Algoritmos y Estructuras de Datos
Segundo Parcial – Viernes 28 de Junio de 2024

LU	Apellido y Nombre	#Hojas	E1	E2	E3	E4	Nota	Corrigió

- Es posible tener una hoja (2 carillas), escrita a mano, con los anotaciones que se deseen, más los apuntes del campus
- Cada ejercicio debe entregarse en **hojas separadas**
- Incluir en cada hoja nombre y apellido, número de libreta y número de hoja
- El parcial se aprueba con 70 puntos

E1. Complejidad [20 pts]

Responder, justificando adecuadamente su respuesta:

a) Decimos que la complejidad de insertar en una lista enlazada ordenada es $O(n)$ en el peor caso, pero eso sólo es cierto si podemos comparar dos elementos en $O(1)$ y lo mismo para la operación de copia. Calcule la complejidad de insertar en una lista enlazada ordenada *sin aliasing* (es decir, copiando en la lista el elemento a insertar y no haciendo una referencia al mismo) si comparar es $O(\log n)$ y copiar es $O(n)$.

b) Si la complejidad en el peor caso de un algoritmo es $\Omega(n)$, ¿es verdad que la complejidad de mejor caso no puede ser $O(1)$?

c) Indique si la siguiente expresión es verdadera o falsa y justifique: Si $O(f(n)) \cap \Omega(g(n)) = \emptyset$, entonces $O(g(n)) \cap \Omega(f(n)) = \emptyset$.

E2. Elección de estructuras e invariante de representación [30 pts]

Una relación parcial es una correspondencia entre dos conjuntos en la cual a cada elemento del primer conjunto le corresponde cero o más elementos del segundo conjunto. Especificamos una relación parcial con el siguiente TAD:

```
TAD Relacion<T1,T2> {
     obs elems: conj<tupla<T1, T2>>

     proc relacionar(inout r: Relacion<T1, T2>, in t1: T1, in t2: T2)
          requiere {r = R_0}
          asegura {r = R_0 ∪ {< t1, t2 >}}

     proc relacionadosDer(in r: Relacion<T1, T2>, in t1: T1): Conjunto<T2>
          asegura {(∀ t2 : T2)(< t1, t2> ∈ r.elems ↔ t2 ∈ res)}

     proc relacionadosIzq(in r: Relacion<T1, T2>, in t2: T2): Conjunto<T1>
          asegura {(∀ t1 : T1)(< t1, t2> ∈ r.elems ↔ t1 ∈ res)}
}
```

Queremos las siguientes complejidades para las operaciones:

- relacionar debe ser $O(\log n + \log m)$.
- relacionadosIzq debe ser $O(\log m)$.
- relacionadosDer debe ser $O(\log n)$.

donde n es la cantidad de elementos de $T1$ que tienen al menos un elemento de $T2$ relacionado, y m es la cantidad de elementos de $T2$ que tienen al menos un elemento de $T1$ relacionado.

a) Proponga una estructura para implementar este TAD cumpliendo esas complejidades.

b) Se quiere agregar la operación `másRelacionadoIzq` que dada una relación devuelve el `T1` que tiene más `T2` relacionados (en caso de haber más de un `T1` con la cantidad máxima de relacionados, se puede devolver cualquiera de ellos). Dicha operación debe tener complejidad $O(1)$. Explique cómo modificaría la estructura propuesta en el punto anterior para también resolver este nuevo problema.

c) Escribir en castellano pero en forma precisa el invariante de representación de la estructura (incluyendo los cambios realizados en el punto b)

d) Escribir en castellano, también en forma precisa, la función de abstracción para dicha estructura.

E3. Estructuras [20 pts]

¿Qué estructura elegiría para implementar cada uno de los siguientes conjuntos? Justifique su respuesta indicando qué ventajas tiene la estructura que eligió.

a) un conjunto de a lo sumo 100 números de 64 bits (entre 1 y 18446744073709551615)
b) un conjunto de mediciones en el que se agregan nuevas mediciones muy frecuentemente y se consulta muy poco
c) un conjunto de palabras sobre un alfabeto no acotado en el que se inserta poco y se consulta mucho

E4. Sorting [30 pts]

Dado un arreglo de pares <estudiante, nota>, queremos devolver un arreglo de pares <estudiante, promedio>, ordenado en forma descendente por promedio y, en caso de empate, por número de libreta.

Los estudiantes se representan con un número de libreta (puede considerar que es un número de a lo sumo 10 dígitos), y la nota es un número entre 1 y 100. La cantidad de estudiantes se considera no acotada.

Se pide dar un algoritmo que resuelva el problema con complejidad $O(n + m \log m)$, donde n es la cantidad total de notas (es decir, el tamaño del arreglo de entrada), y m es la cantidad total de estudiantes.

Ejemplo:

```
Entrada: <12395, 72>, <45615, 81>, <12395, 94>, <45615, 100>, <78920, 81>,
         <12395, 90>, <45615, 71>, <78920, 50>
Salida: <12395, 85.3333>, <45615, 84>, <78920, 65.5>
```

a) Describa cada etapa del algoritmo con sus palabras, y justifique por qué cumple con las complejidades pedidas.
b) Escriba el algoritmo en pseudocódigo. Puede utilizar (sin reescribir) todos los algoritmos y estructuras de datos vistos en clase.