Actividad 1 Desarrollar un programa que permita analizar y maximizar la ganancia de compra y venta de acciones utilizando programación dinámica. Contexto: Un inversionista quiere comprar y vender acciones de una empresa con un número máximo de transacciones K. Dado un conjunto de precios históricos de la acción, debe determinar la máxima ganancia posible siguiendo estas reglas: Puede comprar y vender la acción hasta K veces. No puede comprar sin haber vendido antes una acción anterior. Debe encontrar la mejor estrategia para maximizar su ganancia.

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

import java.util.\*;

public class AnalizadorGanancias {

    public static void main(String[] args) {

        String rutaArchivo = "datos.json";

        int maxTransacciones = 2;

        try {

            Map<String, List<Integer>> datos = leerDatosHistoricos(rutaArchivo);

            if (datos.isEmpty()) {

                System.out.println("No hay datos de precios en el archivo.");

                return;

            }

            for (Map.Entry<String, List<Integer>> entrada : datos.entrySet()) {

                String simbolo = entrada.getKey();

                int[] precios = entrada.getValue().stream().mapToInt(i -> i).toArray();

                EstrategiaInversion estrategia = new EstrategiaInversion(maxTransacciones, precios);

                int gananciaMaxima = estrategia.calcularGananciaMaxima();

                System.out.println("La ganancia máxima posible para " + simbolo + " es: " + gananciaMaxima);

            }

        } catch (IOException e) {

            System.err.println("Error al leer el archivo de datos: " + e.getMessage());

        }

    }

    private static Map<String, List<Integer>> leerDatosHistoricos(String rutaArchivo) throws IOException {

        BufferedReader reader = new BufferedReader(new FileReader(rutaArchivo));

        StringBuilder contenido = new StringBuilder();

        String linea;

        while ((linea = reader.readLine()) != null) {

            contenido.append(linea);

        }

        reader.close();

        String json = contenido.toString().replaceAll("[\\[\\]{}]", "").trim();

        String[] registros = json.split("},");

        Map<String, List<Integer>> datos = new HashMap<>();

        for (String registro : registros) {

            String[] campos = registro.replaceAll("[\" ]", "").split(",");

            String simbolo = "";

            int precio = 0;

            for (String campo : campos) {

                String[] claveValor = campo.split(":");

                if (claveValor[0].equals("simbolo")) {

                    simbolo = claveValor[1];

                } else if (claveValor[0].equals("precio")) {

                    precio = Integer.parseInt(claveValor[1].split("\\.")[0]);

                }

            }

            datos.computeIfAbsent(simbolo, k -> new ArrayList<>()).add(precio);

        }

        return datos;

    }

}

class EstrategiaInversion {

    private final int maxTransacciones;

    private final int[] precios;

    public EstrategiaInversion(int maxTransacciones, int[] precios) {

        this.maxTransacciones = maxTransacciones;

        this.precios = precios;

    }

    public int calcularGananciaMaxima() {

        if (precios == null || precios.length == 0 || maxTransacciones == 0) {

            return 0;

        }

        int dias = precios.length;

        int[][] dp = new int[maxTransacciones + 1][dias];

        for (int t = 1; t <= maxTransacciones; t++) {

            int maxDiferencia = -precios[0];

            for (int d = 1; d < dias; d++) {

                dp[t][d] = Math.max(dp[t][d - 1], precios[d] + maxDiferencia);

                maxDiferencia = Math.max(maxDiferencia, dp[t - 1][d] - precios[d]);

            }

        }

        return dp[maxTransacciones][dias - 1];

    }

}

Datos Json

[

    {"simbolo": "AAPL", "dia": 1, "precio": 150.0},

    {"simbolo": "AAPL", "dia": 2, "precio": 155.0},

    {"simbolo": "AAPL", "dia": 3, "precio": 152.0},

    {"simbolo": "AAPL", "dia": 4, "precio": 160.0},

    {"simbolo": "AAPL", "dia": 5, "precio": 158.0},

    {"simbolo": "GOOGL", "dia": 1, "precio": 2800.0},

    {"simbolo": "GOOGL", "dia": 2, "precio": 2850.0},

    {"simbolo": "GOOGL", "dia": 3, "precio": 2820.0},

    {"simbolo": "GOOGL", "dia": 4, "precio": 2900.0},

    {"simbolo": "GOOGL", "dia": 5, "precio": 2875.0},

    {"simbolo": "MSFT", "dia": 1, "precio": 300.0},

    {"simbolo": "MSFT", "dia": 2, "precio": 310.0},

    {"simbolo": "MSFT", "dia": 3, "precio": 305.0},

    {"simbolo": "MSFT", "dia": 4, "precio": 315.0},

    {"simbolo": "MSFT", "dia": 5, "precio": 312.0},

    {"simbolo": "AMZN", "dia": 1, "precio": 3400.0},

    {"simbolo": "AMZN", "dia": 2, "precio": 3450.0},

    {"simbolo": "AMZN", "dia": 3, "precio": 3420.0},

    {"simbolo": "AMZN", "dia": 4, "precio": 3500.0},

    {"simbolo": "AMZN", "dia": 5, "precio": 3475.0}

]