Estructuras de Datos y Algoritmos 1 - ST0245
Primer Parcial - Martes (032)

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

Marzo 10 de 2020

1 Recursión 30%

En ciberseguridad, cuando se realiza un ataque por fuerza bruta para encontrar una contraseña, se debe tener en cuenta que los caracteres de una contraseña se pueden repetir y, además, que el número de caracteres de la contraseña es variable. Dado una cadena de caracteres S –compuesta por letras mayúsculas en inglés– puedes intercambiar cualesquiera dos caracteres de S o remover cualquier número de caracteres de S. La pregunta que debes resolver es: ¿Cuál es el número de nuevas cadenas de caracteres que se pueden formar al aplicar esas operaciones? Por ejemplo, si $S = AAB$, entonces la respuesta es 8 (A, B, AA, BA, AAB, ABA, BAA). El siguiente código resuelve el problema, pero le faltan algunas líneas; por favor, complétalas. Gracias.

```
1  public int solve(String tiles) {
2    int[] chars = new int[26];
3    for (char ch: tiles.toCharArray()){
4      chars[ch - 'A']++;
5    }
6    return dfs(chars);
7  }
8  int dfs(int[] arr) {
9    int res = 0;
10   for (int i = 0; i < 26; i++) {
11     if(arr[i] == 0) continue;
12     res = ..............;
13     arr[i] = ..............;
14     res = res + ..............;
15     arr[i]++;
16   }
17   return res;
18 }
```

A (10%) Completa la línea 12 .....

B (10%) Completa la línea 13 .....

C (10%) Completa la línea 14 .....

El ciclo en la función `solve` calcula cuantas veces aparece la letra 'A' en `tiles` y lo coloca en la posición 0 del arreglo `chars`, luego cuantas veces aparece la letra 'B' en `tiles` y lo coloca en la posición 1 del arreglo `chars`, y así, sucesivamente, hasta la letra 'Z' en la posición 25. La instrucción `continue` omite las líneas 12 a la 15 y salta a la siguiente iteración del ciclo; es decir, continúa a una nueva iteración del ciclo donde $i = i + 1$.

2 Notación O 20%

Una de las promesas de la biología computacional es la cura el cancer. Los algoritmos exactos para muchos de los problemas de biología computacional tienen una complejidad asintótica, para el peor de los casos, exponencial. Si un algoritmo permite detectar el cancer en un ADN, pero dura 1 trillón de años en ejecutarse, no sirve para el problema.

A (10%) Dadas las siguientes proposiciones, selecciona la(s) correcta(s). Pueden haber varias correctas.

(a) Si $1 < c_1 < c_2$, entonces n^{c_1} es $O(n^{c_2})$.

(b) Si $c_1 > 1$, entonces $\log_{c_1}(n)$ es $O(n)$ y n es $O(\log_{c_1}(n))$.

(c) Si $c_2 > 1$ y $c_1 > 0$, entonces n^{c_1} es $O({c_2}^n)$, pero ${c_1}^n$ NO es $O(n^{c_1})$.

(d) Si $1 < c_1 < c_2$, entonces ${c_1}^n$ es $O({c_2}^n)$ y ${c_2}^n$ es $O(n^{c_1})$.

Considera que c_1 y c_2 son constantes.

B (10%) ¿Es cierto que $\frac{1}{1000}n^3$ es $O(1000n^2)$?

(a) Verdadero

(b) Falso

3 Listas enlazadas 20%

En la vida real, las listas enlazadas se utilizan para implementar el mecanismo de deshacer y rehacer en *Microsoft Word* y *Adobe Photoshop*. Dada una lista simplemente enlazada (lista) de n nodos, `head`, se requiere determinar si esta contiene algún ciclo de una forma particular. Una lista contiene un ciclo de esta forma si algún nodo v, contiene una referencia a otro nodo u y a través de u es posible llegar a v, usando un nodo intermedio p. Por ejemplo, la lista $head = A- > B- > C- > B- > E- > C$ contiene un ciclo de esa forma. El siguiente código determina si una lista tiene un ciclo de esa forma, pero faltan algunas líneas; por favor, complétalas. ¡Muchas gracias! Eres muy amable.

```
1  class Node {Node next; int data;}
2  boolean solve(Node head){
3    Node slow = head;
4    Node fast = head;
```

```
5     while(fast!=null||.....){
6       slow = slow.next;
7       fast = ................;
8       if(slow == fast){
9         return true;
10    } }
11    return false;
12  }
```

A (10%) ¿Cuál es la complejidad asintótica, en el peor de los casos, del algoritmo anterior? .....

B (10%) Completa la línea 7 .....
Y Completa la línea 5 .....

4 Vectores dinámicos 10%

En la vida real, una variación de un *vector dinámico* (en Java, *ArrayList* y en Python, *list*), llamado *Gap Buffer*, se utilizó para implementar el editor *Emacs*. Dado un vector dinámico a de n enteros, un par de índices (i, j) donde $1 \leq i, j \leq n$ e $i < j$, encuentra un $z = a_i + a_j + i - j$ tan grande como sea posible. Por ejemplo, si $a = [8, 1, 5, 2, 6]$, la respuesta es $z = 8 + 5 + 1 - 3 = 11$, donde $a_i = 8, a_j = 5, i = 1, j = 3$. El siguiente código encuentra el valor de z, pero le faltan algunas líneas; por favor, complétalas.

```
1  int solve(ArrayList<Integer> a){
2    int temp = 0;
3    int res = 0;
4    for(int i = 1; i < a.size(); ++i){
5      res = Math.max(res, ..........);
6      temp = Math.max(temp, a.get(i)+i);
7    }
8    return res;
9  }
```

A (10%) Completa la línea 5 .....
Y ¿Cuál es la complejidad asintótica, en el peor de los casos, del algoritmo anterior? .....

Ten en cuenta que, para el cálculo de z, los índices i, j empiezan su valor en 1 y no en 0. En Java, el método `a.get(i)` retorna el elemento en la posición i del vector a.

5 Complejidad 20%

En la vida real, las grandes empresas de tecnología, solicitan calcular la complejidad de los algoritmos que preguntan en sus entrevistas técnicas; por ejemplo, *Google*, *Facebook* y *Riot Games*. Estas entrevistas se realizan para acceder a una práctica laboral como lo hizo Santiago Zubieta, en 2015, y Juan M. Ciro, en 2019, para hacer sus prácticas en Google y Facebook, respectivamente. ¡Ambos eran estudiantes de la Universidad Eafit! Para prepararnos para esas entrevistas, para cada uno de los siguientes códigos, determina la ecuación que representa su complejidad para el peor caso. Considera que C es una constante.

A (10%) Primer misterio

```
1  void mistery1(int n){
2    int res = 0;
3    for(int i = 0; i < n; ++i){
4      for(int j=0;j<n*n*n; ++j){
5        res = res * 2;
6  } } }
```

(a) $T_1(n) = n^3 \times C$.
(b) $T_2(n) = n^4 \times C$.
(c) $T_3(n) = n^5 \times C$.
(d) $T_4(n) = n^6 \times C$.

B (10%) Segundo misterio

```
1  void mistery2(int n){
2    int res = 0;
3    for(int i = 0; i < n; ++i){
4      for(int j=0;j<2*n-1; ++j){
5        res = res * 3;
6  } } }
```

(a) $T_1(n) = n \times C$.
(b) $T_2(n) = n^2 \times C$.
(c) $T_3(n) = n^3 \times C$.
(d) $T_4(n) = n^4 \times C$.

6 Complejidad (2% extra)

(2 %) En la vida real, los memes se utilizan en presentaciones orales como una estrategia para *romper el hielo*. 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.