

**Ingeniería de Sistemas y Computación**

**ISIS1105 – Diseño y análisis de algoritmos**

**Sección 1. Profesor: Jorge Duitama**

**Semestre: 2018-10**

**Tarea 6**

# Objetivos

1. Entender conceptos de programación dinámica
2. Comparar diferentes algoritmos para solucionar un problema
3. Practicar programación

# Parte 1. El problema de las vueltas

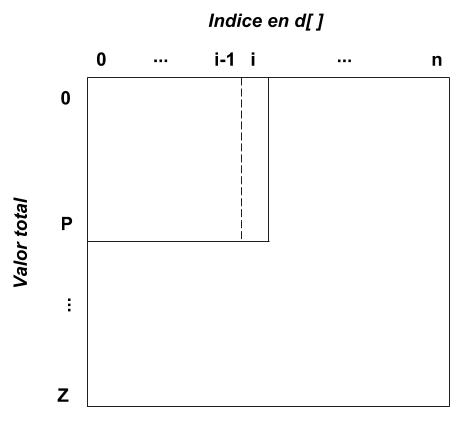
Se desea construir un algoritmo que dada una cantidad total de dinero P y unas denominaciones de monedas d1, d2, … dn, determine la cantidad mínima de monedas m1, m2, …, mn, tal que la suma total de dinero sea igual a P.

Para este problema, descargar el proyecto adjunto y realizar los siguientes pasos:

1. Formalizar el problema de las vueltas como un problema de programación lineal. Describir la función a minimizar y una ecuación que represente la restricción.

Encontrar tal que:

1. Derivar una función que represente el problema y una ecuación de recurrencia que describa la función. Argumentar por qué la ecuación de recurrencia efectivamente representa la función diseñada.
2. Crear una clase que implemente la interfaz CoinChangeCalculator implementando directamente la ecuación de recurrencia como una función recursiva. Calcular la complejidad temporal de este algoritmo.
3. Crear una clase que implemente la interfaz CoinChangeCalculator implementando un algoritmo voraz que escoja en cada paso la moneda con mayor denominación. Calcular la complejidad temporal de este algoritmo.
4. Dibujar un grafo de necesidades de acuerdo con la ecuación de recurrencia planteada en el segundo punto.



1. De acuerdo con el grafo de necesidades, crear una clase que implemente la interfazCoinChangeCalculator implementando un algoritmo de programación dinámica para resolver el problema. Calcular la complejidad temporal de este algoritmo.
2. Utilizar el programa disponible en la clase ExampleCoinChange para probar los algoritmos. Probar los diferentes algoritmos con valores totales 1000, 100000 y un millon y con 3 distintos conjuntos de denominaciones y generar una tabla con el tiempo que necesitó cada algoritmo.
3. Describir un valor y un conjunto de denominaciones en el que el algoritmo voraz no encuentra la solución óptima.

# Parte 2: Otros problemas de programación dinámica

Para los siguientes problemas realice los siguientes pasos:

1. Describa una función con la que se pueda representar el problema
2. Defina una ecuación de recurrencia para calcular dicha función que exprese lasolución en términos de soluciones a subproblemas relacionados c. Dibuje el grafo de necesidades relacionado con la ecuación

d. Diseñe un algoritmo de programación dinámica en GCL que permita obtener cualquier valor de la ecuación de recurrencia

**Nota:** No es obligatorio (aunque si es recomendable) desarrollar una implementación del algoritmo.

1. Dado un arreglo a de números naturales y un número total T, decidir si existe unconjunto C de índices del arreglo tal que:

(+ i | i ∈ C : a[i]) = T

Ejemplo de entrada: a=[15,28,3,12,12] y T = 30

1. Para procesar archivos de texto es muy común partir una cadena en subcadenas deacuerdo con un caracter de corte específico (espacio, tab, coma, etc). Dada una cadena a de tamaño n y una ocurrencia del caracter de corte en una posición i, vamos a asumir que el costo de generar las dos cadenas a[0..i-1] y a[i+1..n) es igual a n porque sería necesario recorrer la cdena completa para copiarla en las dos subcadenas.

Dada una cadena a de tamaño n y un caracter de corte c que ocurre k veces en a, determinar la secuencia de k cortes que minimiza el costo total de la operación.

Ejemplo de entrada. a=“Diseño+de+algoritmos” c=”+”.

Costo de cortar de izquierda a derecha: 20 + 13 = 33

Costo de cortar de derecha a izquierda: 20 + 9 = 29

1. Considere el siguiente juego para dos personas. Se organiza en una fila un número Npar de monedas. Los jugadores por turno escogen una moneda de alguno de los dos extremos y se la quedan. Esto se repite hasta que ya no quedan monedas. El jugador que termine con más dinero gana.

Escriba un algoritmo que le ayude al primer jugador (al primero que retira una moneda) a encontrar la combinación de jugadas que debe hacer para maximizar la cantidad de dinero que tiene al final, sin importar como juegue el segundo jugador.