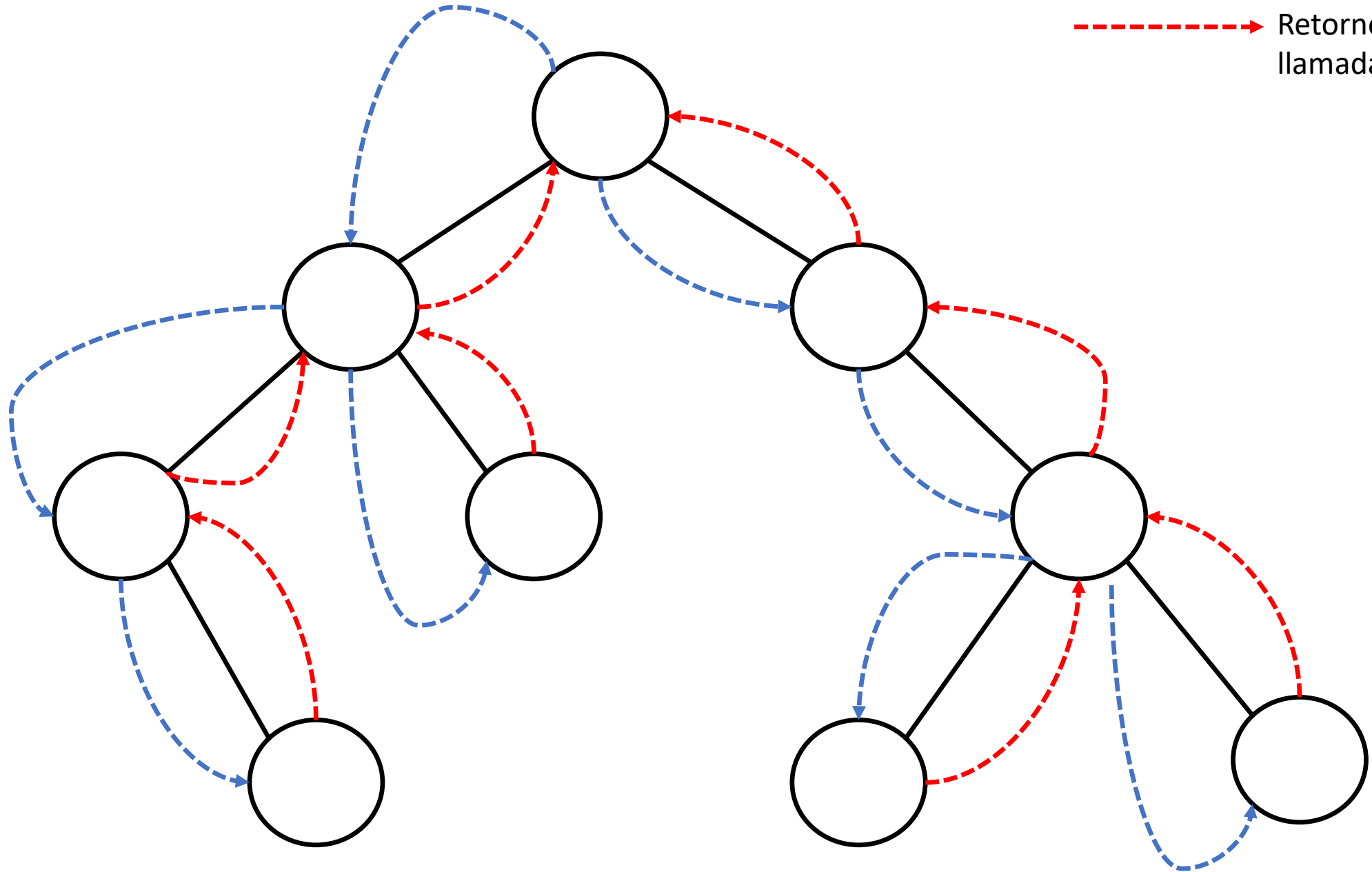


Algoritmos de árboles binarios basados en su recorrido

-----> Llamada recursiva
-----> Retorno de llamada recursiva



Algoritmos de árboles binarios basados en su recorrido

de TNodeArbolBinario elMetodo (): de algún tipo

COM

HACER ALGO EN PREORDEN, por ejemplo inicializar variables locales, o comprobar algo y salir devolviendo

Si hijoIzq != nulo entonces

resultadoIzquierdo = hijoIzq.elMetodo()

fin si

HACER ALGO EN INORDEN, por ejemplo decidir si se llama al hijo derecho o no

hijoDer != nulo entonces

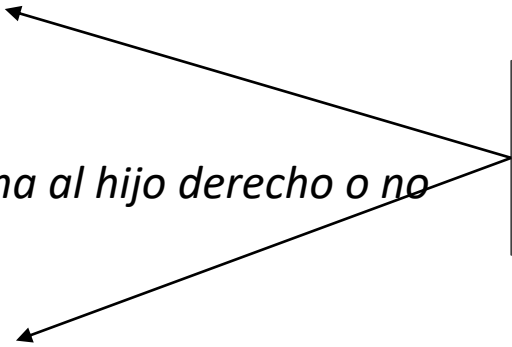
resultadoDerecho = hijoDer.elMetodo()

fin si

HACER ALGO EN POSTORDEN, por ejemplo componer ambos resultados y salir devolviendo

FIN

Se divide el problema en dos subproblemas menores y no superpuestos (disjuntos)



de TNodeArbolBinario cantidadHojas(): entero

COM

Si hijoIzq == nulo Y hijoDer == nulo entonces
 devolver 1

fin si

hojasIzquierdo = 0

hojasDerecho = 0

Si hijoIzq != nulo entonces
 hojasIzquierdo = hijoIzq.cantidadHojas()

fin si

Si hijoDer != nulo entonces
 hojasDerecho = hijoDer.cantidadHojas()

fin si

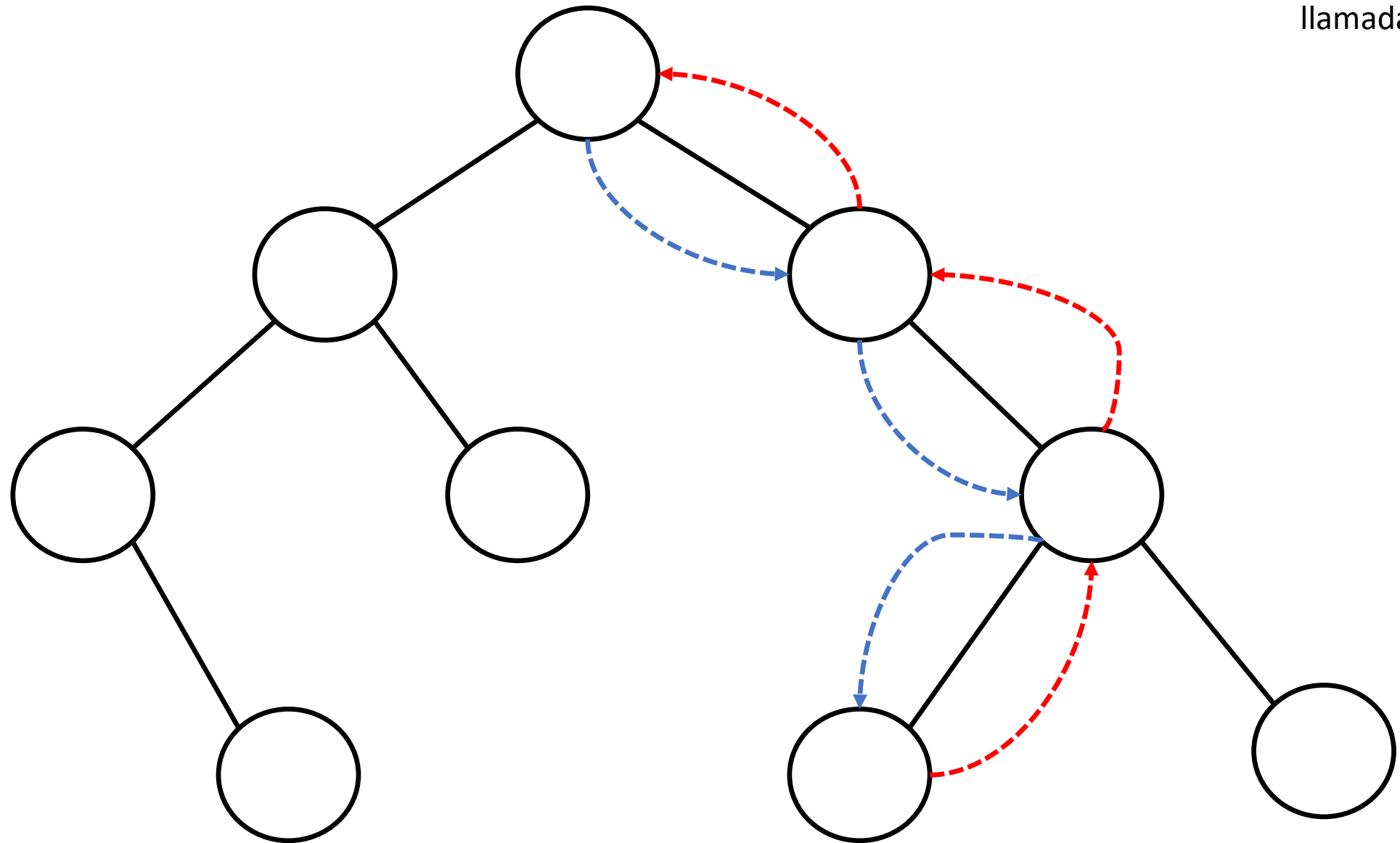
devolver hojasIzquierdo + hojasDerecho

FIN

Algoritmos de árboles binarios basados en su recorrido,
En general también pueden ser aplicados a
Árboles binarios de búsqueda

Algoritmos de árboles binarios de búsqueda basados en su propiedad

-----> Llamada recursiva
-----> Retorno de llamada recursiva



Algoritmos de árboles binarios de búsqueda basados en su propiedad: búsqueda, búsqueda e inserción, búsqueda y eliminación

de TNodeArbolBinarioBusqueda elMetodo (unaClave): de algún tipo

COM

inicializar variable **resultado** del tipo apropiado a devolver // eventualmente, inicializar otras variables locales

Si unaClave == etiqueta entonces

*// unaClave está en el árbol, eventualmente hacer algo, por ejemplo devolver un valor de **resultado***

Sino

Si unaClave < etiqueta entonces *// hay que buscar en el subárbol izquierdo, si lo tiene*

Si hijoIzq != nulo entonces *// tiene subárbol izquierdo, llamar recursivamente*

resultado = hijoIzq.elMetodo(unaClave)

sino *// no tiene subárbol izquierdo*

*// unaClave no está en el árbol, eventualmente hacer algo, por ejemplo devolver un valor de **resultado***

fin si

Sino *// unaClave es mayor que etiqueta, hay que buscar en el subárbol derecho, si lo tiene*

Si hijoDer != nulo entonces *// tiene subárbol izquierdo, llamar recursivamente*

resultado = hijoDer.elMetodo(unaClave)

sino *// no tiene subárbol derecho*

*// unaClave no está en el árbol, eventualmente hacer algo, por ejemplo devolver un valor de **resultado***

fin si

fin si

Fin si

*// eventualmente componer (calcular, procesar, preparar) el **resultado** a devolver*

devolver resultado

FIN

Una u otra llamada, nunca las dos. En cada paso descarta una parte del espacio de búsqueda