

Algoritmos y Estructuras de Datos. Examen Final. [28 de Julio de 2005]

[Ej. 1] [Clases (30 puntos)]

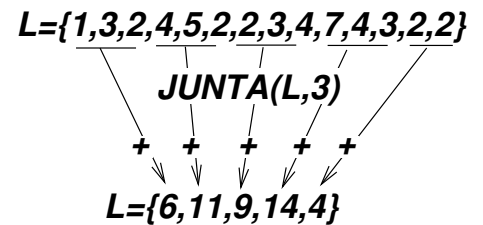
Escribir las primitivas que se indican del TAD `list<>` `insert(x,p)`, `erase(p)`, `retrieve(p)/*p, next()/p++`, `list()` y `~list()`.

[Ej. 2] [Programación (total = 70 puntos)]

a) [junta (25 puntos)]

Escribir una función `void junta(list<int>&L,int n);` que dada una lista `L`, agrupa de `a` `n` elementos dejando su suma (ver figura).

Restricciones: No usar ninguna estructura auxiliar. Prestar atención a no usar posiciones inválidas después de una supresión. El algoritmo debe tener un tiempo de ejecución $O(n)$, donde n es el número de elementos en la lista original.



b) [ord-nodo (25 puntos)]

Escribir una función predicado `bool ordnodo(tree<int> &A);` que verifica si cada secuencia de hermanos del subárbol del nodo `n` (perteneciente al árbol ordenado orientado `A`) están ordenadas entre sí, de izquierda a derecha. Por ejemplo, para el árbol $(3\ 5\ (6\ 1\ 3)\ (7\ 4\ 5))$ debería retornar `true`, mientras que para $(3\ 9\ (6\ 1\ 3)\ (7\ 4\ 2))$ debería retornar `false`, ya que las secuencias de hermanos $(9\ 6\ 7)$ y $(4\ 2)$ NO están ordenados. Se sugiere el siguiente algoritmo: para un dado nodo retornar `false` si: 1) sus hijos no están ordenados o 2) algunos de sus hijos contiene en su subárbol una secuencia de hermanos no ordenada (recursividad).

c) [es-completo (20 puntos)]

Escribir una función predicado `bool es_completo(btree<int> &A);` la cual retorna verdadero si el árbol binario es completo.

[Ej. 3] [operativos (total = 80 puntos)]

a) [rec-arbol (30 ptos)] Dibujar el árbol ordenado orientado cuyos nodos, listados en orden previo y posterior son

- `ORD_PRE = {C, Z, Q, R, A, M, P, K, L, T}`,
- `ORD_POST = {Z, A, P, M, K, L, R, T, Q, C}`.

b) [huffman (30 ptos)] Dados los caracteres siguientes con sus correspondientes probabilidades, construir el código binario y encodar la palabra **VELOCIRAPTOR** $P(R) = 0,05$, $P(V) = 0,2$, $P(E) = 0,1$, $P(L) = 0,05$, $P(O) = 0,2$, $P(T) = 0,15$, $P(C) = 0,1$, $P(P) = 0,05$, $P(I) = 0,05$, $P(A) = 0,05$. Calcular la longitud promedio del código obtenido.

c) [misc-arbol (10pt)]: Dado el árbol $(c\ q\ (t\ (r\ u\ (v\ z))))$,

- 1) Cuál es el nodo que está a la vez a la izquierda de `v` y no es descendiente de `r`?
- 2) Particione el árbol con respecto al nodo `q`, es decir indique cuales son sus antecesores y descendientes propios, derecha e izquierda.

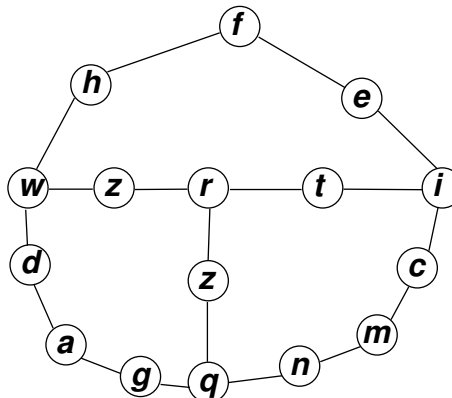
d) [colorear-grafo (10 ptos)]

Apellido y Nombre: _____

Carrera: _____ DNI: _____

[Llenar con letra mayúscula de imprenta GRANDE]

Colorear el siguiente grafo, utilizando una estrategia heurística para tratar de usar el menor número de colores posibles.



[Ej. 4] [Preguntas (total = 20 puntos, 5 puntos 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!!**

Sea el árbol (a b (c d e)). ¿Cuál de las opciones es verdadera?

- ☐ Tiene dos raíces y altura 2.
- ☐ Tiene 3 hojas y altura 3.
- ☐ Tiene 3 hojas y altura 2.
- ☐ Tiene 4 nodos a profundidad 1 y 3 hojas.

¿Cuál es el tiempo de ejecución para la función `find(x)` en conjuntos implementados con árboles binarios de búsqueda?

- ☐ Caso promedio $O(n)$, peor caso $O(n)$.
- ☐ Siempre $O(n \log n)$.
- ☐ Caso promedio $O(n \log n)$, peor caso $O(n^2)$.
- ☐ Caso promedio $O(\log n)$, peor caso $O(n)$.

Sea L una lista conteniendo los elementos (1, 3, 4, 2, 5, 6). Después de aplicar las siguientes líneas

```
list<int>::iterator p,q;  
p = L.begin();  
q = ++p;  
p = L.erase(q);  
p++;  
q = p;  
q++;
```

¿Cuál de las siguientes opciones es verdadera?

- ☐ *p=2, *q=5.
- ☐ *p=2, q es inválido.
- ☐ *p=4, *q=2
- ☐ *p=4, q es inválido.

¿Cuál es el tiempo de ejecución para intersección de conjuntos por vectores de bits? (N es el número de elementos en el conjunto universal, n el número de elementos en el conjunto dado.)

- ☐ $O(N \log N)$
- ☐ $O(n)$
- ☐ $O(N)$
- ☐ $O(n \log n)$