Examen de Algorítmica y Complejidad (Plan 2014)

10 de noviembre de 2023

Nº matrícula:_____ **Nombre:** _____

Apellidos: _____

Problema (**2.5 puntos**). La estación de esquí Sweet Snowflake acaba de instalar varios teleféricos[1] cuya capacidad en personas (*numP*) y peso máximo (*pMax*) varía dependiendo de la instalación. El gerente de la estación quiere optimizar el uso de los teleféricos, de forma que en cada viaje se consiga que un teleférico transporte exactamente *numP* personas con un peso total exacto de *pMax*. Ejemplo: dado un teleférico con *numP*=4, *pMax*=320 y el siguiente array de pesos de personas que están esperando a subir al teleférico:

0	1	2	3	4	5	6	7	8	9
45	**80**	**52**	**90**	**67**	**60**	**90**	**84**	**33**	**28**

la selección de las personas cuyos pesos están en las posiciones {1,3,5,6} consigue que se trasporten exactamente 4 personas con un peso total de 320 Kg. Sin embargo, dado un teleférico con *numP*=5, *pMax*=350 y el array de pesos anterior, no hay ninguna combinación de 5 personas con un peso total de 350Kg.

SE PIDE: Implementar un algoritmo en Java, basado en el ***esquema*** de ***Backtracking***[2], que ofrezca esta funcionalidad[3]. El algoritmo deberá tener la siguiente cabecera:

```
boolean[] teleferico(int[] pesos, int numP, int pMax)
```

donde *pesos* es un array que contiene los pesos de los personas, *numP* es el número de personas que admite el teleférico y *pMax* es el peso máximo soportado por el teleférico. El método deberá devolver un vector de valores *boolean*, con tantos elementos como tenga el vector *pesos*, con valor *true* en las posiciones de las persona que hayan sido seleccionadas para entrar en el teleférico. Si el algoritmo no encuentra solución devolverá *null*. Se podrán implementar todos los métodos adicionales que se consideren necesarios (las clases Java Booleano y Entero no es necesario implementarlas).

```
boolean[] teleferico(int[] pesos, int numP, int pMax){
    boolean[] solucion = new boolean[pesos.length];
    for (int i=0; i<solucion.length;i++) solucion[i]=false;
    Booleano exito = new Booleano(false);

    teleféricoAux(pesos, numP, pMax, 0, solucion, exito);

    if (exito.getValor())
        return solucion;
    else return null;
}
```

[1] Teleférico: Sistema de transporte consistente en cabinas suspendidas de un cable de tracción, que en las estaciones de esquí se utiliza para trasportar a los esquiadores de manera rápida desde la base de la montaña hasta las pistas de esquí.

[2] Desarrollar un algoritmo que no esté basado en la estrategia de Backtracking conllevará una puntuación de 0 en el ejercicio.

[3] Desarrollar un algoritmo que genere un árbol de estados inválido para dar solución al problema conllevará una puntuación de 0 en el ejercicio.

```
void telefericoAux(int[] pesos, int faltaPersonas, int faltaPeso, int nivel,
                   boolean[] solucion, Booleano exito){
    if (nivel == pesos.length){
        if ((faltaPersonas==0) && (faltaPeso==0))
            exito.setValor(true);
    }
    else{
        int c=0;
        // c=0 No se selecciona pesos[nivel] para entrar en el funicular
        // c=1 Sí se selecciona pesos[nivel] para entrar en el funicular
        while ((c<2) && !exito.getValor()){
            if ((c==0) || (faltaPersonas>0 && pesos[nivel]<=faltaPeso)){
                faltaPersonas = faltaPersonas - c;
                faltaPeso = faltaPeso - (pesos[nivel]*c);
                solucion[nivel] = (c==1);
                nivel++;

                telefericoAux(pesos, faltaPersonas, faltaPeso, nivel,
                              solucion, exito);

                if (!exito.getValor()){
                    nivel--;
                    solucion[nivel]=false;
                    faltaPeso = faltaPeso + (pesos[nivel]*c);
                    faltaPersonas = faltaPersonas + c;
                }
            }
            c++;
        }
    }
}
```