Examen de Algorítmica y Complejidad (Plan 2014)

11 de noviembre de 2022

Nº matrícula:_____ **Nombre:** _____

Apellidos: _____

1) **Problema** (**2.5 puntos**). Se dispone de dos contenedores, cada uno capaz de soportar un peso máximo, y un conjunto de productos, cada uno con un peso característico. Se pretende distribuir **todos** los productos entre los dos contenedores de forma que el reparto del peso sea lo más equilibrado posible. Ejemplo: dado un peso máximo de 10 para cada contenedor y el siguiente array de pesos de los productos:

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

el reparto más equilibrado distribuye los objetos {0,1,2,4,6,7} en el contenedor 1 (con un peso total de 10) y {3,5} en el contenedor 2 (con un peso total de 9).

SE PIDE: 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:

```
int[] distribucionCarga2 (int[] pesos, int pMax)
```

donde *pesos* es un array que contiene los pesos de los productos y *pMax* es el peso máximo soportado por cada contenedor. El método deberá devolver un vector de valores *int*[3], con tantos elementos como tenga el vector *pesos*, con valor *1 o 2* según el objeto haya sido asignado al contenedor 1 o al 2 (**ningún producto puede quedar sin asignar**). Se podrán implementar todos los métodos y clases adicionales que se consideren necesarios.

```
int[] distribucionCarga2 (int[] pesos, int pMax){
    int[] distribucion = new int[pesos.length];
    int[] mejorDistribucion = new int[pesos.length];
    int[] pesoAcumulado = {0,0};
    Entero mejorDiferencia= new Entero(Integer.MAX_VALUE);
    for (int i=0; i<distribucion.length;i++){
        distribucion[i]=0;   mejorDistribucion[i]=0;
    }
    distribucionCarga2Aux(pesos, pMax, 0, distribucion, pesoAcumulado,
                          mejorDistribucion, mejorDiferencia);
    return mejorDistribucion;
}
```

[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.
[3] NO será necesario contemplar el escenario de que los objetos no quepan en los dos contenedores.

```
void distribucionCarga2Aux(int[] pesos, int pMax, int nivel, int[] distribucion,
                           int[] pesoAcumulado,
                           int[] mejorDistribucion, Entero mejorDiferencia){
    if (nivel==pesos.length){
        int diferencia = Math.abs(pesoAcumulado[0]-pesoAcumulado[1]);
        if (diferencia < mejorDiferencia.getValor()) {
            mejorDiferencia.setValor(diferencia);
            for (int i = 0; i < distribucion.length; i++)
               mejorDistribucion[i] = distribucion[i];
        }
    } else{
        for (int c=0; c<2; c++){
            // c=0 se asigna pesos[nivel] al contenedor 1;
            // c=1 se asigna al contenedor 2
            if ((pesoAcumulado[c] + pesos[nivel]) <= pMax){
                distribucion[nivel] = c + 1;
                pesoAcumulado[c] = pesoAcumulado[c] + pesos[nivel];
                nivel++;

                distribucionCarga2Aux(pesos, pMax, nivel, distribucion,
                                      pesoAcumulado,
                                      mejorDistribucion, mejorDiferencia);

                nivel--;
                distribucion[nivel] = 0;
                pesoAcumulado[c] = pesoAcumulado[c] - pesos[nivel];

            }
        }
    }
}
```