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

FICH
UNL

Algoritmos y Estructuras de Datos.
Recuperatorio Parciales. [2019-11-21]

1. **[ATENCIÓN 1]** Para aprobar deben obtener un **puntaje mínimo** del 60 % en las preguntas de teoría y 50 % en clases y operativos.
2. **[ATENCIÓN 2]** Escribir cada ejercicio en **hoja(s) separada(s)**. Es decir todo `CLAS2` en una o más hojas **separadas**, `OPER2` en una o más hojas **separadas**, `PREG2` en una más hojas **separadas**, etc...
3. **[ATENCIÓN 3]** Encabezar las hojas con **sección, Nro de hoja (relativo a la sección), apellido, y nombre, ASI**: `CLAS1, Hoja #2/3 TORVALDS, LINUS`

[Ej. 1] [CLAS1 (W=20pt)]

a) Dado el siguiente fragmento de código:

```
1 tree<char> T;
2 lisp2tree("(m (j k) r t))", T);
3 tree<char>::iterator p = T.begin().lchild().right().right();
```

Presente un croquis de todas las estructuras **`tree<T>::cell`** e **`tree<T>::iterator`** presentes en memoria luego de la ejecución y los punteros que las relacionan.

b) Demuestre a partir de un ejemplo porqué no es posible utilizar un puntero a celda **`cell*`** como iterador de un AOO.

c) Implemente los métodos **`iterator_t erase(iterator_t p)`** y **`iterator_t erase(iterator_t p, iterator_t q)`** de la clase lista doblemente enlazada.

[Ej. 2] [OPER1 (W=20pt)]

a) **[rec-arbol]**: Dibujar el **`AOO`** cuyos nodos, listados en orden previo y posterior son

```
1         ORD-PRE = (A B F G H C D E J),
2         ORD-POST = (G H F B C J E D A),
```

luego particionar el conjunto de nodos respecto al nodo **`C`**.

b) **[operaciones]**: Sea el árbol **`G=(5 (3 4 6) (2 7) 1 (9 8 10))`**. Después de hacer:

```
1         auto n = G.find(1);
2         n++;
3         n = n.lchild();
4         n = n.lchild();
5         n = D.insert(n,11);
```

¿Cómo queda el árbol? ¿Se produce un error?

c) **[Notación $O(\cdot)$]**. Para cada una de las funciones $T_1, \ldots, T_5$ determinar su velocidad de crecimiento (expresarlo con la notación $O(\cdot)$) y ordenarlas de forma creciente.

$$T_1 = n^3\,(2n^2 + 5\,n^3) + 2\log_{10} n + \sqrt[2]{n}$$
$$T_2 = 4n! + 2^n + 5 \cdot 2^n$$
$$T_3 = 7 \cdot 4^n + 4^4 + 2n^4$$
$$T_4 = 4.2 \log n + 5^n + 10n!$$
$$T_5 = 40^9 + 8 \log n + \log(50) n^2$$

d) **[size]**: Suponga que cierto procesador tiene una frecuencia de reloj de 1 Ghz, y simplifique el análisis suponiendo que por cada diez ciclos de reloj se realiza una operación en particular. Estime de qué tamaño puede ser el problema que se puede resolver en diez segundos usando un algoritmo que requiere $T(n)$ operaciones, con los siguientes valores de $T(n)$: $log(n)$, n y $2n$.

e) **[orden]**: Dada la siguiente función:

```
1     set<pair<int,int>> f(set<int>& A,set<int>& B){
2         set<pair<int,int>> C;
3         for(int a:A)
4            for(int b:B)
5                C.insert(pair(a,b));
6        return C;
7     }
```

- Renombrar la función con un nombre alusivo a la tarea que efectúa.
- Determinar el orden algorítmico de **f** respecto al tamaño de los conjuntos **A** y **B**.

[Ej. 3] [PREG1 (W=20pt)]

a) ¿Cuantas posiciones **no dereferenciables** puede haber en una **lista**? ¿Y en un **árbol**?

b) Supongamos que tenemos un árbol (AOO) `T=(6 (0 1 3) 2 7)` y un iterator `p` apuntando al 2. ¿Cómo queda `T` si hacemos una inserción: `T.insert(p,9);`

c) En una **correspondencia**: ¿puede haber una **misma clave** con **diferentes valores**? Por ej. `M=(5->8, 5->9)` ¿Puede haber **diferentes claves** con el **mismo valor**? Por ej. `M=(4->3, 8->3)`

d) Sea el árbol `(r (a (p t u)) (b q z))`. Cuáles de los siguientes son **caminos**?

- `(p a r)`
- `(r a p)`
- `(a r b z)`
- `(a p u)`

e) ¿Cuál es el **tiempo de ejecución** (mejor/promedio/peor) de las siguientes funciones? (asumir listas implementadas con **celdas enlazadas por punteros o cursores**)

1) `list<T>::begin(),`
2) `list<T>::insert(p,x),`
3) `list<T>::clear(p,x),`
4) `map<K,V>::find(x),` para la implementación con vectores ordenados,
5) `map<K,V>::find(x),` para la implementación con listas ordenadas.

f) ¿Como se define la **notación asintótica** $T(n) = O(f(n))$? ¿Porqué decimos que $(n+1)^2 = O(n^2)$ si siempre es $(n+1)^2 > n^2$?

[Ej. 4] [CLAS2 (W=20pt)]

a) Implementar una función
`bool closedhashtable_insert(vector<T>& table, unsigned int (*hashfunc)(T), T x)`
que inserta el elemento `x` en la tabla de dispersión cerrada `table` utilizando la función de dispersión `hashfunc` y redispersión lineal, retornando un booleano indicando si la inserción fue o no exitosa.

b) Implemente el método `iterator_t splice(iterator_t to, iterator_t from)` de la clase árbol binario.

c) ¿Cómo se define un `iterator_t` de la estructura conjunto (`set<T>`) implementado a partir de un árbol binario de búsqueda?

[Ej. 5] [OPER2 (W=20pt)]

a) **[hf-decode]** Utilizando el código Lisp
`(. (. E O) (. (. (. (. K M) L) (. W (. F S))) (. B U)))`
desencodar el mensaje: `11000011010111100110110`.

b) **[huffman]**: Dados los caracteres siguientes con sus correspondientes probabilidades, construir el código binario utilizando el algoritmo de Huffman y encodar la palabra **PARALELISMO**, $P(P) = 0.01, P(A) = 0.05, P(R) = 0.04, P(M) = 0.10, P(U) = 0.20, P(O) = 0.20, P(E) = 0.05, P(S) = 0.20, P(I) = 0.05, P(L) = 0.10$. Calcular la longitud promedio del código obtenido.

c) **[hash]**: Insertar los enteros **{2, 7, 5, 12, 7, 33, 9}** en una tabla de dispersión cerrada con B=9 cubetas, con función de dispersión **h(x) = x%B** y redispersión linear. Luego eliminar el elemento **2** e insertar el elemento **38**, en ese orden. Mostrar la tabla resultante.

d) **[heap-sort]**: Dados los enteros **(3,12,10,4,5,15,7)** ordenarlos por el método de **montículos** (**heap-sort**). Mostrar el montículo (minimal) antes y después de cada inserción/supresión.

[Ej. 6] [PREG2 (W=20pt)]

a) Explique el concepto de estrategia **por búsqueda exhaustiva**. Si le piden colorear un grafo de 200 vértices, ¿utilizaría una estrategia exhaustiva para resolver el problema? Justifique.

b) Si tenemos un iterator **q** dereferenciable en un **set<>** ¿es posible asignarle un valor? Es decir, dado el siguiente código, ¿puede falla línea **(1)**? ¿porqué?

```
set<string> S;
//... llena S
auto q = S.begin();
// (Asumir que q es dereferenciable)
*q = "Spiderman"; // (1)
```

c) Escriba las instrucciones necesarias para crear el siguiente **árbol binario (AB)**: **T=(3 . (4 2 .))**

d) Explique que operaciones realizan (semántica) las dos versiones de **erase** para conjuntos.

```
1  void erase(iterator p);   // (1)
2  int erase(key_t x);       // (2)
```

Explicar qué son los valores de retorno, y porqué una retorna un entero y la otra no. ¿Cuando puede fallar la versión de iterator (1)? ¿Puede fallar la versión (2) si la clave **x** no tiene asignación?

e) Discuta el número de **intercambios** que realizan los algoritmos de ordenamiento lentos en el peor caso.

f) ¿Porqué el árbol binario de una códificación binaria para compresión debe ser un **árbol binario lleno (FBT)**? (Recordemos que un FBT es áquel cuyos hijos son hojas o nodos interiores con sus dos hijos.)

mstorti@galileo/tpl2-2019-88-gff840e36/Sun Nov 24 18:14:16 2019 -0300