

Estructuras de Datos 1 - ST0245

Segundo Parcial Grupo 032 (Martes)

Nombre
Departamento de Informática y Sistemas
Universidad EAFIT

Mayo 8 de 2018

Para propósitos de éste parcial se considerará la implementación de un árbol binario y sus respectivos recorridos.

```
1 //Arbol binario
2 class BNode{
3     BNode izq;
4     BNode der;
5     int val;
6 }
7
8 //Recorridos
9 void preorden(BNode nodo){
10     if(nodo != null){
11         System.out.println(nodo.val);
12         preorden(nodo.izq);
13         preorden(nodo.der);
14     }
15 }
16 void posorden(BNode nodo){
17     if(nodo != null){
18         posorden(nodo.izq);
19         posorden(nodo.der);
20         System.out.println(nodo.val);
21     }
22 }
23 void inorden(BNode nodo){
24     if(nodo != null){
25         inorden(nodo.izq);
26         System.out.println(nodo.val);
27         inorden(nodo.der);
28     }
29 }
```

```
7     }
8     int k = 0;
9     while(s.size() > 0){
10         if(k == x){
11             System.out.println(s.pop());
12             break;
13         }
14         k = k + 2;
15         s.pop();
16     }
17 }
```

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

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

b (10%) ¿Qué valor imprime el algoritmo anterior cuando $x = 8$ y $n = 20$?

- (i) 6
- (ii) 8
- (iii) 12
- (iv) 3

c (10%) ¿Cuál es la complejidad asintótica, en el peor de los casos, de encontrar un elemento en un pila con n elementos, en otras palabras, decir si un elemento está o no está en la pila?

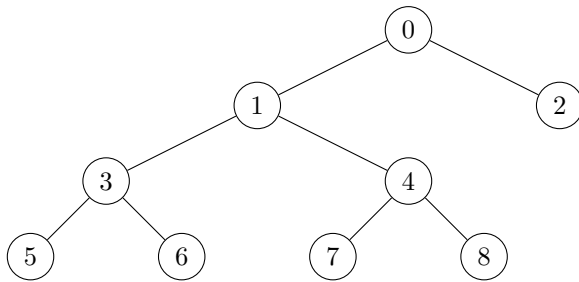
- (i) $O(1)$
- (ii) $O(n)$
- (iii) $O(\log n)$
- (iv) $O(n^2)$

1 Pilas 30%

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(int n, int x){
2     Stack<Integer> s = new Stack();
3     for(int i = 0; i < n; i++){
4         if(i % 3 == 0){
5             s.push(i);
6         }
7     }
8 }
```

2 Árboles 30%



a (10%) Sean A y B las salidas de los recorridos pre-orden y pos-orden del árbol binario anterior, respectivamente. Determine el número de elementos para los cuales se cumple que $A_i = B_i$ para $1 \leq i \leq 8$.

- (i) 3
- (ii) 2
- (iii) 4
- (iv) 0

b (10%) ¿Cuál es la salida del recorrido in-orden del árbol binario anterior?

- (i) 5, 3, 6, 1, 7, 4, 8, 0, 2
- (ii) 0, 1, 3, 5, 6, 4, 7, 8, 2
- (iii) 5, 6, 3, 7, 8, 4, 1, 2, 0
- (iv) 5, 6, 3, 1, 7, 4, 8, 0, 2

c (10%) ¿Es un *árbol binario de búsqueda* el árbol anterior?

- (i) Sí
- (ii) No

3 Colas 30%

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 calcular(int k){
2     Queue<Integer> q = new Queue();
3     for(int i = 0; i < k; ++i){
4         if(k % 3 == 0 && i % 3 == 0){
5             q.add(k - i);
6         }
7     }
8     int j = 0;
9     while(q.size() > 0){
10        if(j == 3){
11            System.out.println(q.poll());
12            break;
13        }
14        q.poll();
  
```

```

15         j++;
16     }
17 }
  
```

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

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

b (10%) ¿Qué imprime el algoritmo anterior cuando $k = 21$?

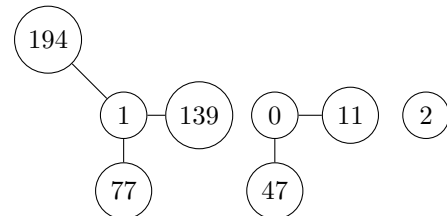
- (i) 6
- (ii) 9
- (iii) 12
- (iv) 3

c (10%) ¿Cuál es la complejidad asintótica, en el peor de los casos, de adicionar un dato a una cola de n elementos?

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

4 Grafos 10%

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.

$0 \rightarrow 11 \rightarrow 47$
 $1 \rightarrow$
 $2 \rightarrow$
 $11 \rightarrow$
 $47 \rightarrow$
 $77 \rightarrow$
 $139 \rightarrow$
 $194 \rightarrow$