ut4_pd3

Ejercicio 1

Desarrolla un algoritmo (ambas partes, método de Árbol y método de Nodo), para Insertar un
nuevo Nodo en el Árbol (este nuevo nodo se ha de brindar como parámetro al método del Arbol).

De acuerdo a los lineamientos para desarrollo de algoritmos en seudocódigo presentados en
clase, deberás desarrollar:

1. Descripción en lenguaje natural del algoritmo solicitado
2. Identificación de precondiciones y postcondiciones correspondientes
3. Escritura del algoritmo en seudocódigo formal
4. Análisis detallado del orden del tiempo de ejecución del algoritmo

Solución

Insertar de TArbolBB

lenguaje natural

si la raiz es nula, inserto el elemento y lo asigno como la raiz del arbol. Sino, llamo al método insertar desde la raiz.

precondiciones

-

postcondiciones

- inserto el nuevo elemento en el arbol

Insertar de TNodoBB

lenguaje natural

Visito cada nodo y comparo la etiqueta del nodo a insertar con la del nodo actual

Si es menor, si su hijo izquierdo es nulo, inserto el nuevo elemento como hijo izquierdo del nodo actual. Sino, llamo al método insertar desde su hijo izquierdo.

Si es mayor, visito su subarbol derecho. Si su hijo derecho es nulo, inserto el elemento asignandolo como hijo derecho del nodo actual. Sino, llamo al método insertar desde su hijo derecho.

Sino, es igual a la etiqueta del nodo actual. Quiere decir que el nodo ya existe en el arbol y no se inserta.

```
insertar de TElelementoAB(TElementoAB nodo)
if etiqueta de nodo < etiqueta this    // O(1)
    if hijoIzq es null                   // O(1)
        hijoIzq <- nodo                    // O(1)
        return true                        // O(1)
    finsi
    hijoIzq.insertar(nodo)               // O(m)
finSi
if etiqueta de nodo > etiqueta this    // O(1)
    if hijo der es null                  // O(1)
        hijoDer <- nodo                    // O(1)
        return true                        // O(1)
    finSi                                // O(1)
    hijoDer.insertar(nodo)               // O(m)
finSi                                   // O(1)
return false                            // O(1)
```

- El tiempo de ejecución en el peor caso es **O(n)** (cuando el árbol está desbalanceado).
- En el mejor caso, si el árbol está balanceado, el tiempo es **O(log n)**.

Ejercicio 2

Desarrolla un algoritmo (ambas partes, método de Arbol y método de Nodo), para contar
todas las hojas que tiene el Árbol.

De acuerdo a los lineamientos para desarrollo de algoritmos en seudocódigo presentados en
clase, deberás desarrollar:

1. Descripción en lenguaje natural del algoritmo solicitado
2. Identificación de precondiciones y postcondiciones correspondientes
3. Escritura del algoritmo en seudocódigo formal
4. Análisis detallado del orden del tiempo de ejecución del algoritmo

Solución

Orden (N) porque se ejecuta en cada nodo

TArbolBB

lenguaje natural

si es vacio, devuelvo 0, sino devuelvo cant hojas

precondiciones

poscondiciones

- no cambia el estado del arbol

seudocodigo

```
de TArbolBB cantHojas()
    si es nulo  // O(1)
        devuelvo 0   // O(1)
    sino
        devolver raiz.cantidad de hojas()  // O(m)
```

de TElemento AB

lenguaje natural

recorro los hijos recursivamente y analizo si son hojas o no. Si son hojas, aumento el contador. Si no, sigo visitando hasta encontrar una hoja

precondiciones

poscondiciones

- no cambia el estado del arbol

seudocodigo

```
de TElementoAB cantHojas()
    contador = 0   // O(1)
    si no es hoja  // O(1)
        si hijo izq no es nulo  // O(1)
            contador += hijo izq.cantHojas()   // O(1)
        si hijo der no es nulo   // O(1)
            contador += hioj der.cantHojas()   // O(1)
        return contador   // O(1)
    finSi
    contador++    // O(1)
    return contador;  // O(1)
```

cantidad de repeticiones = n (se repite para todos los nodos del árbol)

orden del método = n * orden del bloque(1) = n

Ejercicio 3

Dado un Árbol binario de búsqueda que almacena elementos con claves de tipo entero
desarrolla un algoritmo, que Calcule la Suma de las claves de todos los elementos del árbol.
De acuerdo a los lineamientos para desarrollo de algoritmos en seudocódigo presentados en
clase, deberás desarrollar:

1. Descripción en lenguaje natural del algoritmo solicitado
2. Identificación de precondiciones y postcondiciones correspondientes

3. Escritura del algoritmo en seudocódigo formal
4. Análisis detallado del orden del tiempo de ejecución del algoritmo

Solución

TArbolBB

lenguaje natural

si es vacio, devuelvo 0, sino devuelvo suma de las claves

precondiciones

poscondiciones

- no cambia el estado del arbol

seudocodigo

```
de TArbolBB cantHojas()
    si es nulo  // O(1)
        devuelvo 0   // O(1)
    sino
        devolver raiz.sumaClaves()  // O(m)
```

TElementoAB

lenguaje natural

visito el nodo. Si no es hoja, visito todos los nodos del subarbol izquierdo y los de subarbol derecho, sumando las etiquetas de los nodos. Luego sumo la etiqueta del nodo actual y retorno la suma. Si es hoja, retorno el valor de la clave del nodo.

```
de TElememento AB cantHojas()
    suma = 0;
    if no es hoja
        if hijo izq no es nulo
            suma += hijoIzq.cantHojas()
        Finsi
```

```
        if hijo der no es nulo
            suma += hijoDer.sumaHojas()
        finSi
        Devolver suma += clave
    finSi
    devolver clave
```

Ejercicio 4

Desarrolla un algoritmo que, recibiendo como parámetro un entero que indica un cierto nivel,
devuelva la cantidad de nodos del árbol que se encuentran en dicho nivel.
De acuerdo a los lineamientos para desarrollo de algoritmos en seudocódigo presentados en
clase, deberás desarrollar:

1. Descripción en lenguaje natural del algoritmo solicitado
2. Identificación de precondiciones y postcondiciones correspondientes
3. Escritura del algoritmo en seudocódigo formal
4. Análisis detallado del orden del tiempo de ejecución del algoritmo

Solución

TArbolBB

lenguaje natural

visito los nodos, si pertenece al nivel q quiero, entonces lo cuenta. Si llego al nivel y no hay ningun nodo, entonces devuelvo 0.

precondiciones

poscondiciones

- no cambia el estado del arbol

seudocodigo

```
de TArbolBB cantHojas()
    si es nulo  // O(1)
        devuelvo 0   // O(1)
    sino
        devolver raiz.cantNodosNivel(nivel, 0)  // O(m)
```

TElementoAB

lenguaje natural

visito el nodo. voy a acumular en un contador la cantidad de nodos por nivel. Si el nivel actual es distinto al que busco entonces visito sus hijos. Si no tiene hijos, devuelve 0 (no hay nodos en ese nivel). Si el nodo actual es del mismo nivel que busco, entonces se devuelve 1. Entonces al volver de la llamada recursiva, va acumulando los nodos de nivel en contador.

precondiciones

poscondiciones

- no cambia el estado del arbol

seudocodigo

```
cantNodosNivel(int nivel, nivel actual)
nivel = nivel actual
cantHijos = 0
if nivel actual != nivel
    nivelActual++
    if hijo izq no es null
        cantHijos += hijoIzq.cantNodosNivel(nivel, nivelActua
    if hijo der no es null
        cantHijos += hijoDer.cantNodosNivel(nivel, nivelActua
    return cantidad hijos
finSi
return 1;
```