Algoritmos y Estructuras de Datos II

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Examen Final

23 de febrero 2023

Integrante	LU	Correo electrónico
Ezequiel Rueda Sanchez	522/16	ezequiel.ruedasanchez@gmail.com

Reservado para la cátedra

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

1. Ejercicio 1

Responder Verdadero o Falso. Justificar.

a) Lo que hace que una operación sea un observador básico es que deba escribirse en base a los generadores. **Falso**

Lo que hace que una operación sea un observador básico es que es una funcion que permite ver detalles particulares de las instancias.

b) Sí una operación rompe la congruencia debe ser transformada en observador básico. **Falso**

La afirmacion es falsa porque transformarla en observador basico es una opcion. La otra opcion puede ser eliminarla de las operaciones y se evita romper la congruencia.

c) Dos instancias del mismo TAD pueden ser observacionalmente iguales y aun así ser distinguibles por una operación. **Verdadero**

Esta afirmacion es verdadera y la justifico con el siguiente ejemplo. Supongamos que tenemos el TAD NEGOCIO y el observador basico `totalVendido: negocio` $\rightarrow$ `nat` y otra operacion `ventas: negocio` $\rightarrow$ `secu(producto)`.

En este caso, dados dos negocios n_1 y n_2, la igualdad observacional nos dice que `totalVendido(`n_1`)` $=_{\text{obs}}$ `totalVendido(`n_2`)`.

Pero, podemos obtener que `ventas(`n_1`)` $\equiv < jeans, campera, remera >$ y `ventas(`n_2`)` $\equiv < bermuda, campera, camisa >$ donde el valor de las bermudas y jeans son iguales y el valor de las remeras y camisas tambien obtiendo el mismo total vendido pero las ventas son distintas, lo cual produce una inconsistencia.

d) Sí un enunciado dice "Siempre que vale A sucede inmediatamente B y B no puede suceder de ninguna otra maneraz la correspondiente axiomatización incluye las operaciones A y B entonces el TAD está mal escrito. **No lo se**

2. Ejercicio 2

Responder Verdadero o Falso. Justificar.

1) La precondición y la postcondición de las operaciones de la interfaz tiene que cumplir el invariante de representación. **Verdadero**

Esta afirmacion es verdadera porque el invariante de representacion es una funcion booleana que indica que condiciones en la estructura de representacion son validas. La estructura de representacion se va a utilizar para la implementacion de los algoritmos y por lo tanto, estos deben cumplir la precondicion y satisfacer la postcondicion.

2) La complejidad de las operaciones determina el invariante de representación. **Falso**

La complejidad de las operaciones determina la estructura de representacion que se va a elegir para la implementacion de los algoritmos.

3) El invariante determina las complejidades de las operaciones. **Falso**

La estrucutura de representacion elegida va a determinar la complejidad de las operaciones.

3. Ejercicio 3

Indique y justifique si son verdaderas o falsas las siguientes afirmaciones:

a) El análisis "del potencial" y el "del banquero" son técnicas para demostrar la *complejidad promedio de una operación de una estructura de datos. **NO SE**

b) La complejidad del caso promedio de una operación es siempre es menor o igual a la complejidad del peor caso. **Falso**

Vamos a probarlo con el siguiente ejemplo. Supongamos que tenemos una operacion $T(n)$ cuyo algoritmo es implementado recursivamente y se resuelve con la siguiente ecuacion de recurrencia:

$T(n) = aT(\frac{n}{c}) + f(n)$ con $n > 1$

Luego, en el peor caso obtenemos que $a = 2$, $c = 2$ y $f(n) = log(n)$. Entonces, por el Teorema Maestro, $f(n) \in O(n^{log_c(a)-\epsilon})$ para $\epsilon > 0$, es decir, $log(n) \in O(n^{log_2(2)-\epsilon}) = log(n) \in O(n)$ y por lo tanto, $T(n) = \Theta(n^{log_c(a)})$, o sea, $T(n) = \Theta(n)$.

Ahora,en el caso promedio obtenemos que $a = 2$, $c = 2$ y $f(n) = n$. Entonces, por el Teorema Maestro, $f(n) \in \Theta(n^{log_c(a)})$ para $\epsilon > 0$, es decir, $n \in \Theta(n^{log_2(2)-\epsilon}) = n \in \Theta(n)$ y por lo tanto, $T(n) = \Theta(log(n)\ n^{log_c(a)})$, o sea, $T(n) = \Theta(n\ log(n))$.

Como $\Theta(n) < \Theta(n\ log(n))$, concluimos que no siempre se cumple que la complejidad del caso promedio de una operación es siempre es menor o igual a la complejidad del peor caso.

c) Las Skip lists son estructuras de datos que tienen un peor caso de inserción y búsqueda igual a los AVLs. **Verdadero**

Esta afirmacion es verdadera porque las busquedas e insercion en un AVL tiene costo $O(log(n))$ donde n representa la cantidad de elementos del arbol. Ahora, para realizar cualquier busqueda o insercion en una skip list de n elementos, a lo sumo vamos a tener que visitar un maximo de 2 nodos por nivel y como tenemos $log_2(n)$ niveles, hay que visitar $2log_2(n)$ nodos pero se puede acotar por $O(log(n))$ que coincide con la complejidad de los AVL's.

d) La operación de inserción en el método de acceso secuencial indexado tiene un peor caso $O(log(n))$. **Falso**

La operación de inserción en el método de acceso secuencial indexado tiene un peor caso $O(n)$ con n el tamaño del arreglo dado que una nueva insercion puede implicar una reorganizacion completa de las paginas de los discos

4. Ejercicio 4

Decidir si las siguientes afirmaciones son Verdaderas o Falsas. Justificar o dar un contraejemplo.

1) Si f es $O(g)$ y g es $\Omega(f)$, f es $\Theta(g)$. **Falso**

Lo probamos con un contraejemplo. Supongamos que $f = n$ y $g = n^2$. Luego, $n \in O(n^2)$ y $n^2 \in \Omega(n)$ pero $n \in \Theta(n^2)$ es absurdo.

2) Si f es $O(n)$, entonces para cualquier entrada f es $\Omega(n)$. **Falso**

Lo probamos con un contraejemplo. Supongamos que $f = log(n)$. Entonces, $log(n) \in O(n)$ pero $log(n) \in \Omega(n)$ es absurdo.

3) Si f es $\Omega(n)$, entonces para cualquier entrada f es $\Theta(n)$. **Falso**

Lo probamos con un contraejemplo. Supongamos que $f = n^2$. Entonces, $n^2 \in \Omega(n)$ pero $n^2 \in \Theta(n)$ es absurdo.

4) La complejidad del mejor caso de un algoritmo para un cierto problema es menor que cualquier limite inferior para el problema. **Falso**

Supongamos el caso de `insertionSort`. Sabemos que la complejidad en el mejor caso es $O(n)$. Ahora, un limite inferior para este problema podria ser $log(n)$ y sabemos que $n > log(n)$ lo cual prueba que la afirmacion del enunciado no es verdadera.

5) Sea S arreglo de claves representado por max-heap. Sean $S[i]$ y $S[j]$ claves del heap tal que $i < j$ y $S[i] < S[j] \rightarrow$ el arreglo obtenido intercambiar de $S[i]$ y $S[j]$ sigue siendo max-heap. **Falso**

Supongamos que $S = [89, 67, 84, 66, 65, 82, 83, 1, 43, 21, 5, 79, 70]$. Ahora, tomo $i = 9$ y $j = 12$ con $S[i] = 21 < S[j] = 70$.

Al realizar el intercambio propuesto, obtenemos $S = [89, 67, 84, 66, 65, 82, 83, 1, 43, 70, 5, 79, 21]$ y no sigue siendo un max-heap porque el nodo 65 ahora tiene como hijo al nodo 70 y como $70 > 65$, se rompe la condicion de max-heap.

6) Sea S arreglo de claves representado por max-heap. Sean $S[i]$ y $S[j]$ claves del heap tal que $i < j$ y $S[i] > S[j] \rightarrow$ el arreglo obtenido intercambiar de $S[i]$ y $S[j]$ sigue siendo max-heap. **Falso**

Supongamos que $S = [89, 67, 84]$. Ahora, tomo $i = 0$ y $j = 1$ con $S[i] = 89 > S[j] = 67$.

Al realizar el intercambio propuesto, obtenemos $S = [67, 89, 84]$ y no sigue siendo un max-heap porque el nodo 67 ahora tiene como hijo al nodo 89 y como $89 > 67$, se rompe la condicion de max-heap.