

Estructuras de Datos 1 - ST0245

Segundo Parcial Grupo 033 (Lunes)

Nombre
Departamento de Informática y Sistemas
Universidad EAFIT

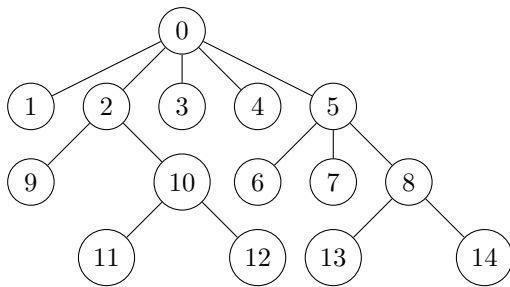
Mayo 7 de 2018

Para propósitos de este parcial se considerarán las siguientes implementaciones de árbol binario y árbol n-ario.

```
1 //Arbol binario
2 class BNode{
3     BNode izq;
4     BNode der;
5     int val;
6 }
7 //Arbol n-ario
8 class NNode{
9     ArrayList<NNode> hijos;
10    int val;
11 }
```

1 Árboles 40%

Considere un árbol n-ario cuyos nodos están enumerados de 0 hasta $n - 1$ y su raíz es el nodo 0. Como un ejemplo, este un árbol n-ario donde $n = 15$.



Para un **sub-árbol** cuyo nodo raíz es el nodo u , el **tamaño** del sub-árbol u , denotado como $tam(u)$, es la cantidad de nodos que éste contiene; particularmente, $tam(0) = n$. Queremos determinar cuantos nodos contiene cada sub-árbol de nuestro árbol n-ario. Ayúdanos a completar las siguientes líneas.

```
1 void calcular(int [] tam,
2               NNode nodo){
3     tam[.....] = 1;
4     int hijos = nodo.hijos.size();
5     for(int i = 0; i < hijos; i++){
6         int siguiente = nodo.get(i).val;
```

```
7         calcular(tam, siguiente);
8         tam[.....] += tam[nodo.val];
9     }
10 }
11 }
12 int [] calcular(int n, NNode raiz){
13     int [] tam = new int [.....];
14     calcular(tam, raiz);
15     return tam;
16 }
```

a (10%) Completa la línea 3

b (10%) Completa la línea 8

c (10%) Completa la línea 13

d (10%) ¿El árbol anterior es un *árbol binario de búsqueda*?

(i) Verdadero

(ii) Falso

2 Pilas 20%

El método **push(i)** ingresa el elemento i al tope de la pila. El método **pop()** retira el elemento en el tope de la pila y retorna su valor. Considere el siguiente método:

```
1 void metodo(Stack<Integer> s){
2     for(int i = 10; i >= 0; i--){
3         if(i % 2 == 0){
4             s.push(i);
5         }
6     }
7     System.out.print(s.pop());
8     while(s.size() > 0){
9         System.out.print(" " + s.pop());
10    }
11 }
```

a (10%) ¿Cuál es la salida del método anterior?

(i) 10, 8, 6, 5, 4, 2, 0

(ii) 2, 4, 6, 8, 10

(iii) 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

(iv) 0, 2, 4, 6, 8, 10

- b (10%) ¿Cuál es la complejidad asintótica, en el peor de los casos, de añadir un nuevo elemento a una pila que contiene n elementos?

- (i) $O(1)$
- (ii) $O(n)$
- (iii) $O(n \log n)$
- (iv) $O(n^2)$

3 Colas 20%

El método `add(i)` agrega el elemento i al inicio de la cola. El método `poll()` retira el elemento al final de la cola y retorna su valor. Considere el siguiente método:

```

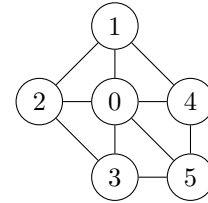
1 void metodo(Queue<Integer> q){
2     for(int i = 2; i * i <= 25; i++){
3         q.add(i);
4     }
5     System.out.print(q.poll());
6     while(q.size() > 0){
7         System.out.print(" " + q.poll());
8     }
9 }
```

- a (10%) ¿Cuál es la salida del algoritmo anterior?
- (i) Los enteros positivos menores que 26 en orden ascendente.
 - (ii) Los enteros positivos menores que 26 en orden descendente.
 - (iii) 2, 3, 4, 5
 - (iv) 5, 4, 3, 2
- b (10%) ¿Cuál es la complejidad asintótica, en el peor de los casos, de retirar un elemento de una cola que contiene n elementos?
- (i) $O(1)$

- (ii) $O(\log n)$
- (iii) $O(n)$
- (iv) $O(n \log n)$

4 Grafos 20%

Considera el siguiente grafo:



- a (10%) Completa su representación utilizando listas de adyacencia:

Pista: La pregunta NO es calcular la clausura transitiva del grafo, sino la lista de las adyacencias, es decir, la lista de los vecinos.

$0 \rightarrow [1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5]$

$1 \rightarrow$

$2 \rightarrow$

$3 \rightarrow$

$4 \rightarrow$

$5 \rightarrow$

- b (10%) ¿Cuál es la complejidad asintótica, en el peor de los casos, de determinar si dos vértices están unidos por una arista en la representación de un grafo en **matrices de adyacencia**?

- (i) $O(1)$
- (ii) $O(n)$
- (iii) $O(n^2)$
- (iv) $O(\log n)$