ÁRBOLES Y GRAFOS

Los arboles ya no son estructuras lineales y por lo tanto ya puedes moverte y desviarte en algún punto. En estas estructuras te olvidas de las estructuras de datos lineales y pasamos a unas mas flexibles, pueden ser más o menos complejos por las múltiples dimensiones.

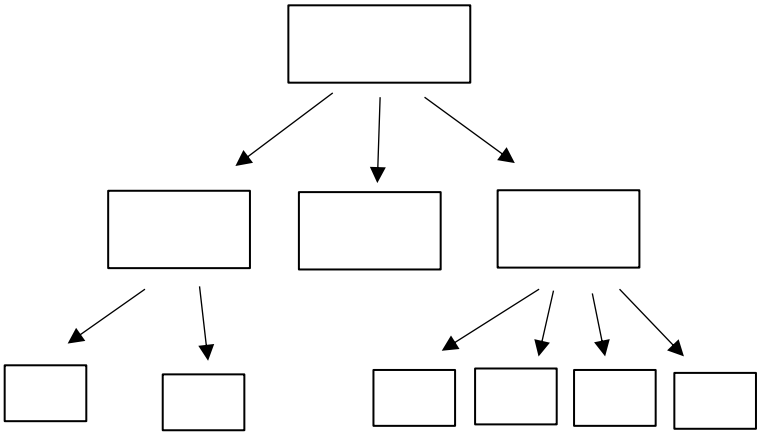
ARBOLES

- Genéricos
- De búsqueda [Binarios]

GRAFOS

Estructura de datos que permiten tener una relación jerárquica padre e hijo(dimensiones).

ROOT(comienzo del árbol)



Está compuesto por nodos considerados hijos que a su vez pueden tener más.

Hay limitaciones, un nodo de un determinado nivel del árbol solo puede conectar con nodos de nivel inferior. En un árbol no hay ciclos, no te puedes quedar enganchado en un recorrido. Solo padres únicos de 1 nodo salvo en un caso, el nodo raíz que no tiene padre.

En los nodos existe el grado que dice cuántos hijos tiene, grado1,2,3… `Los nodos que tiene grado 0 son nodos hojas, especiales.

Los árboles no me dan una serie de posibles a la hora de recorrerlos. En un árbol no hay orden si no lo especifico ya que no es obligatorio. Normalmente los asociamos a estructuras que tiene orden, pero no es necesario. Cuando tenemos árboles con nombre se llaman arboles de búsqueda.

Cuando recorremos árboles, hay 3 tipos de recorridos:

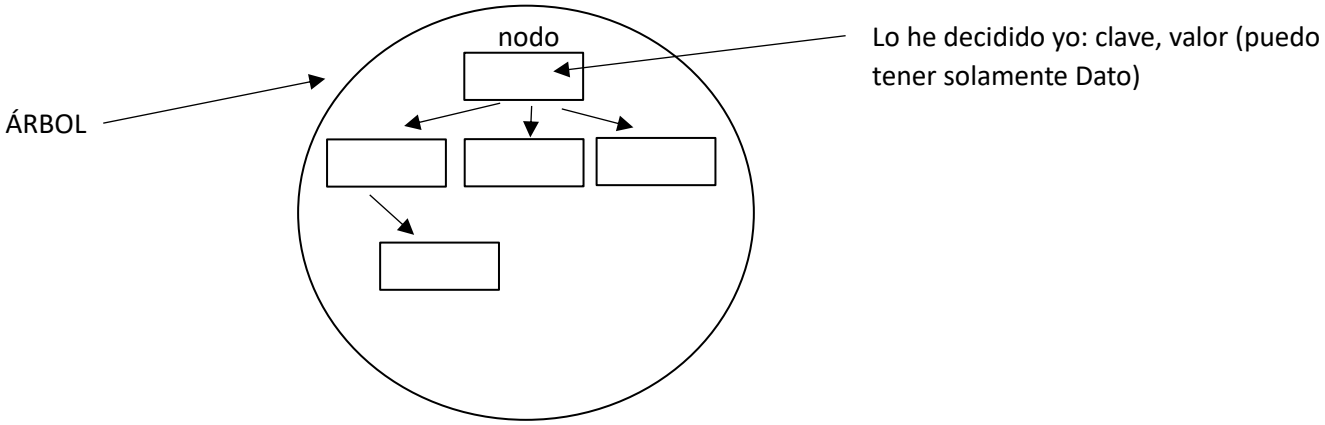
- En pre-orden
- Orden central
- Post orden

} Traducir estructura de árbol a una lineal

PROGRAMACIÓN ÁRBOL GENÉRICO (estructura similar listas)

Usamos un tipo especial de nodo (parecido al diccionario)

Estructura clases:



Class ArbolGenérico <k, v> {

Nodogenerico<k, v> raíz;

….

Class NodoGenerico <k, v>{

K clave;

V valor;

Lista <Nodogenerico <k, v>>

…

Class Iterador

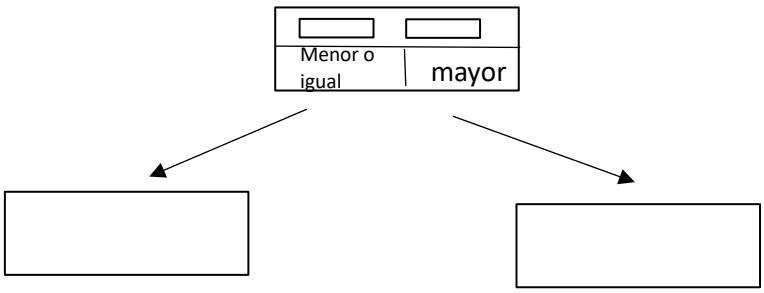
….

Vamos a usar la Búsqueda para poner un orden al árbol.

BUSQUEDA[BINARIOS]

- Con orden
- Máximo grado 2

Aporta cierta facilidad a la hora de trabajar.



El igual puede permitir duplicados.

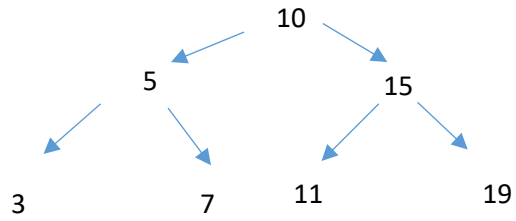
```
Class ArbolBinarioDeBusqueda <k, v>{
        Nodo Árbol BB <k, v> raíz;
…
Class NodoABB <k, v>{
        K clave
        V valor
        NodoABB <k, v> menor;
        NodoABB <k, v> mayor;
…
Class ArbolBinarioDeBusqueda <k, v>{
        NodoABB <k, v> raiz
        Protected boolean addNodoRaiz(k clave, v valor){{
                NodoABB <k, v> n= newNodoABB(K,V);
                This.raiz=n;
        }
        Protected boolean addOtrosNodos (k clave, v valor){
                NodoABB candidato= buscar NodoAInsertar (clave, tis.raiz)
                If (candidato==null)
                        Return false
                NodoABB<K, v> n=new NodoABB(K,v)
                If candidato.getclave()> clave ;
                        Candidato setMenor(n) ;
                Else
                        Candidato.setMayor(n) ;
                Return true
                }
Class NodoABB <k, v>{
        K clave;
        V valor;
        NodoABB<k, v> menor;
        NodoABB<k, v> mayor;
```

```
        NodoABB(k clave, v valor){
                This.clave=clave;
                This.valor=valor;
        }
}
NodoABB <k, v>  buscarNodoAintersare(k.c.buscadi, nodoABB <k, v>, nodo){
        If (nodo.getclave()=c.busqueda)
                Return null;
        If (nodo.getClave()=c.busqueda)
                If (nood.getMenor()!= null)
                        Return busarNodoAInsertar(c.busqueda, nodo.getmenor)
                Else
                        Return nodo
        Else
                If (nodo.getMayor ()!= null)
                        Return buscarNodoAInsertar (c.busqueda,nodod.getMenor)
                Else
                        Return nodo
}
Public Boolean addNodo (k clave, v valor)
        If (this.raiz==null)
                Return addNodoRaiz(clave,valor)
        Else
                Return sddOtrosNodos(clave, valor)
}
```

ESTRUCTURA DE DATOS DE UN ÁRBOL(RECORRIDOS)

Un árbol se compone de múltiples subárboles, es decir estructura recursiva. Hay tres tipos de nodos: raíz, intermedios y hojas.

Árbol:



RECORRIDO PREORDEN

```
Void Preorden (N){
        If (n!= null){
                System.outprintln(N.dato);
                Preorden(N.gteMayor());
                Preorden(N.getMayor());
         }
}
```

10, 5, 3, 7, 15, 11, 19

RECORRIDO CENTRAL

```
Void Central (N) {
        If (N!= null){
                Central(N.getMenor());
                System.outprintln(N.dato);
                Central(N.getMayor());
        }
}
```

3, 5, 7, 10, 11, 15, 19

RECORRIDO POST ORDEN

```
Void PostOrden (N){
        If (N!= bull)
                PostOrden(N.getMenor());
                PostOrden(N.getMayor());
```

```
            System.out.println(n.dato);
}
```

¿Por qué hay 3 recorridos?

En el post orden, es una aplicación directa de porque los árboles, para poder descomponer las matemáticas de forma lógica para resolver los componentes. Todo de industria está compuesto por árboles, siempre hay aplicación gráfica.

El preorden se quiere para cuando no va de componentes que construir son de elementos que tengo que partir. Cada vez que entro a un hijo, ya has soltado al padre.

¿Qué pasaría si llego y quiero quitar el 7?

La estructura del árbol se desequilibra.

Necesitamos establecer las estrategias para el borrado:

- Borro un nodo que no tiene nada-> lo desenganchamos. (no sé si es mayor o menor)
  ```
  If (n.padre.getMenor()==N)
          N.padre.setMenor(null);
  Else
          N.padre.setMayor(null);
  ```
- Borro un elemento que contiene un elemento con hijos y el otro no.
  ```
  If (n.padre.getMenor()!=null)
          Hijo= N.getMenor();
  Else
          Hijo= N.getMayor();
  If (n.padre.getMenor()==N)
          N.padre.setMenor(hijo);
  Else
          N.padre.setMayor(hijo);
          Hijo.setPadre(N.padre)
  ```
- Borro un elemento que tiene 2 hijos
  (De rama de pequeños busco el mayor y de rama de mayores busco el menor.)

ÁRBOLES AVL

ESTRUCTURAS

En el momento que ya hay un árbol no se puede volver a estructurar y equilibrar. Lo que puedes hacer es coger los nodos y crear otro. Las modificaciones que acepta un árbol son inserciones y eliminaciones y es lo que puede variar la estructura del árbol.

Todos los valores que se van a usar son los valores que se pueden calcular. Una de las características es que, normalmente la estructura, solemos cumplimentarla con nodos nuevos.

La altura de una hoja siempre es 1 porque es si mismo, la altura de un subárbol es la altura del camino más largo. Las alturas se calculan de abajo a arriba. Factor de balanceo=la resta de la altura del subárbol derecho menos el izquierdo o viceversa.

Si un nodo solo tiene 1 hijo, entonces tiene altura 2.

Altura= 1+ máx. (h izquierda, h derecha)

ROTACIONES

Cuando un nodo tiene un subárbol desequilibrado, entonces todos los que hay por encima también van a estar desequilibrados.

El factor de balanceo lo hemos dejado como número enteros entonces uso el signo del dato del factor de balanceo para saber si son rotaciones simples o dobles. Si el signo del factor de balanceo coincide es una rotación doble sino son simples. A derechas positivos, a izquierda negativos.

Los signos son los que nos dan como hacer la rotación, el truco es que el árbol esta desbalanceado y por tanto sabes por los signos donde queda el hueco para rotarlo.

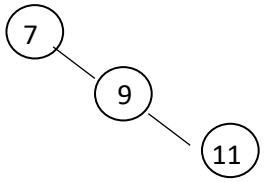
Si en una rotación ya hay un hijo, el hijo pasa a ser nieto.

PROCESO DE CREACIÓN DE UN ÁRBOL:

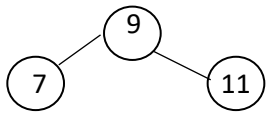
[7,9,11,8,10,12,5,14]

- 7: altura 3 F.balanceo -2
- 9: altura 2 F.balanceo -1
- 11:(Aquí se desbalancea el árbol)
- 8:
- 10:
- 12:
- 5:
- 14:

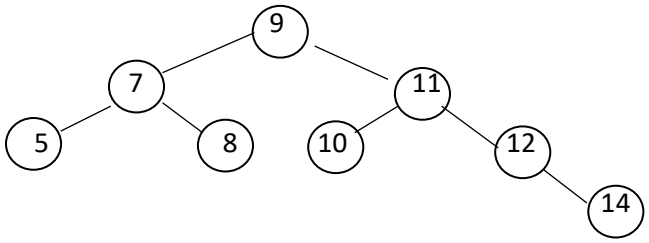
Pasamos de :



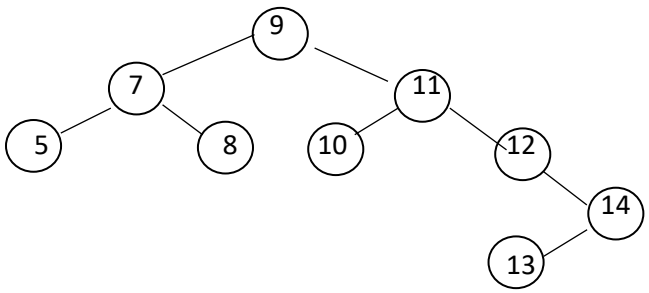
A:



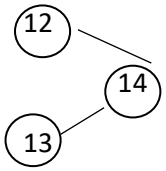
QUEDA:



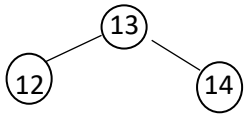
Si le añadimos el 13 tiene desbalance:



Hay un desbalance a su qué vamos a sacar el subárbol 12,14,3 y vamos a operar con él.



Pasa a ser:



El problema es que puedes tener subárboles muy grandes por debajo. Los borrados son un poco más complicados. ¿Qué puede suceder? Que al hacer un movimiento se desequilibre las ramas anteriores entonces hay que revisarlo para cambiar valores como la altura y factor de balanceo.

Se mira de abajo a arriba y cuando se encuentra un desbalanceo lo cambiamos y seguimos subiendo hasta que se llega a la raíz.

GRAFOS

Un grafo es una cosa muy importante que sirve para representar muchas cosas. Hoy en día en informática la mayor parte de la información del mundo la tenemos almacenada en grafos, grafos de conocimiento se llaman.

Los grafos son estructuras de datos muy flexibles. Hasta ahora las estructuras de datos hemos visto limitan como introducir información y por tanto hay una jerarquía de como entrar, algún tipo de estructuración.

Existen muchos tipos de relaciones entre las distintas entidades y se representan por grafos. Estructuras matemáticas que pueden representar el mundo real y por tanto podemos tener una estructura abstracta del problema con la cual tenemos algoritmos.

A nivel matemático tenemos una serie de nodos con unas conexiones

Hay muchos tipos de grafos, ¿Cómo implementamos las distintas representaciones de graos? Pueden ser dirigidos o no dirigidos. ¿Cómo puedo representarlos?

Los voy a representar en función de unos vértices que en informática llamamos nodos y una serie de arcos también llamados aristas que en informática solemos decir relaciones que conectan nodos.

¿Cómo lo llevamos a una estructura de datos que pueda trabajar? Originalmente usábamos estructuras de datos típicas de ordenadores, las matrices.

En la diagonal principal esta de ceros y solo hay datos por encima, estoy reseñando que tengo un grafo no dirigido. Si mi grafo fuese dirigido diría que puedo ir por una dirección, pero no por otra. Las celdas vacías dicen que no existe conexión directa.

Si yo quiero que mi grafo sea dirigido, entonces tengo que añadir información por debajo de la diagonal principal.

No tiene por qué poner los mismos valores en la parte superior que la inferior, porque “ir desde Teruel hasta Guadalajara puedo ir por una carretera y volver a otra”.

Diferencia con métodos anteriores como un árbol. En un árbol la información la tenemos en los notos, pero en un grafo la tenemos en los notos y también en los arcos.

Eso nos permite añadir una segunda dimensión a la información que nosotros añadimos. La información que nos da un grafo tiene una parte dinámica y una parte estática.

Teniendo los mismos nodos estáticos, pero puedo alterar la información solo con alterar los arcos disponibles o la información que tienen.

Los arcos anotados son los que nosotros anotamos todos los datos que queramos en los arcos.

Entonces ahora las matrices sé que me quedan cortas porque solo puedes añadir un dato. Los grafos cuando se aplican a la vida real no tienen porque solo tener un camino.

Modelar en datos el mundo real.

Google tiene un grafo de conocimiento porque cuando tú buscas cervantes, Google te dice nacido en no sé qué y enlace a Google maps y toda la información está conectada.

Los grafos de conocimiento nos permiten relaciona roto tipo de información, establecer una estructura de relaciones y después filtrarla.

El pensamiento en grafos matemático e informático se diferencian en el problema que se resuelve. Es un campo de investigación activo. El enfoque es distinto.

En informática queremos representar de la forma más fielmente posible la realidad para después aplicar ciertos algoritmos. Son limitadas entonces nos vamos a dedicar a ver cuáles son los distintos tipos de implementaciones (4) y algorítmica.