

Algoritmos y Estructuras de Datos.

Recuperatorio Globalizador. Tema 1a. [2 de Julio de 2002]

Ej. 1.- Escribir las funciones primitivas del TAD LISTA con celdas simplemente enlazadas por punteros ó cursores. Es decir, implementar en Pascal los siguientes procedimientos/funciones listadas abajo. Incluir todas las definiciones de tipo necesarias.

- (a) INSERTA(x,p,L),
- (b) LOCALIZA(x,L),
- (c) RECUPERA(p,L),
- (d) SUPRIME(p,L),
- (e) SIGUIENTE(p,L),
- (f) ANULA(L),
- (g) PRIMERO(L), y
- (h) FIN(L).

Ej. 2.- Dos árboles ordenados T y T' se dicen “*semejantes*” si tienen la misma estructura. Formalmente esto significa que

- ambos son vacíos, ó
- ambos son no vacíos y los subárboles de sus hijos correspondientes son semejantes.

Informalmente decimos que T y T' tienen ambos la misma “*forma*”. **Consigna:**
Escriba una función

```
function SEMEJANTE(T,T': nodo) : boolean;
```

que retorna **true** si los árboles son semejantes y **false** caso contrario. Usar las primitivas de árbol ordenado orientado siguientes:

- (a) PADRE(n,A)
- (b) HIJO_MAS_IZQ(n,A)
- (c) HERMANO_DER(n,A)

Ej. 3.- Ejercicios operativos:

- (a) **Árboles:** Dibujar el árbol ordenado orientado cuyos nodos, listados en orden previo y posterior son
 - ORD_PRE = {S, T, A, V, F, X, G, Z, H},
 - ORD_POST = {A, T, F, G, X, H, Z, V, S}.

- (b) **Árboles de Huffman:** Dados los caracteres siguientes con sus correspondientes probabilidades, contruir el código binario y encodar la palabra MUNDIAL
 $P(d) = 0.2, P(i) = 0.25, P(a) = 0.25, P(l) = 0.2,$
 $P(m) = 0.03, P(u) = 0.03, P(n) = 0.02, P(f) = 0.02.$
 Calcular la longitud promedio del código obtenido.
- (c) **Tablas de dispersión:** Insertar los números 9, 22, 32, 15, 14, 42, 24, 11, 34 en una tabla de dispersión cerrada con $B = 10$ cubetas, con función de dispersión $h(x) = x \bmod 10$ y estrategia de redistribución lineal.

Ej. 4.- Preguntas: [Responder según el sistema “multiple choice”, es decir marcar con una cruz el casillero apropiado. **Atención:** Algunas respuestas son intencionalmente “descabelladas” y tienen puntajes **negativos!**]

- (a) El tiempo de inserción y supresión de un elemento para el TAD LISTA implementado con listas simplemente enlazadas por cursores, en cualquier posición dentro de la misma es (n es el número de elementos en la lista.)

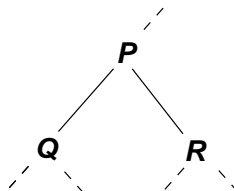
- ☐ $T(n) = O(1)$
☐ $T(n) = O(n)$
☐ $T(n) = O(\sqrt{n})$
☐ $T(n) = O(1/n)$

- (b) El tiempo de ejecución para el algoritmo de clasificación por montículos (“heapsort”) es $O(n \log(n))$ (n es el número de elementos a ordenar) ...

- ☐ ... siempre.
☐ ... a veces.
☐ ... nunca.
☐ ... en el mejor caso.

- (c) Una de las representaciones más eficientes para el TAD COLA DE PRIORIDAD es la implementación por montículos. El montículo es un árbol binario que satisface la condición de ser “parcialmente ordenado”. Si P, Q, R son las etiquetas del nodo y sus dos hijos, la condición de parcialmente ordenado se expresa como (Nota: Consideramos un montículo “minimal”):

- ☐ $Q + R \leq \infty.$
☐ $Q \leq P \leq R.$
☐ $P \leq \frac{1}{2}(Q + R).$
☐ $P \leq \text{le}Q, R.$



- (d) ¿Cuál de los siguientes algoritmos de clasificación es el más rápido en el caso promedio?

- ☐ Clasificación rápida (“Quick-sort”)
☐ Selección

TEMA: 1a

Apellido y Nombre: _____

Carrera: _____ DNI: _____
[Llenar con letra mayúscula de imprenta GRANDE]

Universidad Nacional del Litoral
Facultad de Ingeniería y Ciencias Hídricas
Departamento de Informática
Algoritmos y Estructuras de Datos

- ☐ Burbuja (*“bubble-sort”*)
- ☐ Clasificación telescópica (*“hubble-sort”*)