

## Algoritmos y Estructuras de Datos. 3er Parcial. Tema: 2A. [24 de Junio de 2004]

### [Ej. 1] [Clases (20 puntos)]

Dado el siguiente archivo de cabecera `hash_set.h` para el TAD diccionario con tablas de dispersión abiertas, se pide escribir la implementación de las siguientes funciones del correspondiente archivo `set.cpp`

- `erase(iterator_t p)`
- `clear()`
- `insert(const key_t& x)`
- `find(const key_t& x)`
- `erase(const key_t& x)`

---

```

1  #ifndef HASH_SET_H
2  #define HASH_SET_H
3
4  #include <list>
5  #include <vector>
6  #include <pair.h>
7
8  typedef int key_t;
9  typedef int (*hash_fun)(key_t x);
10
11 class hash_set;
12
13 class iterator_t {
14     friend class hash_set;
15     private:
16         int bucket;           // numero de cubeta
17         std::list<key_t>::iterator p; // posicion en la lista
18         iterator_t(int bucket_a, std::list<key_t>::iterator p_a)
19             : bucket(bucket_a), p(p_a) { }
20     public:
21         iterator_t() { };
22         bool operator==(iterator_t q);
23         bool operator!=(iterator_t q);
24 };
25
26
27 class hash_set {
28     private:
29         int B;                // cantidad de cubetas
30         int count;            // cantidad de elementos en el conjunto
31         hash_fun h;           // puntero a la funcion de hash
32         std::vector< std::list<key_t> > v; // vector de cubetas
33     public:
34         hash_set(const hash_set& s);
35         hash_set(int B_a, hash_fun h_a) : B(B_a), v(B), h(h_a), count(0) { }
36         iterator_t begin();
37         iterator_t end();
38         iterator_t next(iterator_t p);
39         key_t retrieve(iterator_t p);
40         std::pair<iterator_t, bool> insert(const key_t& x);
41         iterator_t find(const key_t& x);
42         int erase(const key_t& x);

```

---

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

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

[Llenar con letra mayúscula de imprenta GRANDE]

```
43 void erase(iterator_t p);  
44 void clear();  
45 int size();  
46 };  
47  
48 #endif /* HASH_SET_H */
```

[Ej. 2] [Programación (total = 50 puntos)]

a) [en-todos (30 puntos)]

Escribir una función predicado `bool en_todos(vector< set<int> > &v)`; que retorna verdadero si existe al menos un elemento que pertenece a todos los conjuntos `v[j]`. Por ejemplo, si

$$v[0] = \{0, 2, 3, 4, 5\}, \quad v[1] = \{0, 1, 5, 7\}, \quad v[2] = \{2, 3, 5, 6, 7\} \quad (1)$$

entonces `en_todos(v)` debe retornar `true` ya que 5 está en los tres conjuntos. Por el contrario, si

$$v[0] = \{0, 2, 3, 4, 5\}, \quad v[1] = \{0, 1, 7\}, \quad v[2] = \{2, 3, 5, 6, 7\} \quad (2)$$

entonces `en_todos(v)` debe retornar `false`.

*Sugerencia:* generar el conjunto que es la intersección de todos los `v[j]` y finalmente verificar si es vacío o no.

b) [mediana (20 puntos)]

Escribir una función `int mediana(list<int> &L)`; que retorna la mediana de los valores contenidos en la lista `L`. Recordemos que la mediana de una serie de  $n$  valores consiste en el valor que queda en la posición  $n/2$  después de ordenarlos. Por ejemplo, si  $L = (3, 2, 4, -1, 0)$  la mediana es 2. Asumir que todos los elementos en  $L$  son distintos.

*Sugerencia:* Insertar los elementos en un conjunto temporario  $A$  y después buscar la posición apropiada, recorriéndolo con un iterator. Recordemos que al iterar sobre un conjunto los elementos aparecen en forma ordenada de menor a mayor.

[Ej. 3] [operativos (total = 20 puntos)]

a) [huffman (8 puntos)]: Dados los caracteres siguientes con sus correspondientes probabilidades, contruir el código binario y encodar la palabra **TEMPLATE**

$$P(T) = 0,3, P(E) = 0,2, P(M) = 0,15, P(P) = 0,15, P(L) = 0,05, P(A) = 0,05, P(W) = 0,05, P(Q) = 0,05$$

Calcular la longitud promedio del código obtenido.

b) [abb (8 ptos)] Dados los enteros  $\{12, 18, 5, 23, 22, 15, 20, 21, 24, 13\}$

insertarlos, en ese orden, en un “árbol binario de búsqueda”. Mostrar las operaciones necesarias para eliminar los elementos 18, 20 y 23 en ese orden.

c) [chash (4 ptos)] Insertar los enteros  $(14, 27, 24, 15, 34, 41, 57, 67, 55, 27)$  en una tabla de dispersión cerrada con  $B=10$  cubetas con función de dispersión  $h(x) = x \% B$  y redispersión lineal. Mostrar como queda la tabla después de realizar las inserciones.

[Ej. 4] [Preguntas (total = 10 puntos, 2.5puntos 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!!**

¿Donde se encuentra el mínimo en un árbol binario de búsqueda?

- ☐ ... En el nodo lo más profundo posible.
- ☐ ... En un nodo ubicado lo más a la izquierda posible.
- ☐ ... En un nodo ubicado lo más a la derecha posible.
- ☐ ... En la raíz

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

---

¿Cuál de los siguientes árboles es un árbol binario de búsqueda?

- ☐ ... (10 (5 . 8) (12 . 9))
- ☐ ... (10 (5 . 8) (12 9 .))
- ☐ ... (10 (5 . 11) 12)
- ☐ ... (10 (5 . 8) 12)

---

¿Cuál es el tiempo de ejecución de **find(x)** en el TAD diccionario por tablas de dispersión abiertas, en el caso promedio?

- ☐ ...  $O(1 + n/B)$
- ☐ ...  $O(1 + B/n)$
- ☐ ...  $\log(n/B)$
- ☐ ...  $O(1 + n^2/B)$

---

¿Cuál es el tiempo de ejecución para **erase(x)** en el TAD conjunto por árbol binario de búsqueda, en el peor caso?

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