

# RECORRIDO GRAFOS

BFS      VS      DFS

USAMOS : Queue      Stack

CUANDO : Buscamos Raiz      Buscamos Hojas

COMP :  $\Theta(\max(A,V))$        $\Theta(\max(A,V))$

MEMOR : -      +

¿ Cómo elegir cual usar ?

→ Distancia k de un vertice : BFS

→ Último nodo : DFS



# RECORRIDOS GRAFOS

**BFS :** Recorrido en anchura, visitamos todos los nodos distancia k antes de pasar a los k+1

**Complejidad:**  $\Theta(\max(\text{Vert}, \text{Arist}))$

grafo = List[ArrayList]

**Pseudocódigo :**

List<Int>[] BFS (grafo)

Tenemos una lista para guardar las salidas y con un for rellenaremos las posiciones de la lista con el grafo y la posición, usando en el bucle grafo.length

List<Int> BFS (grafo, vertice)

Declaramos la salida como ArrayList y un boolean[] visitados

Queue<Int> cola = LinkedList()

ponemos el vertice como visitado y lo añadimos a la salida y a la cola

while (!cola vacia)

guardamos en un auxiliar cola.remove que es Int

for (Int ady : grafo[aux])

if (!visit[ady])

lo ponemos como visitado, lo añadimos a la salida y a la cola.

Devolvemos el ArrayList

**DFS:** Recorremos el grafo en profundidad.

**Complejidad:**  $\Theta(\max\{Vert., Arist.\})$

**Pseudocódigo:**  $grafo = \text{List}[\text{ArrayList}]$

**List <Int> DFS (grafo)**

Tenemos una lista para guardar las salidas con un for rellenamos la lista llamando a DFS con el grafo y la posición ( $i$ )

**List <Int> DFS (grafo, vertice)**

Tenemos un ArrayList de salida y un array de visitados

**Stack <Int> pila = new Stack<>()**

Ponemos el vértice como visitado y lo añadimos a salida y a pila

**while (!pila vacia)**

Guardamos en un entero el aux con pila.pop()

**for (int ady : grafo [aux])  
if (!visitado [ady])**

Lo ponemos en visitado, lo añadimos a salida y pila (pila.push())

Devolvemos la salida