

Estructuras de Datos no Lineales

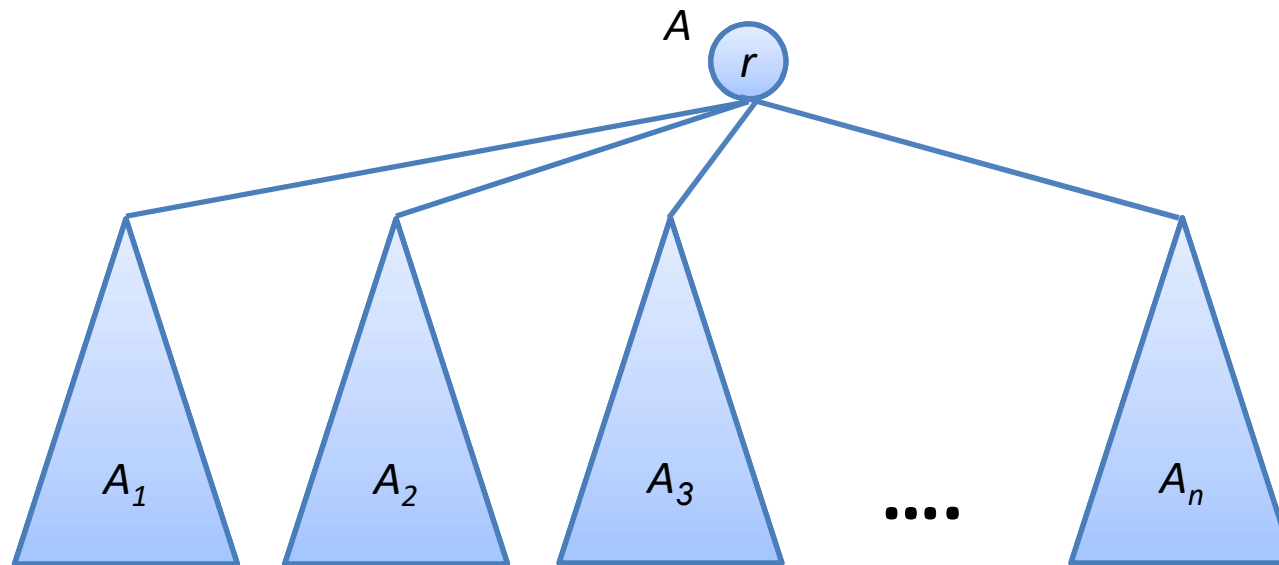
1.4. Recorridos de árboles generales

José Fidel Argudo Argudo
José Antonio Alonso de la Huerta
M^a Teresa García Horcajadas



Versión 2.0

Recorridos en profundidad de árboles generales

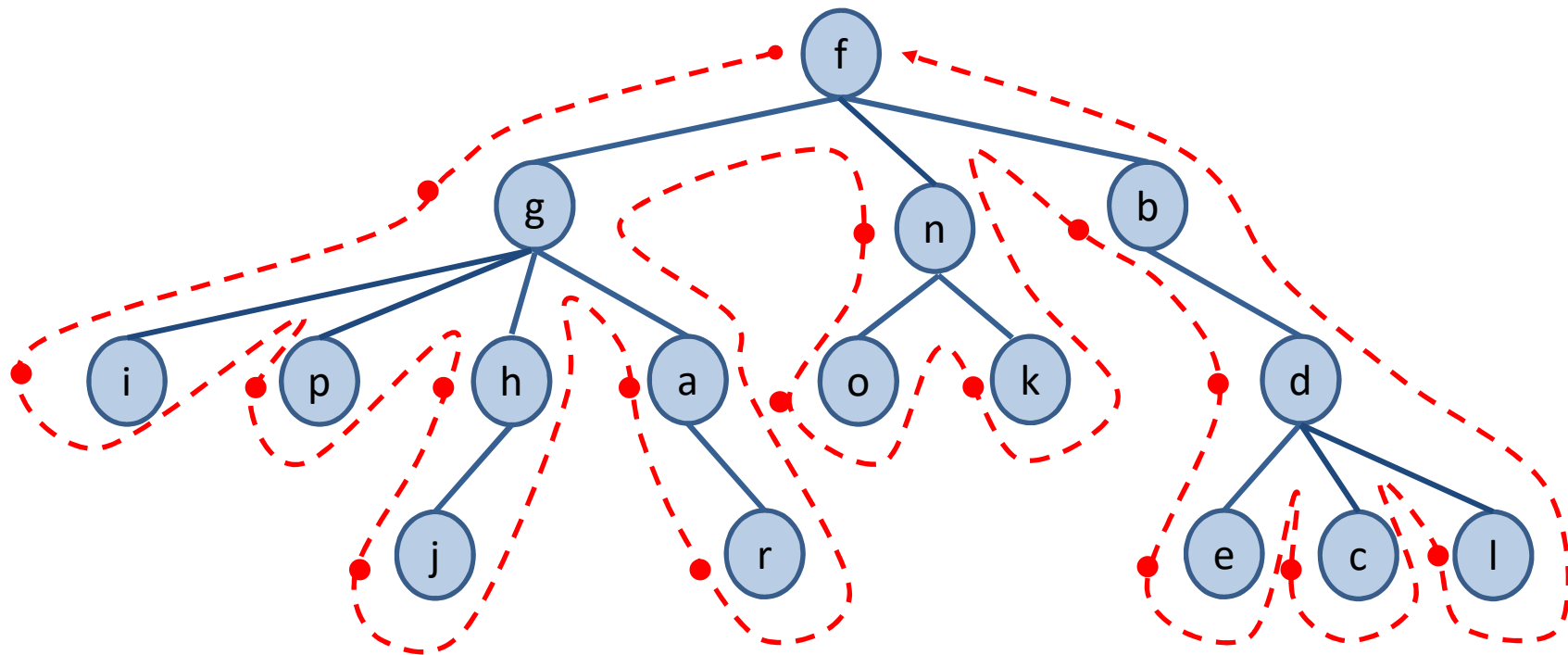


Preorden(A) = r Preorden(A_1) Preorden(A_2) Preorden(A_3)... Preorden(A_n)

Inorden(A) = Inorden(A_1) r Inorden(A_2) Inorden(A_3)... Inorden(A_n)

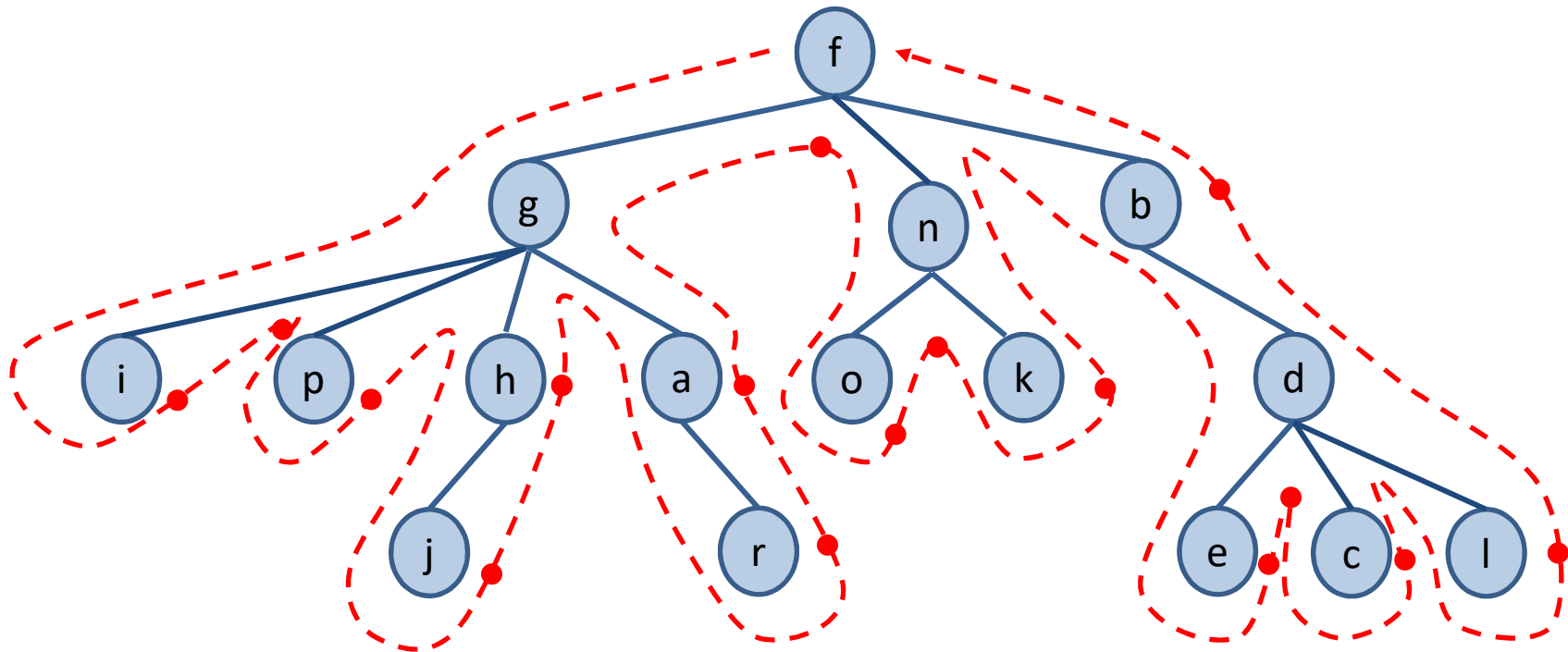
Postorden(A) = Postorden(A_1) Postorden(A_2) Postorden(A_3)...Postorden(A_n) r

Recorrido en preorden de árboles generales



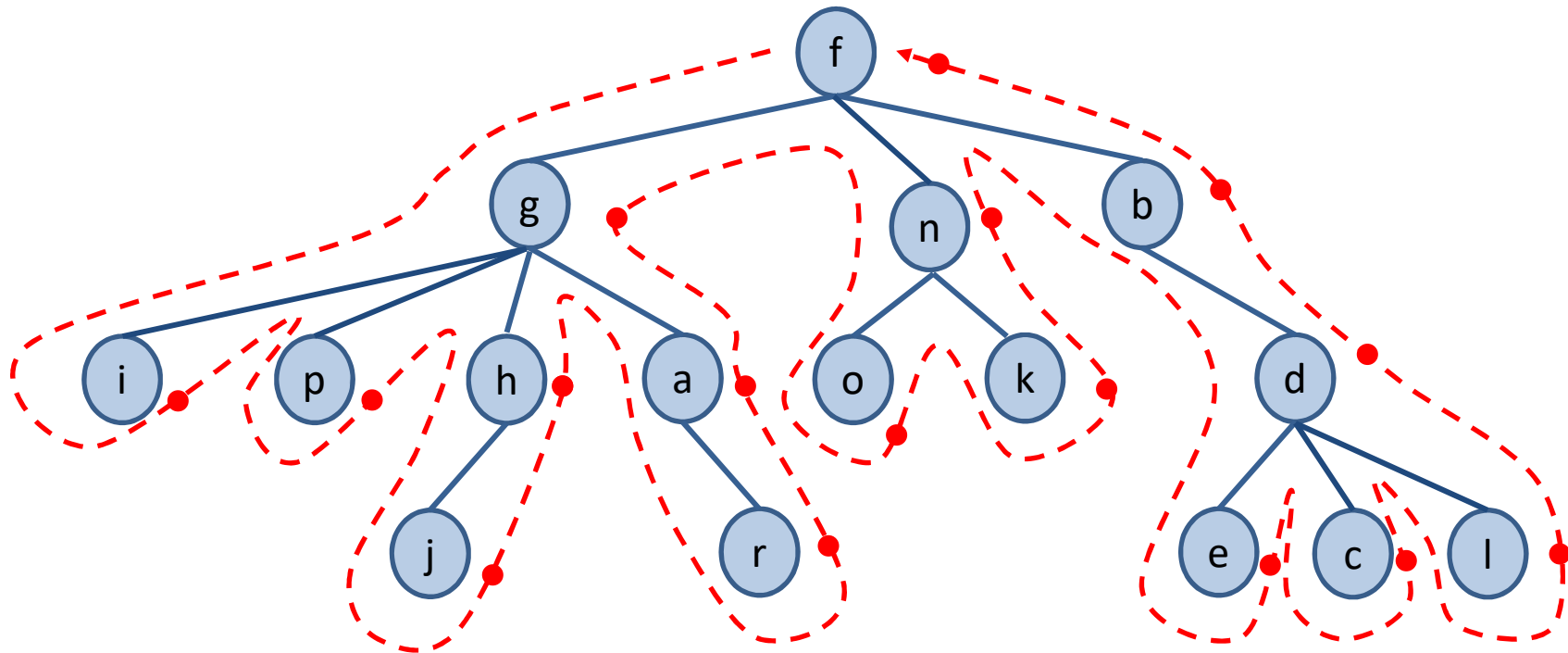
Preorden: fgiphjarnokbdecl

Recorrido en inorden de árboles generales



Inorden: igpjhrafonkedclb

Recorrido en postorden de árboles generales



Postorden: ipjhragoknecldbf

Implementación recursiva de recorridos en profundidad de árboles generales

```
template <typename T>
void preordenAgen(typename Agen<T>::nodo n, const Agen<T>& A,
                 void (*procesar)(typename Agen<T>::nodo, const Agen<T>&))
// Recorrido en preorden del subárbol cuya raíz es el nodo n
// perteneciente al árbol A. Cada nodo visitado se procesa mediante
// la función procesar().
{
    if (n != Agen<T>::NODO_NULO) {
        procesar(n, A);
        n = A.hijoIzqdo(n);
        while (n != Agen<T>::NODO_NULO) {
            preordenAgen(n, A, procesar);
            n = A.hermDrcho(n);
        }
    }
}
```

```

template <typename T>
void inordenAgen(typename Agen<T>::nodo n, const Agen<T>& A,
                void (*procesar)(typename Agen<T>::nodo, const Agen<T>&))
// Recorrido en inorden del subárbol cuya raíz es el nodo n
// perteneciente al árbol A. Cada nodo visitado se procesa mediante
// la función procesar().
{
    if (n != Agen<T>::NODO_NULO)
    {
        typename Agen<T>::nodo hijo = A.hijoIzqdo(n);
        if (hijo != Agen<T>::NODO_NULO)
        {
            inordenAgen(hijo, A, procesar);
            procesar(n, A);
            while ((hijo = A.hermDrcho(hijo)) != Agen<T>::NODO_NULO)
                inordenAgen(hijo, A, procesar);
        }
        else
            procesar(n, A);
    }
}

```

```

template <typename T>
void postordenAgen(typename Agen<T>::nodo n, const Agen<T>& A,
                  void (*procesar)(typename Agen<T>::nodo, const Agen<T>&))
// Recorrido en postorden del subárbol cuya raíz es el nodo n
// perteneciente al árbol A. Cada nodo visitado se procesa mediante
// la función procesar().
{
    if (n != Agen<T>::NODO_NULO) {
        typename Agen<T>::nodo hijo = A.hijoIzqdo(n);
        while (hijo != Agen<T>::NODO_NULO) {
            postordenAgen(hijo, A, procesar);
            hijo = A.hermDrcho(hijo);
        }
        procesar(n, A);
    }
}

template <typename T>
void escribirNodo (typename Agen<T>::nodo n, const Agen<T>& A)
{
    if (n != Agen<T>::NODO_NULO)
        std::cout << A.elemento(n) << ' ';
}

```


Implementación iterativa del recorrido en preorden de árboles generales

```
#include "pilaenla.h"
template <typename T>
void preordenAgen2(typename Agen<T>::nodo n, const Agen<T>& A,
                  void (*procesar)(typename Agen<T>::nodo, const Agen<T>&))
{
    Pila<typename Agen<T>::nodo> P; // Pila de nodos de un árbol.
    do {
        if (n != Agen<T>::NODO_NULO) {
            procesar(n, A);
            if (A.hermDrcho(n) != Agen<T>::NODO_NULO)
                P.push(A.hermDrcho(n));
            n = A.hijoIzqdo(n);
        }
        else if (!P.vacia()) {
            n = P.tope(); P.pop();
        }
    } while (!(n == Agen<T>::NODO_NULO && P.vacia()));
}
```

Implementación del recorrido en anchura o por niveles de árboles generales

```
#include "colaenla.h"
template <typename T>
void recNivelesAgen(typename Agen<T>::nodo n, const Agen<T>& A,
                  void (*procesar)(typename Agen<T>::nodo, const Agen<T>&))
{
    Cola<typename Agen<T>::nodo> C; // Cola de nodos de un árbol.
    if (n != Agen<T>::NODO_NULO) {
        do {
            if (!C.vacia()) { n = C.frente(); C.pop(); }
            procesar(n, A);
            typename Agen<T>::nodo hijo = A.hijoIzqdo(n);
            while (hijo != Agen<T>::NODO_NULO) {
                C.push(hijo);
                hijo = A.hermDrcho(hijo);
            }
        } while (!C.vacia());
    }
}
```