Estructura de Datos 20/21

Práctica 1 - Sobre análisis de algoritmos

***The Maximum Contiguous Subsequence Sum Problem* (MCS)**

Objetivos

- Introducir el análisis de algoritmos de una forma práctica
- Reflexionar sobre la importancia de entender los detalles de los problemas para diseñar algoritmos más o menos eficientes.

Descripción del laboratorio

Para alcanzar los objetivos del laboratorio, abordaremos el problema MCS:

maximum contiguous subsequence sum problem

Given (possibly negative) integers $A_1, A_2, \ldots, A_N$, find (and identify the sequence corresponding to) the maximum value of $\sum_{k=i}^{j} A_k$. The maximum contiguous subsequence sum is zero if all the integers are negative.

El MCS consiste en encontrar, dado una lista de nombres enteros (positivos y negativos), una subsecuencia **contigua** (posición inicial, posición final, máximo) tal que la suma de los números que la componen sea el **máximo** de todas las sumas de las posibles subsecuencias de la lista de nombres enteros dada.

Nota: El inicio y final de la lista no son subsecuencia contiguas (es decir, no es circular). Si la suma es negativa, o la lista de entrada está vacía, entonces el máximo es 0.

Por ejemplo, supongamos la siguiente lista de entrada {-2, **11, -4, 13**, -5, 2}. La solución del problema MCS es {1, 3, 20}.

Subsecuencia	Suma	Máximo	Inicio y final
-2	-2	0	
-2+11	9	9	
-2+11-4	5	5	
-2+11-4+13	18	18	
-2+11-4+13-5	13	13	
-2+11-4+13-5+2	15	15	
11	11	11	
11-4	7	7	
11-4+13	**20**	**20**	**i=1, j=3**
11-4+13-5	15	15	
11-4+13-5+2	17	17	
-4	-4	0	
-4+13	9	9	
-4+13-5	4	4	

-4+13-5+2	6	6	
13	13	13	
13-5	8	8	
13-5+2	10	10	
-5	-5	0	
-5+2	-3	0	
2	2	2	

El problema MCS resulta interesante para el análisis de algoritmos porque admite diferentes soluciones, algunas más eficientes que otras. Un algoritmo, seguramente el primero que podemos plantear, consiste en realizar una búsqueda exhaustiva (en Inglés, *brute force algorithm*), como el siguiente.

```
public static void maxSubsequenceSum_cubic (int [] a){
       int maxSum = 0;
       int thisSum;
       int seqStart = 0;
       int seqEnd = 0;
       int numIte = 0;
       for (int i=0; i < a.length; i++){
           numIte += 1;
           for (int j=i; j < a.length; j++){
               numIte += 1;
               thisSum = 0;
               for (int k = i; k <= j; k++){
                    numIte += 1;
                    thisSum += a[k];
                }
                if (thisSum > maxSum){
                    maxSum = thisSum;
                    seqStart = i;
                    seqEnd = j;
                }
           }
       }
       System.out.println("Solucion de orden cubico");
       System.out.println("Maximo: " + maxSum);
       System.out.println("Inicio: " + seqStart + " Final: " + seqEnd);
       System.out.println("Numero de iteraciones: " + numIte);
}
```

Este algoritmo tiene un tiempo de ejecución $O(N^3)$ – orden cúbico. Para N grandes, el algoritmo se vuelve poco practicable. ¿Podemos encontrar una mejor solución?

Trabajo a realizar

Implementar en Java dos algoritmos que solucionen el problema MCS:

- Un algoritmo será de orden cuadrado, $O(N^2)$
- Un algoritmo será de orden lineal, O(N)

No tenéis que hacer nada a nivel de código máquina o utilizar estructuras más o menos avanzadas – no se han visto en la asignatura de momento. Lo que tenéis que hacer es del estilo del código fuente del enunciado.

Entrega

Fecha: ver tarea correspondiente en el Campus Virtual.

Material: un documento comprimido (ZIP) con el proyecto IntelliJ IDEA, con los siguientes ficheros, en la tarea correspondiente del campus virtual:

- Un fichero Java con la implementación de los dos algoritmos y un menú sencillo para utilizarlos (por ejemplo, "1. Algoritmo orden cúbico. 2. Algoritmo orden cuadrado. 3. Algoritmo orden lineal"). El fichero Java tiene que tener el nombre de los integrantes del grupo. Ver Anexo I para código ejemplo de menú.
- Un documento DOC / RTF / PDF con la explicación de los algoritmos y su análisis de complejidad. Extensión máxima: 1 página. Con el nombre de los integrantes del grupo.

Criterios generales de evaluación

- Práctica no entregada o entregada fuera de plazo = 0
- Entrega parcial <= 3
- Entrega completa, entre 4 y 10. Se valorará el diseño y claridad del código, la funcionalidad, juego de pruebas, y el análisis de los algoritmos (documento bien organizado y sin faltas de ortografía).

Anexo I – Ejemplo de menú por consola

```
public static void showMenu(){
        int opcio = 0;
        Scanner scan = new Scanner(System.in);
        while (opcio != 4){
            System.out.println("Testcase 1");
            System.out.println("Testcase 2");
            System.out.println("Testcase 3");
            System.out.println("4. Sortir");
            System.out.println("La teva opcio");
            opcio = scan.nextInt();
            if (opcio == 1)
                System.out.println("Cridar mètode que toqui");
        }
}
```