Examen de Algorítmica y Complejidad (Plan 2014)

26 de noviembre de 2021

Nº matrícula:_____ **Nombre:** _____

Apellidos: _____

Problema. Dado un array de números enteros, se pide obtener el subconjunto de valores del array con mayor número de elementos cuya suma sea igual a 0. Ejemplo:

0	1	2	3	4	5	6	7	8
-4	**3**	**5**	**-5**	**0**	**1**	**8**	**10**	**-21**

En este ejemplo, el subconjunto de valores del array con mayor número de elementos cuya suma es igual a 0 es el formado por {3, 5, -5, 0, 8, 10, -21} (tiene 7 elementos).

Implementar un algoritmo en Java, basado en el ***esquema*** de ***Selección óptima***[1], que ofrezca esta funcionalidad[2]. El algoritmo deberá tener la siguiente cabecera:

```
boolean[]subcSuma0MaxElem (int[] v)
```

donde *v* es el array de valores. El método deberá devolver un vector de valores *boolean*, con tantos elementos como tenga el vector *v*, con valor *true* en las posiciones de los valores de *v* seleccionados por el algoritmo y *false* en el resto. Si ningún subconjunto de valores cumple la condición, el vector de *boolean* contendrá *false* en todas sus posiciones. Se podrán implementar todos los métodos y clases adicionales que se consideren necesarios.

```
public boolean[] subcSuma0MaxElem(int[] v){
    boolean[] solucion = new boolean[v.length];
    boolean[] mejorSolucion = new boolean[v.length];
    for (int i=0; i<solucion.length; i++) {
        solucion[i] = false;
        mejorSolucion[i] = false;
    }
    int numElem=0;
    Entero mejorNumElem = new Entero(0);
    int acumulado=0, nivel=0;

    subcSuma0MaxElemAux(v, numElem, solucion, mejorNumElem,
                        mejorSolucion, nivel, acumulado);

    return mejorSolucion;
}
```

[1] Desarrollar un algoritmo que no esté basado en la estrategia de Selección Óptima conllevará una puntuación de 0 en el ejercicio.
[2] Desarrollar un algoritmo que genere un árbol de estados inválido para dar solución al problema conllevará una puntuación de 0 en el ejercicio.

```java
public void subcSuma0MaxElemAux(int[] v, int numElem,
       boolean[] solucion, Entero mejorNumElem,
       boolean[] mejorSolucion, int nivel, int acumulado){

    if (nivel==v.length){
        if ((acumulado == 0) && (numElem>mejorNumElem.getValor())){
            mejorNumElem.setValor(numElem);
            for (int i=0; i<mejorSolucion.length; i++)
                mejorSolucion[i]=solucion[i];
        }
    } else{
        for (int c=0; c < 2; c++){
           // c=0 no se añade v[nivel] a la solución
           // c=1 sí se añade v[nivel] a la solución

            // dado que hay números positivos y negativos, siempre
            // es aceptable No añadirlo o añadirlo
            solucion[nivel] = (c==1);
            acumulado = acumulado + (v[nivel]*c);
            numElem = numElem + c;
            nivel = nivel + 1;
            subcSuma0MaxElemAux(v, numElem, solucion, mejorNumElem,
                                mejorSolucion, nivel, acumulado);
            nivel = nivel - 1;
            numElem = numElem - c;
            acumulado = acumulado - (v[nivel]*c);
            solucion[nivel] = false;
        }
    }
}



public class Entero {
    private int valor;
    public Entero(int valor){
        this.valor=valor;
    }

    public int getValor(){
        return this.valor;
    }

    public void setValor(int valor){
        this.valor=valor;
    }
}
```