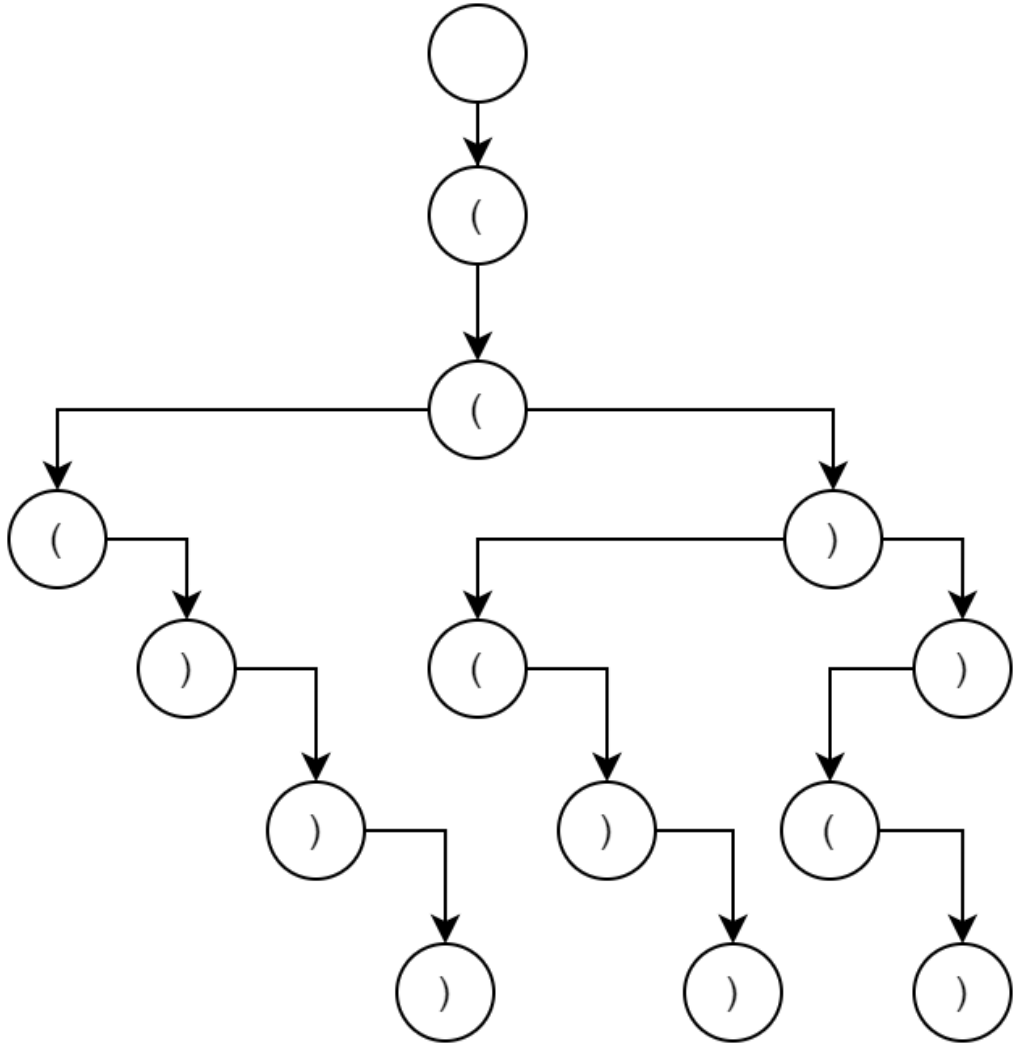
PROBLEMAS

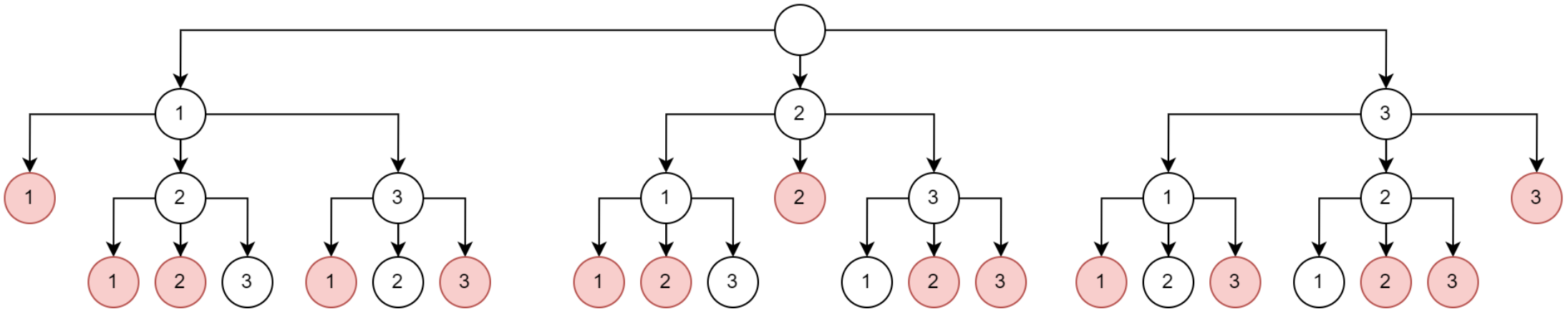
Letter Combinations of a Phone Number – LeetCode

Generate Parentheses – LeetCode

Permutations – LeetCode


a
b
c
d
e
f
d
e
f
d
e
f




1
2
3
1
2
3
1
2
3
1
2
3
1
2
3
1
2
3
1
2
3
1
2
3
1
2
3
1
2
3

PROGRAMACIÓN DINÁMICA

Es un método para reducir el tiempo de ejecución de un algoritmo mediante la utilización de subproblemas superpuestos y subestructuras óptimas, como se describe a continuación.

Memoización (Top-down)

Tabulación (Bottom-up)

```c
int fib(int n)
{
   if (n <= 1)
      return n;
   return fib(n-1) + fib(n-2);
}
```

Recursion : Exponential

```c
f[0] = 0;
f[1] = 1;

for (i = 2; i <= n; i++)
{
    f[i] = f[i-1] + f[i-2];
}

return f[n];
```

Dynamic Programming : Linear

PROBLEMAS

Fibonacci Number – LeetCode

Climbing Stairs – LeetCode

0	1	2	3	4	5	6	7	8	9	10
0	1	1	2	3	5	8	13	21	34	55