Unidad Temática 4: Trabajo Obligatorio Árboles Binarios

Santiago Panozzo – Algoritmos y Estructuras de Datos – UCU 2023

Tipo de Datos Abstracto Árbol Binario: Obtener menor clave

Operación, funcionalidad, comportamiento: Obtener el elemento de menor clave.

Objetivo: dado un árbol binario, obtener su menor clave.

Lenguaje natural: Recorrer el árbol binario de hijo izquierdo en hijo izquierdo, hasta llegar a un nodo que no posea hijo izquierdo y retornarlo.

Precondiciones:

- El árbol no debe estar vacío.

Postcondiciones:

- El árbol no se verá modificado.
- Se retornará el elemento de menor valor del árbol.

Pseudocódigo:

```
1  Inicio
2  Auxiliar = raíz
3  Mientras auxiliar != nulo:
4      Si auxiliar.hijo_izquierdo == nulo:
5          Retornar auxiliar
6      Auxiliar = auxiliar.hijo_izquierdo
7  FinMientras
8  Fin
```

Tiempo de ejecución:

- **Mejor caso:** Si el árbol solo tiene un nodo, el algoritmo retorna al pasar por la raíz. La complejidad de tiempo es de O(1).
- **Peor caso:** Si el árbol está desbalanceado completamente hacia la izquierda, con una sola rama, y tiene múltiples nodos, el algoritmo deberá recorrer todos los nodos hasta llegar al último de la única rama. En este caso la complejidad de tiempo es de O(n), donde n es el número de nodos.

Tipo de Datos Abstracto Árbol Binario: Obtener mayor clave

Operación, funcionalidad, comportamiento: Obtener el elemento de mayor clave.

Objetivo: dado un árbol binario, obtener su mayor clave.

Lenguaje natural: Recorrer el árbol binario de hijo derecho en hijo derecho, hasta llegar a un nodo que no posea hijo derecho y retornarlo.

Precondiciones:

- El árbol no debe estar vacío.

Postcondiciones:

- El árbol no se verá modificado.
- Se retornará el elemento de mayor valor del árbol.

Pseudocódigo:

```
1  Inicio
2  Auxiliar = raíz
3  Mientras auxiliar != nulo:
4      Si auxiliar.hijo_derecho == nulo:
5          Retornar auxiliar
6      Auxiliar = auxiliar.hijo_derecho
7  FinMientras
8  Fin
```

Tiempo de ejecución:

- **Mejor caso:** Si el árbol solo tiene un nodo, el algoritmo retorna al pasar por la raíz. La complejidad de tiempo es de O(1).
- **Peor caso:** Si el árbol está desbalanceado completamente hacia la derecha, con una sola rama, y tiene múltiples nodos, el algoritmo deberá recorrer todos los nodos hasta llegar al último de la única rama. En este caso la complejidad de tiempo es de O(n), donde n es el número de nodos.

Tipo de Datos Abstracto Árbol Binario: Obtener clave inmediata anterior

Operación, funcionalidad, comportamiento: Obtener la clave inmediata anterior de un elemento de un árbol binario.

Objetivo: dado un árbol binario y un elemento perteneciente al mismo, retornar la clave inmediata anterior a este.

Lenguaje natural: recorrer los elementos del árbol binario de búsqueda utilizando dos punteros, uno que represente el nodo actual, y otro que represente el anterior, hasta que el puntero actual llegue a la etiqueta buscada. Si la clave es menor que el puntero actual, nos moveremos al hijo izquierdo del puntero actual; si la clave es mayor que el puntero actual, el puntero anterior pasará a valer lo mismo que el puntero actual y el puntero actual se moverá a su propio hijo derecho (esto es porque el anterior solo será menor que sus hijos en caso de que se mueva al hijo derecho). Cuando el puntero actual llegue a la etiqueta buscada, retornará el mayor descendiente de su hijo izquierdo, ya que es el más cercano a si mismo. En caso de que no tenga hijos izquierdos, retornará el puntero anterior, ya que representa el mayor número que sea menor que el actual.

Precondiciones:

- El árbol no debe estar vacío.
- La etiqueta buscada debe estar en el árbol.
- La etiqueta dada no debe ser el menor elemento del árbol.

Postcondiciones:

- El árbol no se verá modificado.
- Se retornará el elemento inmediato anterior al dado.
- Se retornará nulo de no existir el elemento anterior al buscado.

Pseudocódigo:

```
1  Inicio
2  anterior = nulo
3  actual = nodo
4  Repetir Siempre:
5      Si actual.etiqueta == buscado:
6          Si actual.hijoIzq == nulo:
7              Retornar anterior
8          SiNo Retornar actual.hijoIzq.Mayor()
9      SiNo Si actual.etiqueta > buscado:
10         Si actual.hijoIzq != nulo:
11             actual = actual.hijoIzq
12     SiNo Si actual.etiqueta < buscado:
13         Si actual.hijoDer != nulo:
14             anterior = actual
15             actual = actual.hijoDer
16     FinSi
17 FinRepetir
18 Fin
```

Tiempo de ejecución:

- **Mejor caso:** Si el árbol solo tiene un nodo, el algoritmo retorna al pasar por la raíz y comprobar su hijo izquierdo. La complejidad de tiempo es de O(1).
- **Peor caso:** En el peor caso, el algoritmo tiene que recorrer hasta el último nivel del árbol. Sin importar el camino que tenga que tomar, la complejidad de tiempo es de O(n) donde n es la cantidad de *niveles* que tiene el árbol.

Tipo de Datos Abstracto Árbol Binario: Obtener cantidad de nodos por nivel

Operación, funcionalidad, comportamiento: Obtener la cantidad de nodos en un nivel dado.

Objetivo: Dado un árbol binario y un nivel, retornar la cantidad de nodos del árbol en dicho nivel.

Lenguaje natural: Recorrer el árbol binario de manera recursiva, almacenando un contador y un registro del nivel actual. Al encontrar un nodo cuyo nivel sea el buscado, sumar uno al contador y regresar al nodo padre. Hacer esto para cada nodo y sus hijos derecho e izquierdo (si los tiene), menos para los nodos cuyo nivel sea superior al buscado.

Precondiciones:

- El árbol no debe estar vacío.

Postcondiciones:

- El árbol no se verá modificado.
- Se retornará la cuenta de todos los nodos en el nivel buscado.

Pseudocódigo:

```
1  Inicio
2  Definir CantidadHijosNivel(nivel, actual, contador):
3      Si actual == nivel:
4          Retornar contador + 1
5      Si HijoIzq != nulo:
6          Contador = HijoIzq.CantidadHijosNivel(nivel, actual+1, contador)
7      Si HijoDer != nulo:
8          Contador = HijoDer.CantidadHijosNivel(nivel, actual+1, contador)
9      Retornar Contador
10 Fin
```

Tiempo de ejecución:

- **Mejor caso:** En el mejor caso, el nivel buscado es el nivel 1, en cuyo caso solo pasará por la raíz y retornará en el contador en 1. La complejidad de tiempo es O(1) en este caso.
- **Peor caso:** En el peor caso, no existe el nivel buscado, en cuyo caso el algoritmo recorrerá todo el árbol buscando el nivel y solo retornará cuando llegue a todas las hojas del último nivel. En este caso la complejidad de tiempo es O(n) donde n es el número de nodos del árbol.

Tipo de Datos Abstracto Árbol Binario: Imprimir hojas con su nivel

Operación, funcionalidad, comportamiento: Imprimir en pantalla todas las hojas del árbol con su respectivo nivel.

Objetivo: dado un árbol binario, imprimir todos los nodos que no tengan hijos, con su nivel.

Lenguaje natural: recorrer todos los nodos del árbol binario de manera recursiva, si el nodo actual no tiene hijos, imprimir su clave, datos, y el nivel actual con respecto a la raíz.

Precondiciones:

- El árbol no debe estar vacío.

Postcondiciones:

- El árbol no se verá modificado.
- Si había nodos que no tuviesen hijos, fueron impresos.

Pseudocódigo:

```
1  Inicio
2  Definir ImprimirHojas(nivelActual):
3      Si nodo.hijo_izquierdo == nulo y nodo.hijo_derecho == nulo:
4          Imprimir nodo, nivel_actual
5      Si nodo.hijo_izquierdo != nulo:
6          Nodo.hijo_izquierdo.ImprimirHojas(nivelActual + 1)
7      Si nodo.hijo_derecho != nulo:
8          Nodo.hijo_derecho.ImprimirHojas(nivelActual + 1)
9  Fin
```

Tiempo de ejecución:

- **Mejor caso:** En el mejor caso, el árbol solo tiene un nodo, por lo que solo pasa por la raíz y la imprime como hoja al no tener hijos. En este caso la complejidad de tiempo es O(1).
- **Peor caso:** En el peor caso, las hojas están todas en el último nivel del arbol, en cuyo caso el algoritmo debe recorrer todos los nodos. La complejidad de tiempo en este caso es de O(n) donde n es el número de nodos en el árbol.

Tipo de Datos Abstracto Árbol Binario: Verificar Árbol de Búsqueda

Operación, funcionalidad, comportamiento: Verificar que el árbol es de búsqueda.

Objetivo: dado un árbol binario, recorrerlo comprobando que sea de búsqueda.

Lenguaje natural: recorrer todos los nodos del árbol binario de manera recursiva, recibiendo cual es el mínimo nodo que puede contener y cual es el máximo nodo que puede contener (en función de los nodos anteriores). En caso de la raíz, todo nodo descendiente suyo debe estar comprendido entre el mínimo y el máximo nodo del árbol. Para cada otro nodo, si tiene un hijo izquierdo, el máximo nodo descendiente de ese hijo debe ser menor que el nodo padre. Si el nodo tiene un hijo derecho, el mínimo nodo descendiente de ese hijo debe ser mayor que el nodo padre. Si no se cumple alguna de estas condiciones se retornará falso.

Precondiciones:

- El árbol no debe estar vacío.

Postcondiciones:

- El árbol no se verá modificado.
- Se retornará un valor verdadero o falso dependiendo de si el árbol es de búsqueda o no.

Pseudocódigo:

```
1  Inicio
2  Verificar = falso
3  Definir VerificarBusqueda(mínimo, máximo):
4      Si etiqueta < mínimo o etiqueta > máximo:
5          Verificar = Falso
6      Si nodo.hijo_izquierdo != nulo:
7          Verificar = Verificar y hijo_izquierdo.VerificarBusqueda(etiqueta, máximo)
8      Si nodo.hijo_derecho != nulo:
9          Verificar = Verificar y hijo_derecho.VerificarBusqueda(minimo, etiqueta)
10 Retornar Verificar
11 Fin
```

Tiempo de ejecución:

En todo caso el algoritmo deberá recorrer todo el árbol buscando nodos que no cumplan con la propiedad de búsqueda. No afecta a la complejidad de tiempo el hecho de que se cumpla o no esta propiedad. La complejidad de tiempo es de O(n) donde n es el número de nodos del árbol.