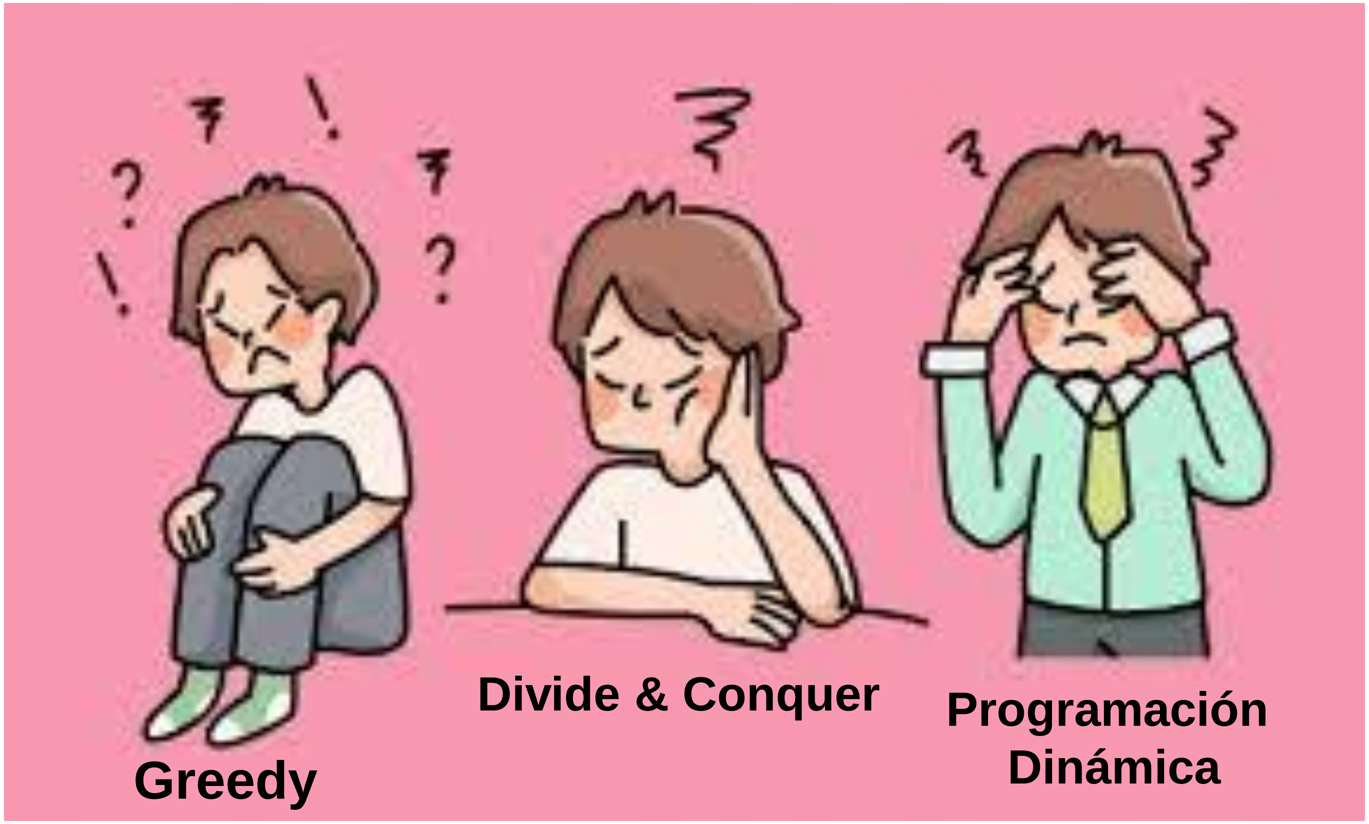
TPN°7: Vuelta atrás
Backtracking

Algoritmos y Estructuras de Datos II

BACKTRACKING



BACKTRACKING → Estudio exhaustivo de posibles soluciones

BACKTRACKING

BACKTRACKING

- **No utilizan estrategia** en la búsqueda de las soluciones

  Proceso de prueba y error en el cual se construye gradualmente una solución

- Similar al **recorrido en profundidad**

ALGORITMO dfs (v)
P1. visitado (v) ← verdadero
P2. ESCRIBIR (v)
P3. PARA cada vértice w adyacente a v HACER
    SI NOT visitado (w) ENTONCES
        dfs (w)
P3. FIN

BACKTRACKING

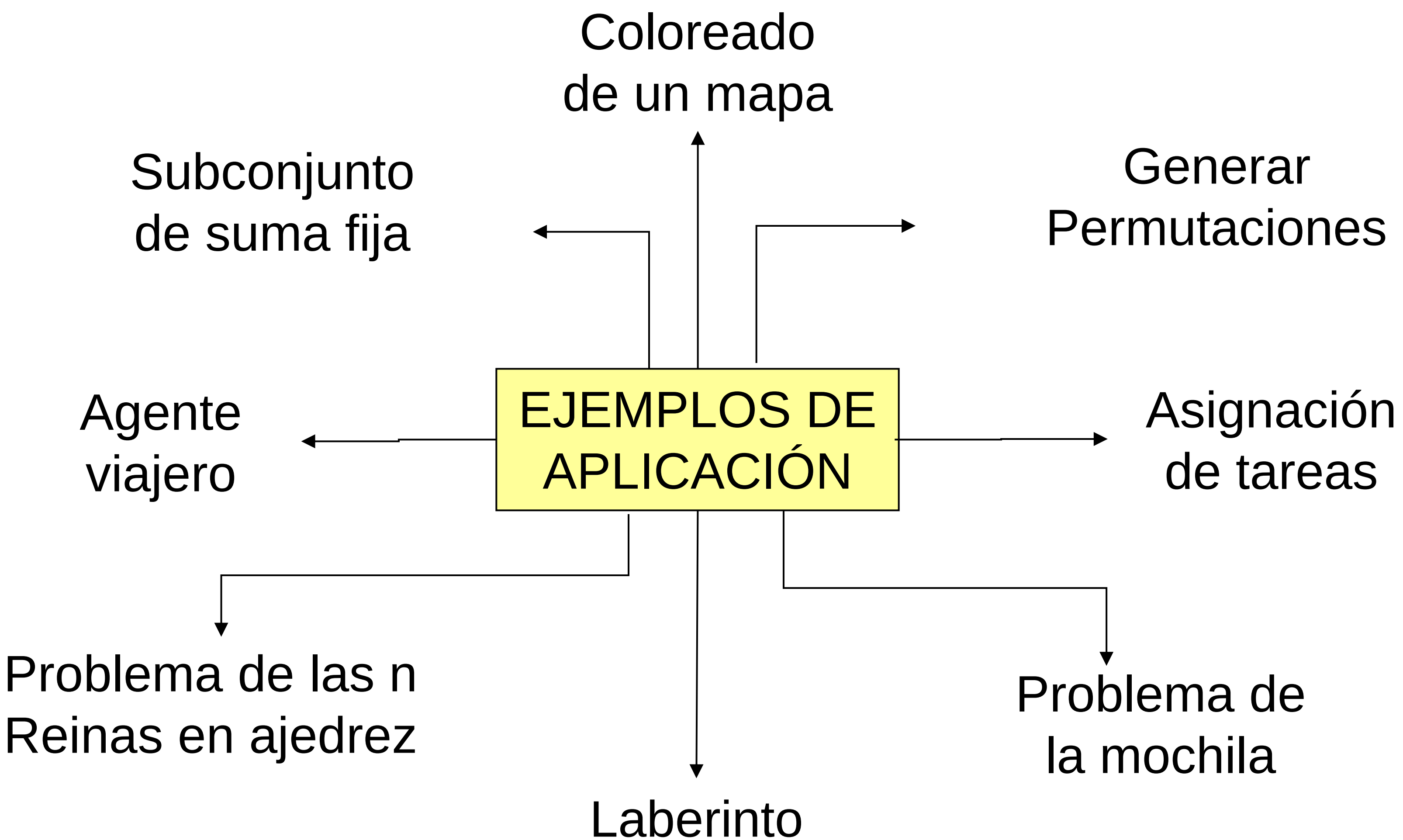
BACKTRACKING

PROBLEMA

SOLUCION → **VECTOR** de componentes
(x1, x2, ..., xn)

xi *:* *se* elige en cada etapa de entre
un conjunto finito de valores

BACKTRACKING



BACKTRACKING
Mochila Múltiple

```
Función mochila (i, M): tipo x peso → beneficio
// globales: n, b y p
// entrada: elementos de tipos i a n y con peso máximo M.
// salida: bmax el beneficio de la mejor carga.

        bmax←0
        Para k=i hasta n hacer
                si p(k) ≤ M entonces
                        bmax← max (bmax, b(k)+mochila(k, M-p(k))
        retorna bmax
```

Examina posibilidades dentro del nivel k

Baja un nivel en el árbol

Mochila Múltiple

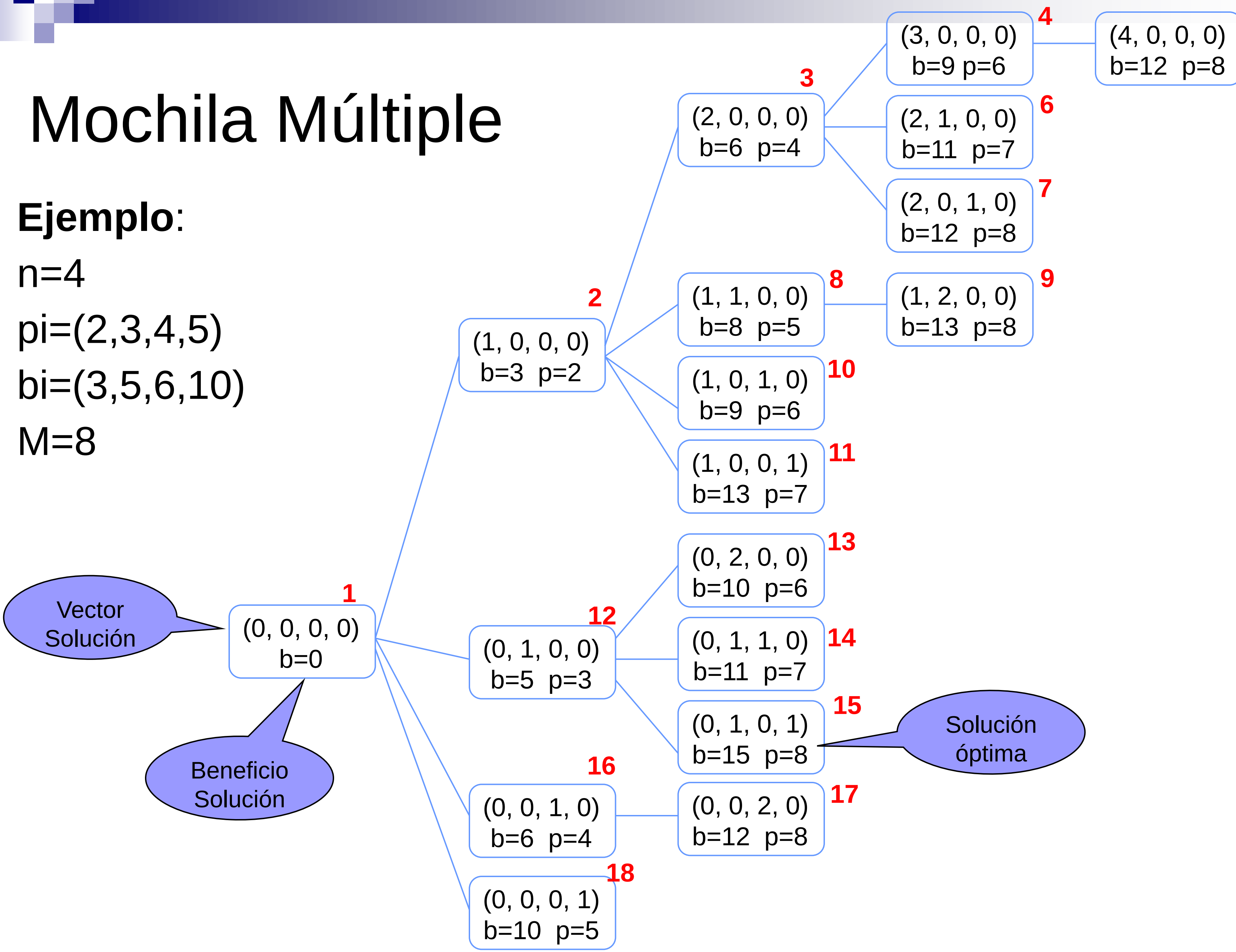
Ejemplo:

n=4

pi=(2,3,4,5)

bi=(3,5,6,10)

M=8



BACKTRACKING
Mochila 0/1

¿Cómo modificamos el algoritmo del problema de la Mochila Múltiple para adaptarlo al problema de la Mochila 0/1?

```
Función mochila (i, M): tipo x peso → beneficio
// globales: n, b y p
// entrada: elementos de tipos i a n y con peso máximo M.
// salida: bmax el beneficio de la mejor carga.

        bmax←0
        Para k=i hasta n hacer
                si p(k) ≤ M entonces
                        bmax← max (bmax, b(k)+mochila(k, M-p(k))
        retorna bmax
```

Mochila 0/1

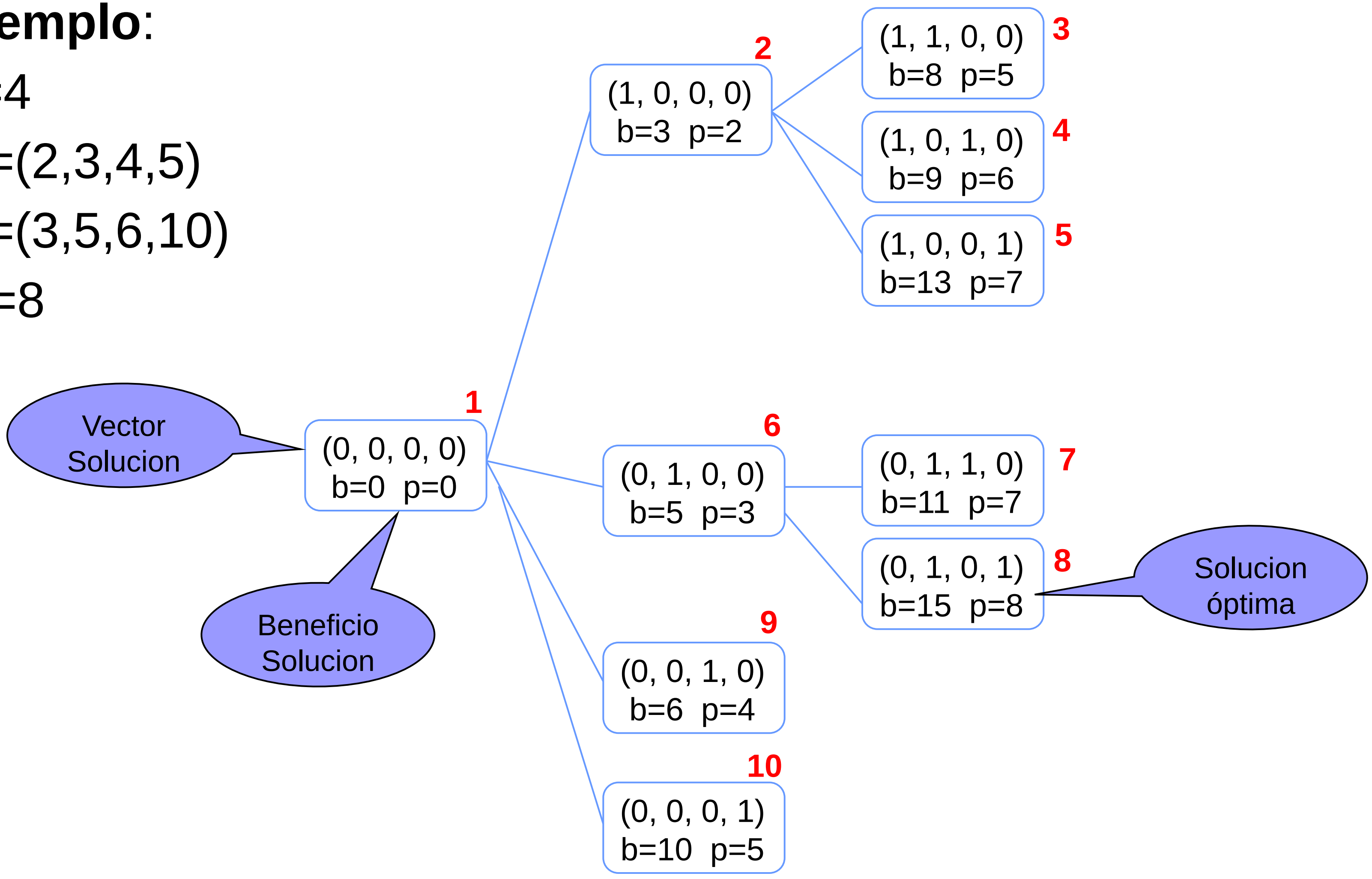
Ejemplo:

n=4

pi=(2,3,4,5)

bi=(3,5,6,10)

M=8



BACKTRACKING
Mochila 0/1

¿Cómo modificamos el algoritmo del problema de la Mochila Múltiple para adaptarlo al problema de la Mochila 0/1?

```
Función mochila (i, M): tipo x peso → beneficio
// globales: n, b y p
// entrada: elementos de tipos i a n y con peso máximo M.
// salida: bmax el beneficio de la mejor carga.

        bmax←0
        Para k=i hasta n hacer
                si p(k) ≤ M entonces
                        bmax← max (bmax, b(k)+mochila(k, M-p(k))
        retorna bmax
```

k+1

Expresión aritmética de valor M

Dado un número entero M y un vector V de n números naturales, se quiere determinar si existe una forma de insertar entre los n números del vector (en el mismo orden en que están colocados en el vector) operadores de suma y resta de forma tal que se obtenga una expresión aritmética con el valor de M como resultado final. Se quiere comprobar si es posible llegar a una solución, y en ese caso mostrar la o las expresiones de suma M.

DATOS:

M = 7

n = 4

V = [7,2,5,3]

POSIBLES EXPRESIONES

$$\left\{ \begin{array}{l} 7 + 2 + 5 + 3 = 17 \text{ ✗} \\ 7 + 2 + 5 - 3 = 11 \text{ ✗} \\ 7 + 2 - 5 - 3 = 1 \text{ ✗} \\ 7 - 2 + 5 - 3 = 7 \text{ ✓} \\ \ldots \end{array} \right.$$

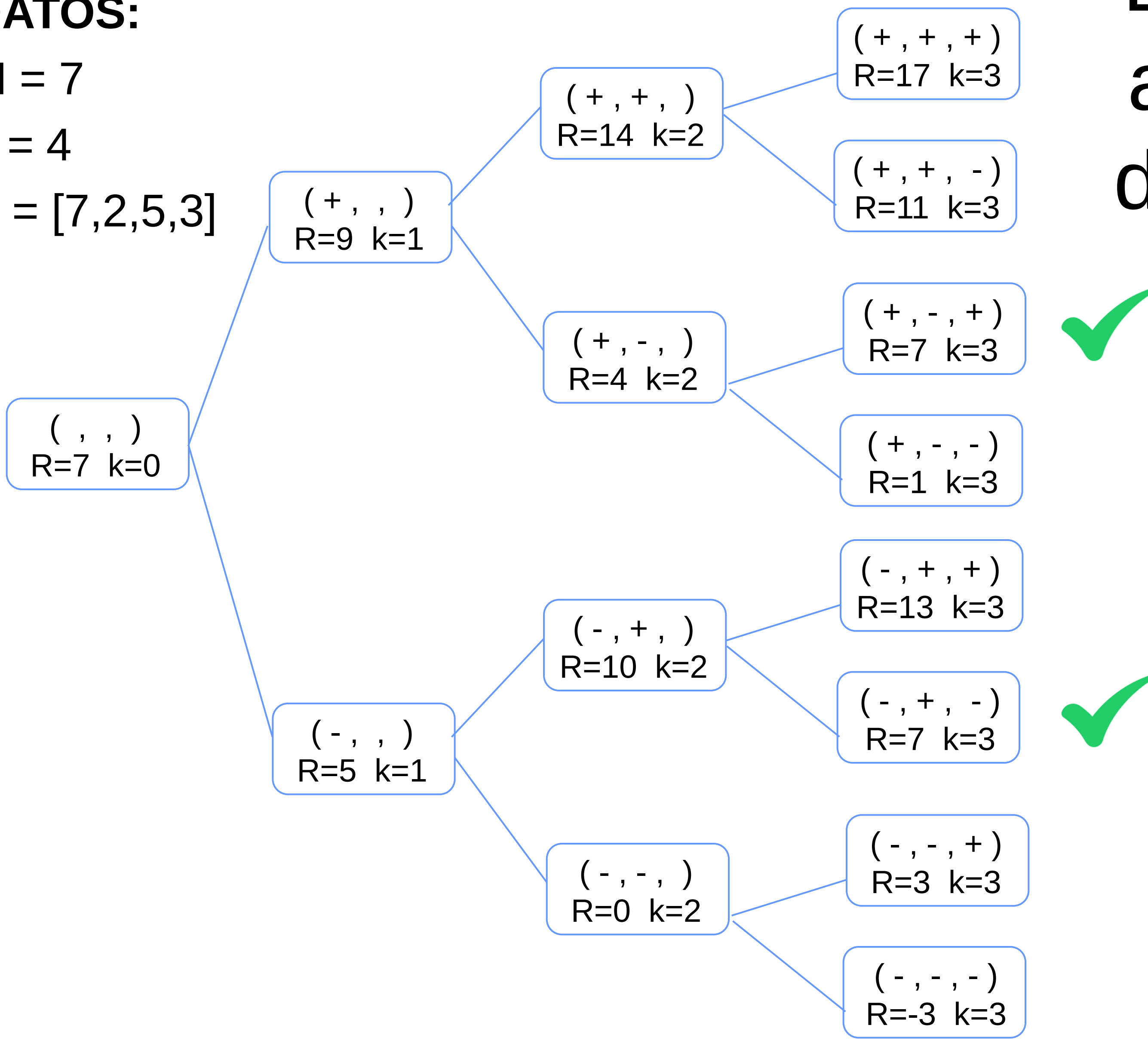
Expresión aritmética de valor M

DATOS:

M = 7

n = 4

V = [7,2,5,3]

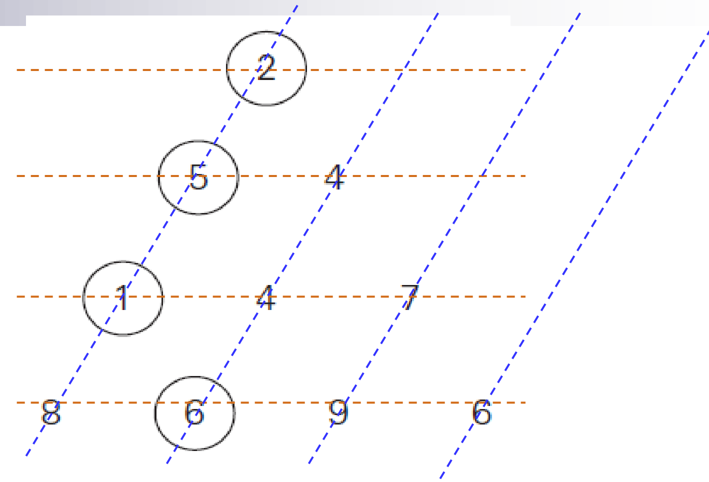


SALIDA

7 + 2 - 5 + 3

7 - 2 + 5 - 3

SUMA MINIMA TRIANGULO



Variables Globales

T_{n+1}

Columnas

Filas (niveles)

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

C_{n+1}

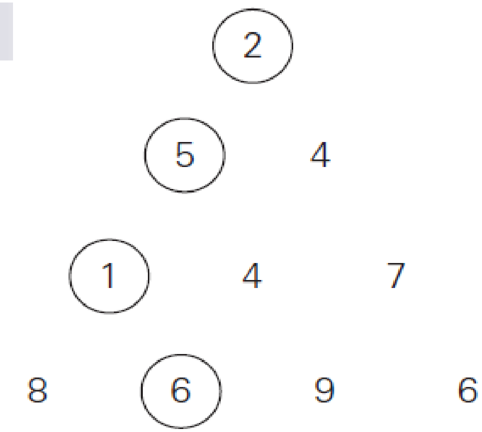
	0	1	2	3	4
0	∞	∞	∞	∞	∞
1	∞	2	∞	∞	∞
2	∞	∞	∞	∞	∞
3	∞	∞	∞	∞	∞
4	∞	∞	∞	∞	∞

SUMA MINIMA TRIANGULO

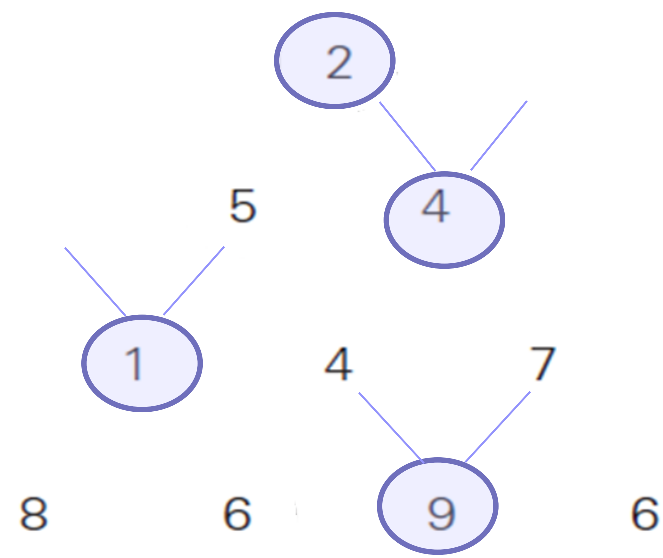
```
      2
    5   4
  1   4   7
8   6   9   6
```

```
FUNCION minSumTriangulo(n): entero≥1 → entero

    SI (n = 1) ENTONCES
        RETORNA T[1][1]
    SINO
        minimo ← ∞
        HACER n VECES (i = 1,..n)
            minimo ← min(minimo, MSD(n, i))
        RETORNA minimo
```

SUMA MINIMA TRIANGULO




```
FUNCION MSD(fila, col): ent.≥0 x ent.≥0 → ent

      // COMPLETAR
```



Preguntas…
…y a practicar…