Alberi binari di ricerca

Luca Becchetti

Presentazione tratta dalle slide che accompagnano il testo Data Structures and Algorithms in Java, 6th edition, by M. T. Goodrich, R. Tamassia, and M. H. Goldwasser, Wiley, 2014

Mappe ordinate

- Chiavi totalmente ordinate
- Esiste un ordinamento delle coppie definito dalle chiavi
  - Es.: implementato mediante array ordinato
- Ricerca binaria efficiente
- Ricerche di tipo nearest neighbour
- V. lezione precedente

Limiti delle tabelle ordinate

- Ricerca ha complessità O(log n) ma ...
- Inserimenti/cancellazioni hanno costo O(n) nel caso peggiore
  - Spostamento elementi
- Poco usate in pratica
  - Dimensioni piccole
  - Inserimenti/cancellazioni rari

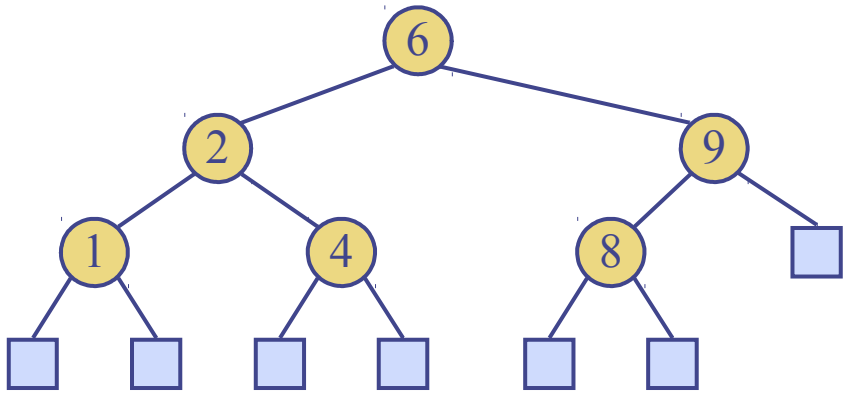
Alberi binari di ricerca

Albero binario di ricerca

- Albero binario ai cui nodi interni sono associate coppie (k, v)
- Proprietà *chiave*:
  - Se se u e w sono rispettivamente nel sottoalbero sinistro e destro di v:

    $$key(u) \leq key(v) \leq key(w)$$
- I nodi esterni non hanno elementi associati

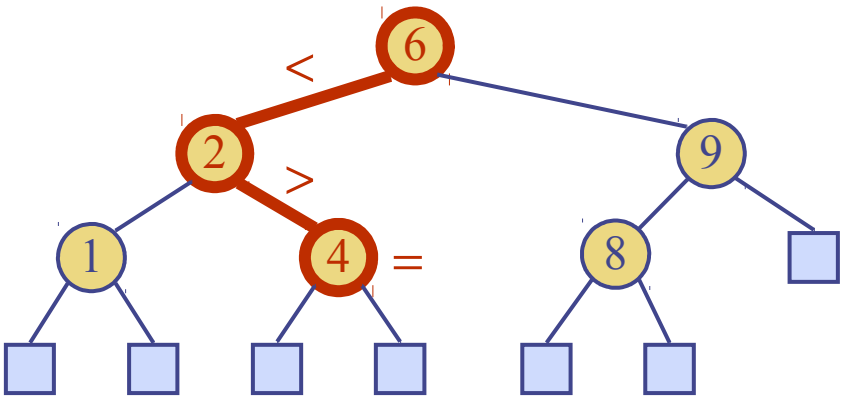
- Visita simmetrica (in-order)
- Visita le chiavi in ordine crescente
  - *Dimostrare (più avanti)*



Ricerca di una chiave in un BST

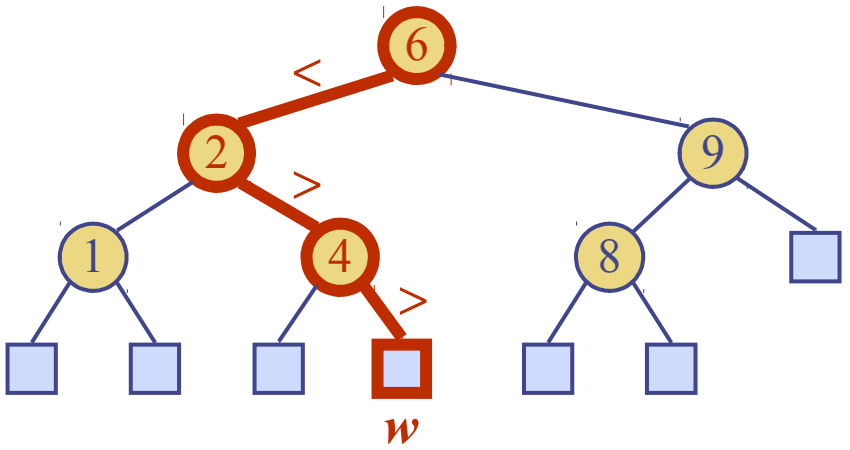
- Si segue un percorso dalla radice
- Chiave assente se viene raggiunta una foglia
- Esempio: get(4) → TreeSearch(4, root)
- Esempio: get(5)
  - Si raggiunge il figlio destro di 4 (foglia)
  - 4 è la chiave più grande tra quelle più piccole di 5
- I BST consentono di effettuare ricerche di tipo nearest neighbour

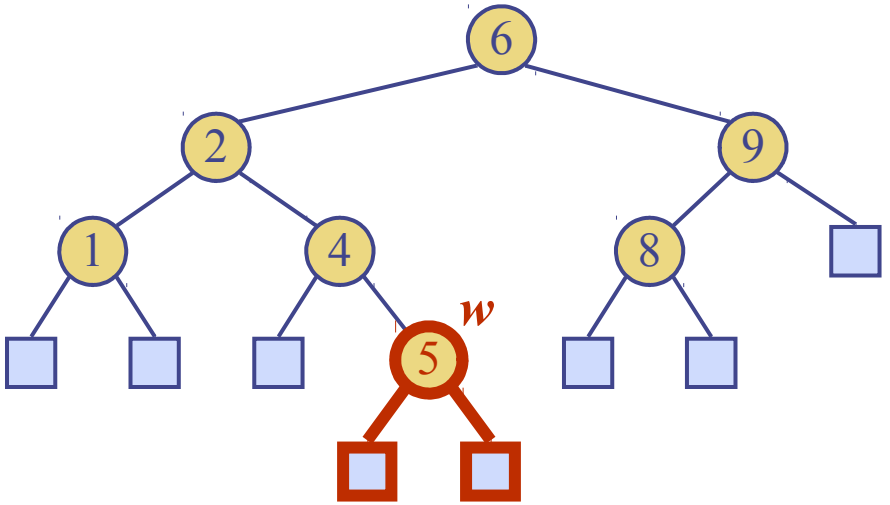
```
Algorithm TreeSearch(k, v)
    if T.isExternal (v)
        return v
    if k < key(v)
        return TreeSearch(k, left(v))
    else if k = key(v)
        return v
    else { k > key(v) }
        return TreeSearch(k, right(v))
```



Inserimento di una chiave

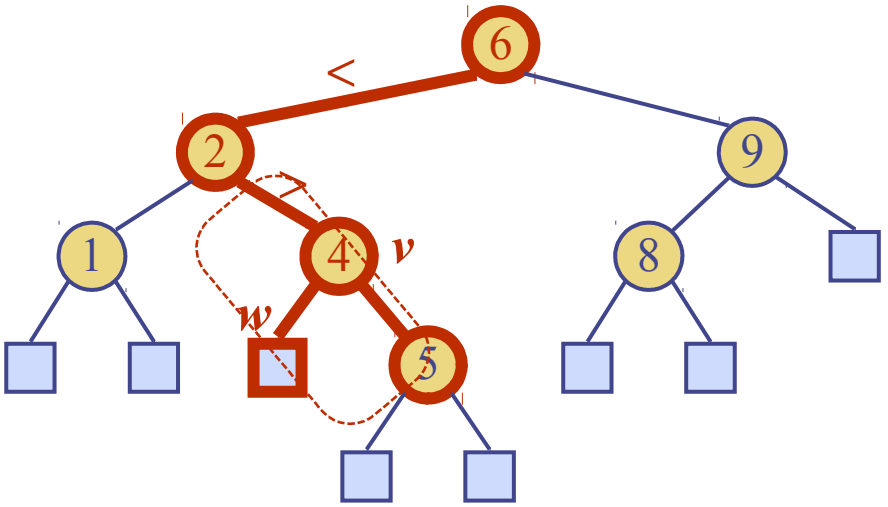
- put(k, o)
  - Si cerca un nodo con chiave k
    - treeSearch(k, root)
- Se k non è presente
  - Sia w la foglia raggiunta
- Si inserisce (k, o) nel nodo w e si espande w
- Esempio: put(5, 0)

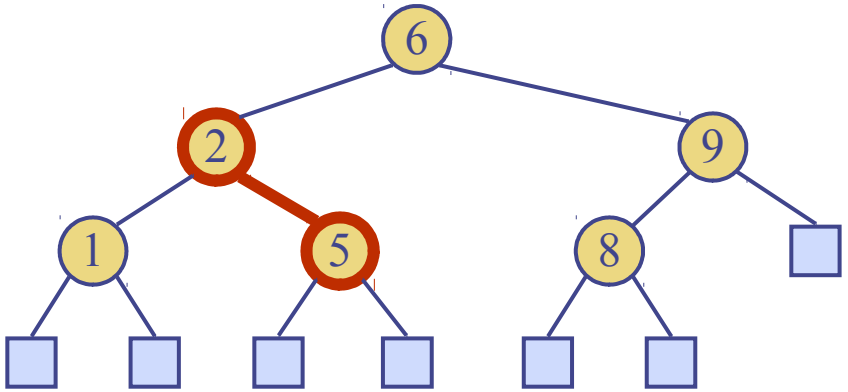




Cancellazione

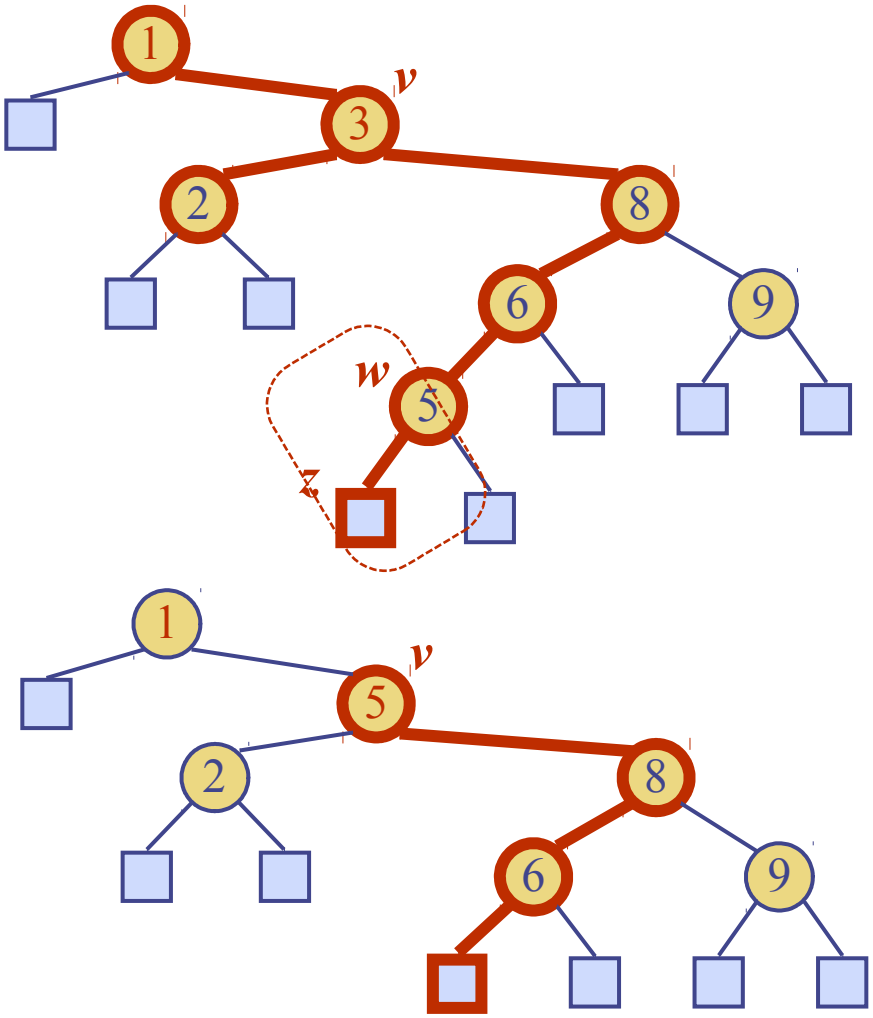
- remove(k)
  - treeSearch(k, root)
- Se presente
  - Sia v il nodo associato
- Se v ha come figlio una foglia w
  - Si rimuovono v e w
- Esempio: remove(4)





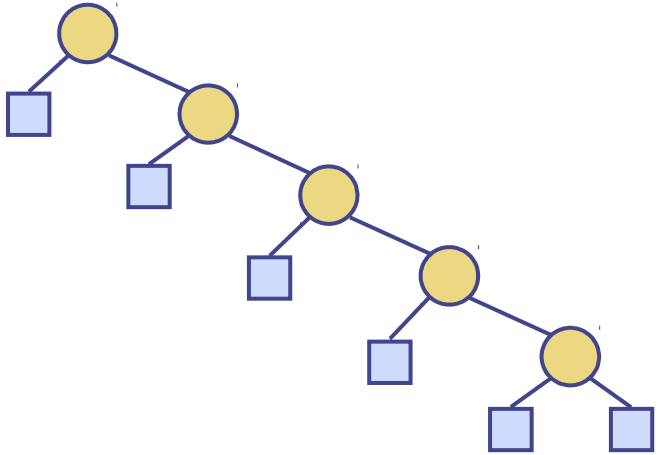
Cancellazione/cont.

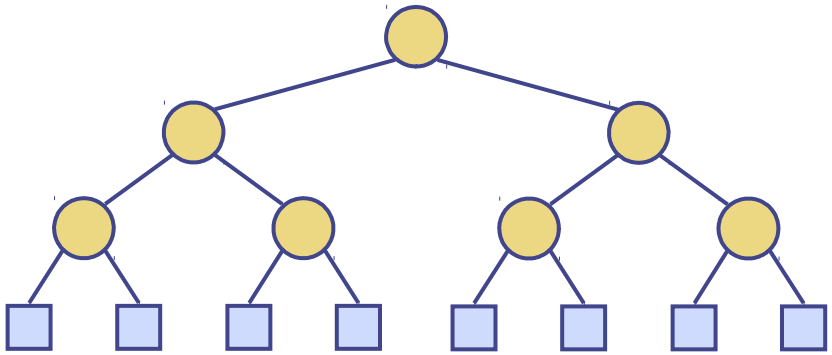
- k è associata a un nodo i cui figli sono *interni*
  - Trova il nodo w con chiave immediatamente successiva a k
  - Copia key(w) (e il valore associato) in v
  - Rimuovi w e il suo figlio sinistro (quest'ultimo deve essere una foglia)
- Esempio: remove(3)



Prestazioni

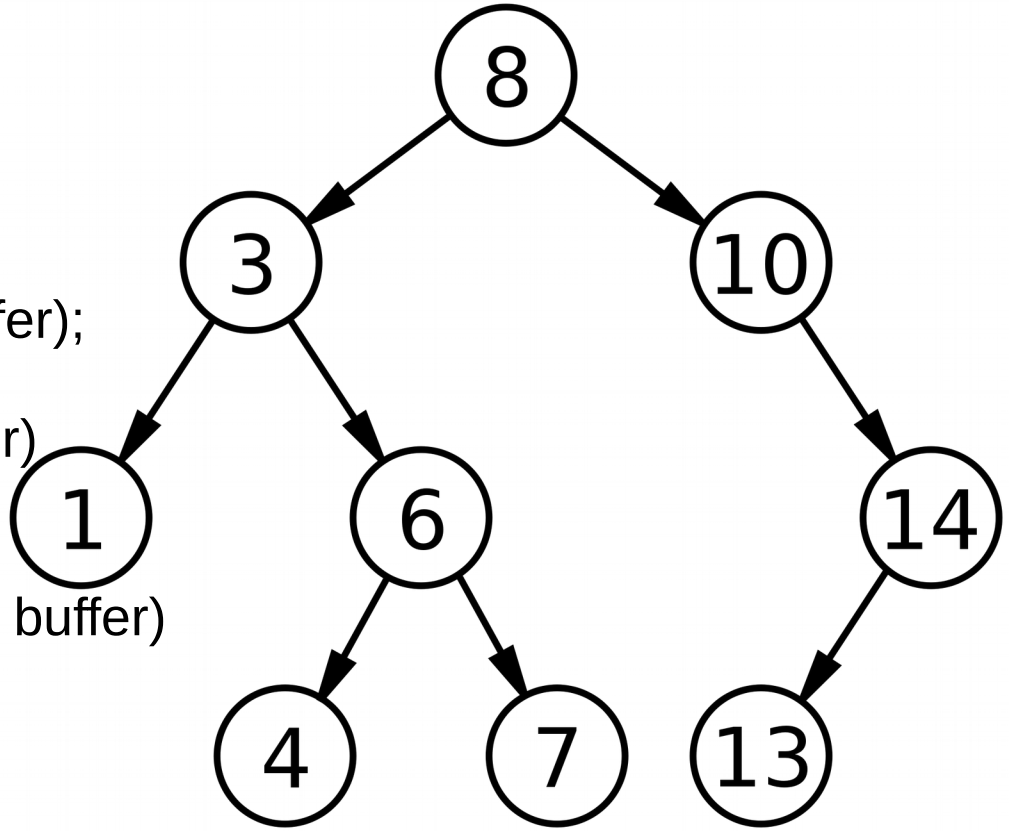
- Le operazioni get, put e remove hanno complessità O(h)
  - h = altezza albero
- Lo spazio richiesto è O(n)
- h = O(n) nel caso peggiore e O(log n) nel caso migliore





Range query – subMap(v, k_1, k_2)

```
Algorithm subMap(v, k1, k2, buffer) {
    If (v == null)
        return;
    If (k1 > v.key)
        subMap(v.rightChild(), k1, k2, buffer);
    else {
        subMap(v.leftChild(), k1, k2, buffer)
        If (k2 >= v.key) {
            buffer.add(v.pair);
            subMap(v.rightChild(), k1, k2, buffer)
        }
    }
}
```



Esempio: subMap(root, 2, 10, buffer)

→ buffer conterrà le coppie con chiavi 3, 4, 6, 7, 8, 10

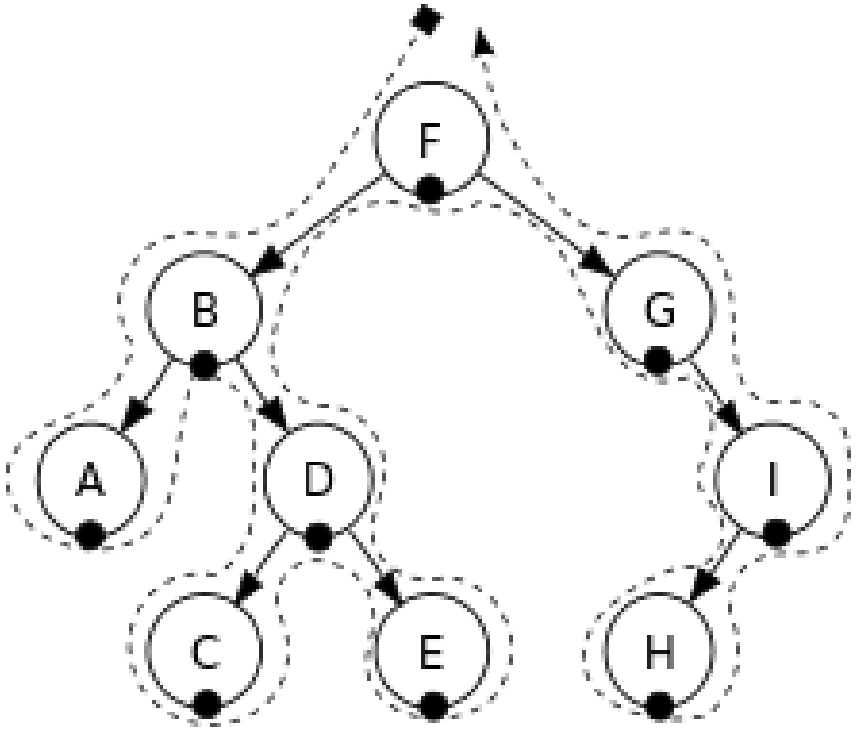
Range query - efficienza

- La complessità nel caso peggiore è O(h + s)
  - h = altezza dell'albero
  - s = numero di chiavi nell'intervallo [k1, k2]
- Prova:
  - L'algoritmo ricorre in ogni nodo con chiave nell'intervallo [k1, k2]
  - L'algoritmo non ricorre in nodi con chiave esterna all'intervallo

Proprietà delle visite in-order e post-order
Visita di Eulero e espressioni aritmetiche

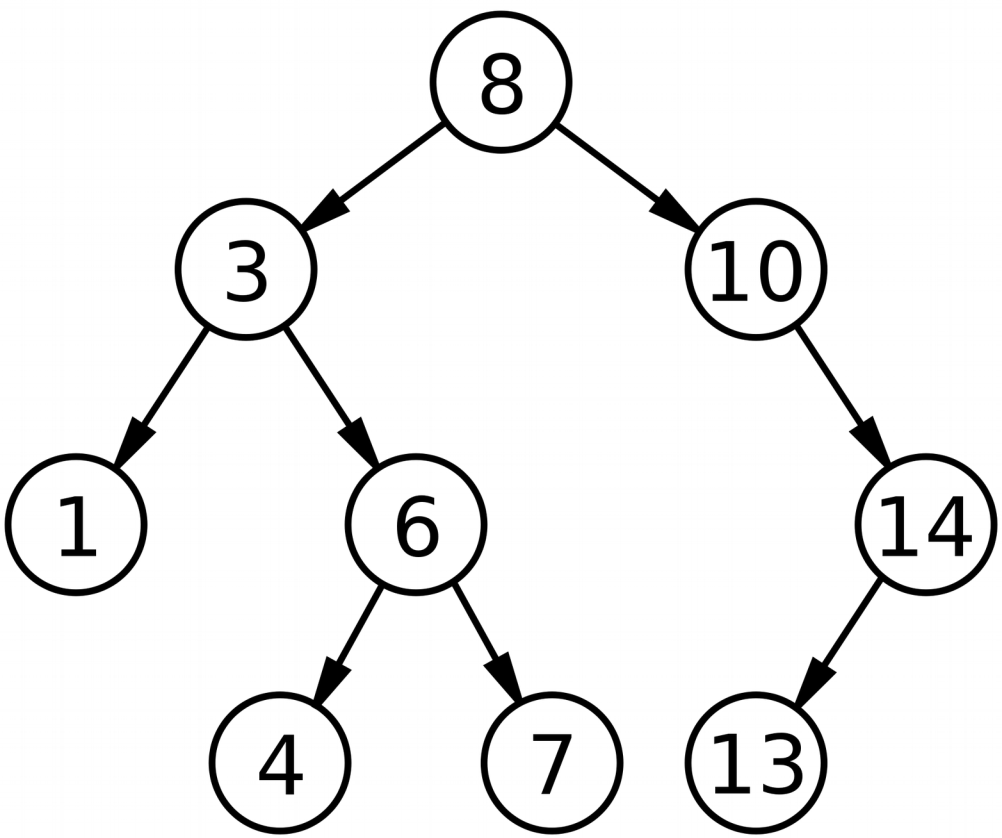
Visita in-order

```
Algorithm inorder(v) {
    if (v == null)
        return;
    inorder(v.leftChild());
    visit(v);
    inorder(v.rightChild());
}
```



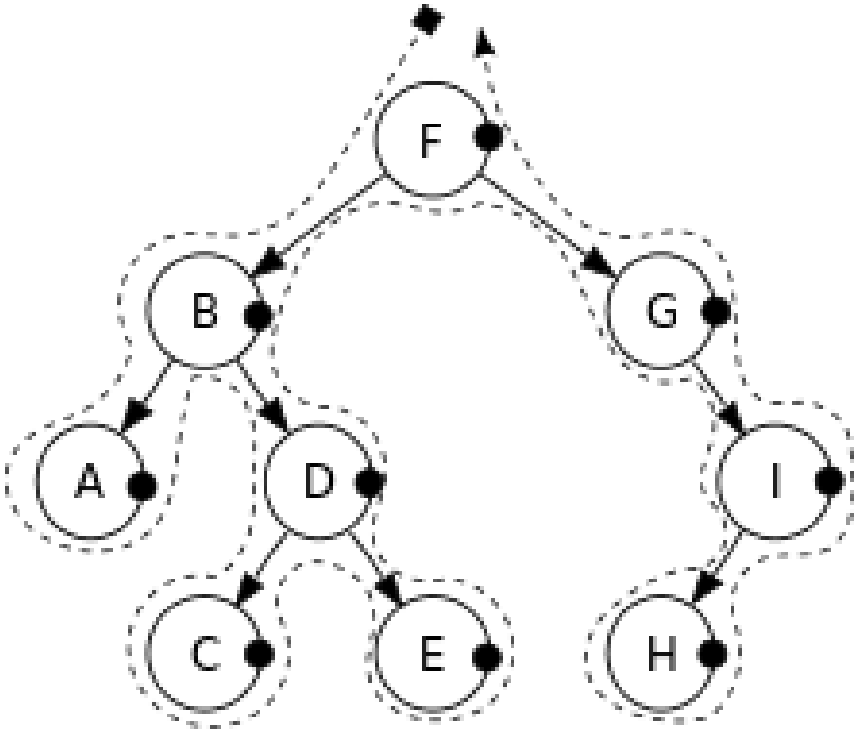
Visita in-order

- Se abbiamo un BST, corrisponde a visitare i nodi in ordine crescente rispetto alle chiavi
- Dimostrare
  - Suggerimento: induzione



Visita in post-ordine (alberi binari)

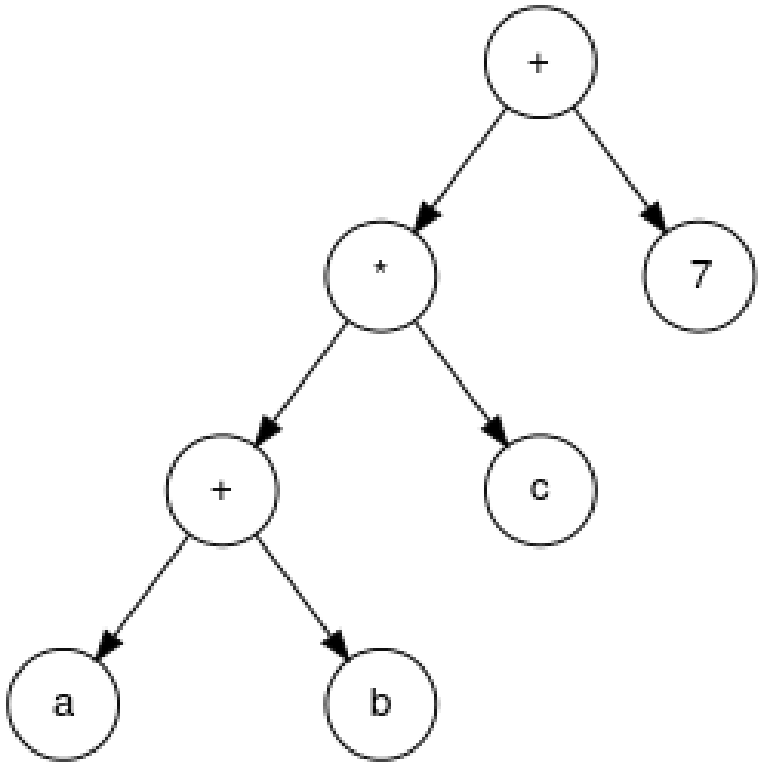
```
Algorithm postOrder(v) {
    if (v == null)
        return;
    postOrder(v.leftChild());
    postOrder(v.rightChild());
    visit(v);
}
```



Ordine di visita: A C E D B H I G F

Visita in post-ordine e valutazione di espressioni aritmetiche

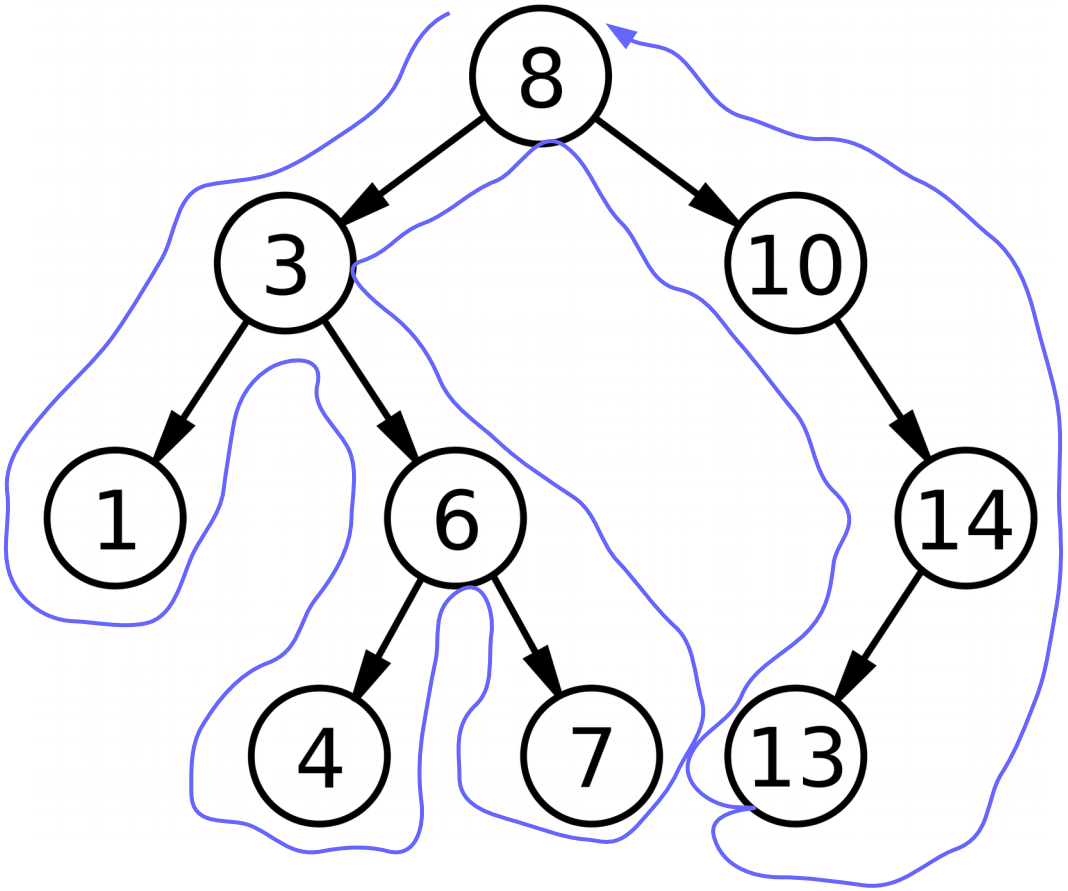
```
Algorithm eval(v) {
    If (v is a leaf)
        return v.val;
    x = eval(v.leftChild());
    y = eval(v.rightChild());
    return x v.op y;
```



Calcola (a + b)*c + 7

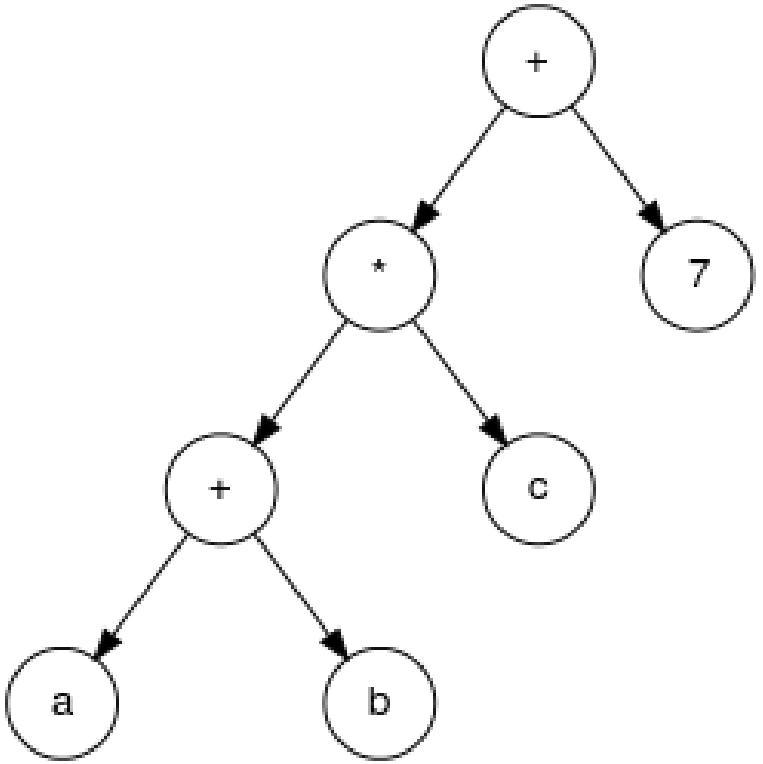
Visita di Eulero

- Si visita ogni nodo dell'albero 3 volte
  - Da sopra
  - Da sotto
    - Sx
    - Dx



Visita di Eulero ed espressioni aritmetiche

```
Algorithm eulerTour(v) {
        If (v is a leaf) // O foglia o 2 figli
                print v.val;
                return
        print(“(”);
        eulerTour(v.leftChild());
        print(v.val); // Numero o operatore aritmetico
        eulerTour(v.rightChild());
        print(“)”);
}
```



Stampa (((a + b) * c) + 7)