Problema B

## Identificación

Juan Pablo Campos 201630726

Santiago Beltrán20 201632541

## Algoritmo de solución

Especificación del problema:

suma(r)= (Σk|0 ≤k<n: a[k])

cumple(r)= (∀i| 1 ≤i<n: (r[i-1]ϵ A ∧ r[i]ϵB) ∨ (r[i-1]ϵ B)

Pre: {n >0 ∧ C > 0}

Pos:{ (capitalActual = (max r| |r| = n : suma(r)) ∧ cumple(r)}

El algoritmo cuenta con dos métodos auxiliares: construirMatrizy maximoCapital. El primero se encarga de construir la estructura para la siguiente ecuación de recurrencia simplificando el problema a obtener el número máximo de los dos arreglos con la restricción dada en el enunciado:

b es true si el número anteriormente escogido es del arreglo A, false de lo contrario

f(i,b) =

max(B[i] + f(i+1,false) , A[i] + f(i+1,true) si b=false

B[i] + f(i+1, false) si b=true

B[i] si i=n ∧ b=true

max(B[i],A[i]) si i=n ∧ b=false

Dicha recurrencia es trasladada a programación dinámica con una matriz de nx2 en la cual la fila hace referencia al índice i, la columna 0 a b en true y la 1 en lo contrario. Con esto en mente se llenan los casos base de primero que corresponde a los elementos de la última fila y a partir de allí se empieza a llenar la matriz m hacia arriba. Se crea otra matriz sel la cual guarda los números que se escogerían en el índice i a partir de la optimización realizada en la otra matriz. Por último, se recorre la matriz sel y se agregan a la lista que será retornada los elementos que se deben escoger para maximizar su suma.

La lista es recorrida y por cada elemento se llama el método maximoCapital que se encarga de retornar el dinero que Juan obtiene realizando la operación ⎣z\*(1+rX/100)⎦.

## Análisis complejidad temporal/espacial

O(n) = n

S(n) = n

Se tiene que el número de columnas de la matriz es constante (2) por lo que el espacio necesario sería de 2n + otras variables y listas temporales. De igual forma, cada matriz es recorrida una vez en donde se hacen operaciones que se pueden asumir constantes como comparación, suma/resta, acceso a arreglo y el método Math.max().

## Comentarios del desempeño

Tener un orden de complejidad espacial y temporal lineal resulta ser un buen desempeño para el programa realizado. En el caso del desempeño es el mejor orden que se puede obtener porque como mínimo se tarda un tiempo n leyendo las entradas