

OperacionesSeq

11. Sea $v = (v_1, v_2, \dots, v_n)$ un vector de números naturales, y sea $w \in \mathbb{N}$. Se desea intercalar entre los elementos de v las operaciones $+$ (suma), \times (multiplicación) y \uparrow (potenciación) de tal manera que al evaluar la expresión obtenida el resultado sea w . Para evaluar la expresión se opera de izquierda a derecha **ignorando la precedencia de los operadores**. Por ejemplo, si $v = (3, 1, 5, 2, 1)$, y las operaciones elegidas son $+$, \times , \uparrow y \times (en ese orden), la expresión obtenida es $3 + 1 \times 5 \uparrow 2 \times 1$, que se evalúa como $((3 + 1) \times 5) \uparrow 2) \times 1 = 400$.

- Escribir una formulación recursiva que sea la base de un algoritmo de PD que, dados v y w , encuentre una secuencia de operaciones como la deseada, en caso de que tal secuencia exista. Explicar su semántica e indicar cuáles serían los parámetros para resolver el problema.
- Diseñar un algoritmo basado en PD con la formulación de a) y dar su complejidad temporal y espacial auxiliar. Comparar cómo resultaría un enfoque *top-down* con uno *bottom-up*.
- (Opcional) Formalizar el problema y demostrar que la función recursiva es correcta.

Igualas precedencia operaciones: ninguna operación pasa más que otra. Se evalúan de forma serial.
Ej: $3 + 1 \times 5 = 4 \times 5 = 20$ y NO a 8

Me piden 2 cosas: saber si \exists secuencia de op → PD
dudarla si \exists → ramo a hacer de la función de PD

PD

- c) ¿Qué ramo a dudar? Verdadero o falso X Secuencia de operaciones
- c) ¿Qué resultados? v, w
- c) ¿Cómo hacer con las operaciones? pidiendo diferentes cosas

Sea $v = (3, 1, 5, 2, 1)$ y $w = 8 \rightarrow$ NO \exists

↓
tomo el primer elemento → no hago nada y paso al siguiente → No puedo no hacer NADA

→ digo armado $3 +$

→ " " " $3 \times$

→ " " " $3 \uparrow$

quiero intercalar siempre elementos

segundo elemento → no hago nada

→ $3 + 1 \rightarrow 3 + 1 +$

→ $3 + 1 \times$

→ $3 + 1 \uparrow$

$$\begin{array}{c}
 \rightarrow 3 \times 1 \rightarrow 3 \times 1 + \\
 \downarrow 3 \times 1 \times \\
 \downarrow 3 \times 1^{\uparrow} \\
 \downarrow 3^1 \rightarrow 3^1 1 + \\
 \downarrow 3^1 1 \times \\
 \downarrow 3^1 1^{\uparrow}
 \end{array}$$

Voy a tener que recorrer v y fijarme cuando $w=0 \rightarrow$ devolver true si pasa → cuando lo implemente → voy a tener un vector como variable global con la reg de operaciones

$$OS(v, w, i, b) = \begin{cases} \text{true} \\ \text{false} \\ OS(v, w, i, v[i]) \\ OS(v, w, i+1, b+v[i]) \vee OS(v, w, i+1, b \times v[i]) \vee \\ OS(v, w, i+1, b^{\uparrow} v[i]) \end{cases}$$

me fijo con los 3 op posibles toda
los resultados

i = |v| y $w \cdot b = 0$ → $\begin{matrix} \text{recorrer todo} \\ \uparrow \text{la elemento} \\ \text{y encontre} \end{matrix}$
 i = |v| y $w \cdot b \neq 0$ → $\begin{matrix} \text{recorrer todo} \\ \downarrow \text{no lo} \\ \text{encontre} \end{matrix}$
 i = 0

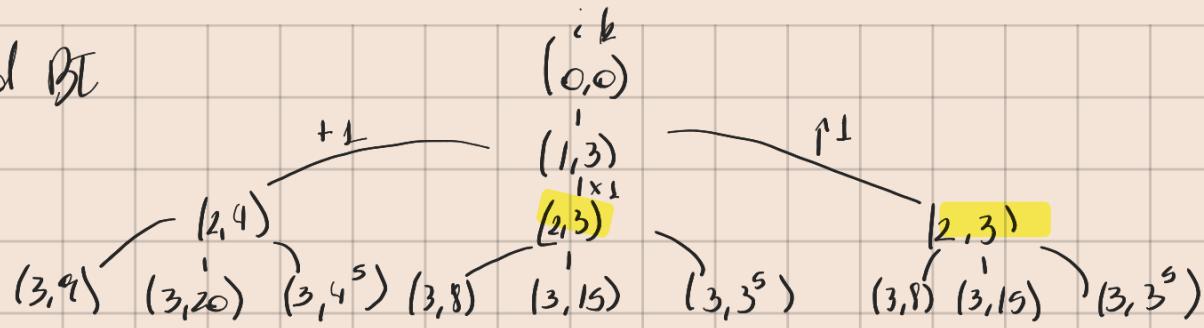
inicializo b ≠ 0
 Hago este XQ
 si me pasan
 un vector
 valido
 N indiferencia
 b si ya llamo
 como v[0]

i assume
 |v| > 1 no tiene
 falta

Puedo rendir llamando $OS(v, w, 0, 0)$:

$OS(v, w, 1, v[1])$ si aviso que $v \neq \emptyset$ no puede tener una entrada

Árbol BT



Vemos que hay \log_2 de subproblemas

→ # llamadas recursivas = $2(3^n)$

submatrices: tengo n número y para cada 1, 3 operaciones $\rightarrow O(3n)$

$$n \ll 3^n \text{ True}$$

Necesito definir una estructura de memorización. Puedo tener 3 matrices de $n \times n$ y en i una almacenar los valores de $i+j, i \times j, i^T j$ en las pos. i, j
o 3 matrices de $n \times n$ una elem., $n \times$ multiplicación elem., $n \times$ elemento

← Se me ocurre a la medida la complejidad espacial. En el momento que un resultado sea mayor a W ya devuelvo falso y no llego a aplicar \oplus

$$OS(v, w, i, b) = \begin{cases} \text{true} \\ \text{false} \\ OS(v, w, i, v[i]) \\ OS(v, w, i+1, b+v[i]) \vee OS(v, w, i+1, b \times v[i]) \vee \\ OS(v, w, i+1, b^T v[i]) \end{cases}$$

$$\begin{aligned} i &= |v| \text{ y } w \cdot b = 0 \\ i &\neq |v| \text{ y } w \cdot b \neq 0 \\ i &= 0 \end{aligned}$$

Si hago esto
puedo tener
una matriz de
 $n \times n$

Con este cambio puedo hacer 1 sola matriz de $n \times n$

Vector MyoreOp = \leftrightarrow , matrix M de $n \times w_{in} - 1$
OS(v, w, i, k, op)

if $i = v.size$

if $w.k = 0$

MyoreOp = op

else if $w.k < 0$ then

OS(v, w, v.size, k)

else if $i = 0$ then

OS(v, w, 1, v[i])

else

// Suma

if $v[i] + k \leq w$

if $pushback("suma")$

if $M[i][v[i] + k] = -1$

$M[i][v[i] + k] = OS(v, w, i+1, v[i]+k, op)$

esto está mal

↑ tengo que guardarme algo en la matriz

Matriz Multiplicación -1s

OS(v, w, i, k, qf)

if $i = w-1$

if $w-k = 0$ then

return qf;

else

return -1

else if $w-k < 0$ then

return -1

else if $i=0$ then

OS(v, w, i+1, v[i], qf)

else

// sumar

qf.pushback(+)

return OS(v, w, i+1, v[i] + k)

qf.popback

// producto

qf.pushback("x")

return OS(v, w, i+1, v[i] * k)

qf.popback

// elevación

qf.pushback("↑")

return OS(v, w, i+1, v[i] ↑ k)

qf.popback

if $v[i]+k \leq w$ then

if $M[i][v[i]+k] = -1$

$M[i][v[i]+k] = qf.pushback("+")$

OS(v, w, i+1, v[i] + k)

return $M[i][v[i]+k]$

else

return -1

↓

Hago esto mismo para multiplicación
y potencia

función del ciclo for

return $M[i][k]$