Tarea 4

Objetivos

1. Diseñar ecuaciones de recurrencia para resolver problemas con 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 d_1, d_2, ... d_n, determine la cantidad mínima de monedas m_1, m_2, ..., m_n, 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 describiendo sus entradas, salidas, precondición y postcondición

2. Definir una función que represente el valor a minimizar y una ecuación de recurrencia que permita calcular esta función. Argumentar por qué la ecuación de recurrencia efectivamente calcula la función diseñada.

3. Crear una clase que implemente la interfaz CoinChangeCalculator implementando directamente la ecuación de recurrencia como una función recursiva.

4. 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.

5. Dibujar un grafo de necesidades de acuerdo con la ecuación de recurrencia planteada en el segundo punto

6. De acuerdo con el grafo de necesidades, crear una clase que implemente la interfaz CoinChangeCalculator implementando un algoritmo de programación dinámica para resolver el problema. Calcular la complejidad temporal de este algoritmo.

7. 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.

8. 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:

a. Formalizar el problema describiendo sus entradas, salidas, precondición y postcondición
b. Definir una función con la que se pueda representar el valor a optimizar o a contar en el problema o que represente la respuesta en el caso de problemas de si o no.
b. Definir una ecuación de recurrencia para calcular dicha función que exprese la solución en términos de soluciones a subproblemas relacionados
c. Dibujar el grafo de necesidades relacionado con la ecuación
d. Diseñar un algoritmo de programación dinámica que permita obtener cualquier valor de la ecuación de recurrencia. Puede escribir el algoritmo en el lenguaje de su elección.

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 un conjunto C de índices del arreglo tal que:

$(+ i \mid i \in C : a[i]) = T$

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

2. Dada una matriz (no necesariamente cuadrada) de unos y ceros, encontrar la cantidad de filas y columnas de la submatriz cuadrada más grande, tal que todos los elementos de dicha submatriz sean iguales a 1.

3. Desarrollar un programa que cuente la cantidad de números de N dígitos en base 4 que no tengan ceros adyacentes. Por ejemplo, para N=10 un número válido sería 3011203320 mientras que uno inválido sería 2113002021

4. El profesor de un colegio decidió repartirle dulces a sus N estudiantes de acuerdo con la nota que sacaron en un examen. Para que ninguno se sienta mal, le va a repartir por lo menos un dulce a cada estudiante. Para facilitar este proceso, decide sentar los estudiantes en una sola fila. Sin embargo, como los estudiantes hablan entre ellos,

cada estudiante sabe su propia nota y la de sus dos vecinos. De esta forma, un estudiante que haya sacado más nota que un vecino espera recibir más dulces que dicho vecino. El profesor necesita entonces un programa que le ayude a calcular cuántos dulces necesitaría repartir como mínimo para cumplir estas condiciones.

Nota: Si dos estudiantes que sean vecinos sacaron la misma nota, deben recibir la misma cantidad de dulces.

Ejemplo:

Nota	3.5	4	4.5	4.5	2.5	3	3
Dulces	1	2	3	3	1	2	2

Mínimo de dulces: 14

5. En un juego de video el protagonista tiene que atravesar un recorrido lineal de N metros. En cada metro pueden ocurrir una de dos cosas:

1. Hay un trampolín que le permite saltar una cantidad de metros entre 2 y K hacia adelante. El protagonista puede decidir si usa el trampolín para saltar o si simplemente camina un metro hacia adelante.
2. Hay un abismo en el que si cae, pierde el juego.

Se debe desarrollar un programa que determine si existe algúna forma de llegar al final del recorrido. Se debe llegar exactamente al metro N porque después de este metro hay un abismo.

Ejemplos:

Suponiendo que cada posición se representa por la cantidad de metros que permite saltar el trampolín que está en esa posición y -1 indica que en esa posición hay un hueco, el siguiente es un ejemplo en el que las casillas coloreadas muestran cómo se podría llegar al final. Se colorean de amarillo las posiciones en las que se debe caminar y en verde las posiciones en las que se debe saltar:

4	5	9	2	-1	-1	3	-1	3	2	2	-1	3	4	2	-1	3	0

Por el contrario, para el siguiente ejemplo no es posible llegar al final:

4	3	9	2	-1	-1	3	-1	3	2	2	-1	3	4	2	-1	3	0