

# Visita di alberi binari

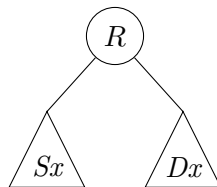
## 1 Attraversamento di alberi binari

Un albero binario è costituito da

$R$ : la radice;

$Sx$ : il sottoalbero sinistro;

$Dx$ : il sottoalbero destro.



Come qualsiasi albero con radice ordinato, un albero binario può essere attraversato

- in ordine anticipato:  $R Sx Dx$ ;
- in ordine posticipato:  $Sx Dx R$ ;
- per livelli (non esprimibile con questa notazione).

Inoltre, per gli alberi binari è definito un altro metodo di attraversamento, chiamato **visita simmetrica** o in **ordine simmetrico (in-order)**:  $Sx R Dx$ ,

### 1.1 Implementazione in Java

```
public class BinTree<Item> {  
    private Node r;  
    private class Node {  
        Item item;  
        Node sx;  
        Node dx;  
    }  
  
    public boolean isEmpty() {
```

```

        return r == null;
    }

    public void visita(int i) {
        switch (i) {
            case 1: preOrder(r); break;
            case 2: inOrder(r); break;
            case 3: postOrder(r); break;
            case 4: levelOrder(r); break;
        }
    }

    private void preOrder(Node r) {
        if (r != null) {
            r.visit();
            preOrder(r.sx);
            preOrder(r.dx);
        }
    }

    private void inOrder(Node r) {
        if (r != null) {
            inOrder(r.sx);
            r.visit();
            inOrder(r.dx);
        }
    }

    private void postOrder(Node r) {
        if (r != null) {
            postOrder(r.sx);
            postOrder(r.dx);
            r.visit();
        }
    }

    private void levelOrder(Node r) {
        Queue<Node> q = new Queue<Node>();
        if (r != null) q.enqueue(r);

        while (!q.isEmpty()) {
            r = q.front(); q.dequeue();
            r.visit();
            if (r.sx != null) q.enqueue(r.sx);
        }
    }

```

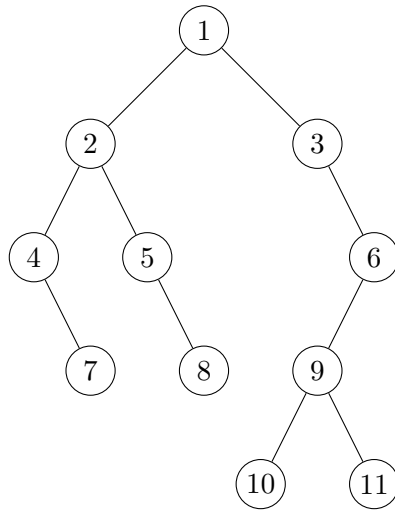
```

        if (r.dx != null) q.enqueue(r.dx);
    }
}

```

Tutte queste operazioni richiedono tempo  $\Theta(n)$  e spazio  $O(n)$  per un albero con  $n$  nodi.

## 1.2 Esempio



Visita	Ordine
<i>Pre-order</i>	1, 2, 4, 7, 5, 8, 3, 6, 9, 10, 11
<i>In-order</i>	4, 7, 2, 5, 8, 1, 3, 10, 9, 11, 6
<i>Post-order</i>	7, 4, 8, 5, 2, 10, 11, 9, 6, 3, 1
<i>Level-order</i>	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11