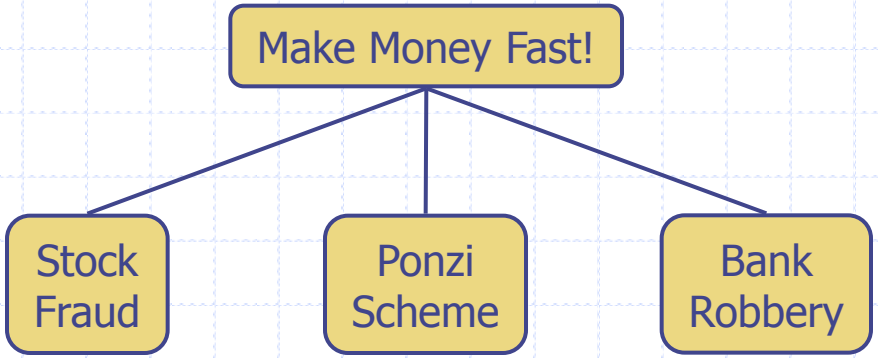
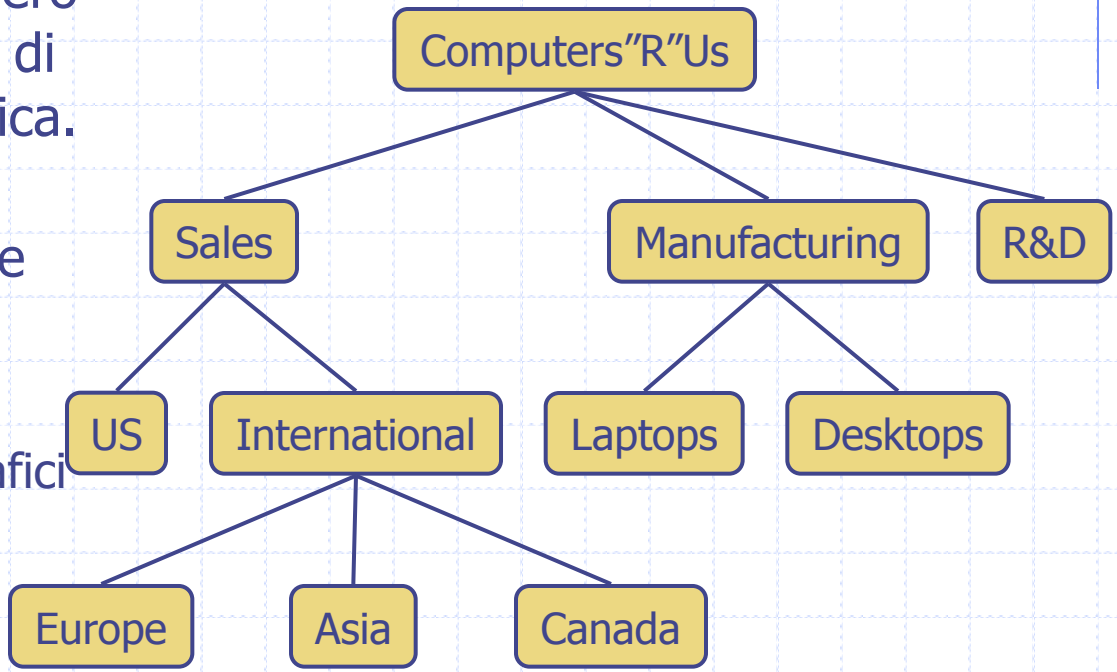
Alberi



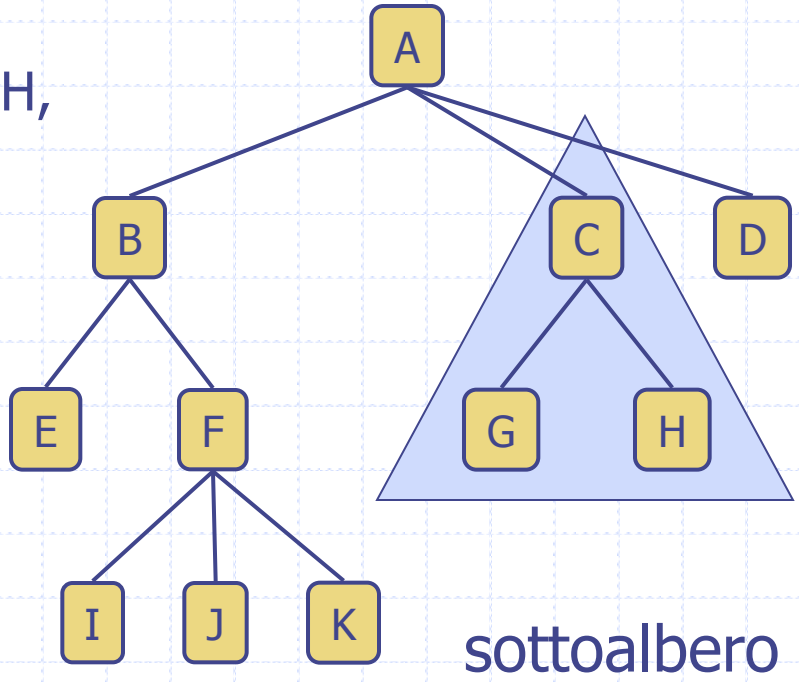
Cosa è un Albero?

- In informatica, un albero è un modello astratto di una struttura gerarchica.
- Un albero consiste di nodi con una relazione padre-figlio.
- Applicazioni:
  - Organizzazine di grafici
  - File systems
  - Ambienti di programmazione



Terminologia degli Alberi

- Radice : nodo senza genitore (A)
- Nodo interno: condo con almeno un figlio (A, B, C, F)
- Nodo esterno (anche chiamato foglia): nodo senza (E, I, J, K, G, H, D)
- Antenato di un nodo: genitore, nonno, bisnonno, etc.
- Profondità : numero di antenati
- Altezza di un albero: massima profondità di un nodo (3)
- Discendente di un nodo: figlio, nipote, bisnipote, etc.
- Sottoalbero: albero consistente di un nodo e dei suoi discendenti



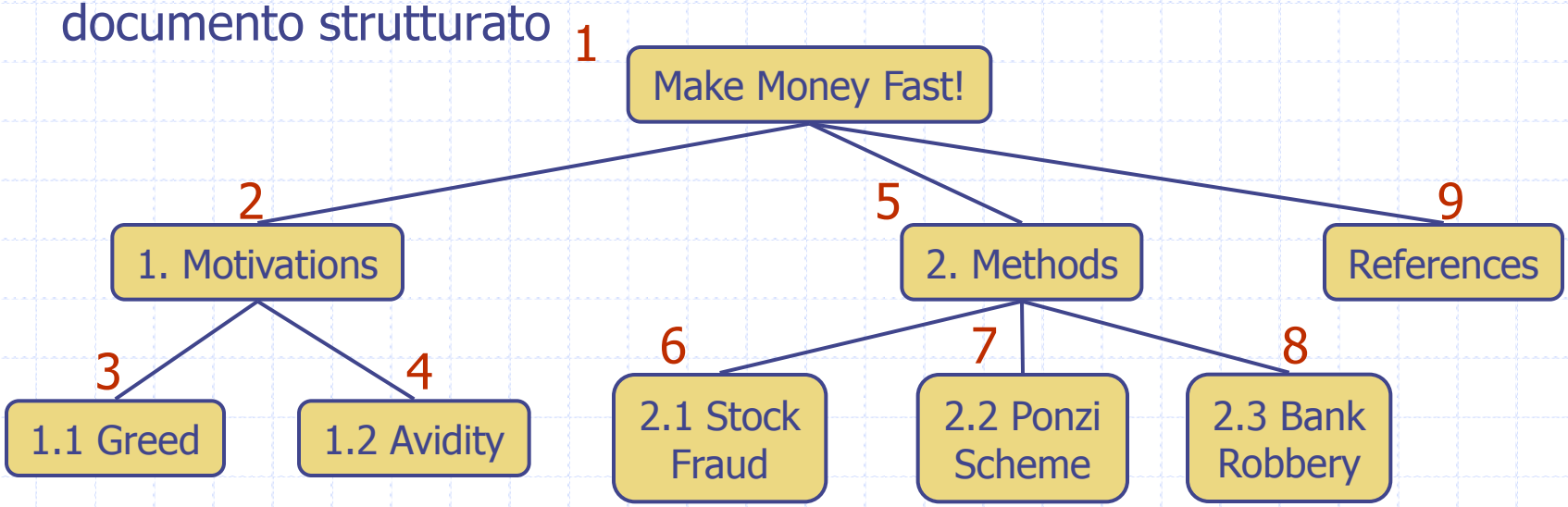
TDA Tree(§ 7.1.2)

- Si usano le posizioni per astrarre i nodi
- Metodi generici:
  - integer size()
  - boolean isEmpty()
  - Iterator elements()
  - Iterator positions()
- Metodi di accesso:
  - position root()
  - position parent(p)
  - positionIterator children(p)
- Metodi di query:
  - boolean isInternal(p)
  - boolean isExternal(p)
  - boolean isRoot(p)
- Metodi di aggiornamento :
  - object replace (p, o)
- Ulteriori metodi di aggiornamento possono essere definiti attraverso strutture dati che implementano il TDA Tree

Visita in Preordine

- Una visita attraversa i nodi di un albero in maniera sistematica
- Nella visita in preordine, un nodo è visitato prima dei suoi discendenti
- Applicazione: stampa di un documento strutturato

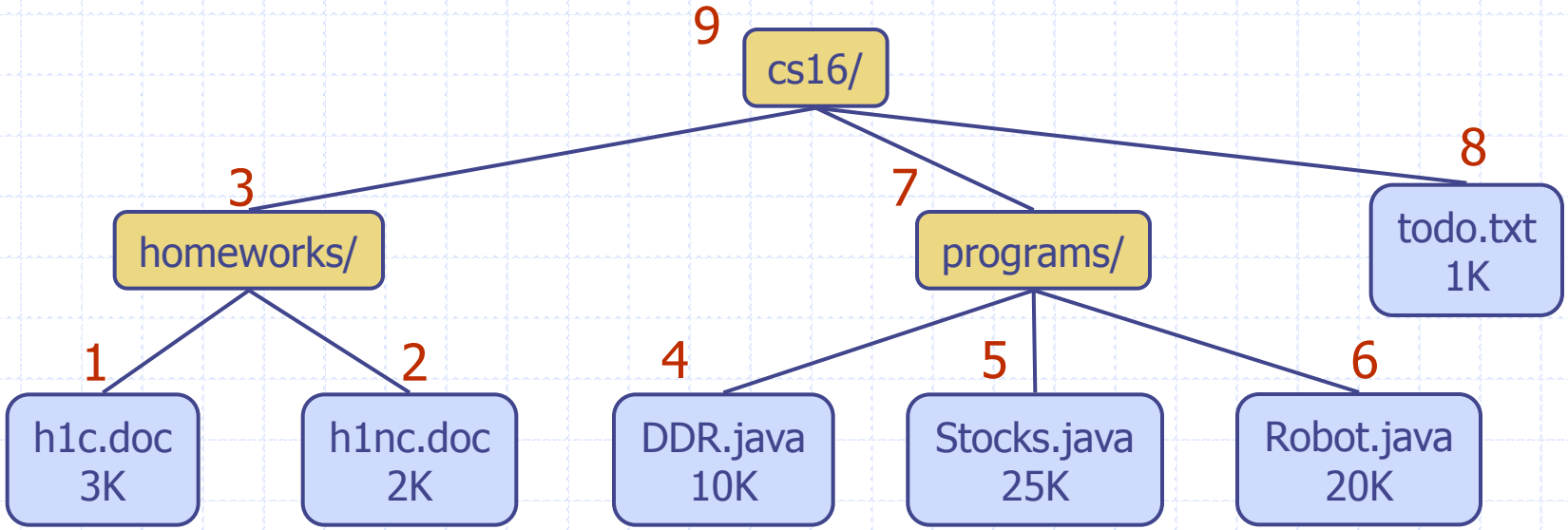
Algorithm ***preOrder(v)***
    visit(v)
    for each child ***w*** of ***v***
        preorder **(*w*)**



Visita in Postordine

- Nella visita in postordine , un nodo è visitato dopo i suoi discendenti
- Applicazione: calcola lo spazio usato dai files in una directory e nelle sue sottodirectory.

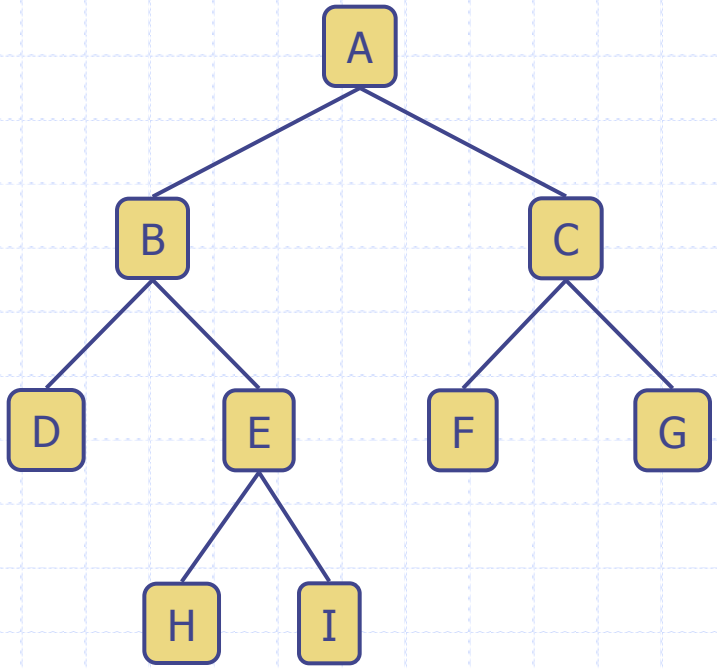
Algorithm *postOrder*(*v*)
  for each child *w* of *v*
    postOrder (*w*)
  visit(*v*)



Alberi Binari (§ 7.3)

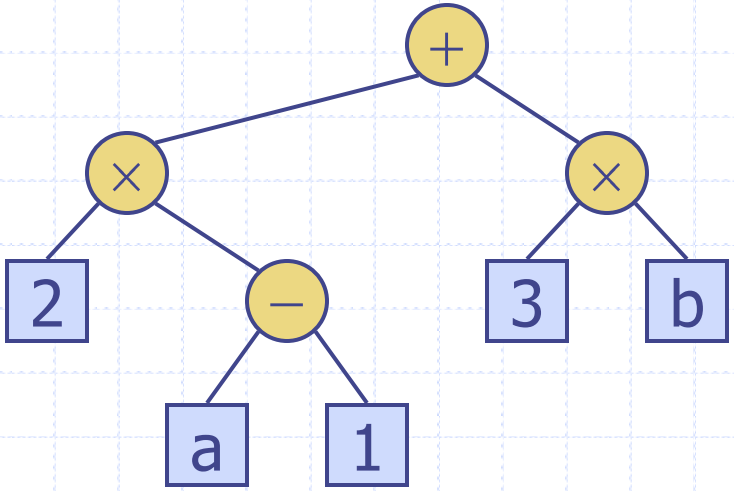
- Un albero binario è un albero con le seguenti proprietà:
  - Ogni nodo interno ha al più due figli (esattamente due per alberi binari cmpleti)
  - I figli di un nodo sono una coppia ordinata
- Chiamiamo i figli di un nodo interno figlio sinistro e figlio destro
- Una definizione alternativa basata sulla ricorsione: un albero binario è
  - un albero formato da un nodo singolo, oppure
  - un albero la cui radice possiede una coppia ordinata di figli, ognuno dei quali è un albero binario

- Applicazioni:
  - espressioni aritmetiche
  - processi di decisione
  - ricerca



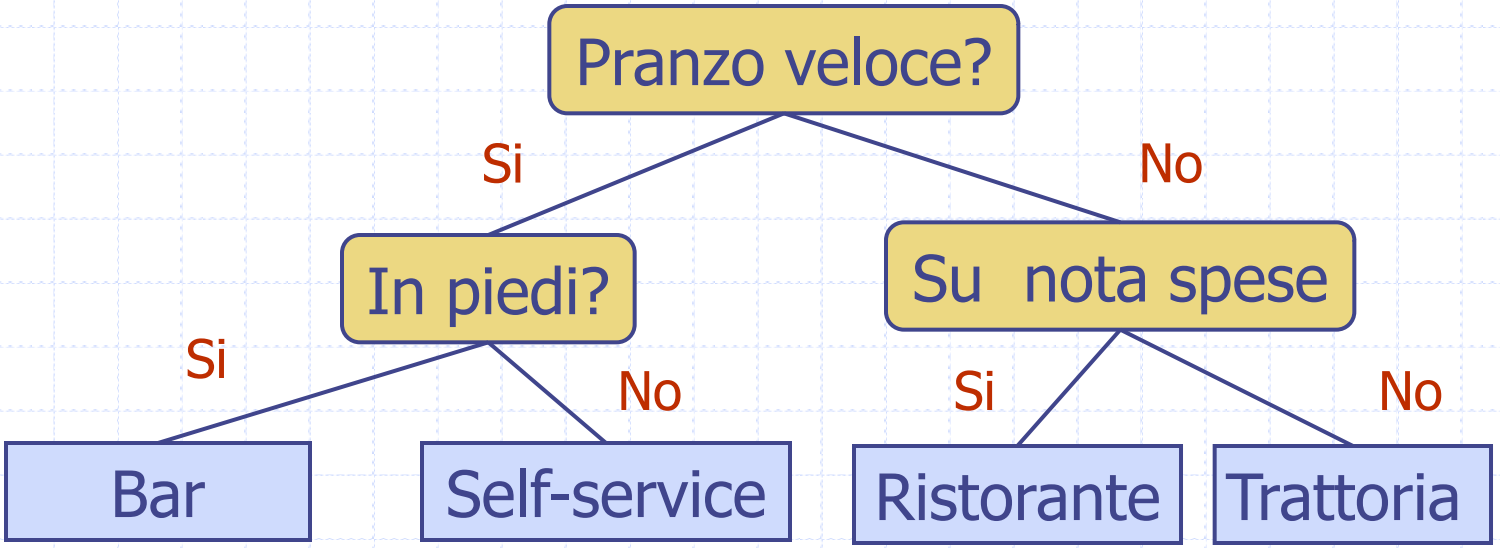
Aritmetiche

- Alberi binari associati con un'espressione aritmetica
  - nodi interni: operatori
  - nodi esterni: operandi
- Esempio: albero di espressione aritmetica per $(2 \times (a - 1) + (3 \times b))$



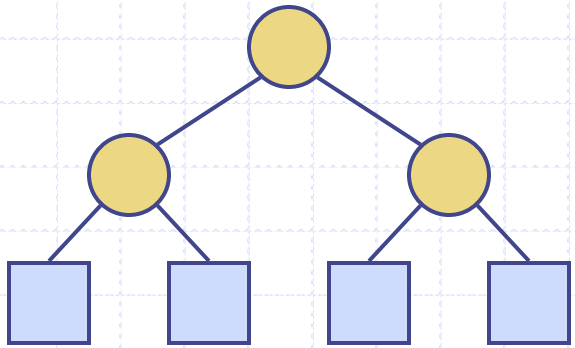
Alberi di Decisione

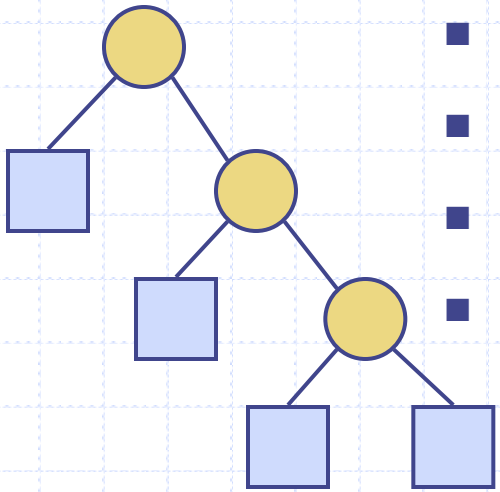
- Albero binario associato con un processo di decisione
  - Nodi interni: domande con risposta si/no
  - Nodi esterni: decisioni
- Esempio: scelta del ristorante



Proprietà degli Alberi Binari Completi

- Notazione
  - n numero di nodi
  - e numero di nodi esterni
  - i numero di nodi interni
  - h altezza





- Proprietà:
  - $e = i + 1$
  - $n = 2e - 1$
  - $h \leq i$
  - $h \leq (n - 1)/2$
  - $e \leq 2^h$
  - $h \geq \log_2 e$
  - $h \geq \log_2 (n + 1) - 1$

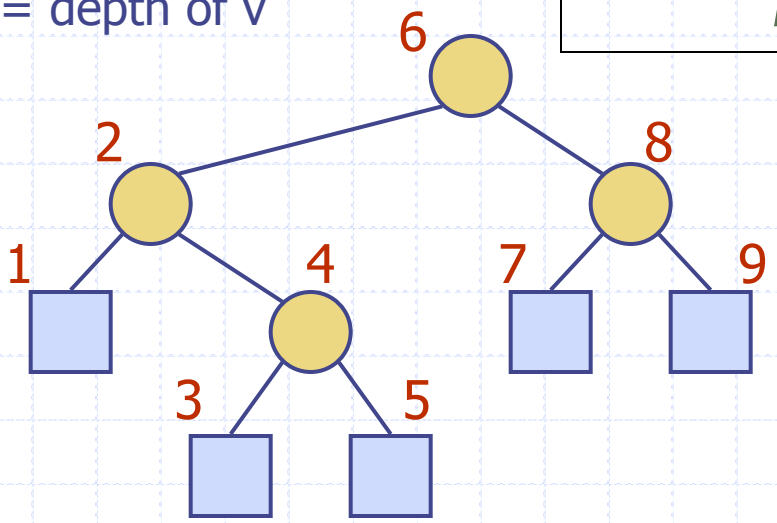
TDA BinaryTree (§ 7.3.1)

- Il TDA BinaryTree estende il TDA Tree, cioè eredita tutti i metodi del TDA Tree
- Metodi addizionali:
  - position left(p)
  - position right(p)
  - boolean hasLeft(p)
  - boolean hasRight(p)
- I metodi di aggiornamento possono essere definiti attraverso le strutture dati che implementano il TDA Tree

Visita Inordine

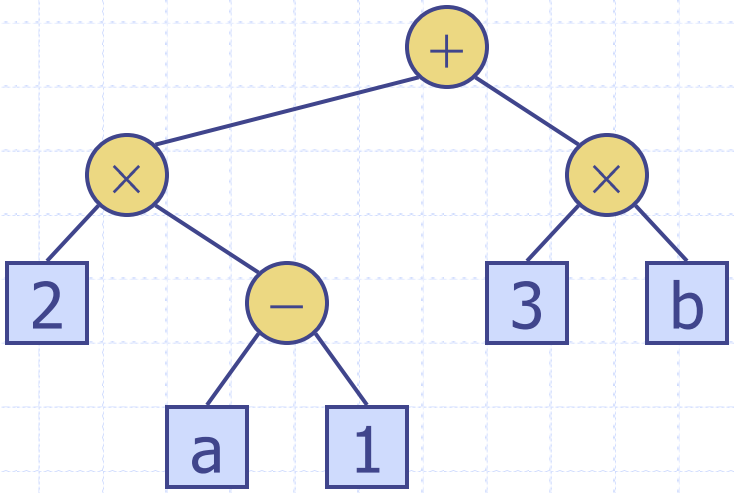
- Nella visita Inordine un nodo è visitato dopo il suo sottoalbero sinistro e prima del suo sottoalbero destro
- Applicazione: disegna un albero binario
  - x(v) = inorder rank of v
  - y(v) = depth of v

Algorithm *inOrder*(*v*)
  if *hasLeft* (*v*)
    inOrder (*left* (*v*))
  visit(*v*)
  if *hasRight* (*v*)
    inOrder (*right* (*v*))



Stampa di Espressioni Aritmetiche

- Specializzazione di una visita inordine
  - Stampa l'operando o l'operatore quando si visita un nodo
  - Stampa"(" prima di visitare il sottoalbero sinistro
  - Stampa")" dopo aver visitato il sottoalbero destro



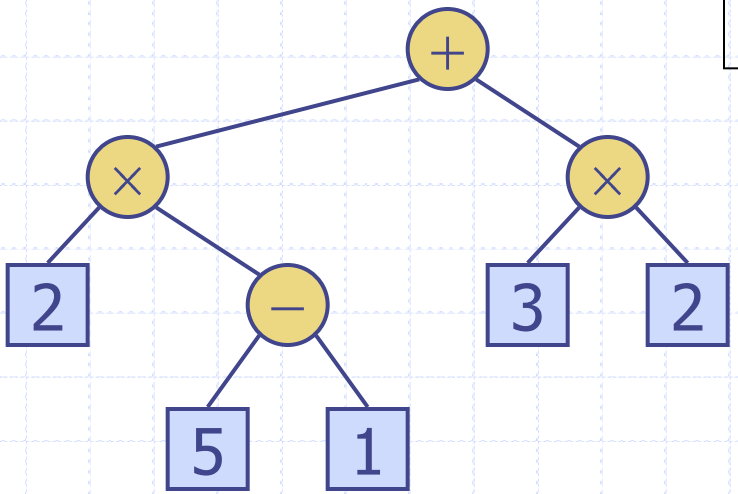
Algorithm *printExpression*(*v*)

```
if hasLeft (v)
    print("(")
    inOrder (left(v))
print(v.element ())
if hasRight (v)
    inOrder (right(v))
    print (")")
```

$((2 \times (a - 1)) + (3 \times b))$

Valutazione di Espressioni Aritmetiche

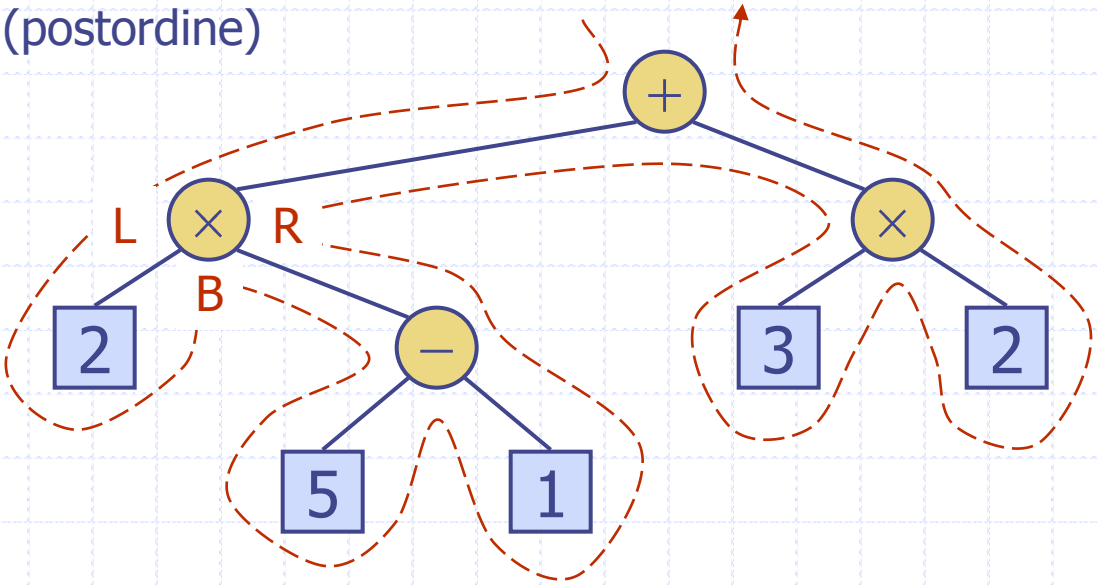
- Specializzazione della visita in postordine
  - metodo ricorsivo che restituisce il valore di un sottoalbero
  - Quando un nodo interno è visitato, si combinano i valori dei sottoalberi



Algorithm *evalExpr*(*v*)
  if *isExternal* (*v*)
    return *v.element* ()
  else
    $x \leftarrow$ *evalExpr*(*leftChild* (*v*))
    $y \leftarrow$ *evalExpr*(*rightChild* (*v*))
    $\lozenge \leftarrow$ operator stored at *v*
    return $x \lozenge y$

Visita con Cammino Euleriano

- Generica visita di un albero binario
- Include come casi particolari le visite in preordine, postordine e inordine
- Percorre l'albero visitando ogni nodo tre volte
  - da sinistra(preordine)
  - da sotto (inordine)
  - da destra (postordine)



Template Method Pattern

- Algoritmo generico che può essere specializzato attraverso la redifinizione di alcuni passi
- Implementato attraverso una classe astratta Java
- Metodi di visita che possono essere rifiniti in una sottoclasse
- Template method eulerTour
  - Ricorsivamente chiamato dal figlio sinistro e dal figlio destro
  - Un Result object con campi leftResult, rightResult e finalResult che tengono traccia delle chiamate ricorsive a eulerTour

```
public abstract class EulerTour {
    protected BinaryTree tree;
    protected void visitExternal(Position p, Result r) { }
    protected void visitLeft(Position p, Result r) { }
    protected void visitBelow(Position p, Result r) { }
    protected void visitRight(Position p, Result r) { }
    protected Object eulerTour(Position p) {
        Result r = new Result();
        if tree.isExternal(p) { visitExternal(p, r); }
            else {
                visitLeft(p, r);
                r.leftResult = eulerTour(tree.left(p));
                visitBelow(p, r);
                r.rightResult = eulerTour(tree.right(p));
                visitRight(p, r);
                return r.finalResult;
            } …
```

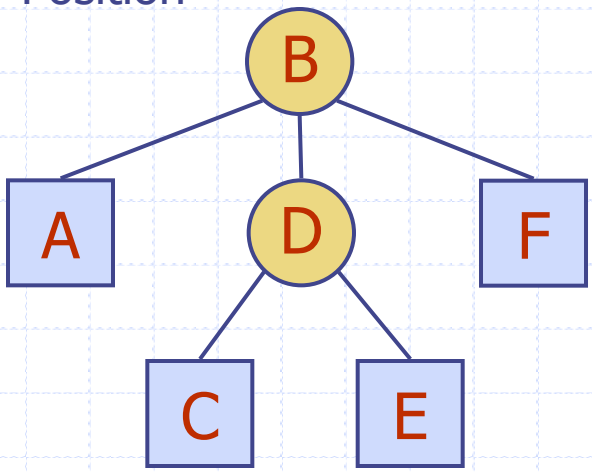
Specializzazione di EulerTour

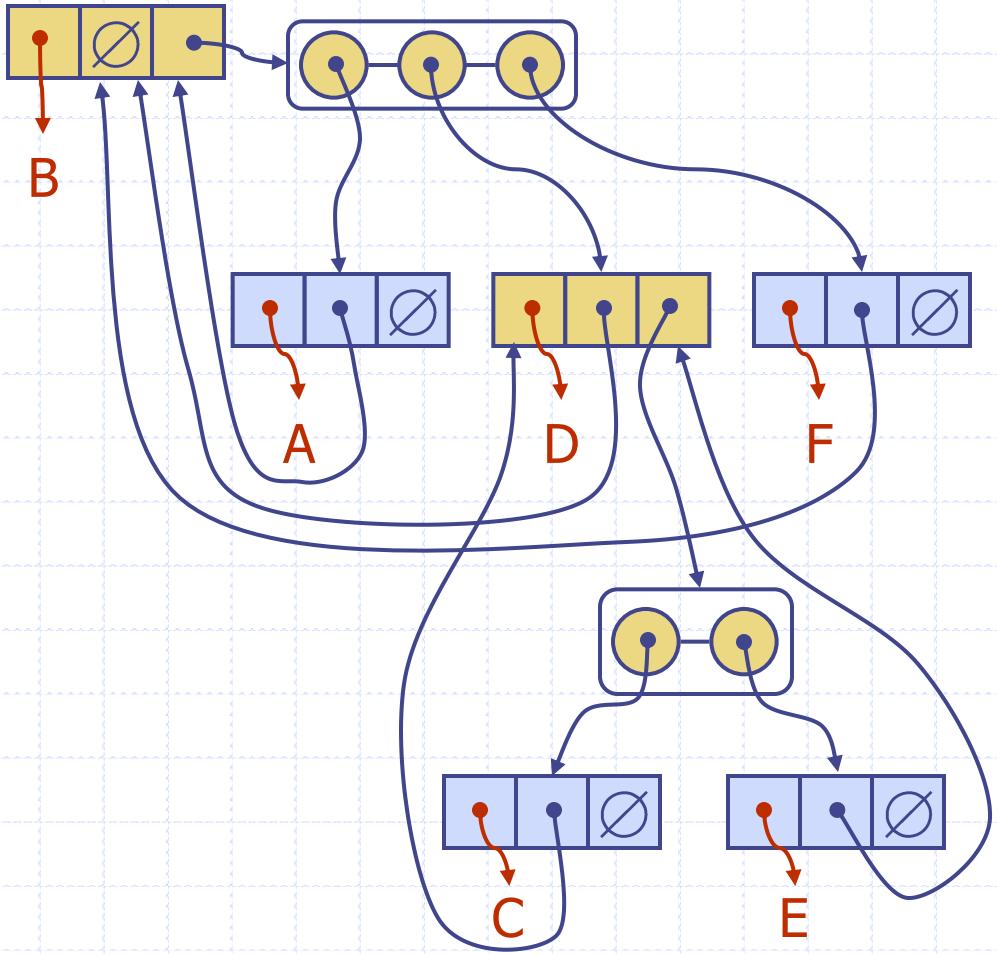
- Mostriamo come specializzare la classe EulerTour per valutare espressioni aritmetiche
- Assunzioni
  - Nodi esterni memorizzano oggetti Integer
  - Nodi interni memorizzano oggetti Operator che supportano il metodo operation (Integer, Integer)

```
public class EvaluateExpression
                    extends EulerTour {

  protected void visitExternal(Position p, Result r) {
    r.finalResult = (Integer) p.element();
  }

  protected void visitRight(Position p, Result r) {
    Operator op = (Operator) p.element();
    r.finalResult = op.operation(
                        (Integer) r.leftResult,
                        (Integer) r.rightResult
                        );
  }

  …

}
```

Struttura Collegata per Alberi

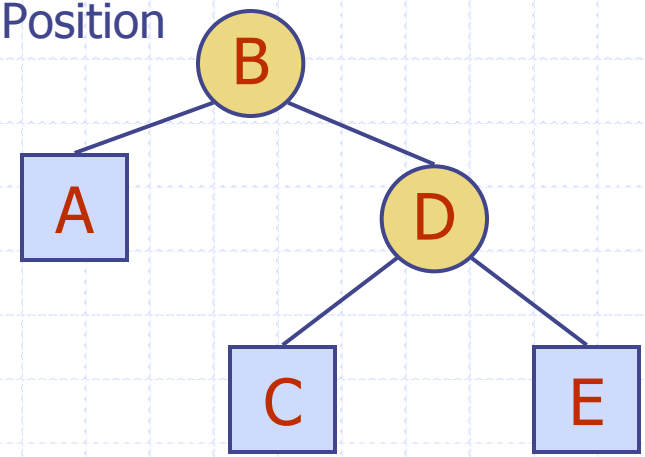
- Un nodo è rappresentato da un oggetto Node che memorizza
  - Elemento
  - Nodo genitore
  - Sequenza di nodi figli
- Oggetti Node implementano il TDA Position

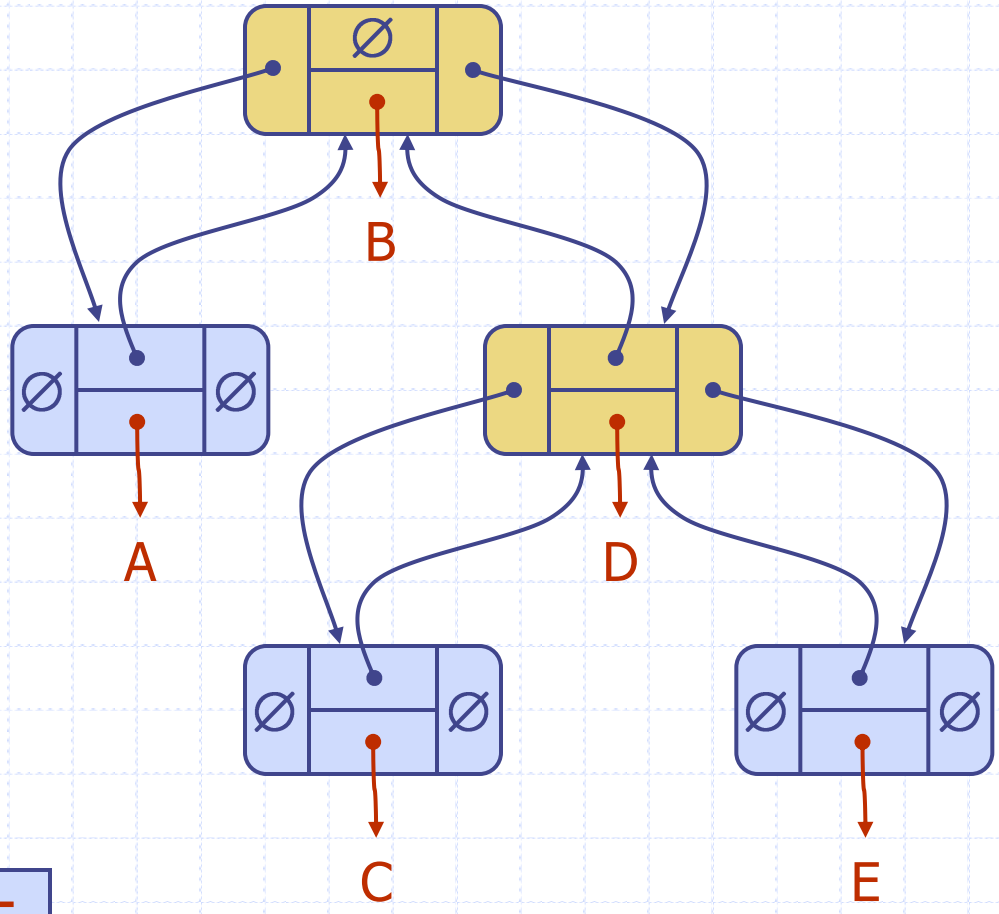




Struttura Collegata per Alberi

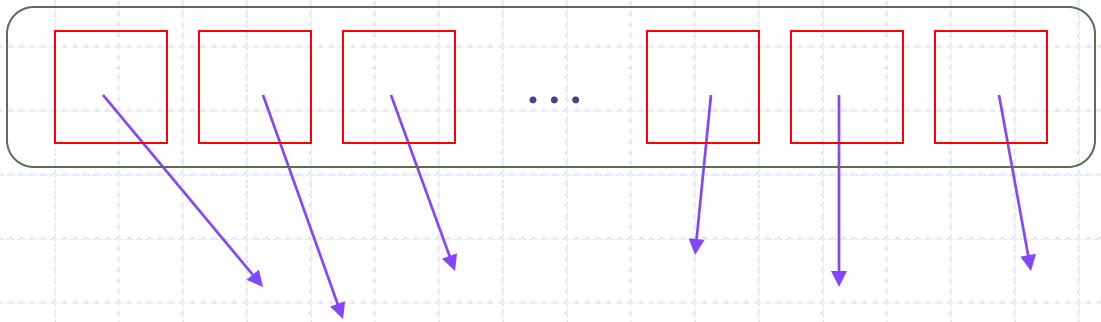
- Un nodo è rappresentato da un oggetto Node che memorizza
  - Elemento
  - Node genitore
  - Node figlio sinistro
  - Node figlio destro
- Oggetti Node implementano il TDA Position

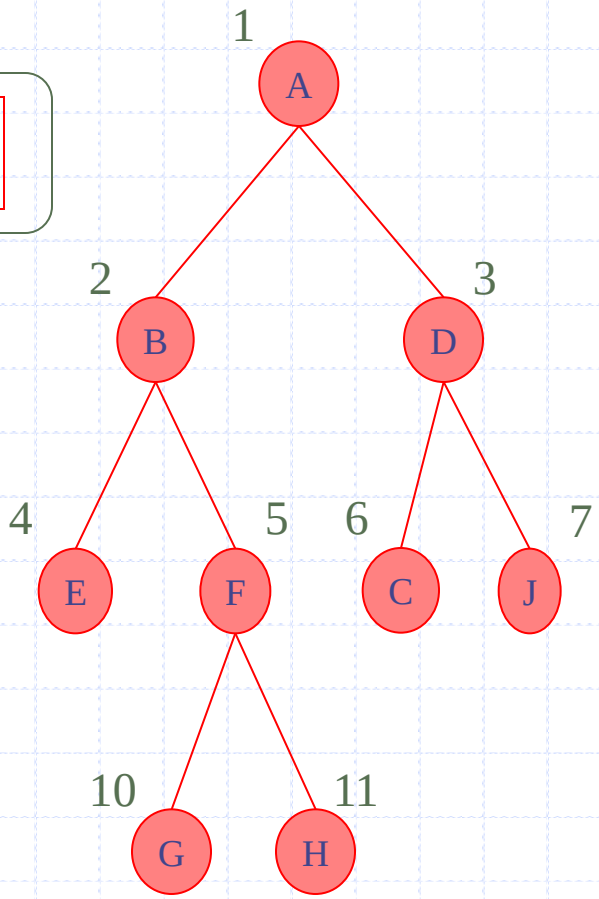




Rappresentazione di Alberi Binari con Array

- Nodi memorizzati in un array





- rank(node) definito come:
  - rank(root) = 1
  - se node è figlio sinistro di parent(node), rank(node) = 2 rank(parent(node))
  - se node è figlio destro di parent(node), rank(node) = 2 rank(parent(node))+1