# Visita di alberi binari

Strategia per analizzare tutti i nodi dell'albero

## VISITA IN PROFONDITÀ
(**Depth First Search - DFS**)

- La visita di un albero visita ricorsivamente i sottoalberi
- Varianti: pre-order, in-oder, post-order
- Si usa la ricorsione (implicitamente una pila)

## VISITA IN AMPIEZZA
(**Breadth First Search - BFS**)

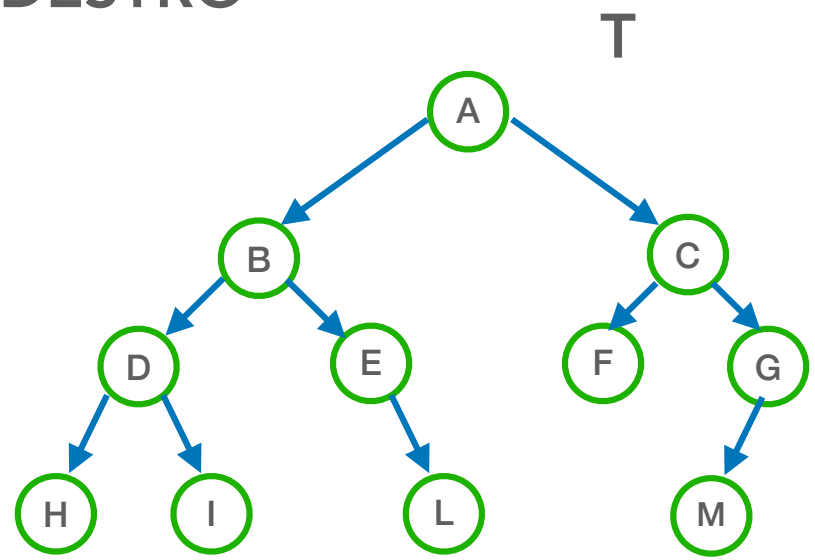
- Si visita un livello dopo l'altro, partendo dalla radice
- Si usa una coda esplicitamente

Costo computazionale $\in O(n \cdot f(n))$,
dove $O(f(n))$ è il costo dell'analisi di un singolo nodo

# Visita DFS: pre-order

Prima visita la **RADICE**, poi il sottoalbero **SINISTRO**,
infine il sottoalbero **DESTRO**

```
DFS_pre_order(t)
if NOT (t = NIL)
 then
  // analisi di t
  print t.val
  DFS_pre_order(t.left)
  DFS_pre_order(t.right)
```




Stampa la seguente sequenza:
A,B,D,H,I,E,L,C,F,G,M

Chiamata principale: **`DFS_pre_order(T)`**

chiamate ricorsive:

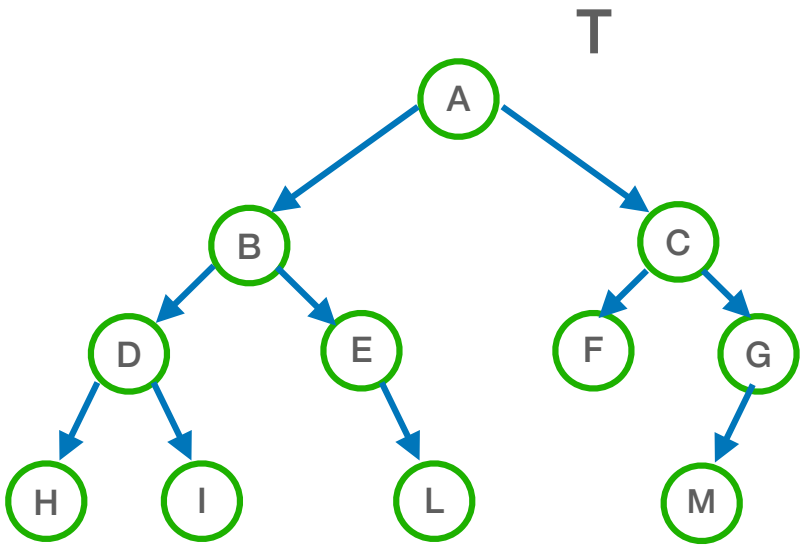
**(A(B(D(H()())(I()()))(E()(L()()))))(C(F()())(G(M()())())))**

# Visita DFS: in-order

Prima visita il sottoalbero **SINISTRO**, poi la **RADICE**,
infine il sottoalbero **DESTRO**

```
DFS_in_order(t)
if NOT (t = NIL)
 then
  DFS_in_order(t.left)
  // analisi di t
  print t.val
  DFS_in_order(t.right)
```

Chiamata principale: **`DFS_in_order(T)`**




Stampa la seguente sequenza:
H,D,I,B,E,L,A,F,C,M,G

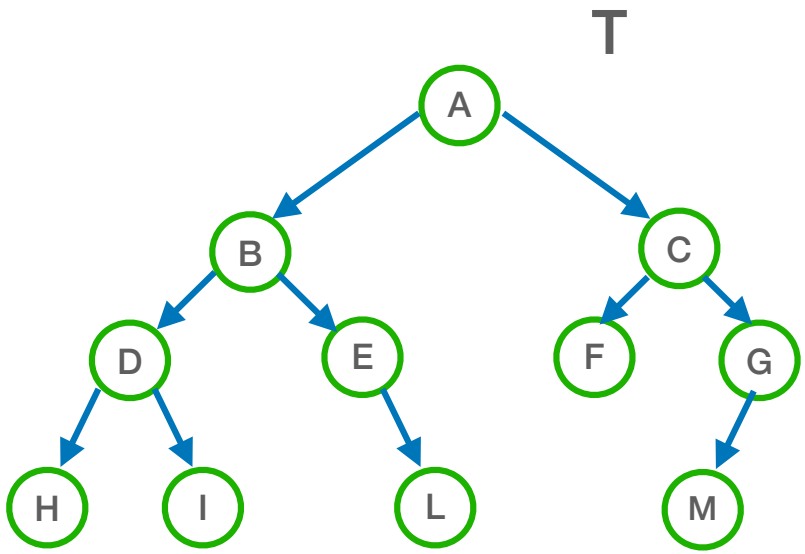
chiamate ricorsive

((((()H())D(()I()))B(()E(()L())))A((()F())C((()M())G())))

# Visita DFS: post-order

Prima visita il sottoalbero **SINISTRO**, poi il sottoalbero **DESTRO**, infine la **RADICE**

```
DFS_post_order(t)
if NOT (t = NIL)
 then
  DFS_post_order(t.left)
  DFS_post_order(t.right)
  // analisi di t
  print t.val
```




Chiamata principale: **`DFS_post_order(T)`**

Stampa la seguente sequenza:
H,I,D,L,E,B,F,M,G,C,A

chiamate ricorsive

((((()()H)(()()I)D)(()(()()L)E)B)((()()F)((()()M)G)C)A)

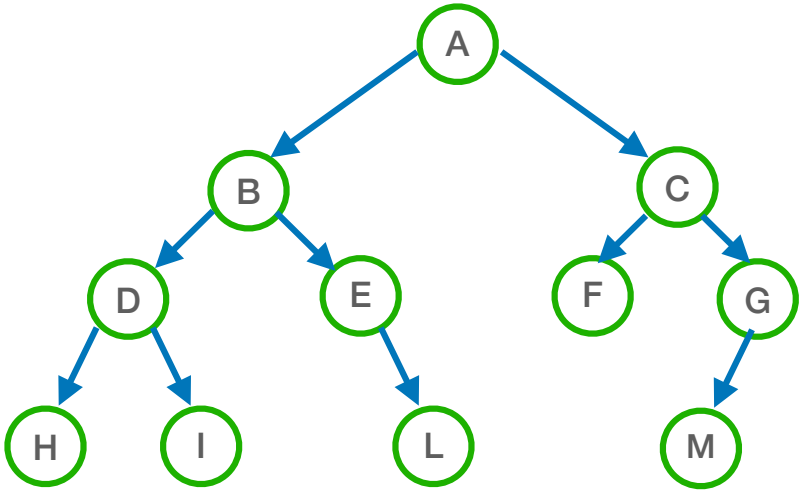
# Visita BFS

Prima visita la **RADICE**, poi i FIGLI della radice,
poi i FIGLI dei FIGLI della radice, poi i FIGLI dei FIGLI dei FIGLI della radice… e cosi' via…

Usiamo una coda

```
BFS(T)
if (NOT T = NIL)
 Q := new_queue()
 enqueue(Q,T)
 while NOT is_empty_queue(Q)
  u := dequeue(Q)
  // analisi di u
  print u.val
  if NOT (u.left = NIL)
   then enqueue(Q,u.left)
  if NOT (u.right = NIL)
   then enqueue(Q,u.right)
```

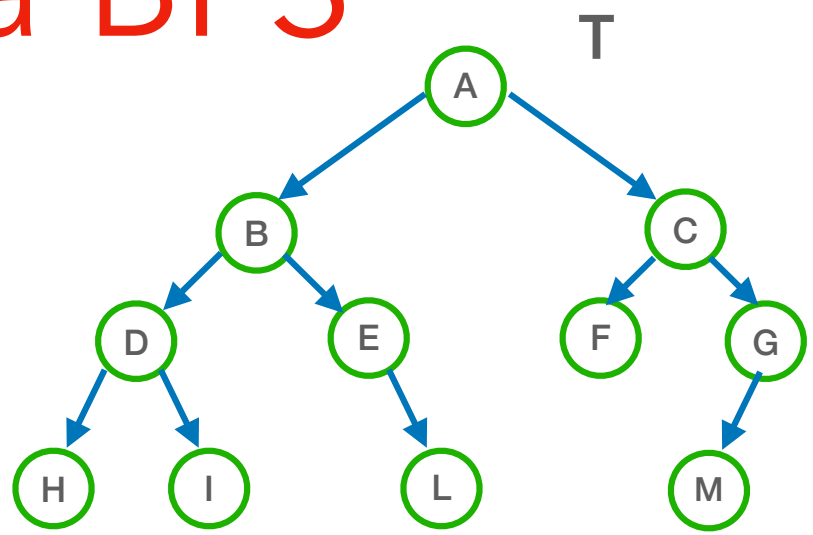
T




Stampa la seguente sequenza:
A,B,C,D,E,F,G,H,I,L,M

# Visita BFS

```
BFS(T)
if (NOT T = NIL)
  Q := new_queue()
  enqueue(Q,T)
  while NOT is_empty_queue(Q)
    u := dequeue(Q)
    // analisi di u
    print u.val
    if NOT (u.left = NIL)
      then enqueue(Q,u.left)
    if NOT (u.right = NIL)
      then enqueue(Q,u.right)
```




| CODA (inizio iterazione): | CODA (dopo dequeue): | |
|---|---|---|
| 1. Q = <A> | 1. Q = <> | 1. print A |
| 2. Q = <B,C> | 2. Q = <C> | 2. print B |
| 3. Q = <C,D,E> | 3. Q = <D,E> | 3. print C |
| 4. Q = <D,E,F,G> | 4. Q = <E,F,G> | 4. print D |
| 5. Q = <E,F,G,H,I> | 5. Q = <F,G,H,I> | 5. print E |
| 6. Q = <F,G,H,I,L> | 6. Q = <G,H,I,L> | 6. print F |
| 7. Q = <G,H,I,L,> | 7. Q = <H,I,L,> | 7. print G |
| 8. Q = <H,I,L,M> | 8. Q = <I,L,M> | 8. print H |
| 9. Q = <I,L,M> | 9. Q = <L,M> | 9. print I |
| 10.Q = <L,M> | 10.Q = <M> | 10.print L |
| 11.Q = <M> | 11.Q = <> | 11.print M |
| 12.Q = <> | | |