Análisis y Algoritmos

Roberto Charreton Kaplun
Universidad de Artes Digitales

Guadalajara, Jalisco

Email: idv17c.rcharreton@uartesdigitales.edu.mx

Profesor: Efraín Padilla

Julio 11, 2019

1) Binary Tree

Un árbol binario es una estructura de datos de árbol en la que cada nodo tiene como máximo dos hijos, a los que se hace referencia como el hijo izquierdo y el hijo derecho. Una definición recursiva que usa las nociones de teoría de los conjuntos es que un árbol binario (no vacío) es una tupla (L, S, R), donde L y R son árboles binarios o el conjunto vacío y S es un conjunto único.

Desarrolle el codigo implementando una funcion que inserta, elimina y ordena el vector de datos de manera ascendente.

Insert Node on Tree

```
void CBinaryTrees::insert(node *tree, node *newnode)
{
  if (root == NULL)
  {
    root = new node;
    root->info = newnode->info;
    root->left = NULL;
    root->right = NULL;
    cout << "Root Node is Added" << endl;
    return;
  }
  if (tree->info == newnode->info)
  {
    cout << "Element already in the tree" << endl;
    return;
  }
  if (tree->info > newnode->info)
  {
    if (tree->left != NULL)
    {
      insert(tree->left, newnode);
    }
    else
    {
      tree->left = newnode;
      (tree->left)->left = NULL;
      (tree->left)->right = NULL;
      cout << "Node Added To Left" << endl;
      return;
    }
  }
  else
  {
```

```
    if (tree->right != NULL)
    {
      insert(tree->right, newnode);
    }
    else
    {
      tree->right = newnode;
      (tree->right)->left = NULL;
      (tree->right)->right = NULL;
      cout << "Node Added To Right" << endl;
      return;
    }
  }
}
```

Delete Node of Tree

```
void CBinaryTrees::del(int item)
{
  node *parent, *location;
  if (root == NULL)
  {
    cout << "Tree empty" << endl;
    return;
  }
  find(item, &parent, &location);
  if (location == NULL)
  {
    cout << "Item not present in tree" << endl;
    return;
  }
  if (location->left == NULL && location->right == NULL)
    case_a(parent, location);
  if (location->left != NULL && location->right == NULL)
    case_b(parent, location);
  if (location->left == NULL && location->right != NULL)
    case_b(parent, location);
  if (location->left != NULL && location->right != NULL)
    case_c(parent, location);
  free(location);
}
```

Order Nodes of Tree

```
void CBinaryTrees::inorder(node *ptr)
{
  if (root == NULL)
  {
    cout << "Tree is empty" << endl;
    return;
  }
  if (ptr != NULL)
  {
    inorder(ptr->left);
    cout << ptr->info << "  ";
    inorder(ptr->right);
  }
}
```