TALLER 3, ARBOLES DE BUSQUEDA

Juan Manuel Soto Morales
Sebastian Orjuela Sanchez
Sergio Herrera Tellez

FACULTAD DE INGENIERÍA

INGENIERÍA DE SISTEMAS

2022

DISEÑO DE TADS

tad arbolAVL

- datos mínimos:
    - fijarRaiz,void,despues de ser insertada la raiz se fija como tal en el arbol AVL
    - datoRaiz,tipo template, aquí se obtiene el dato de la raiz del arbol AVL
    - esVacio,booleano,se quiere saber si en el árbol AVL está vacío empezando en false
- operaciones:

- nodoBinario<T>* obtener Raíz(),se obtiene la raiz del arbol AVL
- void preOrden(),recursividad siendo Primero el padre, luego el hijo izquierdo y finalmente el hijo derecho.
- void inOrden(),se hace recursividad Si visitamos primero hijo izquierdo, luego el padre y finalmente el hijo derecho
- void inOrdenHasta(std::list< std::pair< T, int > >& lista, int tam);
- void posOrden(), recursividad siendo Primero hijo izquierdo, luego el hijo derecho y finalmente el padre
- void nivelOrden(), dependiendo del nivel se le inserta el orden
- bool buscar(T& val), busca si la raíz existe en el árbol AVL
- bool insertar(T& val), inserta la raíz en el árbol AVL
- bool eliminar(T& val), elimina una raiz en el arbol AVL
- T& minimo(),se encuentra el mínimo de la raiz del arbol AVL
- T& máximo(), se encuentra el máximo de la raiz del arbol AVL
- int tamano(), se encuentra el tamaño de la raíz del árbol AVL por sus raíces
- int altura(), se encuentra la altura de la raíz del árbol AVL por sus raíces

Tad nodoBinario:

-Datos minimos:

- Dato,Template,Para poder conocer el dato que llega
- hijoIzquierdo,Template, identificar el hijo izq
- hijoDerecho,Template,identificar el hijo der
- altura,Entero,Poder conocer la altura

-Operaciones:

- nodoBinario(),
- nodoBinario(T& val),
- nodoBinario(T& val, nodoBinario<T> *izq, nodoBinario<T> *der)
- ~nodoBinario(),
- bool esHoja(),
- T& obtenerDato(),nos permite obtener el dato para despues utilizarlo
- nodoBinario<T>* obtenerHijoIzq(),nos permite conocer el hijo izquierdo
- nodoBinario<T>* obtenerHijoDer(),nos permite conocer el hijo derecho
- int obtenerAltura(),nos permite encontrar la altura
- void fijarDato(T& val),fija el dato
- void fijarHijoIzq(nodoBinario<T> *izq),fija el hijo izq
- void fijarHijoDer(nodoBinario<T> *der); fija el hijo der
- void fijarAltura(int& h); fija la altura
- void preOrden(std::list< std::pair< T, int > >& listapre);
- void inOrden(std::list< std::pair< T, int > >& listain);
- void inOrdenHasta(std::list< std::pair< T, int > >& lista, int tam, int nivel);
- void posOrden(std::list< std::pair< T, int > >& listapos);
- void nivelOrden();permite conocer el nivel en el que se encuentra
- bool buscar(T& val);permite buscar un dato
- bool insertar(T& val, nodoBinario<T> **nraiz);su funcion es insertar un dato
- bool eliminar(T& val, nodoBinario<T> **nraiz);nos permite eliminar un dato
- T& minimo();conocer el minimo
- T& maximo();conocer el maximo
- int tamano();conocer el tamaño
- void actualizarAlturas();nos permite actualizar las alturas
- nodoBinario<T>* balanceoArbol ();balancea el arbol

DISEÑO GRAFICO DE LOS 2 TADS

TAD NodoBinario

Datos mínimos:
Dato
hijoIzquierdo
hijoDerecho
Altura

Operaciones:
nodoBinario(),
nodoBinario(T& val),
nodoBinario(T& val, nodoBinario<T>*izq,
nodoBinario<T> *der)
~nodoBinario()
bool esHoja()
T& obtenerDato()
nodoBinario<T>* obtenerHijoIzq()
nodoBinario<T>* obtenerHijoDer()
int obtenerAltura()
void fijarDato(T& val)
void fijarHijoIzq(nodoBinario<T> *izq) void
fijarHijoDer(nodoBinario<T> *der) void
fijarAltura(int& h)
void preOrden(std::list< std::pair< T, int > >&
listapre)
void inOrden(std::list< std::pair< T, int > >& listain);
void inOrdenHasta(std::list< std::pair< T, int > >&
lista, int tam, int nivel)
void posOrden(std::list< std::pair< T, int > >&
listapos)
void nivelOrden();
bool buscar(T& val)
bool insertar(T& val, nodoBinario<T> **nraiz)
bool eliminar(T& val, nodoBinario<T> **nraiz)
T& minimo()
T& maximo()
int tamano()
void actualizarAlturas();
nodoBinario<T>* balanceoArbol ()

→

TAD árbolAVL

datos mínimos:
fijarRaiz
datoRaizesVacio

---

operaciones:
nodoBinario<T>* obtener Raíz
void preOrden()
void inOrdenvoid
inOrdenHasta(std::list< std::pair< T, int >
>& lista, int tam);
void posOrden
void nivelOrden
bool buscar(T& val)
bool insertar(T& val)
bool eliminar(T& val)
T& minimo()
T& máximo()
int tamano()
int altura()