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

Algoritmos y Estructuras de Datos. Examen Final. [19 de Diciembre 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. `INSERTA(x,p,L)`, `LOCALIZA(x,L)`, `RECUPERA(p,L)`, `SUPRIME(p,L)`, `SIGUIENTE(p,L)`, `ANULA(L)`, `PRIMERO(L)`, y `FIN(L)`.

Ej. 2.- Escribir una función `function INCLUIDO(nb,na: nodo; B,A: arbol) : boolean`, la cual retorna verdadero si la *estructura* del subárbol del nodo `nb` en el árbol ordenado orientado `B` está incluida dentro de la del subárbol del nodo `na` en el árbol `A`, independientemente de las etiquetas de los nodos correspondientes. Definiéndolo en forma recursiva, el subárbol del nodo `nb` está incluido en el de otro nodo `na` si:

- `nb`$=\Lambda$, **o**
- `nb`$<>\Lambda$, `na`$<>\Lambda$ y los subárboles de los hijos de `nb` están incluidos en los subárboles de los hijos de `na`. (Notar la *recursividad* de la definición).

Utilizar las funciones `PADRE (n,A)`, `HIJO_MAS_IZQUIERDO (n,A)`, `HERMANO_DERECHO (n,A)`, y `ETIQUETA (n,A)`.

Ej. 3.- Ejercicios simples sobre TAD's

(a) Escribir una función `function DMINMAX(L:lista):integer;` que, dada una lista `L` retorna la distancia entre las posiciones del mínimo y del máximo de la lista. La distancia debe ser positiva si el mínimo está antes del máximo y negativa en caso contrario. Por ejemplo, si `L`$=\{5,1,3,2,4,7,6\}$ entonces `DMINMAX(L)` debe retornar 4, ya que las posiciones del mínimo y máximo son 2 y 6, respectivamente. Para `L`$=\{5,9,3,2,4,1,6\}$ debe retornar -4. Si el valor del mínimo aparece repetido se debe tomar la primera posición, mientras que para el máximo se debe tomar la última. Por ejemplo, para `L`$=\{7,5,1,3,2,1,4,7,6\}$ entonces debe retornar 5. Usar las primitivas del `TAD LISTA:RECUPERA(p,L)`, `SIGUIENTE(p,L)`, `PRIMERO(L)`, y `FIN(L)`. No utilizar estructuras auxiliares.

(b) Escribir un procedimiento `procedure JUNTA(var P:pila; m:integer);` que reemplaza los primeros `m` elementos de la pila P por su suma. Si hay menos de `m` elementos en P entonces debe reemplazar todos por su suma. Por ejemplo, si `P`$=\{1,3,2,5,4,3,2\}$, entonces `JUNTA(P,4)` debe dejar `P`$=\{11,4,3,2\}$. Si `P`$=\{1,3,2\}$ entonces `JUNTA(P,4)` debe dejar `P`$=\{6\}$. Usar las primitivas del `TAD PILA`: **Pila:** `ANULA(P)`, `METE(x,P)`, `SACA(P)`, `TOPE(P)` y `VACIA(P)`.

Ej. 4.- [LIBRES] Ejercicios operativos:

(a) **Árboles:** Dibujar el árbol ordenado orientado cuyos nodos, listados en orden previo y posterior son
- `ORD_PRE` $=\{Z, Q, R, T, U, V, W, O\}$.
- `ORD_POST` $=\{Q, T, V, O, W, U, R, Z\}$.

(b) **[LIBRES] Clasificación por montículos.** Dados los enteros $\{12, 11, 14, 8, 16, 3, 9\}$ ordenarlos por el método de *"montículos"* (*"heap-sort"*). Mostrar el montículo (minimal) antes y después de cada inserción/supresión.

Ej. 5.- **[LIBRES] 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) Dadas las funciones $T_1(n) = 0.5n + \sqrt{n}$, $T_2(n) = 0.2n^2 + 2.1 \log n$, $T_3(n) = 3n! + 5.4n^3$ y $T_4(n) = n^{1/2}$ decir cuál de los siguientes ordenamientos es el correcto

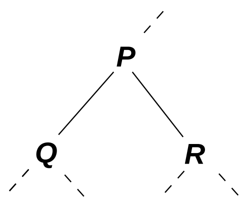
☐ $T_4 < T_3 < T_2 < T_1$
☐ $T_4 < T_1 < T_2 < T_3$
☐ $T_2 < T_1 < T_4 < T_3$
☐ $T_3 < T_4 < T_1 < T_2$

(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) 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"*):

☐ $Q + R \leq \infty$.
☐ $Q \leq P \leq R$.
☐ $P \leq 1/2(Q + R)$.
☐ $P \leq Q, R$.



(d) ¿Cuál es el tiempo de ejecución del procedimiento de clasificación por *incrementos decrecientes* (*shell-sort*) en el caso promedio?

☐ $O(n^{1.3})$
☐ $O(n^{1.5})$
☐ $O(\log n)$
☐ $O(n!)$