UniversidadeVigo

Escola Superior de

de Enxeñaría

Our ense

Campus de Ourense 32004 Ourense España Tel. 988 387 000 Fax 988 387 001

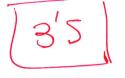
Informática

esei.uvigo.es sdeuig@uvigo.es

AEDII

Prueba Esquemas Algorítmicos

(26/01/2021)



Apellidos: Vila Fonseca Nombre: Sergio

DNI: 53974995K

1.- (3.5 puntos) Implementar el método que se especifica a continuación, partiendo de un valor entero n > 0 queremos obtener una serie de valores enteros positivos $V_1, V_2, ..., V_x$ de tal manera que $\sum V_i = n$ y de forma que $V_1 < V_2 < V_3 ... < V_x$. Es decir, no se pueden repetir valores en la serie, y se comienza añadiendo los valores más pequeños posibles primero. Utiliza una estrategia de **vuelta atrás**. En el anexo está disponible la interface Lista $\leq E$.

Por ejemplo, para n = 11, la serie sería 1,2,3,5. Para n=15, la serie sería 1,2,3,4,5.

public static boolean obtenerSerieValores (int n, Lista<Integer> l)

//Produce: devuelve true si la suma de los valores de la lista es n, falso en otro caso

2.- (3.5 puntos) Implementa el método anterior, pero siguiendo una estrategia voraz. En caso de no encontrar solución devolverá la lista vacía.

public static Lista<Integer> obtenerSerieValores (int n)

//Produce: devuelve una lista de valores enteros diferentes, cuya suma es "n", añadiendo los valores más pequeños posibles primero; si no encuentra solución devuelve la lista vacía

3.- (3 puntos) Sea "v" un array de enteros positivos que se ajusta al perfil de una curva cóncava, es decir, existe una única posición K en el vector tal que:

- a) Los elementos a la izquierda de K están ordenados descendentemente.
- b) Los elementos a la derecha de K están ordenados ascendentemente.

En el ejemplo, K estaría en la posición 6.

100			K							
	9	7	5	4	3	2	1	3	4	7

Implementa el método buscarPosK, que se indica a continuación, de forma eficiente (complejidad $O(\log n)$) para que determine dicha posición K. Puedes considerar que el array está correctamente ordenado. En caso de que el array no se ajuste al perfil de curva cóncava devolverá -1 (esto es, si los elementos están todos ordenados de mayor a menor, o si están todos ordenados de menor a mayor). Implementa este algoritmo siguiendo una estrategia de **divide y vencerás**.

public static int buscarPosK(int[] v, int inicio, int fin)

315

Sin hard



Sergio Vila Fonseca

Universidade de Vigo

Escola Superior de Enxeñería Informática

1-) n = 1 entero, por ejemplo 12 (onjunto candidatos desde el 1 hasta public static boolean obtener Serievalores (int n, Lista xinteger>/) {

boolean objetivo = Salse;
int a = 0;
while (a < n & d lobjetivo) {
 a + +;
 l. Insertar Final (a);
 obtener serie Valores(a+1, 1); X

0

objetivo = true;

else

is (a > n). {

int b = a;

a - -;

1. suprimir (a+1);

1. insertar Final (b);

}

3

No melta

CAMPUS DE OURENSE

```
(2) public static Lista kintegers obtener Serie Valores (int n) }
      int a = 0;
boolean objetivo = Salse;
      while (akn & & ! objetivo) {
        att;
             objetivo = comprobar Suma (a,n)
      is (objetivo) {
     return hacerLista (a);
      else
      return - new Lista cintegers ();
      3
      private boolean comprobar Soma (int i , int j) {
      int b= 0
80r( i ; i = 0; i --){
        b+=1;
      18 (6== 3) {
          return true;
          return Balse;
      private Lista Kinteger > hucerhista (int &) {
      Lista Lintegeral = new Lista (x())
      Sor ( "; 1'50; 1'--) {
           1. insertar Principio (1);
      return 1;
```

Ayuda: Este método sería una adaptación de la búsqueda binaria. Considerando un array de enteros ordenados de menor a mayor, el elemento a buscar y la posición de inicio y fin, el proceso de búsqueda binaria se describe a continuación: se calcula la posición del medio del array. Si el elemento que está en la posición del medio es menor que el elemento a buscar se dirige la búsqueda a la mitad superior (es decir, desde la posición del medio +1 hasta fin). Sin embargo, si el elemento que está en la posición del medio es mayor al elemento buscado, se dirige la búsqueda hacia la mitad inferior (es decir, desde inicio hasta medio -1); en otro caso se ha encontrado el elemento y se devuelve medio. Este proceso se repite hasta que se encuentra el elemento o se llega a la situación de que la posición de inicio es mayor que la de fin, lo cual quiere decir que el elemento no está y se devuelve -1.

Anexo

```
TAD Lista<E>
public interface Iterable<E>{
    public Iterator<E> iterator();
}

public interface Iterator<E>{
    public boolean hasNext();
    public E next() throws NoSuchElementException
}

public interface Lista<E> extends Iterable<E>{
    public boolean es Vacio();
    public int tamaño();
    public boolean contiene (E elemento);
    public void insertarPrincipio (E elemento);
    public void insertarFinal (E elemento);
    public boolean suprimir (E elemento);
    public lteradorLista<E> iteradorLista();
}

public class ListaEnlazada<E> implements Lista<E>{
    ...
}
```