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

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

Mayo 12 de 2020

1 Pilas 20%

En Lenguajes de Programación, se utilizan los árboles para representar la estructura que tiene un programa. Estos árboles se conocen como *árboles sintácticos abstractos (ASC)*. Los ASC se utilizan para detección de plagio en tareas de programación. Dados dos árboles binarios b_1 y b_2, determine si ambos son idénticos. La siguiente implementación usa pilas y le faltan algunas líneas; por favor, complételas. ¡Ayúdanos a detectar el plagio!

```
1  class Node{Node left,right; int data;}
2  boolean identical(Node b1, Node b2){
3    if(b1 == null && b2 == null)
4      return true;
5    if(b1 == null || b2 == null)
6      return false;
7    Stack<Node> s1 = new Stack();
8    Stack<Node> s2 = new Stack();
9    s1.push(b1); s2.push(b2);
10   while(!s1.isEmpty()){
11     Node p1 = s1.pop();
12     Node p2 = s2.pop();
13     if(p1.data!=p2.data)
14       return false;
15     if(p1.left!=null&&p2.left!=null)
16     {
17       ..................;
18       ..................;
19     }
20     else if(p1.left!=null||p2.left!=null)
21       return false;
22     if(p1.right!=null&&p2.right!=null)
23     {
24       ..................;
25       ..................;
26     }else if(p1.right!=null||p2.right!=
         null)
27       return false;
28   }
29   return true;
30 }
```

A (10%) Completa la línea 17 .....
Y Completa la línea 18 .....

B (10%) Completa la línea 24 .....
Y Completa la línea 25 .....

2 Colas 20%

Existe un tipo de árbol binario llamado *partición binaria del espacio (BSP)*. Un BSP se utiliza para determinar qué objetos se deben renderizar en un videojuego 3D; especialmente, en los de acción de primera persona. El primero que lo utilizó fue Doom, desarrollado en 1993. Dado un árbol binario B de n nodos, calcule su altura usando una cola. El siguiente código resuelve el problema, pero faltan algunas líneas; por favor, complételas. ¡Ayuda a ID Software!

```
1  class Node{Node left;Node right;int
       data;}
2  int solve(Node root){
3    if(root == null) return 0;
4    Queue<Node> q = new ArrayDeque();
5    q.add(root);
6    Node front = null;
7    int res = 0;
8    while(!q.isEmpty()){
9      int sz = q.size();
10     for(int i = 0; i < sz; ++i){
11       front = q.poll();
12       if(front.left != null)
13         q.add(front.left);
14       if(front.right != null)
15         q.add(front.right);
16     }
17     res = ..............;
18   }
19   return res;
20 }
```

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

B (10%) Completa la línea 17 .....

3 Tablas de Hash 20%

Cuando un sitio web te solicita tu login y password para autenticarte, el password que se almacena en el servidor es el el hash del password. De esa manera, si un atacante accede a los passwords almacenados en el servidor, estos no le

serán útiles para autenticarse en el sitio web. Para manejar esos passwords, necesitamos implementar una tabla de hash. El siguiente código resuelve el problema, pero es necesario completar algunas líneas; por favor complétalas.

```
1  class Pair {
2      String name;
3      int phone;
4      public Pair(String name, int phone) {
5          this.name = name;
6          this.phone = dato;
7      }
8  }
9  public class HashTable {
10     private ArrayList<LinkedList<Pair>>
           tabla;
11
12     public HashTable(){
13         tabla = new ArrayList(11);
14         for (int i = 0; i < 10; i++) {
15             tabla.add(new LinkedList());
16         }
17     }
18     private int funcionHash(String k){
19         return ((int) k.charAt(0)) % 10;
20     }
21     public void put(String k, int v){
22         Pair nueva = new Pair(k,v);
23         tabla.get(........).add(nueva);
24 }   }
```

A (10%) Completa la línea 23 .....

B (10%) ¿Cuál es la complejidad asintótica, en el peor de los casos, del método insertar (`put`)? Donde n es el número de elementos que hay en tabla.
O(.....)

4 Árboles 30%

En la vida real, el árbol de codificación de Huffman es un tipo de árbol binario que se utiliza para compresión de datos sin pérdidas en formatos como mp3 y jpg. Volviendo a los temas del curso, en este problema, se te entrega un árbol binario y te preguntan encontrar si existe o no un camino desde la raíz hasta un nodo con valor (*value*) de n y, además, si el camino existe, almacenarlo en la lista `path`; si no existe, la lista `path` debe quedar vacía.

```
1  class Node {
2      int data;
3      Node left, right;
4      Node(int value) {
5          data = value;
6          left = right = null;
7  }   }
8  class Tree{
9      private boolean findPath(Node root,
           int n, List<Integer> path) {
10         if (root == null) {
11             return false; }
12         path.add(root.data);
13         if (root.data == n) {
14             return true; }
15         if (root.left != null && findPath(
               root.left, n, path)) {
16             return true; }
17         if (root.right != null && findPath
               (root.right, n, path)) {
18             return true; }
19         path.remove(path.size()-1);
20         return false;
21 }     }
```

A (10%) ¿Por qué se elimina un elemento de la lista en la línea 19? Porque.....

B (10%) Supongamos que, en una entrevista de Google, nos dicen que el árbol es un *árbol binario de búsqueda (BST)*. ¿Cómo se puede hacer más eficiente la búsqueda anterior sabiendo que el árbol es un BST? Se puede.....

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

5 Grafos 10%

En la vida real, un grafo se utiliza para representar los mapas de Google Maps. Para cada una de las siguiente proposiciones, determine si son verdaderas o falsas.

A (10%) Al representar un grafo completo con matrices de adyacencia y con listas de adyacencia, en ambos casos, el consumo de memoria (NO tiempo, sino memoria) es $O(n^2)$

(a) Verdadero

(b) Falso

Cuando representamos un grafo con matrices de adyacencia, la complejidad asintótica, en el peor de los casos, de insertar un nuevo vértice es $O(n)$, donde n es el número de vértices.

(a) Verdadero

(b) Falso

Un grafo completo es áquel donde existe una arista entre cada par de vértices del grafo.