3 Strutture dati

3.1 Implementazione strutture dati elementari

```
bool isEmpty()  // restituisce vero se la pila è vuota
push(ITEM v)  // inserisce v in cima alla pila
ITEM pop()  // estrae l'elemento in cima alla pila e lo restituisce al chiamante
ITEM top()  // legge l'elemento in cima alla pila
```

Ogni volta che si fa una chiamata a funzione si usa implicitamente una pila, in quanto memorizza tutti i record di attivazione delle chiamate effettuare. Sfrutteremo questo meccanismo per visitare gli alberi, anche se non esplicitamente.

Le pile possono essere implmentate come:

- liste bidirezionali, dove il puntatore punta all'elemento top (non utilizzate);
- tramite vettore, dove la dimesione è limitata quindi si crea un *overhead* più basso.

Struttura dati pila basata su vettore

```
ITEM[] A                          // elementi
int n                              // cursore
int m                          // dimesione massima

// crea una pila vuota
STACK Stack(int dim)
    STACK t = new STACK
    t.A = new int[0...dim-1]
    t.m = dim
    t.n = 0
    ritorna t

// leggi l'elemento in cima alla pila
ITEM top
    precondition: n > 0
    ritorna A[n]

// restituisce vero se la pila è vuota
bool isEmpty
    ritorna n == 0

// estrae l'elemento in cima alla pila e lo
   restituisce al chiamante
ITEM pop
    precondition: n > 0
    ITEM t = A[n]
    n --
    ritorna t

// inserisce v in cima alla pila
push(ITEM v)
    precondition: n < m
    n++
    A[n] = v
```

Codice 1: Pila basata su vettore in Java

```java
public class VectorQueue implements Queue {

    /** Element vector */
    private Object[] A;

    /** Current number of elements in the queue */
    private int n;

    /** Top element of the queue */
    private int head;

    public VectorQueue(int dim) {
        n = 0;
```

```
        head = 0;
        A = new Object[dim];
    }

    public boolean isEmpty() {
        return n == 0;
    }

    public Object top() {
        if (n == 0)
            throw new IllegalStateException("Queue is empty");

        return A[head];
    }

    public Object dequeue() {
        if (n == 0)
            throw new IllegalStateException("Queue is empty");

        Object t = A[head];
        head = (head+1) % A.length;
        n = n-1;
        return t;
    }

    public void enqueue(Object v) {
        if (n == A.length)
            throw new IllegalStateException("Queue is full");

        A[(head+n) % A.length] = v;
        n = n+1;
    }
}
```

Codice 2: Coda basata su vettore circolare in Java

```
public class VectorStack implements Stack {

    /** Vector containing the elements */
    private Object[] A;

    /** Number of elements in the stack */
    private int n;

    public VectorStack(int dim) {
        n = 0;
        A = new Object[dim];
    }

    public boolean isEmpty() {
        return n == 0;
    }

    public Object top() {
    if (n == 0)
        throw new IllegalStateException("Stack is empty");

        return A[n-1];
    }

    public Object pop() {
        if (n == 0)
            throw new IllegalStateException("Stack is empty");

        return A[--n];
    }
```

```
    public void push(Object o) {
        if (n == A.length)
            throw new IllegalStateException("Stack is full");

        A[n++] = o;
    }

}
```