Problema A

0. Identificación:

Autores del Problema A

- Tony Santiago Montes Buitrago [202014562] <t.montes@uniandes.edu.co>
- Juan Carlos Marin Morales [202013973] <j.marinm@uniandes.edu.co>

1. Algoritmo de Solución:

Inicialmente se transformó la función r para que fuese más comprensible y manipulable; a la forma:

$$r(0) = A$$

$$r(1) = B$$

$$r(k) = D \cdot r(k-1) + C \cdot r(k-2), \qquad \forall k \geq 2$$

$$cp(r,n) = \big(+k \mid 0 \leq k \leq n : r(k) \cdot r(n-k)\big), \qquad n \geq 0$$

Y realizando varios ejemplos se pudo denotar que, dada la paridad de la multiplicación $r(k) \cdot r(n-k)$ para $0 \leq k \leq n$, se podía hacer el cálculo únicamente de la mitad de los datos, y este resultado multiplicarlo por 2. Sin embargo, para los n pares ($n+1$ impares) ocurre que el elemento de la mitad se debe sumar elevado al cuadrado. Esto se puede ver mejor en el siguiente ejemplo:

$$r = [0,1,1,2,3,5,8,13,21]$$

$$cp(r,5) = 0 \cdot 5 + 1 \cdot 3 + 1 \cdot 2 + 2 \cdot 1 + 3 \cdot 1 + 5 \cdot 0 = 2 \cdot (0 \cdot 5 + 1 \cdot 3 + 1 \cdot 3)$$

$$cp(r,8) = 2 \cdot (0 \cdot 21 + 1 \cdot 13 + 1 \cdot 8 + 2 \cdot 5) + 3^2$$

Dado esto, lo que se hizo para calcular la *convolución ponderada*, lo que se hizo fue sumar hasta la mitad los datos.

Por otra parte, para llenar los datos de r, se utilizó **programación dinámica**, en este caso vimos que esta era la opción más viable y la que más permitía minimizar el tiempo, ya que se trataba de una función recursiva. Para la implementación no fue necesario realizar ecuación de recurrencia (porque ya se da) ni grafo de necesidades, ya que por la forma en que está escrita la ecuación ya se sabe que se deben llenar los datos de izquierda a derecha.

Métodos:

1. `fill_r`

Precondición: Q_1: { $(0 < n < 100) \wedge (-1 \leq A, B, C, D \leq 1)$ }

Postcondición: R_1: { $(\forall\, k \mid 2 \leq k \leq n : r[k] = D * r[k-1] + C * r[k-2]) \wedge r[0] = A \ \wedge r[1] = B$ }

2. `cp`

Precondición: Q_2: { $0 < n < 100$ }

Postcondición: R_2: { $x = (+k \mid 0 \leq k \leq n : r[k] \cdot r[n-k])$ }

2. Análisis de Complejidad:

a. Temporal: ***O(n)***

1. Para llenar el arreglo r se tienen que recorrer solo una vez todos los elementos desde 2 hasta n, ya que la información de los anteriores ya se tendrá almacenada. $O(n-2+1) = O(n-1)$

2. Para realizar la sumatoria del cálculo de la *convolución ponderada* únicamente se recorre la mitad de los datos almacenados en r. $O(n/2)$

Finalmente, la complejidad total será el máximo, es decir: $O(n-1+n/2) = O(n)$

* NOTA: n es el parámetro n que es entrada del problema.

b. Espacial: ***O(n)***

Únicamente se debe tener en cuenta el tamaño del arreglo r, ya que es la única estructura de datos que no tiene un tamaño constante. El tamaño del arreglo siempre será $O(n+1) = O(n)$

3. Comentarios Finales:

Para calcular el tiempo utilizado para medir el rendimiento temporal de la solución realizada, se implementó el decorador *timer*. El tiempo medido para los máximos valores de n,A,B,C,D (99,1,1,1,1) no logró ser perceptible por el tiempo medido (midió 0.000000 segundos), lo cual demuestra que la solución propuesta tiene un buen rendimiento temporal, y por otra parte el rendimiento espacial no es malo, ya que no se requiere más que un arreglo para conseguir una solución óptima.