Università di Pisa

Pietro Ducange

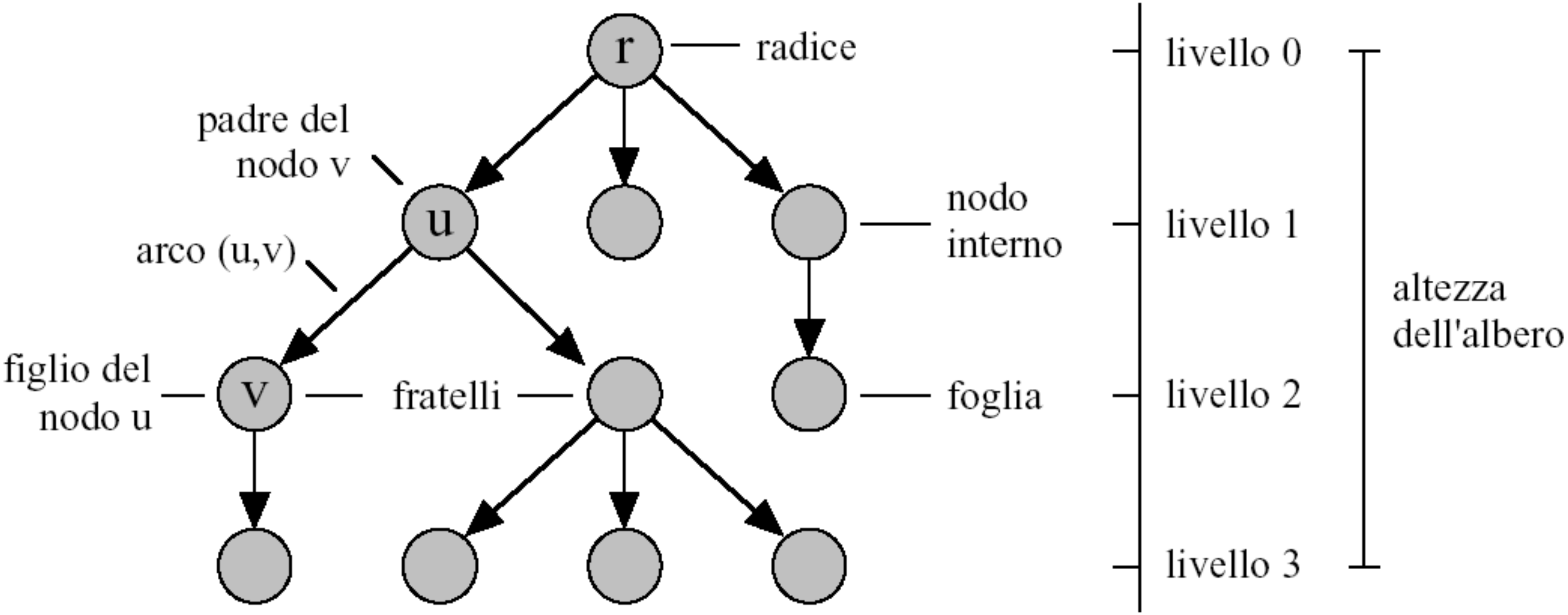
Algoritmi e strutture dati
Alberi Generici e Binari di Ricerca

a.a. 2020/2021

Si ringrazia la prof. Nicoletta De Francesco per aver messo a disposizione la maggior parte delle slide utilizzate nella presente lezione

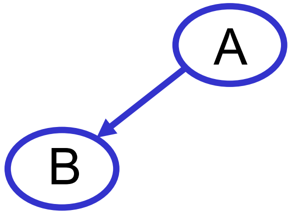
Alberi generici: definizione

- un nodo p è un **albero**
- un nodo + una **sequenza di alberi** A1 .. An è un albero

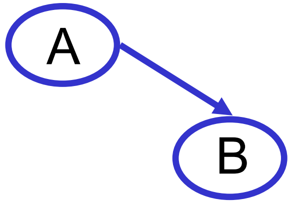


Alberi generici: differenza con alberi binari

alberi binari

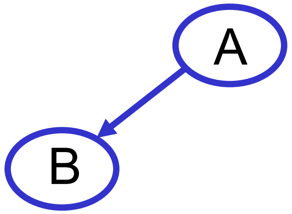


diverso da



sottoalbero destro vuoto **sottoalbero sinistro vuoto**

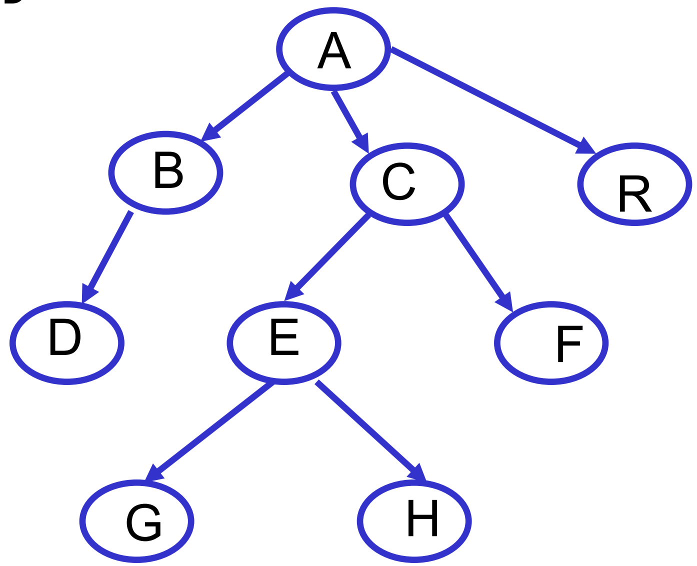
alberi generici



unico albero: radice: A, un sottoalbero

Alberi generici: visite

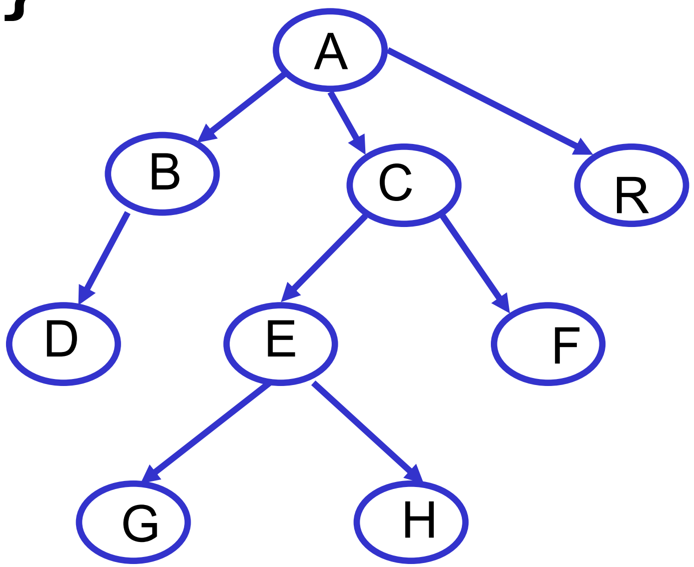
```
void preOrder ( albero ) {

        esamina la radice;
        se l'albero ha n sottoalberi {
                preOrder ( primo sottoalbero);
                ...
                preOrder ( n-esimo sottoalbero);
}
```



A B D C E G H F R

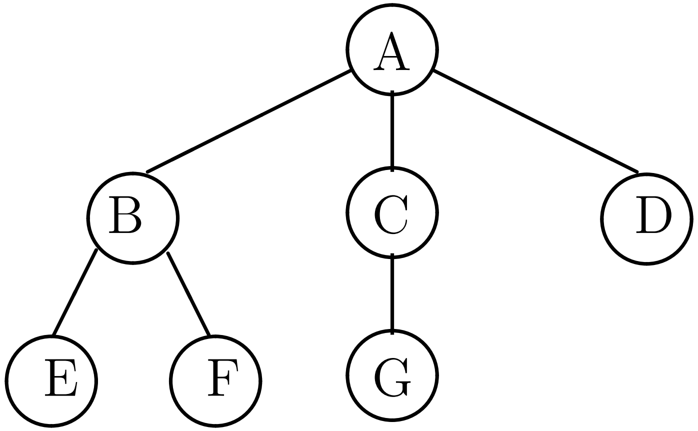
Alberi generici: visite

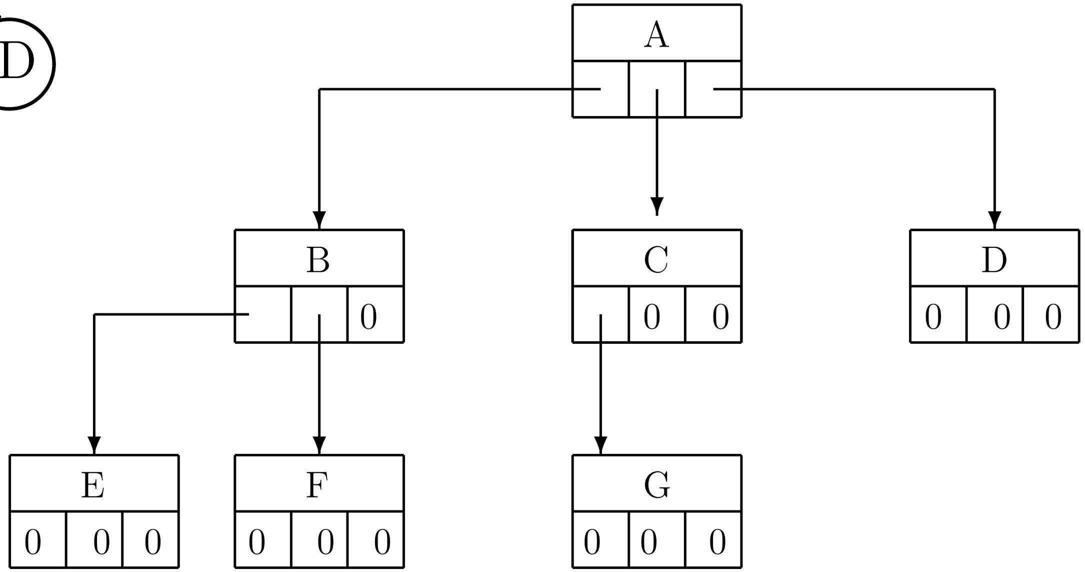
```
void postOrder ( albero ) {

        se l'albero ha n sottoalberi {
                postOrder ( primo sottoalbero);
                ...
                postOrder ( n-esimo sottoalbero);
        esamina la radice;
}
```



D B G H E F C R A

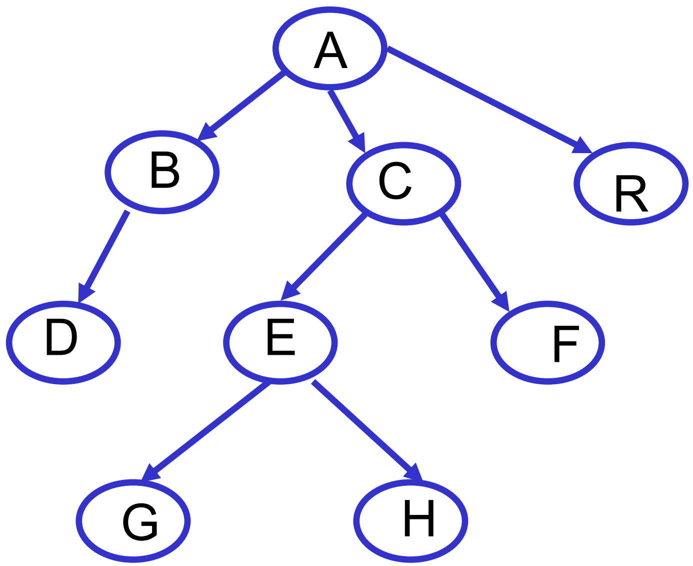
MEMORIZZAZIONE a liste multiple

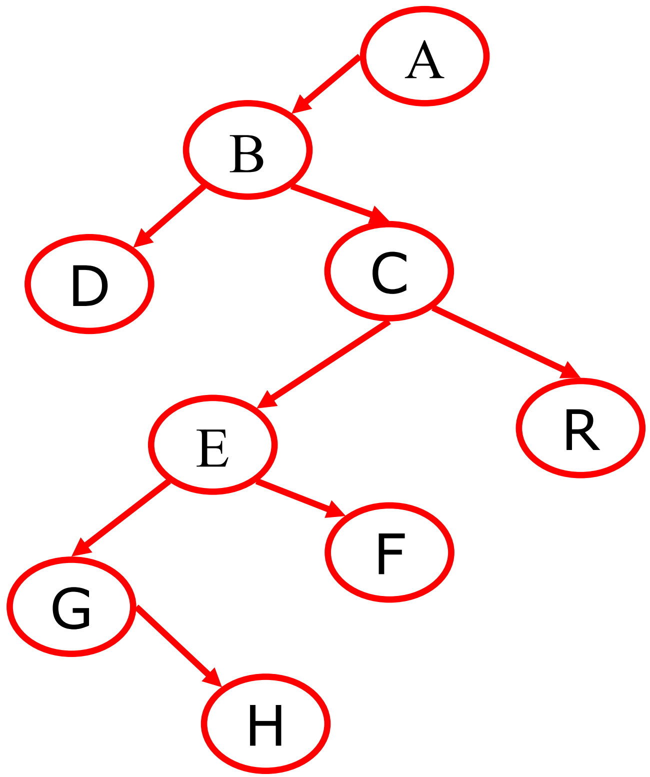




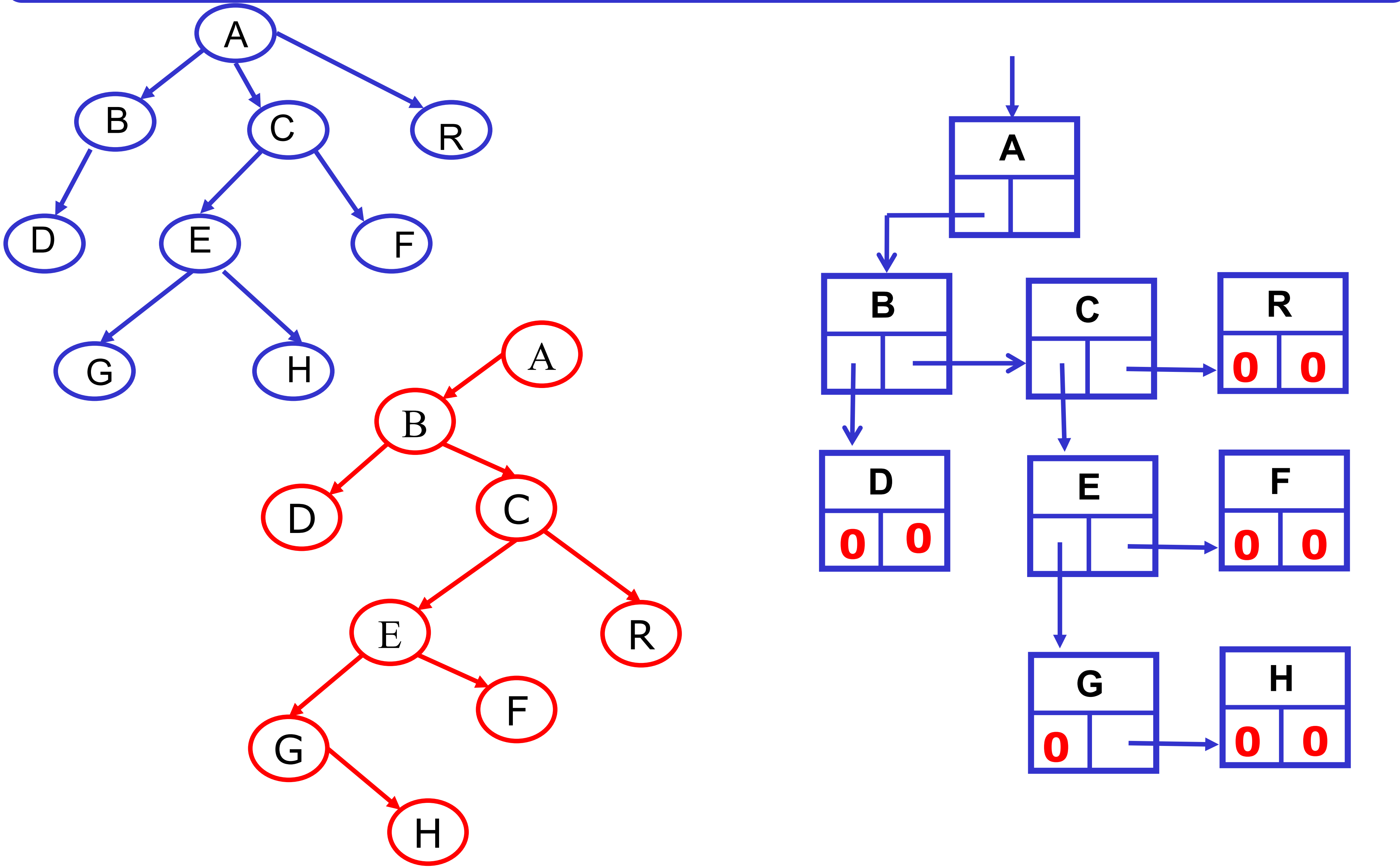
MEMORIZZAZIONE FIGLIO-FRATELLO

- **primo figlio a sinistra**
- **primo fratello a destra**

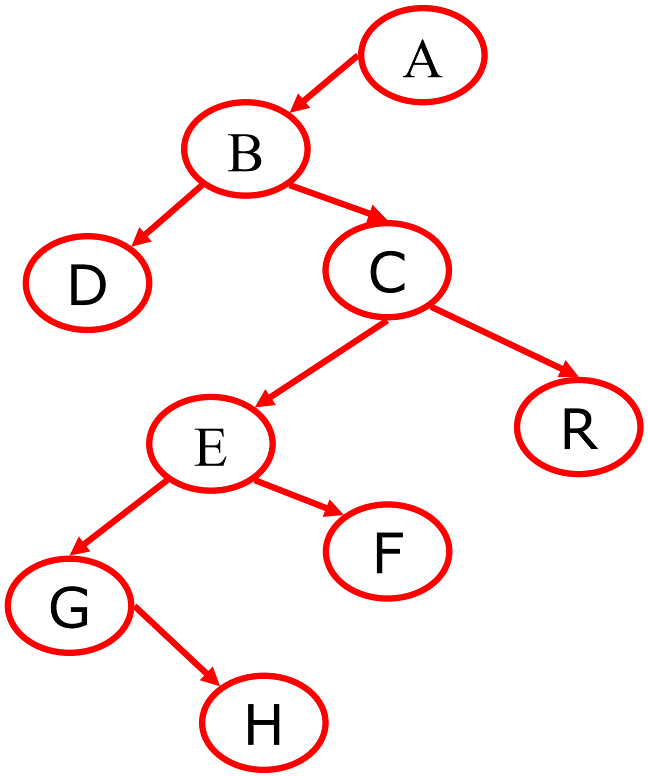




Alberi generici: memorizzazione



Esercizio



Dato il seguente albero binario effettuare:

La visita pre order: ???

La visita simmetrica: ????

Alberi generici: corrispondenza fra visite

Utilizzando la memorizzazione figlio-fratello:

La visita preorder del trasformato corrisponde alla visita preorder dell'albero generico

La visita inorder del trasformato corrisponde alla visita postorder dell'albero generic

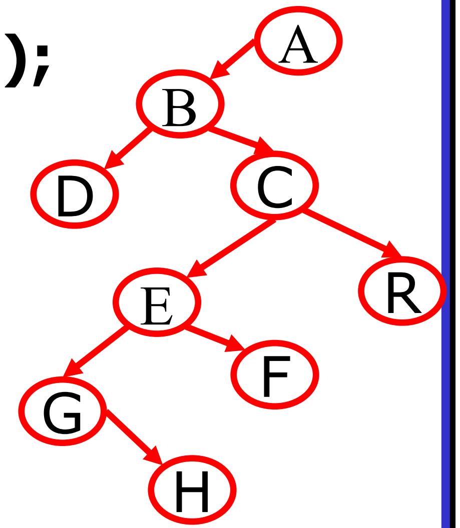
il tempo delle visite in un albero generico è lineare nel numero dei nodi.

Per la ricerca, l'inserimento e la cancellazione di un nodo, il tempo è comunque lineare. Infatti queste operazioni possono essere programmate mantenendo la struttura delle visite.

Esempi di programmi su alberi generici: conta nodi e foglie

conta i nodi (vedi albero binario)

```
int nodes (Node* tree) {
    if (!tree) return 0;
    return 1+nodes(tree->left)+nodes(tree->right);
}
```

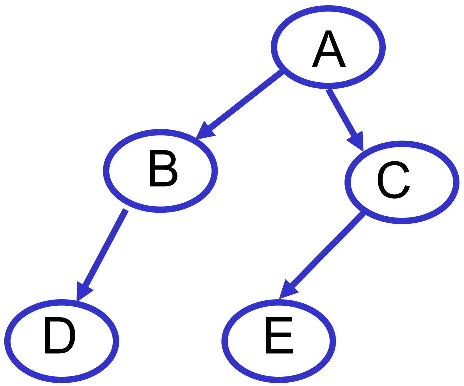


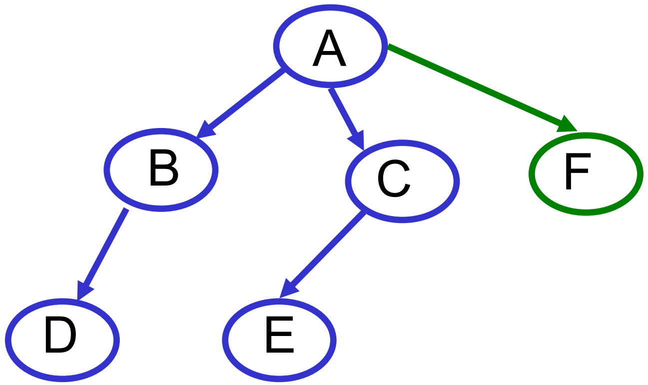
conta le foglie

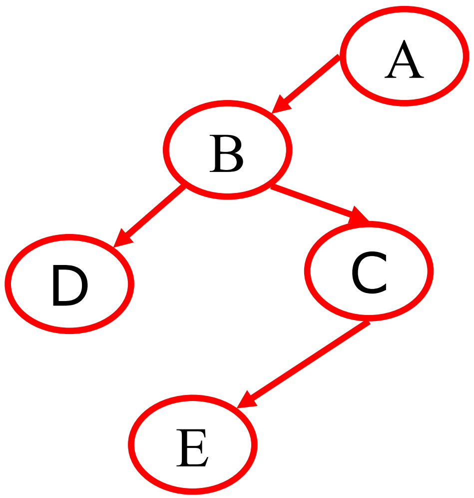
```
int leaves(Node* tree) {
    if (!tree) return 0;
    if (!tree->left) return 1+ leaves(tree->right); // foglia
    return leaves(tree->left)+ leaves(tree->right);
}
```

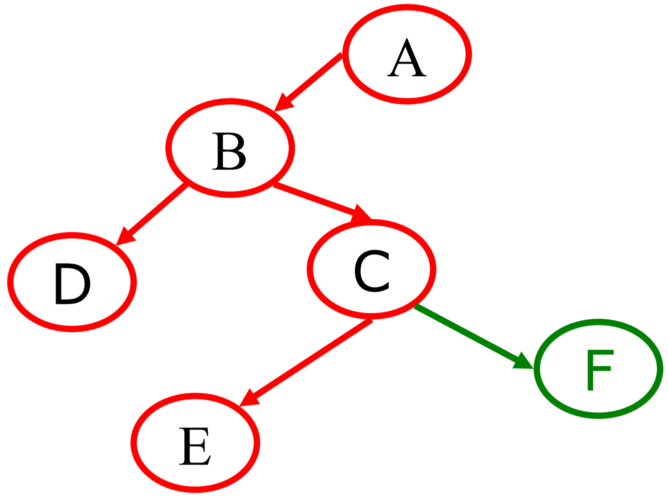
Esempi di programmi su alberi generici: inserimento

Inserisci F come ultimo figlio di A



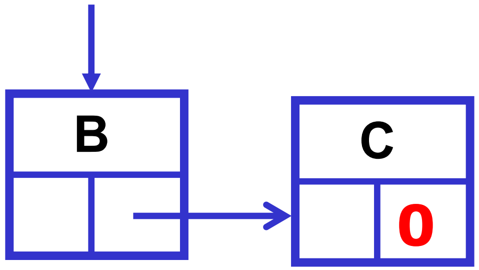






inserisce un nodo in fondo a una lista di fratelli

```
void addSon(InfoType x, Node* &list) {
  if (!list) {                    //lista vuota
    list=new Node;
    list>label=x;
    list->left = list->right = NULL;
  }
  else                            //lista non vuota
     addSon(x, list->right);
}
```

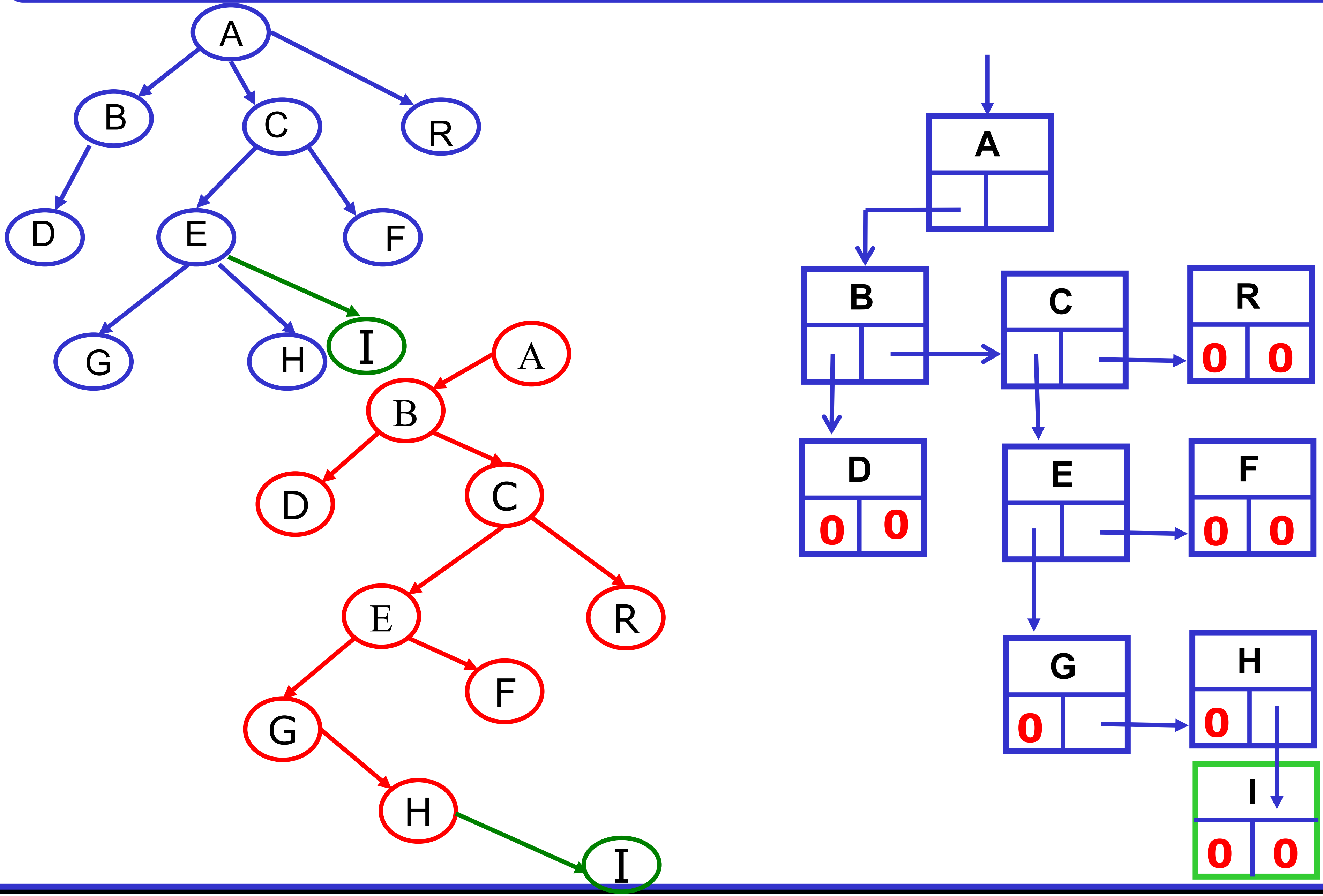


Esempi di programmi su alberi generici: inserimento

inserisce son come ultimo figlio di father.

```
int insert(InfoType son, InfoType father, Node* &tree) {
        Node* a=findNode(father, tree);   // a: puntatore di father
        if (!a) return 0;                 // father non trovato
        addSon(son, a->left);
        return 1;
 }
```

Alberi generici: memorizzazione



Riferimenti Bibliografici

Demetrescu:

Paragrafo 3.3

Cormen:

Paragrafo 10.4

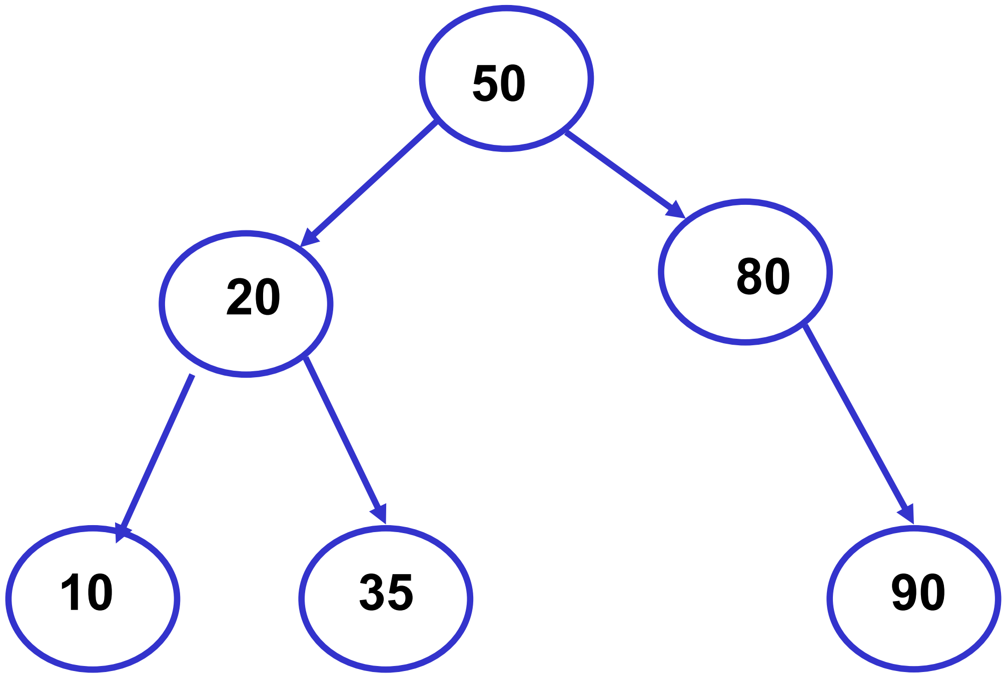
Alberi binari di ricerca: definizione

Un albero binario di ricerca è un albero binario tale che per ogni nodo p:

- i nodi del sottoalbero sinistro di p hanno etichetta minore dell'etichetta di p

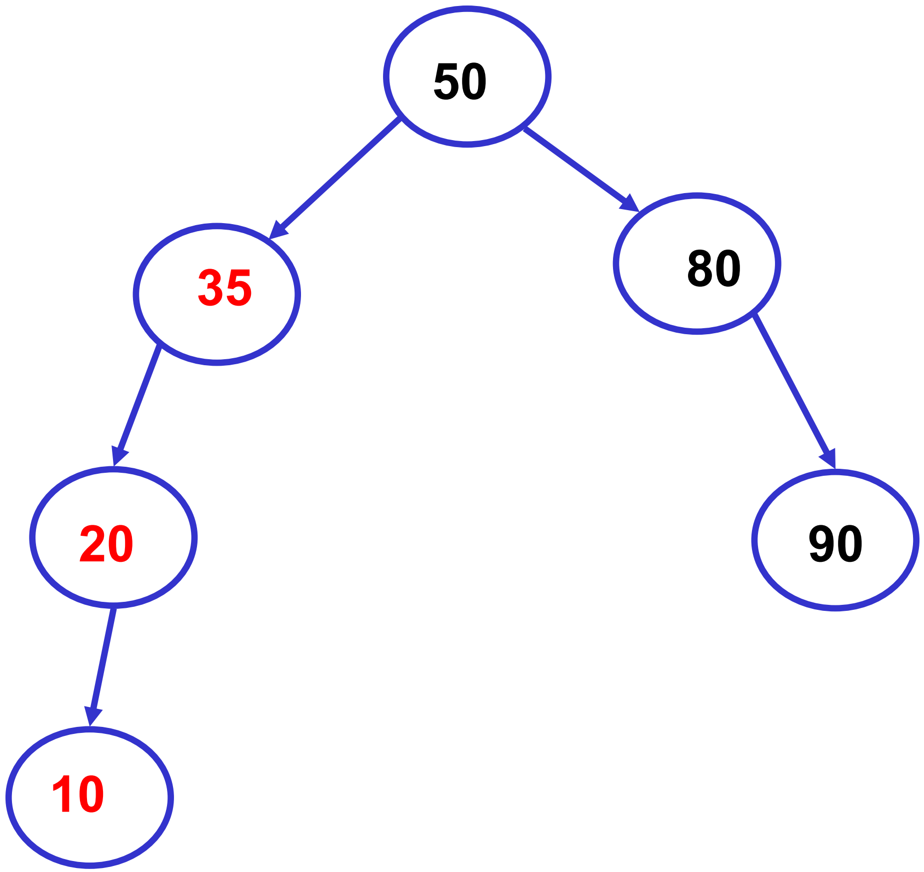
- i nodi del sottoalbero destro di p hanno etichetta maggiore dell'etichetta di p

Un albero binario di ricerca

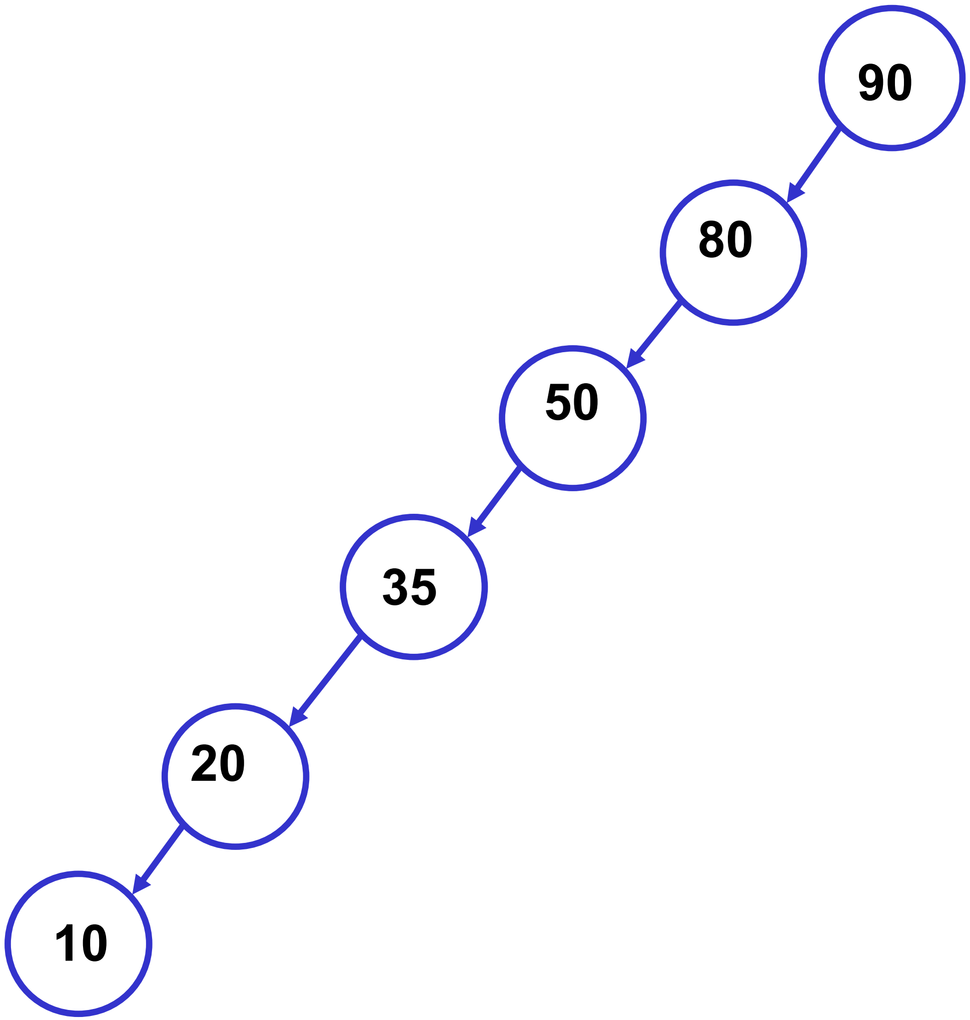


Dalla proprietà base segue che i nodi di un albero binario di ricerca hanno tutti etichette diverse

Un albero binario di ricerca con gli stessi nodi



Un albero binario di ricerca con gli stessi nodi



Alberi binari di ricerca: proprietà e operazioni

- **non ci sono doppioni**
- **la visita simmetrica elenca le etichette in ordine crescente**

OPERAZIONI

- **ricerca di un nodo**
- **inserimento di un nodo**
- **cancellazione di un nodo**

```
Node* findNode (InfoType n, Node* tree) {
  if (!tree) return 0;                       // albero vuoto

  if (n == tree->label) return tree;        // n=radice

  if (n<tree->label)                         // n<radice
     return findNode(n, tree->left);

  return findNode(n, tree->right);          // n>radice

}
```

$T(0)=a$

$T(n)= b + T(k) \quad k < n$

$T(0)=a$

$T(n)= b + T(n/2)$

$O(\log n)$

$T(0)=a$

$T(n)= b + T(n-1)$

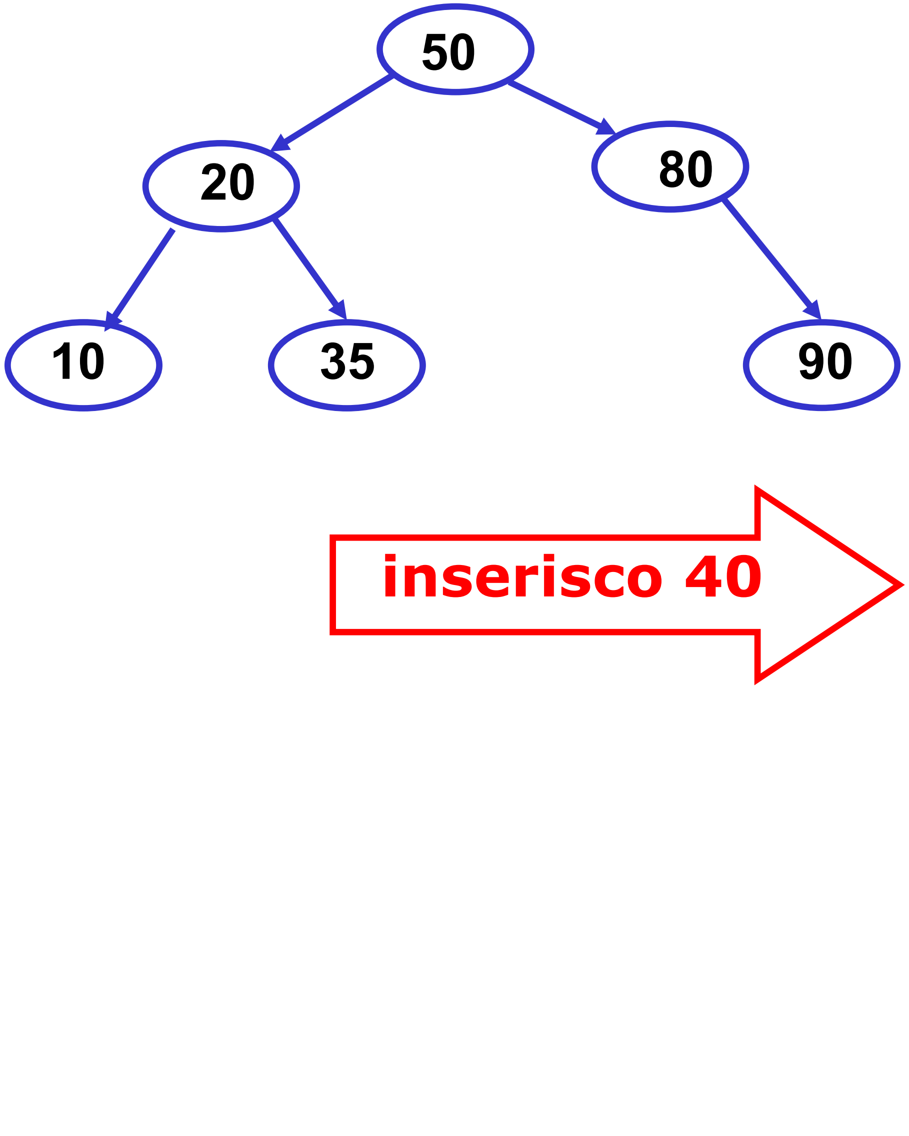
$O(n)$

in media : $O(\log n)$

```
void insertNode (InfoType n, Node* &tree) {
   if (!tree) {                // albero vuoto: creazione nodo
     tree=new Node;
     tree->label=n;
     tree->left = tree->right = NULL; return;
   }
   if (n<tree->label)                       // n<radice
         insertNode (n, tree->left);
   if (n>tree->label)                       // n>radice
         insertNode (n, tree->right);
 }
```

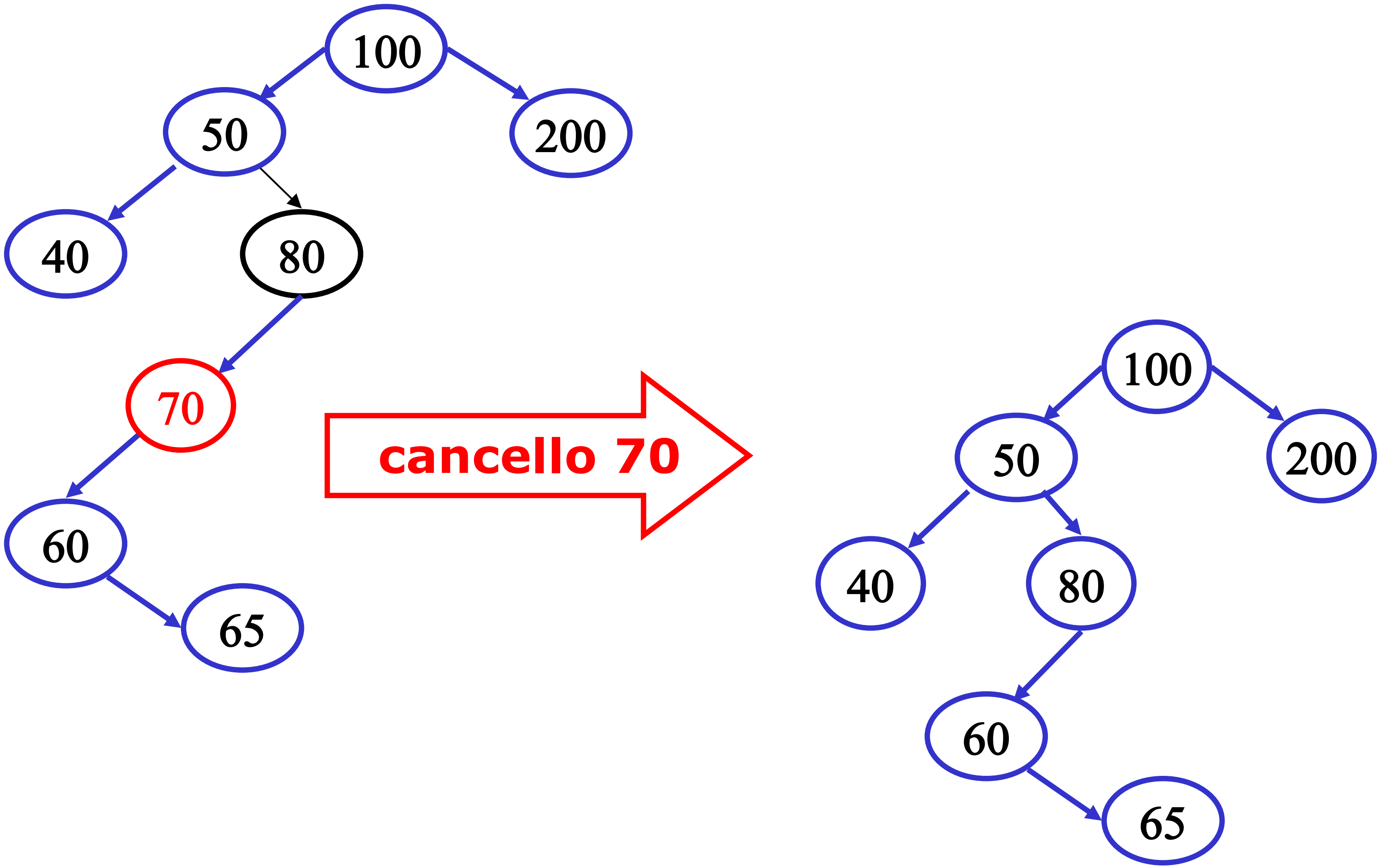
O(log n)

Esempio di inserimento

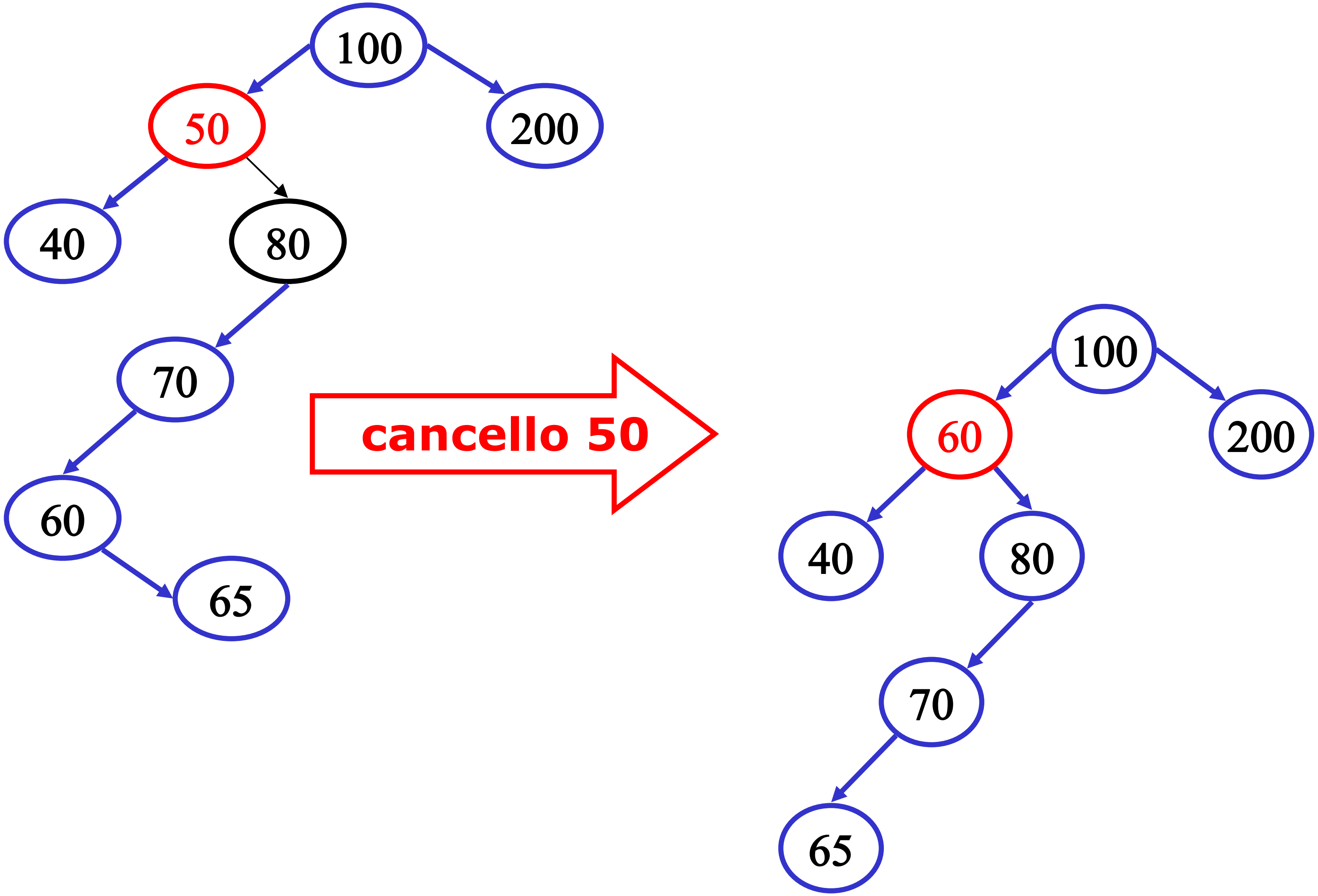


- Prima si cerca il nodo da cancellare effettuando una ricerca come negli algoritmi precedenti.
- Se il nodo viene trovato, sia esso p, possono verificarsi due situazioni diverse.
  - Se p ha un sottoalbero vuoto, il padre di p viene connesso all'unico sottoalbero non vuoto di p
  - Se p ha entrambi i sottoalberi non vuoti si cerca il nodo con etichetta minore nel sottoalbero destro di p, si cancella e si mette la sua etichetta come etichetta di p

Esempio di cancellazione: 1 Caso



Esempio di cancellazione: 2 Caso



restituisce l'etichetta del nodo più piccolo di un albero ed elimina il nodo che la contiene

```
void deleteMin (Node* &tree, InfoType &m) {
   if (tree->left)              //c'è un nodo più piccolo
         deleteMin(tree->left, m);
   else {
     m=tree->label;             //restitusco l'etichetta
     Node* a=tree;
     tree=tree->right;          //connetto il sottoalbero destro di
                                // m al padre di m
     delete a;                  //elimino il nodo
   }
 }
```

```
void deleteNode(InfoType n, Node* &tree) {
  if (tree)
    if (n < tree->label)                    //n minore della radice
            { deleteNode(n, tree->left);  return; }
    if (n > tree->label)                    //n maggiore della radice
            { deleteNode(n, tree->right); return; }
    if (!tree->left)                        //n non ha figlio sinistro
            { Node* a=tree; tree=tree->right; delete a;return;}
    if (!tree->right)                       //n non ha figlio destro
           { Node* a=tree; tree=tree->left; delete a; return;}
    deleteMin (tree->right, tree->label);   //n ha entrambi i figli
 }
```

O(log n)

Domande

In quanto tempo è possibile ricercare una chiave in un albero binario di ricerca di n elementi?

(a) $O(\log n)$
(b) $O(\sqrt{n})$
(c) $O(n)$
(d) $O(n^2)$

In quanto tempo è possibile trovare il minimo in un albero binario di ricerca bilanciato di n elementi?

(a) $O(\log n)$
(b) $O(\sqrt{n})$
(c) $O(n)$
(d) $O(n^2)$

Riferimenti Bibliografici

Demetrescu:

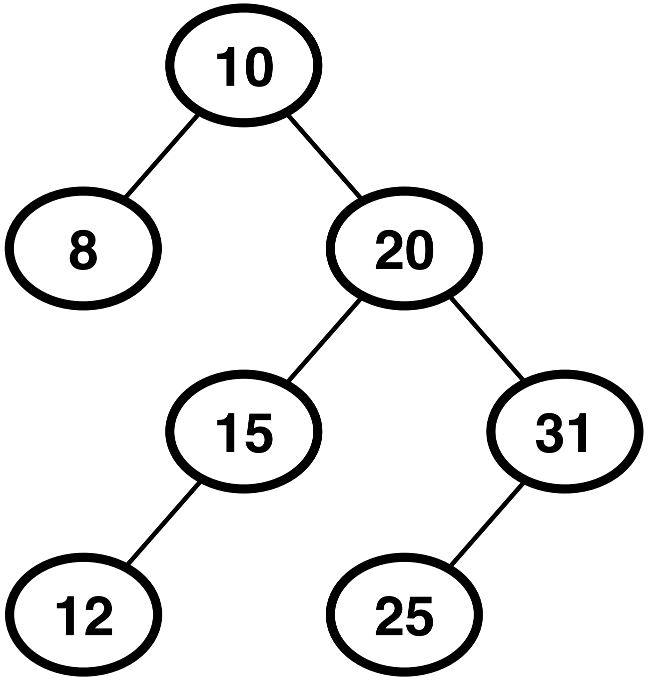
Paragrafo 6.1

Cormen:

Paragrafo 12.1, 12.2 e 12.3

Esercizio 1

Dato il seguente albero binario di ricerca:



Disegnare gli alberi risultanti dopo l'aggiunta del valore 27 e la successiva eliminazione del valore 20.

Altri esercizi

1. Scrivere un programma C++ che dato un albero generico, somma 1 ad ogni sua etichetta

2. Scrivere un programma C++ che somma ad ogni nodo di un albero generico il numero dei suoi figli