**PRÁCTICAS 1° PARCIAL EN MÁQUINA-EDA**

**TP3**

1) Escriba la versión recursiva del algoritmo de búsqueda secuencial en un arreglo. La búsqueda debe ser tal que:

- devuelva la posición de la primera ocurrencia del elemento buscado en el arreglo

- devuelva-1 si el mismo no está presente. Calcule la complejidad e implemente en Java. Compare con el algoritmo de Búsqueda Binaria dado en clase.

**Resolución**

FUNCIONbusquedaArreglo(A, n,buscado):VECTORxENTERO>=0xENTERO🡪ENTERO>=0

SI n=0 entonces

Retornar -1

SINO

SI(A[n]=buscado) entonces

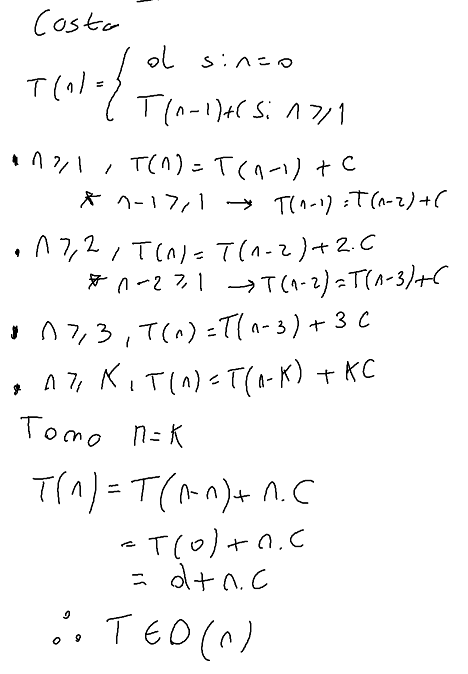
Retornar n

SINO

Retornar busquedaArreglo(A, n-1,buscado)

FIN

-----------VER IMPLEMENTACION-----------



2)

Rediseñar el método mostrar de su clase ListaEnlazada del TP n° 1 para que sea recursiva. Debe conservar la interfaz de la clase.

Calcule la complejidad en notación O grande. Implemente en Java.

**Resolución**

FUNCION mostrar(L):LISTAENLAZADA->Escribir los elementos de los nodos

SI esVacia(L) entonces

escribir(“Lista vacía”)

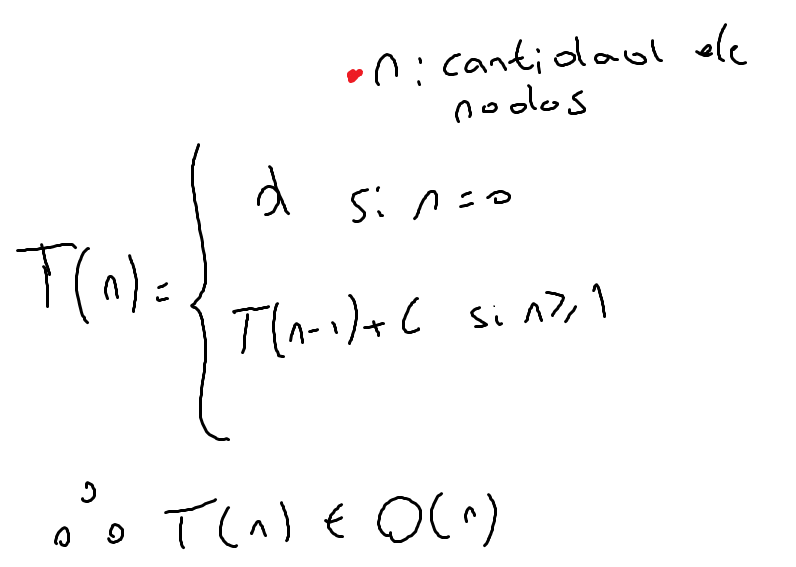
SINO

escribir(primerElemento(L))

mostrar(borrarPrimero(L))

FIN

-----------VER IMPLEMENTACION-----------



3)

Dada una lista enlazada de números enteros, escriba un algoritmo recursivo que calcule la suma de los enteros almacenados en los nodos cuyo valor sea superior a un valor umbral pasado como parámetro.

Calcule la complejidad en notación O grande. Implemente en Java.

FUNCION sumaEnteros(L,valorUmbral):LISTAENLAZADAxENTERO🡪ENTERO

SI esVacia(L) entonces

retornar 0;

SINO

SI primerElemento(L)>valorUmbral

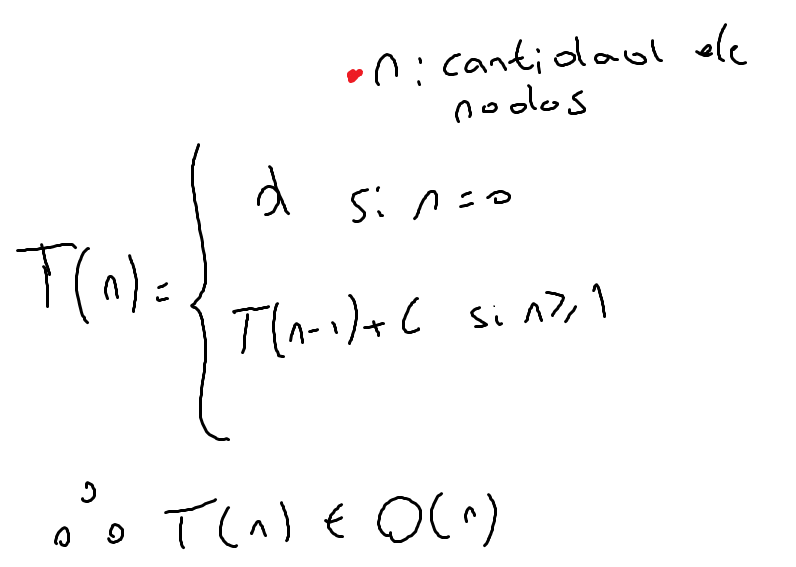
retornar obtenerInformacion(L)+sumaEnteros(borrarPrimero(L),valorUmbral)

SINO

retornar sumaEnteros(borrarPrimero(L),valorUmbral)

FIN

-----------VER IMPLEMENTACION-----------



4)

Diseñe un algoritmo recursivo que dado un arreglo de números enteros devuelva el entero de mayor valor en el arreglo.

FUNCION mayorValor(A,n):VECTORxENTERO>=0🡪ENTERO

SI n=0 entonces

retornar nulo

SINO

ENTERO mayor🡨A[n]

ENTERO anterior🡨mayorValor(A,n-1)

SI mayor>anterior

retornar mayor

SINO

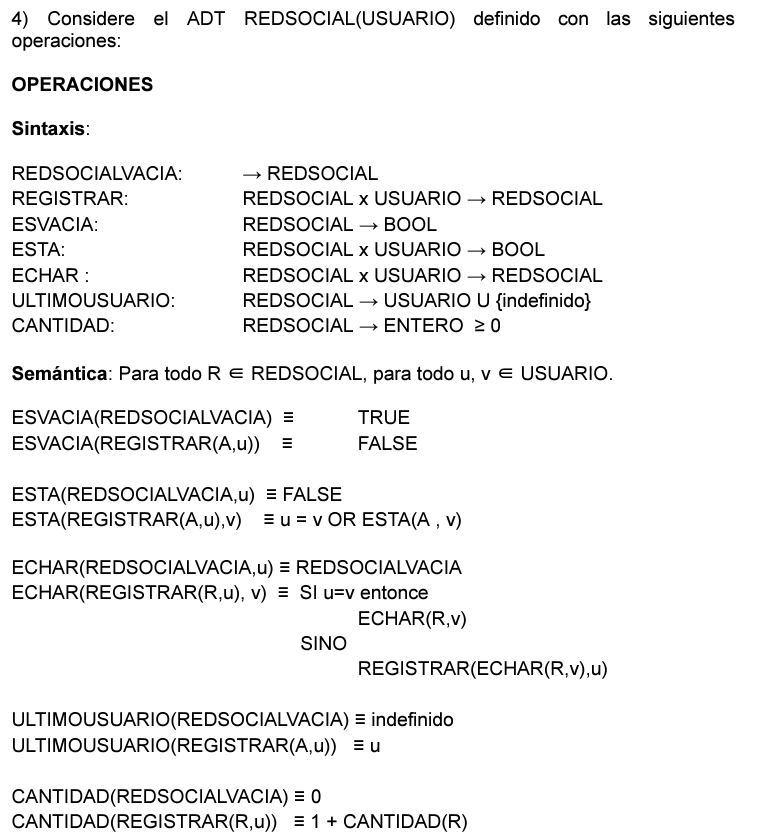
mayor🡨 anterior

retornar mayor

FIN

**TP4**

4)





Dado el ADT REDSOCIAL agregue a la especificación algebraica la siguiente operación:

**esAmigoComun**: recibe 2 redes sociales y un usuario, determinando si el mismo pertenece a ambas. En ese caso debe retornar true.

**Resolución**

**Sintaxis:** ESAMIGOCOMUN:REDSOCIALxREDSOCIALxUSUARIO🡪BOOL

**Semántica:**

Para todo r1,r2 pertenecientes a REDSOCIAL, para todo u1,u2,u3 pertenecientes a USUARIO

ESAMIGOCOMUN(REDSOCIALVACIA,REDSOCIALVACIA,u1)= FALSE

ESAMIGOCOMUN(REGISTRAR(r1,u1),REDSOCIALVACIA,u1)=FALSE

ESAMIGOCOMUN(REDSOCIALVACIA,REGISTRAR(r2,u2),u2)=FALSE

**ESAMIGOCOMUN(REGISTRAR(r1,u1),REGISTRAR(r2,u2),u3)= ESTA(REGISTRAR(r1,u1),u3) AND ESTA(REGISTRA(r2,u2),u3**

1. Implemente en Java el ADT REDSOCIAL(USUARIOS) usando lista simplemente enlazada. USUARIO debe contener los atributos nombre, apellido. La implementación de la operación ESTA debe ser recursiva.

-----------VER IMPLEMENTACION-----------

**TP5**

2) Escriba una implementación en Java del ADT PILA con lista simplemente enlazada. Pruebe todas las operaciones de la Pila.

-----------VER IMPLEMENTACION-----------