Taller arboles

Estructura de datos lineales

Santiago Jose Hernandez

Juan diego carreño

Isaias acosta

Julian Diaz

El taller cuenta con 7 carpetas de arboles, nombradas:

1. Arbol
2. Arbol AVL
3. Arbol Quad-Tree
4. Arbol expresion
5. Arbol KD-Tree
6. Arbol Binario
   de los cuales el primero ya se encuentra arreglado, los otros 6 toca realizarle los TADs y corregir el codigo.

Arbol:

Este arbol ya se encuentra previamente solucionado.

Arbol binario:

TAD arbol binario:

```
TAD ArbolBinario
Nombre: ArbolBinario
Responsabilidad: Gestionar el árbol binario, permite agregar, eliminar y
recorrer el árbol.
Atributos:
    NodoBinario<T>* raiz: la raíz del árbol.
Métodos:
```

```
    ArbolBinario(): constructor, inicializa el árbol vacío.
    bool esVacio(): verifica si el árbol está vacío.
    T& datoRaiz(): retorna el dato en la raíz.
    int altura(): calcula la altura del árbol.
    int tamano(): calcula el tamaño del árbol.
    void insertar(T& val): inserta un valor en el árbol.
    bool eliminar(T& val): elimina un nodo con el valor val.
    NodoBinario<T>* buscar(T& val): busca un valor en el árbol.
    void preOrden(): recorre el árbol en preorden.
    void inOrden(): recorre el árbol en inorden.
    void posOrden(): recorre el árbol en postorden.
    void nivelOrden(): recorre el árbol en nivel orden.
```

TAD nodo binario:

```
TAD NodoBinario
Nombre: NodoBinario
Responsabilidad: Representar un nodo de un árbol binario, con referencias a
sus hijos izquierdo y derecho.
Atributos:
 T dato: el valor almacenado en el nodo.
 NodoBinario<T>* hijoIzq: puntero al hijo izquierdo.
  NodoBinario<T>* hijoDer: puntero al hijo derecho.
Métodos:
  T& obtenerDato(): retorna el dato almacenado.
  void fijarDato(T& val): asigna el valor val al dato.
  NodoBinario<T>* obtenerHijoIzq(): retorna el hijo izquierdo.
 NodoBinario<T>* obtenerHijoDer(): retorna el hijo derecho.
 void fijarHijoIzq(NodoBinario<T>* izq): asigna el hijo izquierdo.
 void fijarHijoDer(NodoBinario<T>* der): asigna el hijo derecho.
 int altura(): calcula la altura del subárbol a partir del nodo.
  int tamano(): calcula el tamaño del subárbol a partir del nodo.
 void insertar(T& val): inserta un valor en el árbol binario.
 NodoBinario<T>* buscar(T& val): busca un nodo con el valor val.
 void preOrden(): recorre el árbol en preorden.
  void inOrden(): recorre el árbol en inorden.
  void posOrden(): recorre el árbol en postorden.
  void nivelOrden(): recorre el árbol en nivel orden utilizando una cola.
 NodoBinario<T>* extremo_izq(): retorna el nodo más a la izquierda.
```

```
 NodoBinario<T>* extremo_der(): retorna el nodo más a la derecha.
```

Arbol AVL:

TAD Arbol AVL:

```
TAD Arbol AVL:
Datos Mínimos:
Dato de tipo entero, dato char op
Comportamiento:
Dependiendo del dato tipo char, si op= A inserta un dato en el árbol, si op
= E elimina un dato del árbol, ademas, muestra el árbol en Inorder, preOrder
y PosOrden.
```

TAD nodo:

```
Datos Minimos:
Los 4 nodos cuentan con pocos datos mínimos, de los cuales destacan dato de
tipo T, el cual se utiliza para los nodos de los arboles
Comportamiento:
Estos nodos se dedican a ingresar los datos al árbol, mediante hijos y nodos
```

Arbol Quad-Tree:

TAD Arbol Quad-Tree:

```
Datos Mínimos:
raiz: NodoBinario, señala al primer nodo del árbol binario.
Comportamiento:
ArbolBinario():
Descripción: Constructor que inicializa el árbol vacío.
Retorno: Ninguno.
getRaiz():
Descripción: Retorna el nodo correspondiente a la raíz del árbol.
```

Retorno: NodoBinario.
esVacio():
Descripción: Retorna verdadero si el árbol no posee nodo raíz.
Retorno: booleano.
datoRaiz():
Descripción: Retorna el contenido de la raíz del árbol.
Retorno: Tipo del dato almacenado en la raíz.
altura():
Descripción: Retorna la altura del árbol.
Retorno: Entero.
tamano():
Descripción: Retorna el número de nodos que posee el árbol.
Retorno: Entero.
insertar(valor):
Descripción: Inserta un valor en el árbol siguiendo los parámetros de orden.
Retorno: Ninguno.
altura(subarbol):
Descripción: Retorna la altura de un subárbol que es enviado por parámetro.
Retorno: Entero.
tamano(subarbol):
Descripción: Retorna el número de nodos que posee un subárbol que es enviado por parámetro.
Retorno: Entero.
insertar(valor, subarbol):
Descripción: Retorna un apuntador a la raíz de un subárbol al cual se le ingresa un valor enviado por parámetro.
Retorno: NodoBinario.
eliminar(valor):
Descripción: Busca un valor en el árbol y lo elimina.
Retorno: Ninguno.
buscar(valor):
Descripción: Busca un valor en el árbol y retorna verdadero si lo encuentra.
Retorno: booleano.
void preOrden(subarbol):
Descripción: Realiza la impresión preorden de un subárbol.
Retorno: Ninguno.
void inOrden(subarbol):
Descripción: Realiza la impresión inorden de un subárbol.
Retorno: Ninguno.
void posOrden(subarbol):

```
Descripción: Realiza la impresión posorden de un subárbol.
Retorno: Ninguno.
void nivelOrden(subarbol):
Descripción: Realiza la impresión por nivel de un subárbol.
Retorno: Ninguno.
void preOrden():
void inOrden():
void posOrden():
void nivelOrden():
```

TAD nodo:

```
Datos Mínimos:

dato: Tipo del dato, almacena la información del nodo.
izquierdo: NodoBinario, señala al hijo izquierdo del nodo.
derecho: NodoBinario, señala al hijo derecho del nodo.
NodoBinario(dato):
  Descripción: Constructor que inicializa un nodo con un valor específico y
sin hijos.
 Retorno: Ninguno.
 getDato():
 Descripción: Retorna el valor almacenado en el nodo.
 Retorno: Tipo del dato almacenado.
setDato(valor):
Descripción: Asigna un valor al nodo.   Retorno: Ninguno.
getIzquierdo():
Descripción: Retorna el nodo correspondiente al hijo izquierdo.
Retorno: NodoBinario (o nullptr si no tiene hijo izquierdo).
setIzquierdo(nodo):
Descripción: Asigna un nodo como hijo izquierdo.
 Retorno: Ninguno.
getDerecho():
Descripción: Retorna el nodo correspondiente al hijo derecho.
Retorno: NodoBinario (o nullptr si no tiene hijo derecho).
setDerecho(nodo):
   Descripción: Asigna un nodo como hijo derecho.
 Retorno: Ninguno.
```

```
 esHoja():
Descripción: Retorna verdadero si el nodo no tiene hijos (es una hoja).
Retorno: booleano.
```

Arbol binario ordenado:

TAD arbol binario ordenado:

```
Datos Mínimos:
    raiz, NodoBinario, señala al primer nodo del árbol binario.
Comportamiento:
    ArbolBinario(): Constructor que inicializa el árbol vacío.
    getRaiz(): NodoBinario, retorna el nodo correspondiente a la raíz del
árbol.
    esVacio(): Retorna verdadero si el árbol no posee nodo raíz.
    datoRaiz(): ?, retorna el contenido de la raíz.
    altura(): entero, retorna la altura del árbol.
    tamano(): entero, retorna el número de nodos que posee el árbol.
    insertar(valor): Inserta un valor en el árbol siguiendo los parámetros
de orden.
    altura(subarbol): entero, retorna la altura de un subárbol que es
enviado por parámetro.
    tamano(subarbol): entero, retorna el número de nodos que posee un
subárbol que es enviado por parámetro.
    insertar(valor, subarbol): Retorna un apuntador a la raíz de un subárbol
al cual se le ingresa un valor enviado por parámetro.
    eliminar(valor): Busca un valor en el árbol y lo elimina.
    buscar(valor): Busca un valor en el árbol y retorna verdadero si lo
encuentra.
    void preOrden(subarbol): Realiza la impresión preorden de un subárbol.
    void inOrden(subarbol): Realiza la impresión inorden de un subárbol.
    void posOrden(subarbol): Realiza la impresión posorden de un subárbol.
    void nivelOrden(subarbol): Realiza la impresión por nivel de un
subárbol.
    void preOrden(): Realiza la impresión preorden del árbol.
    void inOrden(): Realiza la impresión inorden del árbol.
    void posOrden(): Realiza la impresión posorden del árbol.
    void nivelOrden(): Realiza la impresión por nivel del árbol.
```

TAD nodo binario:

```
Datos Mínimos:
    dato, T, posee el contenido del nodo.
    Izq, NodoBinario, apuntador hacia el hijo izquierdo.
    Der, NodoBinario, apuntador hacia el hijo derecho.
Comportamiento:
    NodoBinario(dato): Crea un nodo binario e inicializa su dato con el dato enviado por parámetro.
    NodoBinario(): Crea un nodo binario vacío.
    obtenerDato(): Retorna el dato contenido en el nodo.
    fijarDato(val): Fija el valor del dato en el nodo.
    obtenerHijoIzq(): Retorna el nodo hijo izquierdo.
    obtenerHijoDer(): Retorna el nodo hijo derecho.
    fijarHijoIzq(izq): Establece el nodo hijo izquierdo.
    fijarHijoDer(der): Establece el nodo hijo derecho.
```

Arbol expresión:

TAD arbol expresión:

```
Datos Minimos:
    raiz,NodoExp , indica el inicio del arbol
  operadores, conjunto de operadores validos para el arbol.
Comportamiento:
    ArbolExpresion(),crea un arbolExpresion
    llenarDesdePrefija(expresion),recibe una expresion en Polaca y llena el arbol a partir de esta.
    llenarDesdePosfija (expresion),recibe una expresion en Polaca Inversa y llena el arbol a partir de esta.
    obtenerPrefija(),cadena de caracteres, retorna una cadena de caracteres que expresa el arbol en Polaca.
    obtenerInfija(),cadena de caracteres, retorna una cadena de caracteres
```

```
que expresa el arbol en Infija.
    obtenerPosfija(),cadena de caracteres, retorna una cadena de caracteres
que expresa el arbol en Polaca Inversa.
    Prefija(subarbol),cadena de caracteres,retorna una cadena de caracteres
que expresa el subarbol en Polaca.
    Posfija(subarbol),cadena de caracteres,retorna una cadena de caracteres
que expresa el subarbol en Polaca inversa.
    Infija(subarbol),cadena de caracteres,retorna una cadena de caracteres
que expresa el subarbol en Infija.
    evaluar(),entero, retorna el resultado de la expresion contenida en el
arbol.
    esOp(ope),booleano ,verifica si una cadena de caracteres esta en el
conjunto de operadores validos, si lo esta retorna true.
    eval(NodoExp *), entero,evalua la expresion contenida en un subarbol.
    NodoExp* llenarDesdePrefijaa(vector<string> &q, unsigned int &pos,
NodoExp* actual);
    NodoExp* llenarDesdePosfijaa (vector<string> &q, unsigned int &pos,
NodoExp* actual);
    void tokenizar(const string & s,vector <string> &q);
};
```

TAD nodo expresión:

```
Datos Minimos:
    data,string, posee el contenido del nodo
    left, NodoExp,apuntador hacia el hijo izquierdo
   right, NodoExp,apuntador hacia el hijo derecho
   op, booleano, determina si el contenido es un operador(true), o un
numero(false).
Comportamiento:
    NodoExp(dato),crea un nodo binario e inicializa su dato con el dato
enviado por parametro.
    NodoExp()crea un nodo binario vacio.
//Setters y Getters
   getData()
    setData( val)
    getLeft()
    getRight()
```

```
    setLeft(left)
     setRight(right)
  getOp()
    setOp(left)
```