**TEORIA ARBOLES**

**Árbol**: estructura jerárquica compuesta por una colección finita de objetos.

**Grado de un nodo**: número de descendientes directos (nº hijos).

**Grado de un árbol**: máximo grado que pueden llegar a tener sus nodos (nº max hijos).

**Nodo hoja**: nodo que no tiene descendientes.

**Antecesores de un nodo X**: camino desde la raíz hasta llegar a un nodo.

**Nivel de un nodo**: lugar que ocupa un nodo dentro de la jerarquía (raíz = nivel 1).

**Profundidad de un árbol:** nivel del nodo máximo.

**Bosque**: conjunto de árboles que no comparten nodos entre sí.

**Número máximo de nodos:**

* en el nivel ‘i’ de un árbol binario es 2i-1 para i >= 1.
* de un árbol binario de profundidad k es 2k-1.

**Árbol lleno**: de profundidad k, es aquel que tiene 2k-1 nodos.

**Árbol completo**: de profundidad k, es aquel que está lleno hasta el nivel k-1, y en el ultimo nivel de izquierda a derecha.

**Árbol sesgado**: solo tiene hijos izquierdos/derechos.

**Representación secuencial**: por niveles de izquierda a derecha (espacios en blanco se respeta el número de posición) (array)

**Recorrido inorden(IRD)**: hijo izquierdo + raíz + hijo derecho

**Recorrido preorden(RID)**: raíz + hijo izquierdo + hijo derecho

**Recorrido postorden(IDR)**: hijo izquierdo + hijo derecho + raíz

**Recorrido anchura (jerárquica):** por niveles de izquierda a derecha

**Árbol binario de búsqueda**: árbol donde hijo izquierdo menor que raíz e hijo derecho mayor que raíz.

**Árbol AVL**: árbol con la característica de tener los mismo nodos o diferencia de uno entre el subárbol izquierdo y derecho.

**Factor de equilibrio**: nodo izquierdo – nodo derecho (1 o 0, esta bien; distinto hay que hacer rotaciones)

Cuando nodo izquierdo mayor nodo derecho, sustituimos el nodo mayor del subárbol izquierdo por donde se desequilibra el árbol, he insertamos en orden los elementos.

Cuando nodo derecho mayor que nodo izquierdo, sustituimos el nodo menor del subárbol derecho por donde se desequilibra el árbol, he insertamos en orden los elementos.

Rotación X – X simple: nodo insertado hijo X del subárbol X

Rotación X – Y doble: nodo insertado hijo Y del subárbol X

**HEAP (montículo)**: árbol binario completo donde la raíz es menor que sus hijos, y estos a su vez son HEAP también. Si al insertar no se cumple la condición HEAP, haremos rotaciones hasta que se cumpla esta.

**GRAFOS**

**Grafo**: estructura de datos relacional compuesta por una colección finita de objetos.

**Predecesor** (GD): vértices de origen. A predecesor de B (A 🡪 B)

**Sucesor** (GD): vértices de destino. B sucesor de A (A 🡪 B)

**Grado de un vértice:** numero de aristas que llegan y salen de este.

**Grado de un grafo**: grado del vértice con mayor grado.

**Camino**: recorrido que se puede seguir para llegar de un vértice a otro siguiendo el sentido de las aristas.

**Camino simple**: no pasas dos veces por un mismo vértice.

**Ciclo**: es un camino cerrado, donde el primer vértice y el ultimo son el mismo.

*Todo ciclo es un camino, pero no todo camino es un ciclo.*

**Cadena**: es un camino donde no importa la dirección de las aristas.

**Cadena simple**: no pasas dos veces por un mismo vértice.

**Circuito**: cuando el primer y último vértice de la cadena son el mismo.

*Todo camino es una cadena, pero no toda cadena es un camino.*

*Todo ciclo es un circuito, pero no todo circuito es un ciclo.*

**Camino** -> se tiene en cuenta la dirección de las aristas (ciclo)

**Cadena** -> no se tiene en cuenta la dirección de las aristas (circuito)

**esArbol**() 🡪 cada vértice tenga solo un predecesor y varios sucesores. Además, mi vértice no puede estar en la lista de sucesores de otros vértices

**esFuente**() 🡪 un nodo del que solo salen aristas

**esSumidero**() 🡪 un nodo del que solo entran aristas

**Recorrido en profundidad**: desde un vértice, voy recorriendo todos los demás vértices sin repetirse.

**Recorrido en anchura**: selecciono un vértice y visito todos sus adyacentes, y después los adyacentes de los adyacentes.

**Subgrafo**: subconjunto de un grafo, que no tiene por qué respetar las conexiones.

**Subgrafo inducido**: conserva todas las conexiones entre el conjunto de vértices correspondiente.

**Grafo conexo**: si existe un camino entre cualquier par de vértices (desde cualquier vértice puedo llegar a otro)

**Fuertemente conexo**: existe camino entre cualquier par de vértices

**Débilmente conexo**: existe una cadena entre cualquier par de vértices.

**Vértice aislado**: no tiene conexión con ninguna conexión con otro vértice.

**Grafo completo**: hay una arista entre cualquier par de vértices.

**Puntos de articulación**: vértice que hace que un grafo sea conexo, si este se elimina se quedan dos grafos.

**Grafo biconexo** es un grafo que no tiene puntos de articulación.

**Componente conexa**: subgrafo inducido por subconjunto de vértices que son conexos.

**Árbol de recubrimiento**: estructura que permite llegar a todos los vértices de un grafo sin formar ciclos.

**Árbol de recubrimiento mínimo**: árbol de recubrimiento cogiendo las aristas con menor coste.

**Algoritmo Prim**: desde un vértice, recorro todos los vértices por el camino de menor coste.

**Algoritmo Kruskal**: recorro todos los vértices con las aristas de menor coste sin formar ciclos.

**AVANCE RAPIDO**

**Solución factible**: cualquier solución que satisfaga las restricciones del problema.

**Solución óptima**: solución factible con valor óptimo.

**Algoritmo cambio mínimo**: llegar al precio con las menos monedas posibles

**Algoritmo mochila**: llenar la mochila con porcentajes de los valores.

* S1 🡪 seleccionar en cada caso el mayor beneficio
* S2 🡪 seleccionar en cada caso el menor peso
* S3 🡪 seleccionar en cada caso la relación mayor beneficio/peso.