Tema 5. Árboles

5.0. Contenido y documentación

5.0. Contenido y documentación
5.1. Grafos
5.2. Árboles
5.3. Árboles binarios
5.3.1. Estructura de datos
5.4. Recorridos en profundidad
5.4.1. Preorden
5.4.2. Postorden
5.4.3. Inorden

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/35df65d4-7c6b-4515-bbc5-f7aba8f2dac8/U5_Arboles.pdf

5.1. Grafos

*Definición. Un **grafo** es una estructura de datos $G = (V, A)$ conpuesta de un conjunto de vértices V y un conjunto de aristas A.*

*Definición. Un **grafo dirigido** es aquel cuyas aristas son pares ordenados de vértices.*

*Definición. Un **grafo no dirigido** es aquel cuyas aristas son pares no ordenados de vértices.*

*Definición. Un **camino** de un grafo $G = (V, A)$ es una secuencia de nodos V en los que cada nodo está conectado al siguiente mediante un arco en A.*

5.2. Árboles

*Definición. Un **árbol** con raíz es un grafo dirigido tal que:*
*- Tiene un nodo distinguido, denominado **raíz**, sin arcos incidentes.*
- Cada nodo distinto de la raíz recibe un solo arco.
- Cualquier nodo es accesible desde la raíz.

*Definición. Los **antecesores** de un vértice v son todos los vértices en un camino desde la raíz hasta v.*

*Definición. Los **descendientes** de un vértice v son todos aquellos que tienen a v como antecesor.*

*Definición. Las **hojas** son los nodos terminales del árbol.*

*Definición. Los **nodos intermedios** son todos aquellos que no son hojas.*

*Definición. La **profundidad** o nivel **de un nodo** es el número de ramas entre el nodo y la raiz.*

*Definición. La **profundidad** o altura **de un árbol** es el máximo número de ramas entre la raíz y una hoja del árbol.*

5.3. Árboles binarios

*Definición. Un **árbol binario** es un árbol ordenado con raíz tal que cada nodo tiene como mucho 2 hijos.*

> Propiedad recursiva de los árboles binarios. Cualquier nodo de un árbol binario es a su vez la raíz de un subárbol binario.

5.3.1. Estructura de datos

Los árboles binarios se forman a partir de nodos, que tienen la siguiente etructura:

```
btypedef struct _BTNode{
  void *info; /*contiene los datos del nodo*/
  struct _BTNode *left; /*puntero al hijo izquierdo del nodo*/
  struct _BTNode *right; /*puntero al hijo derecho del nodo*/
}
```

En el caso de la hojas, los campos `left` y `right` son `NULL`.

Por otra parte, la estructura del árbol es:

```
typedef struct _BSTree{
  BTNode *root; /*puntero al nodo raíz*/
}
```

Un árbol está vacío si su nodo raíz es `NULL`.

5.4. Recorridos en profundidad

Cuando se recorre un árbol en profundidad se trata de visitar a los hijos de un nodo antes que a sus hermanos. De forma que para cada nodo se debe:

- Recorrer recursivamente el hijo izquierdo.
- Recorrer recursivamente el hijo derecho.
- Visitar el nodo.

Dependiendo del orden en el que se realicen estos 3 pasos, se dan tres tipos de recorridos en profundidad.

5.4.1. Preorden

Si se sigue un recorrido de orden previo, primero se visita el nodo explorado, después se recorre recursivamente en orden previo el hijo izquierdo, y, por último, el derecho.

```
pre_order (BinaryTree T){
  if bt_isEmpty(T) == TRUE:
    return

  visit(T)
  pre_order(left(T))
  pre_order(right(T))
}
```

5.4.2. Postorden

Si se sigue un recorrido de orden posterior, primero se recorre recursivamente en orden posterior el hijo izquierdo, después el derecho, y, por último, se visita el nodo explorado.

```
post_order (BinaryTree T){
  if bt_isEmpty(T) == TRUE:
    return

  post_order(left(T))
  post_order(right(T))
  visit(T)
}
```

5.4.3. Inorden

Si se sigue un recorrido de orden medio, primero se recorre recursivamente en orden medio el hijo izquierdo, después se visita el nodo explorado, y, por último, se recorre recursivamente en orden medio el hijo derecho.

```
in_order (BinaryTree T){
  if bt_isEmpty(T) == TRUE:
    return

  in_order(left(T))
  visit(T)
  in_order(right(T))
}
```