

## Algoritmos y Estructuras de Datos. Examen Final. [2 de Diciembre de 2004]

[Ej. 1] [clases (20 puntos)] Escribir los siguientes métodos del TAD Conjunto, implementado con árboles binarios de búsqueda `set<>`: `erase(n)`, `find(x)`, `insert(x)` (donde `n` es nodo y `x` elemento). Escribir las declaraciones de la clase y los componentes necesarios para implementar las funciones indicadas.

[Ej. 2] [programacion (total = 80 puntos)]

a) [merge-dict (total = 40 puntos)] Implemente una función

```
void merge_map(const map< string, list<int> > &A,
               const map< string, list<int> > &B,
               map< string, list<int> > &C)
```

que a partir de los diccionarios A y B construya un diccionario C de acuerdo a las siguientes reglas:

- Si una clave `key` esta contenida en A o B pero no en ambos, entonces C debe contener dicha clave y su valor asociado debe ser la lista asociada a `key` en A o B pero sin repeticiones.
- Si una clave `key` esta contenida en A y B a la vez, entonces C debe contener dicha clave y su valor asociado debe ser una lista que contenga todos los elementos de las listas asociadas a `key` en A y B, pero sin repeticiones (es decir, su unión como conjunto,  $C[key] \leftarrow A[key] \cup B[key]$ ).

Por ejemplo, dados:

```
A = { 'XX' : [3,3,1,2,2],      B = { 'YY' : [3,3,4,5,8],
      'YY' : [5,5,4]           'ZZ' : [1,1,9]           }
```

se debe obtener:

```
C = { 'XX' : [3,1,2],
      'YY' : [5,4,3,8]
      'ZZ' : [1,9]           }
```

b) [intersect-tree (total = 40 puntos)] Dados dos árboles binarios A y B, escribir una función `void intersect_tree(btree<int>&A, btree<int>&B, btree<int>&C)` que devuelve en C la intersección entre los dos árboles, es decir aquellos nodos que (estructuralmente) están en los dos árboles y que contienen el mismo valor. Por ejemplo, si  $A=(8 \ (3 \ 6 \ (7 \ 10 \ 11)) \ (5 \ 4 \ .))$  y  $B=(8 \ (3 \ 6 \ (9 \ . \ 3)) \ (5 \ 7 \ .))$  entonces  $C=(8 \ (3 \ 6 \ .) \ 5)$ .

[Ej. 3] [operativos (total=80pts) - LIBRES]

- a) [abb (30 pto)] Dados los enteros {6,5,8,2,10,7,3,1,9,15,0} insertarlos, en ese orden, en un “árbol binario de búsqueda”. Mostrar las operaciones necesarias para eliminar los elementos 6, 3 y 7.
- b) [misc-arbol (20pt)]: Dado el árbol binario (z (w (x k .) (y m n)) p),
- 1) Dibuje el árbol correspondiente.
  - 2) De los nodos que están a la izquierda de `y` ¿Cuál es el que está a mayor profundidad?
  - 3) Particione el árbol con respecto al nodo `w`, es decir indique cuales son sus antecesores y descendientes propios, derecha e izquierda.
  - 4) ¿Es completo?
- c) [heap-sort (30 pto)] Dados los enteros {15,14,17,11,19,6,12} ordenarlos por el método de “montículos” (“heap-sort”). Mostrar el montículo (minimal) antes y después de cada inserción/supresión.

[Ej. 4] [Preguntas (total = 20 puntos, 5puntos por pregunta) - LIBRES] 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!!**

Apellido y Nombre: \_\_\_\_\_

Carrera: \_\_\_\_\_ DNI: \_\_\_\_\_

[Llenar con letra mayúscula de imprenta GRANDE]

Sea el árbol (5 7 (8 6 9)). Después de hacer:

`n = D.find(7);`

`n++;`

`n = n.lchild();`

`n = n.lchild();`

`n = D.insert(n,2);`

¿Cuál de las opciones es verdadera?

☐ (5 7 (8 2 6 9))

☐ Da un error.

☐ (5 7 (8 6 2 9))

☐ (5 7 (8 (6 2) 9))

¿Cuál es el tiempo de ejecución de la operación de sort en montículos?

☐  $O(n \log n)$  sólo en el caso promedio.

☐ Siempre  $O(n \log n)$ .

☐  $O(\log n)$  sólo en el caso promedio.

☐ Siempre  $O(\log n)$ .

¿Cuál es el tiempo de ejecución de `splince(p,q)` para árboles?

☐  $\dots O(n)$

☐  $\dots O(1/n)$

☐  $\dots O(1)$

☐  $\dots O(\log_2 n)$

Sea un tabla de dispersión cerrada con  $B$  cubetas y  $n$  elementos. Asumiendo que la función de dispersión es lo suficientemente buena como para distribuir los elementos en forma uniforme entre las cubetas, el costo medio de una inserción exitosa ( $\alpha = n/B$ ) en promedio

☐  $\alpha/(1 - \alpha)$

☐  $1/(1 - \alpha)$

☐  $\log \alpha/(1 - \alpha)$

☐ cte