Nombre y Apellido:........................................................................ Legajo N°: .........................

# *Examen Final de Estructuras de Datos y Algoritmos (72.34)*

14/12/2016

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Ejercicio 1*** | ***Ejercicio 2*** | ***Ejercicio 3*** | ***Ejercicio 4*** | ***Ejercicio 5*** | *NOTA* |
| **/2** | **/2** | **/2** | **/2** | **/2** |  |

#### *Ejercicio 1*

Se desea almacenar una colección de claves enteras.

Indicar cuál de las siguientes estructuras es más eficiente si se desea realizar búsquedas por rango de claves (por ejemplo todas las que están entre 40 y 50, o entre 100 y 300). Justificar brevemente

1. Linear Hashing
2. Extendible Hashing
3. Ninguna de las anteriores

#### *Ejercicio 2*

#### Se tienen cuatro estructuras distintas que almacenan un conjunto ordenado de valores: arreglo, lista y árbol binario de búsqueda. Indicar el orden de complejidad de cada una de las siguientes operaciones para cada estructura: insertar un nuevo elemento (en el lugar que corresponda según el orden), buscar un elemento por índice, buscar un elemento por valor, y eliminar un valor por índice.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Estructura/Operación | Insertar ordenado | Buscar por índice | Buscar por valor | Eliminar por índice |
| Arreglo |  |  |  |  |
| Lista |  |  |  |  |
| Binary Search Tree |  |  |  |  |

#### *Ejercicio 3*

#### Indique si las siguientes afirmaciones son verdaderas o falsas. Justificar brevemente

1. Una lista lineal doblemente encadenada tiene mayor orden de complejidad espacial que una lista lineal simplemente encadenada.
2. Un árbol B sólo almacena datos en las hojas, los nodos internos se utilizan sólo para claves de búsqueda.
3. *Hill climbing* estocástico permite pasar de un estado V a un estado vecino Vn, aún si la función de evaluación indica que V es mejor solución que Vn.
4. *Simulated Annealing* pasa de un estado V a un estado vecino Vn basándose únicamente en una probabilidad, sin importar si es mejor o peor

#### *Ejercicio 4*

Para cada uno de los siguientes algoritmos, indicar cuál es el peor y el mejor caso (si es que hay):

1. Quicksort (tomando como pivot el primer elemento)
2. Bubblesort (original, sin optimización)
3. Insertion Sort
4. Merge Sort

#### *Ejercicio 5*

Dado el siguiente código, indicar todas las correcciones de funcionamiento, eficiencia, estilo, claridad, etc que deberían hacerse.

/\*\* Lista lineal simplemente encadenada sin orden, que no admite repetidos.

\*\* Cada elemento se inserta al final.

\*/

public class LinearList<T> {

Node first;

Node last;

/\*\* Agrega elemento en la lista, si ya existe no lo agrega. \*/

public void add(T elem) {

Node aux = first;

while ( aux != null && ! aux.head.equals(elem))

aux = aux.tail;

if ( aux == null )

{

last = new Node(elem, null);

if ( first == null )

{

fist = last;

}

}

}

public int size() {

int i = 0;

while ( first != last )

{

i++;

first = first.tail;

}

return i;

}

private class Node {

private T head;

private Node tail;

Node(T head, Node tail) {

this.head = head;

this.tail = tail;

}

}

}