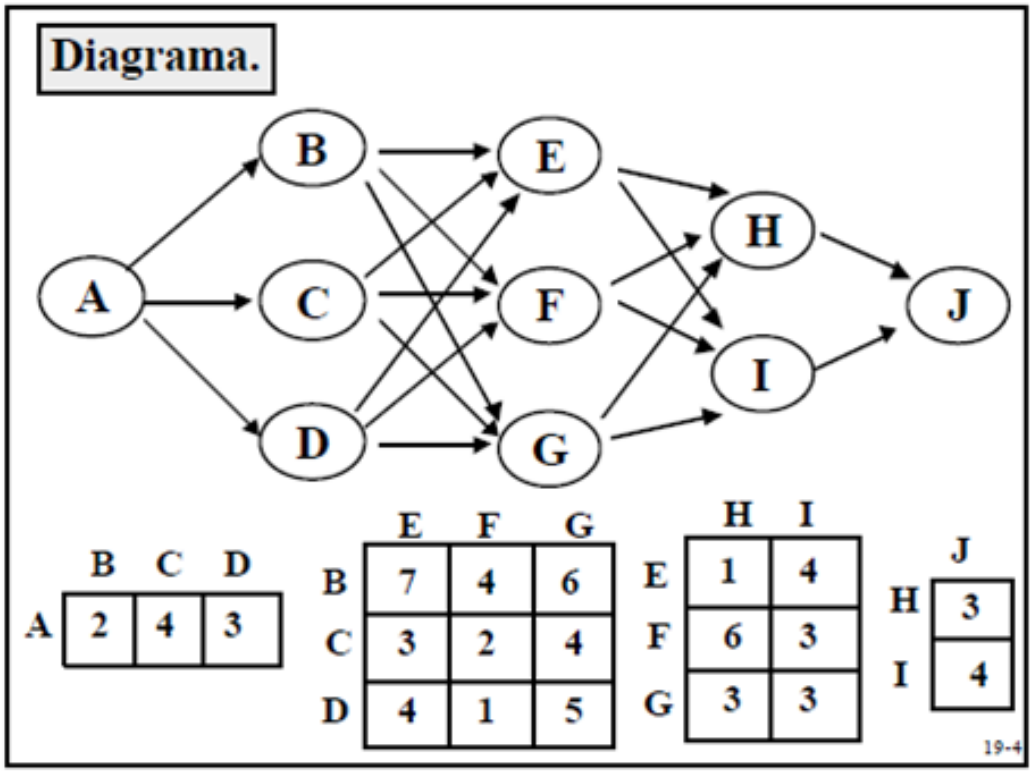
P R E D A



Laura Saltaren Andrade – 47586360G
lsaltaren1@alumno.uned.es

1.Respuesta a las cuestiones teóricas planteadas en este enunciado.

Coste Temporal y Espacial del Algoritmo:
Coste Temporal:
El algoritmo implementa programación dinámica para resolver el problema del cambio de monedas. La complejidad temporal es $O(n \cdot C)$, donde n es el número de tipos de monedas y C es la cantidad total a devolver.
Esto se debe a que, en el peor caso, el algoritmo debe llenar una tabla de tamaño $n \times C$.

Coste Espacial:
El algoritmo utiliza una matriz de tamaño $n \times C$ para almacenar los resultados intermedios, lo que da un coste espacial de $O(n \cdot C)$. Además, se utilizan vectores y variables adicionales, pero su tamaño no depende de la entrada, por lo que no afectan significativamente el coste espacial en términos asintóticos.

Otros Esquemas para Resolver el Problema:
Enfoque Voraz : Podría considerarse un enfoque voraz seleccionar siempre la moneda de mayor valor que sea menor o igual a la cantidad restante. Sin embargo, este enfoque no garantiza la solución óptima, ya que en algunos casos podría no proporcionar el cambio mínimo.
Algoritmo de Vuelta Atrás : Un enfoque de vuelta atrás podría explorar todas las combinaciones posibles de monedas para encontrar la solución óptima. Sin embargo, este enfoque podría ser ineficiente en términos de tiempo, ya que la cantidad de combinaciones crece exponencialmente con el número de tipos de monedas.
Algoritmo de Cambio de Monedas Convencional: Si la cantidad a devolver es relativamente pequeña y los valores de las monedas son (por ejemplo, 1, 5, 10, 25), se podría utilizar un algoritmo más simple basado en restas sucesivas.

2.Un ejemplo de ejecución para distintos tamaños del problema.

Para archivo prueba.txt

3
1 6 10
12

```
C:\Users\lausa>java -jar CambioDinamica.jar -t prueba.txt resultado.txt

 Se ha leido el archivo desde entrada
Iniciando el método setTable...
Fila 0 Columna 1 es : 1
Fila 0 Columna 2 es : 2
Fila 0 Columna 3 es : 3
Fila 0 Columna 4 es : 4
Fila 0 Columna 5 es : 5
Fila 0 Columna 6 es : 6
Fila 0 Columna 7 es : 7
Fila 0 Columna 8 es : 8
Fila 0 Columna 9 es : 9
Fila 0 Columna 10 es : 10
Fila 0 Columna 11 es : 11
Fila 0 Columna 12 es : 12
Fila 1 Columna 1 es : 1
Fila 1 Columna 2 es : 2
Fila 1 Columna 3 es : 3
Fila 1 Columna 4 es : 4
Fila 1 Columna 5 es : 5
Fila 1 Columna 6 es : 1
Fila 1 Columna 7 es : 2
Fila 1 Columna 8 es : 3
Fila 1 Columna 9 es : 4
Fila 1 Columna 10 es : 5
Fila 1 Columna 11 es : 6
Fila 1 Columna 12 es : 2
Fila 2 Columna 1 es : 1
Fila 2 Columna 2 es : 2
Fila 2 Columna 3 es : 3
Fila 2 Columna 4 es : 4
Fila 2 Columna 5 es : 5
Fila 2 Columna 6 es : 1
Fila 2 Columna 7 es : 2
Fila 2 Columna 8 es : 3
Fila 2 Columna 9 es : 4
Fila 2 Columna 10 es : 1
Fila 2 Columna 11 es : 2
Fila 2 Columna 12 es : 2

 Se ha escrito en el archivo de salida
```

Para archivo resultado.txt

Para archivo prueba.txt

4

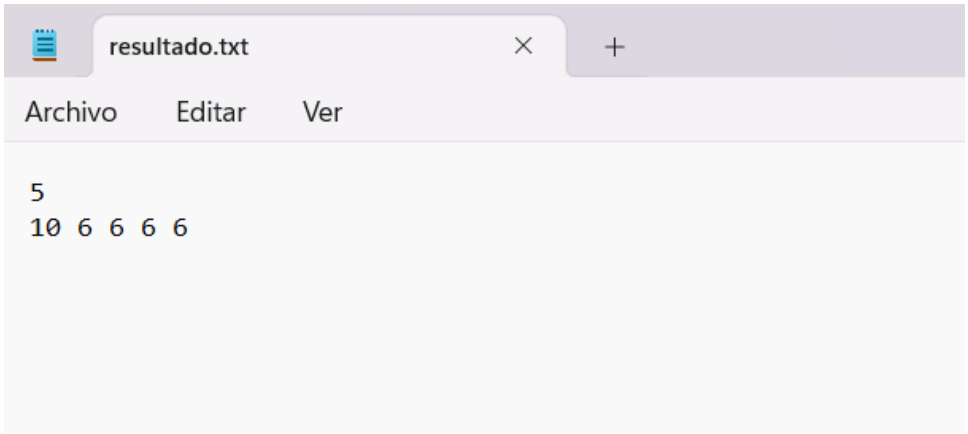
1 6 10 3
34

```
C:\Users\lausa>java -jar CambioDinamica.jar -t prueba.txt resultado.txt

 Se ha leido el archivo desde entrada
Iniciando el método setTable...
Fila 0 Columna 1 es : 1
Fila 0 Columna 2 es : 2
Fila 0 Columna 3 es : 3
Fila 0 Columna 4 es : 4
Fila 0 Columna 5 es : 5
Fila 0 Columna 6 es : 6
Fila 0 Columna 7 es : 7
Fila 0 Columna 8 es : 8
Fila 0 Columna 9 es : 9
Fila 0 Columna 10 es : 10
Fila 0 Columna 11 es : 11
Fila 0 Columna 12 es : 12
Fila 0 Columna 13 es : 13
Fila 0 Columna 14 es : 14
Fila 0 Columna 15 es : 15
Fila 0 Columna 16 es : 16
Fila 0 Columna 17 es : 17
Fila 0 Columna 18 es : 18
Fila 0 Columna 19 es : 19
Fila 0 Columna 20 es : 20
Fila 0 Columna 21 es : 21
Fila 0 Columna 22 es : 22
Fila 0 Columna 23 es : 23
Fila 0 Columna 24 es : 24
Fila 0 Columna 25 es : 25
Fila 0 Columna 26 es : 26
Fila 0 Columna 27 es : 27
Fila 0 Columna 28 es : 28
Fila 0 Columna 29 es : 29
Fila 0 Columna 30 es : 30
Fila 0 Columna 31 es : 31
Fila 0 Columna 32 es : 32
Fila 0 Columna 33 es : 33
Fila 0 Columna 34 es : 34
Fila 1 Columna 1 es : 1
Fila 1 Columna 2 es : 2
Fila 1 Columna 3 es : 1
Fila 1 Columna 4 es : 2
Fila 1 Columna 5 es : 3
Fila 1 Columna 6 es : 2
Fila 1 Columna 7 es : 3
Fila 1 Columna 8 es : 4
Fila 1 Columna 9 es : 3
Fila 1 Columna 10 es : 4
Fila 1 Columna 11 es : 5
Fila 1 Columna 12 es : 4
Fila 1 Columna 13 es : 5
Fila 1 Columna 14 es : 6
Fila 1 Columna 15 es : 5
```

```
Fila 1 Columna 6 es : 2
Fila 1 Columna 7 es : 3
Fila 1 Columna 8 es : 4
Fila 1 Columna 9 es : 3
Fila 1 Columna 10 es : 4
Fila 1 Columna 11 es : 5
Fila 1 Columna 12 es : 4
Fila 1 Columna 13 es : 5
Fila 1 Columna 14 es : 6
Fila 1 Columna 15 es : 5
Fila 1 Columna 16 es : 6
Fila 1 Columna 17 es : 7
Fila 1 Columna 18 es : 6
Fila 1 Columna 19 es : 7
Fila 1 Columna 20 es : 8
Fila 1 Columna 21 es : 7
Fila 1 Columna 22 es : 8
Fila 1 Columna 23 es : 9
Fila 1 Columna 24 es : 8
Fila 1 Columna 25 es : 9
Fila 1 Columna 26 es : 10
Fila 1 Columna 27 es : 9
Fila 1 Columna 28 es : 10
Fila 1 Columna 29 es : 11
Fila 1 Columna 30 es : 10
Fila 1 Columna 31 es : 11
Fila 1 Columna 32 es : 12
Fila 1 Columna 33 es : 11
Fila 1 Columna 34 es : 12
Fila 2 Columna 1 es : 1
Fila 2 Columna 2 es : 2
Fila 2 Columna 3 es : 1
Fila 2 Columna 4 es : 2
Fila 2 Columna 5 es : 3
Fila 2 Columna 6 es : 1
Fila 2 Columna 7 es : 2
Fila 2 Columna 8 es : 3
Fila 2 Columna 9 es : 2
Fila 2 Columna 10 es : 3
Fila 2 Columna 11 es : 4
Fila 2 Columna 12 es : 2
Fila 2 Columna 13 es : 3
Fila 2 Columna 14 es : 4
Fila 2 Columna 15 es : 3
Fila 2 Columna 16 es : 4
Fila 2 Columna 17 es : 5
Fila 2 Columna 18 es : 3
Fila 2 Columna 19 es : 4
Fila 2 Columna 20 es : 5
Fila 2 Columna 21 es : 4
```

```
Fila 2 Columna 22 es : 5
Fila 2 Columna 23 es : 6
Fila 2 Columna 24 es : 4
Fila 2 Columna 25 es : 5
Fila 2 Columna 26 es : 6
Fila 2 Columna 27 es : 5
Fila 2 Columna 28 es : 6
Fila 2 Columna 29 es : 7
Fila 2 Columna 30 es : 5
Fila 2 Columna 31 es : 6
Fila 2 Columna 32 es : 7
Fila 2 Columna 33 es : 6
Fila 2 Columna 34 es : 7
Fila 3 Columna 1 es : 1
Fila 3 Columna 2 es : 2
Fila 3 Columna 3 es : 1
Fila 3 Columna 4 es : 2
Fila 3 Columna 5 es : 3
Fila 3 Columna 6 es : 1
Fila 3 Columna 7 es : 2
Fila 3 Columna 8 es : 3
Fila 3 Columna 9 es : 2
Fila 3 Columna 10 es : 1
Fila 3 Columna 11 es : 2
Fila 3 Columna 12 es : 2
Fila 3 Columna 13 es : 2
Fila 3 Columna 14 es : 3
Fila 3 Columna 15 es : 3
Fila 3 Columna 16 es : 2
Fila 3 Columna 17 es : 3
Fila 3 Columna 18 es : 3
Fila 3 Columna 19 es : 3
Fila 3 Columna 20 es : 2
Fila 3 Columna 21 es : 3
Fila 3 Columna 22 es : 3
Fila 3 Columna 23 es : 3
Fila 3 Columna 24 es : 4
Fila 3 Columna 25 es : 4
Fila 3 Columna 26 es : 3
Fila 3 Columna 27 es : 4
Fila 3 Columna 28 es : 4
Fila 3 Columna 29 es : 4
Fila 3 Columna 30 es : 3
Fila 3 Columna 31 es : 4
Fila 3 Columna 32 es : 4
Fila 3 Columna 33 es : 4
Fila 3 Columna 34 es : 5

 Se ha escrito en el archivo de salida
```

Para archivo resultado.txt

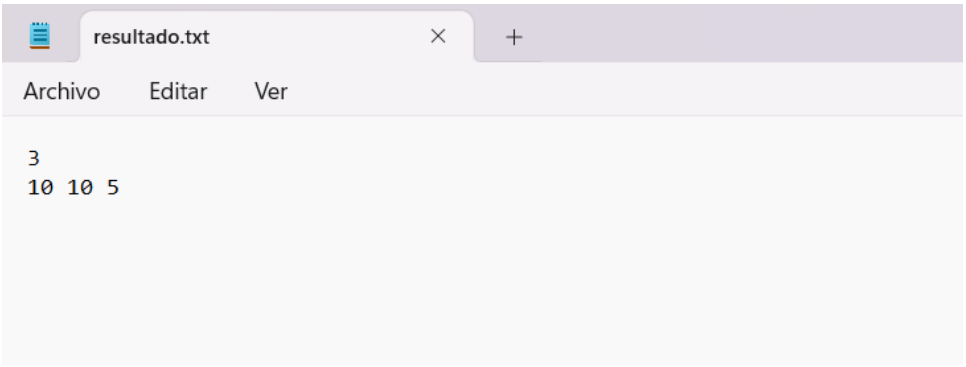


Para archivo prueba.txt

2
5 10
22

```
C:\Users\lausa>java -jar CambioDinamica.jar -t prueba.txt resultado.txt

 Se ha leido el archivo desde entrada
Iniciando el método setTable...
Fila 0 Columna 1 es : 0
Fila 0 Columna 2 es : 0
Fila 0 Columna 3 es : 0
Fila 0 Columna 4 es : 0
Fila 0 Columna 5 es : 1
Fila 0 Columna 6 es : 1
Fila 0 Columna 7 es : 1
Fila 0 Columna 8 es : 1
Fila 0 Columna 9 es : 1
Fila 0 Columna 10 es : 2
Fila 0 Columna 11 es : 2
Fila 0 Columna 12 es : 2
Fila 0 Columna 13 es : 2
Fila 0 Columna 14 es : 2
Fila 0 Columna 15 es : 3
Fila 0 Columna 16 es : 3
Fila 0 Columna 17 es : 3
Fila 0 Columna 18 es : 3
Fila 0 Columna 19 es : 3
Fila 0 Columna 20 es : 4
Fila 0 Columna 21 es : 4
Fila 0 Columna 22 es : 4
Fila 1 Columna 1 es : 0
Fila 1 Columna 2 es : 0
Fila 1 Columna 3 es : 0
Fila 1 Columna 4 es : 0
Fila 1 Columna 5 es : 1
Fila 1 Columna 6 es : 1
Fila 1 Columna 7 es : 1
Fila 1 Columna 8 es : 1
Fila 1 Columna 9 es : 1
Fila 1 Columna 10 es : 1
Fila 1 Columna 11 es : 1
Fila 1 Columna 12 es : 1
Fila 1 Columna 13 es : 1
Fila 1 Columna 14 es : 1
Fila 1 Columna 15 es : 2
Fila 1 Columna 16 es : 2
Fila 1 Columna 17 es : 2
Fila 1 Columna 18 es : 2
Fila 1 Columna 19 es : 2
Fila 1 Columna 20 es : 2
Fila 1 Columna 21 es : 2
Fila 1 Columna 22 es : 2

 Se ha escrito en el archivo de salida
```

Para archivo resultado.txt



3.Un listado del código fuente completo.

Clase principal:

Class CambioDinamica:

```
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Arrays;
import java.util.Scanner;
/**
* Esta clase representa la entrada del programa
* y la ejecución principal del algoritmo de cambio dinámico
* Objetivo: Punto de entrada principal del programa.
* Verificación de Argumentos: Comprueba la cantidad correcta de argumentos y maneja las opciones -t (traza) y -h (ayuda).
* Creación de Instancia de FSalida: Inicializa una instancia de la clase FSalida que gestionará la lógica principal.
* Lectura de Datos de Entrada: Lee el número de tipos de monedas, los valores de las monedas y la cantidad a devolver.
* Resolución del Problema: Llama a métodos de la instancia de FSalida para resolver el problema y mostrar o escribir la solución.
*/
```

```
public class CambioDinamica {
    public static void main(String[] args) {
        // Verificar la cantidad correcta de argumentos
        if (args.length > 4) {
            FSalida.mostrarAyuda();
            return;
        }

        FSalida ficheroSalida;
        boolean trace = false;

        // Verificar la opción de traza
        int startIndex = 0;
        if (args.length > 0 && args[0].equals("-t")) {
            trace = true;
            startIndex = 1;
        }

        // Verificar la opción de ayuda
        if (args.length > startIndex && args[startIndex].equals("-h")) {
            FSalida.mostrarAyuda();
            return;
        }

        // Crear Scanner para leer el archivo de entrada o datos por
pantalla
        Scanner scanner = FEntrada.getScanner(args, startIndex);
        // Leer el número de tipos de monedas
        int n = scanner.nextInt();

        // Leer los valores de las monedas en el conjunto A
        int[] A = new int[n];
```

```
        for (int i = 0; i < n; i++) {
            A[i] = scanner.nextInt();
        }

        // Leer el valor de la cantidad a devolver C
        int C = scanner.nextInt();

        ficheroSalida = new FSalida(n, A, C);
        ficheroSalida.setTable(trace);
        ficheroSalida.getSolution();

        // Nombres de los archivos de salida
        String outputFile = (args.length > (startIndex + 1)) ?
args[startIndex + 1] : null;
        if (outputFile != null) {
            System.out.print("\n Se ha escrito en el archivo de salida
\n");
            ficheroSalida.printSolutionToFile(outputFile);
        } else {
            System.out.print("\n Se va a imprimir la solucion a
continuacion: \n");
            ficheroSalida.printSolution();
        }
    }



}
```

Class FEntrada:

```
import java.util.Scanner;import java.util.Arrays;
import java.io.FileNotFoundException;
import java.io.File;
```

```
/**
* Esta clase representa la entrada de datos y contiene métodos
relacionados con la inicialización de valores
*
* @author Laura Saltaren
*/
public class FEntrada
{
    // Instancia de las variables.
    private int Tmoneda;
    private int[] Vmoneda ;
    private int CantidaDevolver;

    /**
     * Objetivo: Inicializa las variables de entrada con los valores
proporcionados.
     * Parámetros:
     * n: Número de tipos de monedas.
     * A: Valores de los diferentes tipos de monedas.
     * C: Cantidad a devolver.
     */
    public FEntrada(int n, int[]A, int C)
    {
        this.Tmoneda = n;
        this.Vmoneda = new int[Tmoneda];
        this.Vmoneda = A;
        this.CantidaDevolver = C;

    }
    /**
     * Objetivo: Proporciona acceso a los valores almacenados en la
instancia de FEntrada.
     * Return: Devuelven la cantidad a devolver.
```

```
     */
    public int getCantidaDevolver(){
     return CantidaDevolver;
    }
    /**
     * Objetivo: Proporciona acceso a los valores almacenados en la
instancia de FEntrada.
     * Return: Devuelven los valores de las monedas.
     */
    public int [] getVmoneda(){
     return Vmoneda;
    }
    /**
     * Objetivo: Proporciona acceso a los valores almacenados en la
instancia de FEntrada.
     * Return: Devuelven el número de tipos de monedas.
     */
    public int getTmoneda(){
     return Tmoneda;
    }
    /**
     * Objetivo: Devuelve un objeto Scanner para leer datos de entrada
desde un archivo o la entrada estándar.
     * Parámetros:
     * args: Argumentos del programa.
     * startIndex: Índice para comenzar la lectura de argumentos.
     * Return: Un objeto Scanner configurado para leer desde un archivo o
la entrada estándar.
     */
    public static Scanner getScanner(String[] args, int startIndex) {
        Scanner scanner = null;

        if (args.length > startIndex && !args[startIndex].equals("-t")) {
```

```
            // Intenta abrir el archivo si está presente
            try {
                System.out.print("\n Se ha leido el archivo desde entrada
\n");
                String inputFile = args[startIndex];
                scanner = new Scanner(new File(inputFile));
            } catch (FileNotFoundException e) {
                System.err.println("\nError: No se pudo encontrar el
archivo de entrada. Se utilizará la entrada estándar.");
                scanner = new Scanner(System.in);
            }
        }else{
            scanner = new Scanner(System.in);
        }



        return scanner;
    }
}
```

Class FSalida:

```
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Arrays;
import java.util.Scanner;
/**
* Esta clase representa la salida del programa y contiene métodos
relacionados con la lógica principal del algoritmo de cambio dinámico
*
```

```
* @author Laura
*/
public class FSalida
{
    // Se declara el objeto FEntrada
    private FEntrada datosEntrada;
    // Tipo moneda
    private int STmoneda;
    // Cantidad a devolver
    private int SCantidaDevolver;
    // Valor de las diferentes monedas
    private int[] SVmoneda;
    // Se declara  una matriz que sera mi tabla
    private int [][] TablaMonedas;
    // declaracion de vectores
    private int []vectorMonedas;
    private int []vectorSolucion;
    private int []vectorSolucionValue;
    // Entero que sera el numero de monedas final que necesito
    private int AmountCoins;

    // Variable para la opción de traza
    private boolean trace;
    /**
     * Objetivo: Crea una instancia de FEntrada y llama al método
initialize.
     * Parámetros:
     * n: Número de tipos de monedas.
     * A: Valores de los diferentes tipos de monedas.
     * C: Cantidad a devolver.
     */
    public FSalida(int n, int[]A, int C){
```

```
     datosEntrada = new FEntrada(n, A, C);
     initialize();
    }
    /**
     * Objetivo: Inicializa las variables de la clase, configurando la
instancia de FEntrada y otros atributos.
     */
    private void initialize() {
        this.STmoneda = datosEntrada.getTmoneda();
        this.SCantidaDevolver = datosEntrada.getCantidaDevolver();
        this.SVmoneda = datosEntrada.getVmoneda();
        this.vectorMonedas = getVectorMonedas();
        this.vectorSolucion = new int[this.STmoneda];
        this.vectorSolucionValue = new int[10];
        this.TablaMonedas = new int[this.STmoneda][this.SCantidaDevolver +
1];
    }

    public void setTrace(boolean trace) {
        this.trace = trace;
    }

    /**
     * Este metodo coge un vector con x enteros (SVmoneda) de diferentes
valores
     * lo organiza de menor a mayor y lo introduce en orden en otro vector
(vectorMonedas)
     */
    private void setVectorMonedas(){
        Arrays.sort(this.SVmoneda);
        this.vectorMonedas = Arrays.copyOf(this.SVmoneda,
this.SVmoneda.length);
     }
    /**
```

```
     * Este metodo llama al metodo de esta misma clase setVectorMonedas()
     * y devuelve el vector ordenado.
     */
    public int[] getVectorMonedas(){
        setVectorMonedas();
        return this.vectorMonedas;
    }
    /**
     * Objetivo: Rellena una tabla con los valores mínimos de monedas necesarios para alcanzar una cantidad.
     * Parámetros:
     * trace: Indica si se debe imprimir una traza detallada durante la ejecución.
     * Proceso:
     * Utiliza un enfoque dinámico para llenar la tabla con la cantidad mínima de monedas necesarias.
     */
    public void setTable(boolean trace){
        if (trace) {
        System.out.println("Iniciando el método setTable...");
        }
        for(int i = 0; i < this.TablaMonedas.length; i++  ){
        this.TablaMonedas[i][0]=0;
        // System.out.print(" Fila "+i+" Columna 0 es:  " +this.TablaMonedas[i][0]+ "\n");


        }
        for(int i = 0; i < this.TablaMonedas.length; i++  ){
            for(int j=1; j<this.TablaMonedas[0].length; j++){
                if(i==0 && vectorMonedas[i] > j){
                    this.TablaMonedas[i][j]=000;
                }else{
                   if(i == 0){
```

```
                    this.TablaMonedas[i][j]= 1+ this.TablaMonedas[0][j -
this.vectorMonedas[i]];
                    // System.out.print(" Fila "+i+" Columna "+j+" es : "
+this.TablaMonedas[i][j]+"\n");
                   }else{
                    if(j < this.vectorMonedas[i]){
                     this.TablaMonedas[i][j]=this.TablaMonedas[i-1][j];
                     // System.out.print(" Fila "+i+" Columna "+j+" es :
" +this.TablaMonedas[i][j]+"\n");
                    }else{
                     this.TablaMonedas[i][j] =
Math.min(this.TablaMonedas[i-1][j], this.TablaMonedas[i][j -
vectorMonedas[i]]+1);
                     // System.out.print(" Fila "+i+" Columna "+j+" es :
" +this.TablaMonedas[i][j]+"\n");
                    }
                   }
               }
               // Agregar traza si está habilitada
               if (trace) {
               System.out.println("Fila " + i + " Columna " + j + " es :
" + this.TablaMonedas[i][j]);
               }
           }
       }
   }
   /**
    * Objetivo: Calcula la solución a partir de la tabla generada.
    * Proceso:
    * Utiliza la tabla para determinar la combinación óptima de monedas
necesarias para alcanzar la cantidad deseada.
    */
   public void getSolution(){
       this.AmountCoins = 0;
       for(int x = 0; x < this.STmoneda; x++  ){
```

```
        this.vectorSolucion[x]=0;
        // System.out.print("Vector monedas lugar: "+ x +" "+
this.vectorSolucion[x]+"\n");
        }
        int i=this.STmoneda-1;
        int j=this.SCantidaDevolver;
        while(j > 0){
            if(i>0 && this.TablaMonedas[i][j]==this.TablaMonedas[i-1][j]){
                i = i-1;
            }else{
                this.vectorSolucion[i]=this.vectorSolucion[i]+1;
                this.vectorSolucionValue[this.AmountCoins]=
this.vectorMonedas[i];
                this.AmountCoins ++;
                j=j-this.vectorMonedas[i];
            }
        }
    }
    /**
     * Objetivo: Imprime la solución en la consola.
     * Proceso:
     * Imprime la cantidad total de monedas y los valores de las monedas
utilizadas.
     */
    public void printSolution(){
        System.out.print(this.AmountCoins+"\n");
        for(int i=0; i < this.AmountCoins; i++){
            System.out.print(this.vectorSolucionValue[i]+" ");
        }
    }
    /**
     * Objetivo: Escribe la solución en un archivo.
     * Parámetros:
```

```
     * outputFileName: Nombre del archivo de salida.
     */
    public void printSolutionToFile(String outputFileName) {
        try (PrintWriter writer = new PrintWriter(new
FileWriter(outputFileName))) {
            writer.println(this.AmountCoins);
            for (int i = 0; i < this.AmountCoins; i++) {
                writer.print(this.vectorSolucionValue[i] + " ");
            }
        } catch (IOException e) {
            System.err.println("Error al escribir en el archivo de salida:
" + e.getMessage());
        }
    }
    /**
     * Objetivo: Muestra información de ayuda sobre la sintaxis del
programa y las opciones disponibles.
     */
    public static void mostrarAyuda(){
        System.out.println("SINTAXIS:java -jar CambioDinamica.jar [-t][-h]
[fichero entrada] [fichero salida]");
        System.out.println("-t Traza el algoritmo");
        System.out.println("-h Muestra esta ayuda");
        System.out.println("[fichero entrada] Nombre del fichero de
entrada");
        System.out.println("[fichero salida] Nombre del fichero de
salida");
        System.exit(0);
    }
    }
```