Examen de Algorítmica y Complejidad (Plan 2014)

25 de enero de 2023

Nº matrícula:_____ **Nombre:** _____

Apellidos: _____

Problema. Dado un conjunto de números enteros positivos se busca encontrar el subconjunto de mayor tamaño cuya suma sea N, otro número entero positivo objetivo. Ejemplo: dado un N=7 y el siguiente array:

0	1	2	3	4	5	6
5	**10**	**3**	**2**	**1**	**2**	**7**

los elementos {3,2,2} suman 7 y el subconjunto tiene tamaño 3.

SE PIDE: Implementar un algoritmo en Java, basado en el ***esquema Voraz***[1], que encuentre una solución a este problema con ***complejidad***[2] ***O(n²)***. El algoritmo deberá tener la siguiente cabecera:

```
ArrayList<Integer> subconjuntoSuma(int[] numeros, int nObjetivo)
```

donde *numeros* es un array que contiene los elementos y *nObjetivo* es el valor suma objetivo (N). El método deberá devolver el subconjunto de elementos seleccionados o *null* si no se encuentra solución. Se podrán implementar todos los métodos y clases adicionales que se consideren necesarios.

```
ArrayList<Integer> subconjuntoSuma(int[] numeros, int nObjetivo){
    ArrayList<Integer> candidatos = new ArrayList<Integer>();
    ArrayList<Integer> complementario = new ArrayList<Integer>();
    // obtenemos la suma de todos los elementos del vector numeros. Además, los
    // añadimos al ArrayList de candidatos
    int objComplementario = 0;
    for(int i=0; i<numeros.length; i++){
        objComplementario = objComplementario + numeros[i];
        candidatos.add(numeros[i]);
    }
    /* a la suma de todos los valores le restamos el valor nObjetivo. De esta
       manera objComplementario contiene la suma del subconjunto complementario al
       que buscamos*/
    objComplementario = objComplementario -nObjetivo;

    /* El problema se puede resolver seleccionando los elementos para el subconjunto
       complementario, que son los que conseguirán que objComplementario llegue a 0*/
    while(objComplementario>0 && !candidatos.isEmpty()){
        // Elegimos el elemento de mayor valor
        int sel = seleccion(candidatos);
        candidatos.remove((Integer) sel);
        // Solo son aceptables aquellos que no se pasen del ObjComplementario
        if(objComplementario >= sel){
            objComplementario = objComplementario - sel;
            // Recordamos los elementos que forman parte del subjunto complementario
            complementario.add((Integer) sel);
        }
    }
```

1 Desarrollar un algoritmo que no esté basado en la estrategia voraz conllevará una puntuación de 0 en el ejercicio.
2 Desarrollar un algoritmo que encuentre una solución en complejidad superior a $O(n^2)$ conllevará un 0 en el ejercicio.

```
    if(objComplementario==0){
        // Construimos la solución eliminando del array original los elementos
        // seleccionados para el subconjunto complementario
         ArrayList<Integer> solucion = new ArrayList<Integer>();
         for(int i=0; i<numeros.length; i++) solucion.add(numeros[i]);
         for(int i=0; i<complementario.size(); i++)
             solucion.remove((Integer) complementario.get(i));
         return solucion;
    } else
         return null;
}

/* Elegimos el candidato de mayor valor*/
int seleccion(ArrayList<Integer> candidatos){
    int mayor = candidatos.get(0);
    for(int i=1;i<candidatos.size();i++)
         if(mayor<candidatos.get(i))
             mayor = candidatos.get(i);
    return mayor;
}
```