Estructuras de Datos 1 - ST0245
Examen Parcial 1 - Jueves (033)

Nombre .....
Departamento de Informática y Sistemas
Universidad EAFIT

Septiembre 12 de 2019

En las preguntas de selección múltiple, una respuesta incorrecta tendrá una deducción de 0.1 puntos en la nota final. Si dejas la pregunta sin responder, la nota será de 0.1. Si no conoces la respuesta, no adivines.

Recursión 20%

1.1 En la vida real, los palíndromes se utilizan para desarrollar algoritmos de compresión de cadenas de ADN. Para este parcial, considera un algoritmo capaz de decir si una cadena de caracteres es un palíndrome o no. Un palíndrome es una cadena que se lee igual de izquierda a derecha que de derecha a izquierda. A continuación algunos ejemplos:

- Para "amor a roma", la respuesta es `true`
- Para "mamita", la respuesta es `false`
- Para "cocoococ", la respuesta es `true`

El algoritmo `isPal` soluciona el problema, pero le faltan unas líneas. Complétalas, por favor.

```
1 static boolean isPal(String s) {
2  if(s.length() == 0 || s.length() == 1)
3   return ........;
4  if(.........)
5   return isPal(s.substring(1, s.length()
        -1));
6  //else
7  return false;
8 }
```

En Java, el método `s.charAt(i)` permite saber qué caracter hay en la posición i de la cadena s y `s.substring(a,b)` retorna una subcadena de s entre los índices a y $b-1$.

a (10%) Completa la línea 3 .....

b (10%) Completa la línea 4 .....

Complejidad 20%

2.1 (10%) Helmuth implementó el algoritmo de burbuja para ordenar un arreglo de tamaño n. Su algoritmo tiene complejidad $T(n) = c \times n^2$ y toma $T(n)$ segundos para procesar n datos. ¿Cuánto tiempo tardará este algoritmo para para procesar 1000 datos, si sabemos que, para $n = 100$, $T(n) = T(100) = 1\,\text{ms}$? Recuerda que $1\,\text{s} = 1000\,\text{ms}$. Así como en los parciales de Física 1, NO olvides indicar la unidad de medida del tiempo que calcules.

2.2 (10%) Un estudiante afirma que encontró un algoritmo capaz de imprimir todos los subconjuntos de un conjunto de n elementos con una complejidad asintótica, para el peor de los casos, de $O(n^2)$. Recuerda que un conjunto tiene 2^n subconjuntos. Determina si el estudiante tiene o no la razón y en ambos casos justifica tu respuesta.

Complejidad 20%

3.1 Considera el siguiente algoritmo:

```
1 static int count7(int n) {
2      if (n == 0) return 0;
3      if (n % 10 == 7) return 1 + count7(
           n / 10);
4      return count7(n / 10);
5 }
```

a (10%) ¿Cuál es la ecuación de recurrencia que mejor define la complejidad, para el peor caso, del algoritmo anterior? Asume que c es la suma de todas las operaciones que toman un tiempo constante en el algoritmo.

i $T(n) = T(n-1) + c$, que es $O(n)$
ii $T(n) = 4T(n/2) + c$, que es $O(n^2)$
iii $T(n) = T(n-1) + T(n-2) + c$, que es $O(2^n)$
iv $T(n) = T(n/10) + c$, que es $O(\log n)$

b (10%) ¿El algoritmo anterior siempre termina para todo número entero $n \in \mathcal{Z}$?

i Sí
ii No

Notación O 20%

4.1 (10%) Considera el siguiente algoritmo:

```
1 static void f(int n){
2   for (int i = 1; i <=n; i = i * 2) {
3       System.out.println("hola")
4   }
5 }
```

¿Cuál es la complejidad asintótica, para el peor de los casos, del algoritmo $f(n)$, teniendo en cuenta que el código dice `i = i * 2` y NO dice `i = i + 2`?

a $O(n^3)$

b $O(n^2)$

c $O(\log n)$

d $O(n^4 \times \sqrt{n})$

4.2 (10%) Considera las siguientes proposiciones:

A $O(f+g) = O(max(f,g))$

B $O(f \times g) = O(f) \times O(g)$

C Si $h = O(g)$ y $g = O(f)$, entonces $h = O(f)$.

D $O(f \times g) = O(h)$, donde $h = max(f,g)$

¿Cuál(es) de las anteriores proposiciones son verdaderas?

No es necesario justificar tu respuesta, pero, si estás viendo Estructuras Discretas, puedes realizar una demostración directa o por reducción al absurdo. Para demostrar que una proposición es falsa, basta con encontrar un contraejemplo (opcional).

Vectores dinámicos 20%

En este punto utilizaremos una lista implementada con arreglos tambien llamada vector dinámico. Esta lista se conoce, en Java, como ArrayList. Ten en cuenta los siguientes métodos:

- La función `A.add(a)` agrega el elemento a al final de la lista A
- La funcion `A.toString()` convierte la lista A en una cadena de caracteres. Como un ejemplo, convierte la lista $\{1,2,3\}$ en "$[1,2,3]$"

5.1 (10%) ¿Cuál es la salida de `opcional3(3)`, es decir, la salida del método `opcional3` con $n = 3$?

```
1 static void opcional3 (int n){
2   ArrayList<Integer> patron = new
        ArrayList();
3   for(int i = 1; i <= n; i++)      {
4     for(int j = 1; j <= i; j++)    {
5       patron.add(j);
6     }
7   }
8   System.out.println(patron.toString());
9 }
```

a $[1,1,2]$

b $[1,1,2,1,2,3]$

c $[1,1,2,1,2,3,1,2,3,4]$

d $[1,1,2,2,1,2,3]$

5.2 (10%) ¿Cuál es su complejidad asintótica, en el peor de los casos, de `opcional3`? Ten en cuenta que la suma de los números del 1 al n es $n(n+1)/2$.

a $O(n)$

b $O(n^2)$

c $O(n^3)$

d $O(1)$

Complejidad (2% extra)

6.1 (2 %) Con base en los errores que has cometido durante el semestre, escribe un texto para el siguiente meme:

....
....
....
....
....
....
....

Como un ejemplo, ten en cuenta la regla de la suma, regla del producto, notación O o ecuaciones de recurrencia.