

Algoritmos y Estructuras de Datos.

Examen Final. [2 de Octubre de 2003]

Ej. 1.- [Primitivas (20 puntos)] Escribir las funciones del TAD COLA DE PRIORIDAD listadas a continuación, implementado por montículos: ANULA (C), INSERTA (x,C), SUPRIME_MIN (C). *Escribir todos los tipos, definiciones, funciones y procedimientos auxiliares necesarios.*

Ej. 2.- [Ejercicios de programación (total 80 puntos)]

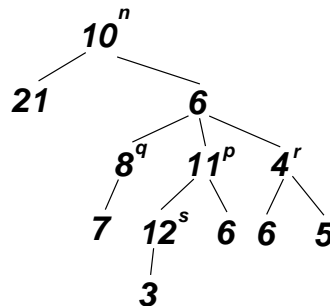
- (a) **[Intercambia secuencia (35 puntos)]** Escribir un procedimiento `procedure INTERCAMBIA_SEC(var L: lista);` que intercambia el grupo de los primeros elementos consecutivos impares por el siguiente grupo de elementos consecutivos pares y así sucesivamente. Por ejemplo si $L=\{1,2,4,5,6,8,7,9,13,2\}$ después de llamar a `INTERCAMBIA_SEC(L)` debe quedar $L=\{2,4,1,6,8,5,2,7,9,13\}$ Restricciones:
- Usar **sólo** una estructura auxiliar. Puede ser una cola o una lista (se sugiere una cola).
 - El algoritmo debe ser $O(n)$
 - Utilizar las primitivas del TAD LISTA: `INSERTA(x,p,L)`, `RECUPERA(p,L)`, `SUPRIME(p,L)`, `SIGUIENTE(p,L)`, `ANULA(L)`, `PRIMERO(L)`, y `FIN(L)`. y del TAD COLA: `ANULA (C)`, `PONE_EN_COLA (x,C)`, `QUITA_DE_COLA (C)`, `VACIA (C)`, y `FRENTE_DE_COLA (C)`.
- (b) **[Encuentra suma (35 puntos)]** Escribir una función `function ENCUESTRA_SUMA(s:real; n:nodo; A:arbol): nodo` que retorna el nodo m del árbol orientado A tal que la suma de las etiquetas de todos los nodos descendientes de m es s . Si no existe un tal nodo, entonces debe retornar Λ . Si hay varios nodos que cumplen la condición debe retornar *el primero en orden previo*. Por ejemplo, para un árbol como el de la figura debe darse

`ENCUESTRA_SUMA(32,n,A)` retorna p

`ENCUESTRA_SUMA(15,n,A)` retorna $\rightarrow q$ (notar que r y s también dan la suma, pero q está primero en orden previo).

`ENCUESTRA_SUMA(15,n,A)` retorna $\rightarrow \Lambda$

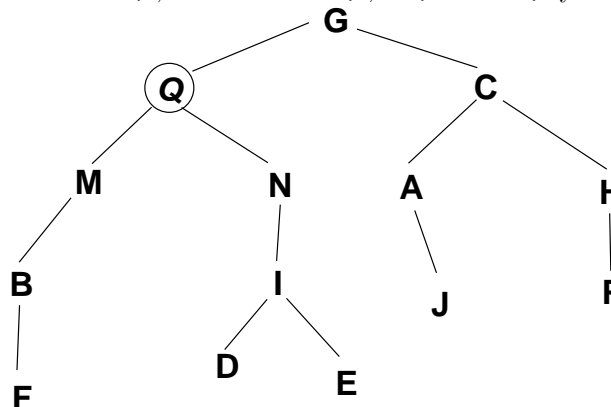
El algoritmo debe ser $O(n)$ donde n es el número de nodos en el árbol. Usar las funciones del TAD ARBOL ORDENADO ORIENTADO: `HIJO_MAS_IZQ(n,A)`, `HERMANO_DER(n,A)` y `ETIQUETA(n,A)`.



- (c) **[Reordena pila (10 puntos)]** Escribir un procedimiento `procedure REORDENA(var P:pila);` que reordena los elementos de una pila de tal forma que quedan los impares en el fondo y los pares arriba. Los pares e impares deben quedar en el mismo orden relativo entre sí. Por ejemplo si $P = \{1, 3, 4, 2, 3, 5, 7, 6, 8, 2, 9\}$ entonces debe quedar $P = \{4, 2, 6, 8, 2, 1, 3, 3, 5, 7, 9\}$. Restricciones:
- Se puede usar sólo dos pilas auxiliares.
 - El algoritmo debe ser $O(n)$
 - Utilizar las funciones del TAD PILA `ANULA(P)`, `METE(x,P)`, `SACA(P)`, `TOPE(P)` y `VACIA(P)`.

Ej. 3.- [LIBRES. Ejercicios operativos (total 80 puntos)] Atención!! Alumnos libres deben completar un mínimo de 70% en cada uno de los ítems

- (a) **[Reconstruir árbol (25 puntos)]** Dibujar el árbol ordenado orientado cuyos nodos, listados en orden previo y posterior son
- $ORD_PRE = \{C, A, R, S, H, Q, T, J, U\}$,
 - $ORD_POST = \{R, H, S, A, J, T, U, Q, C\}$.
- (b) **[Crea (20 puntos)]** Escribir una función `function CREA_ARBOL(A:arbol): nodo` que, usando las funciones del TAD ARBOL ORDENADO ORIENTADO `CREAO(v)`, `CREA1(v, A1)`, ... `CREAn(v, A1, A2, ..., An)` crea el árbol A de la figura más arriba.
- (c) **[Heap-sort (25 puntos)]** Dados los enteros $\{3, 7, 1, 2, 5, 4, 3, 6\}$ ordenarlos por el método de “montículos” (“heap-sort”). Mostrar el montículo (minimal) antes y después de **cada** inserción/supresión.
- (d) **[Particionar árbol (10 puntos)]** Considerando el árbol de la figura, decir cuales son los nodos `DESCENDIENTES(Q)`, `ANTECESORES(Q)`, `IZQUIERDA(Q)` y `DERECHA(Q)`.



Ej. 4.- [LIBRES (20 pts, 5 por pregunta)] 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) ¿Cuál es el tiempo de ejecución para el algoritmo de clasificación por montículos (heap-sort) en el peor caso? (n es el número de elementos a clasificar)

☐ ... $O(n \log n)$

☐ ... $O(1)$

☐ ... $O(n)$

☐ ... $O(n^2)$

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

- (b) ¿Cuál es, en promedio, el número de intentos necesarios para insertar un *nuevo* elemento en una tabla de dispersión cerrada? ($0 < \alpha < 1$ es la “*tasa de llenado*” de la tabla, $\alpha = (\text{ocupados} + \text{suprimidos}) / (\text{total de cubetas})$).

☐ ... $\log \alpha$ ☐ ... α ☐ ... $1/\alpha$ ☐ ... $1/(1 - \alpha)$

- (c) ¿Cuál es el tiempo de ejecución de la función ANTERIOR para listas *simplemente* enlazadas?

☐ ... $O(n^2)$ ☐ ... $O(n)$ ☐ ... $O(\log n)$ ☐ ... $O(1)$

- (d) ¿Cuántos elementos, en promedio, hay en cada cubeta en una tabla de dispersión abierta o cerrada con N elementos y B cubetas?

☐ ... $O(N/B)$ ☐ ... $O(N + B)$ ☐ ... $O(B/N)$ ☐ ... $O((N/B)^2)$