Taller 2 - Programación Dinámica

Nicolás Camacho-Plazas

28 de septiembre de 2020

Parte I
Análisis del problema

El problema requiere que se calculen las particiones binarias cuya diferencia de sumas se mínima dado un conjunto de números naturales. El conjunto de números naturales se definirá como una secuencia S, donde:

$$S = \langle s_1, s_2, \cdots, s_n \rangle = \langle s_i \in \mathbb{N} \rangle$$

Las particiones binarias son dos subconjuntos (A y B) formados a partir de los elementos de S y cuya unión es equivalente al conjunto S, es decir, que todos los elementos de S pertenecerán a A o a B sin pertenecer a los dos; además, si s es la cantidad de elementos en A y m es la cantidad de elementos de B, entonces:

$$\sum_{i=1}^{s} A_i = a$$

$$\sum_{i=1}^{m} B_i = b$$

$$n = a + b$$

$$x = |a - b|$$

Donde x es el mínimo valor posible de todas las particiones binarias que se pueden formar a partir de la secuencia S.

Parte II
Diseño del algoritmo

1. Entradas

- Una secuencia $S = \langle s_1, s_2, \cdots, s_n \rangle = \langle s_i \in \mathbb{N} \rangle$

2. Salidas

- Los subconjuntos disjuntos A y B formados a partir de S cuya unión es equivalente a S y cuyo valor x es el mínimo posible, siendo x la diferencia entre la suma de todos los elementos del subconjunto A y la suma de los elementos de B.

Parte III
Algoritmo

Algorithm 1 Partición binaria con mínima diferencia de sumas totales

```
1: procedure binPartitionBT(S)
2:     total = 0
3:     for i < |S| do
4:         total = total + S[i]
5:     end for
6:     Suponga que M es una matriz booleana de tamaño (|S| + 1)x(total + 1)
7:     for i ≤ |S| do
8:         for j ≤ total do
9:             if j == 0 then
10:                M[i][j] = true
11:            else
12:                M[i][j] = false
13:            end if
14:        end for
15:    end for
16:    Suponga que BT es una matriz de enteros de tamaño (|S| + 1)x(total + 1)x(total)
17:    for i = 1; i < |S| + 1 do
18:        for j = 1; j < total + 1 do
19:            if (j - S[i - 1]) == true then
20:                if M[i - 1][j - S[i - 1]] == true then
21:                    M[i][j] = true
22:                    BT[i][j] = BT[i][j] Union BT[i - 1][j - S[i - 1]]
23:                    BT[i][j] Union S[i - 1]
24:                else if M[i - 1][j] == true then
25:                    BT[i][j] = true
26:                    BT[i][j] = BT[i][j] Union BT[i - 1][j]
27:                end if
28:            else
29:                M[i][j] = M[i - 1][j]
30:                BT[i][j] = BT[i][j] Union BT[i - 1][j]
31:            end if
32:        end for
33:    end for
34:    suma = total/2
35:    found = false
36:    while suma ≥ 0 and found == false do
37:        if M[|S|][suma] == true then
38:            found = true
39:        else
40:            suma = suma - 1
41:        end if
42:    end while
43:    secuenciaA = BT[|S|][suma]
44:    secuenciaB = S y no secuenciaA
45:    return [secuenciaA, secuenciaB]
46: end procedure
```

Parte IV
Complejidad

Por inspección de código se concluye que la complejidad tienen un orden de $O(nm)$, donde $n = |S|$ y $m = \sum_0^i S_i$.

Parte V
Invariante

- La tabla de memoización M se llena correctamente, es decir, si $M[i][j] = true$, se puede alcanzar el valor j con los elementos pertenecientes a $\langle S_0, S_1, S_2, ..., S_{i-1} \rangle$