Titolo unità didattica: Strutture dati dinamiche gerarchiche [P2_09]

Titolo: alberi tramati (threaded trees) [4-AT]

Rappresentazione ed algoritmi di visita

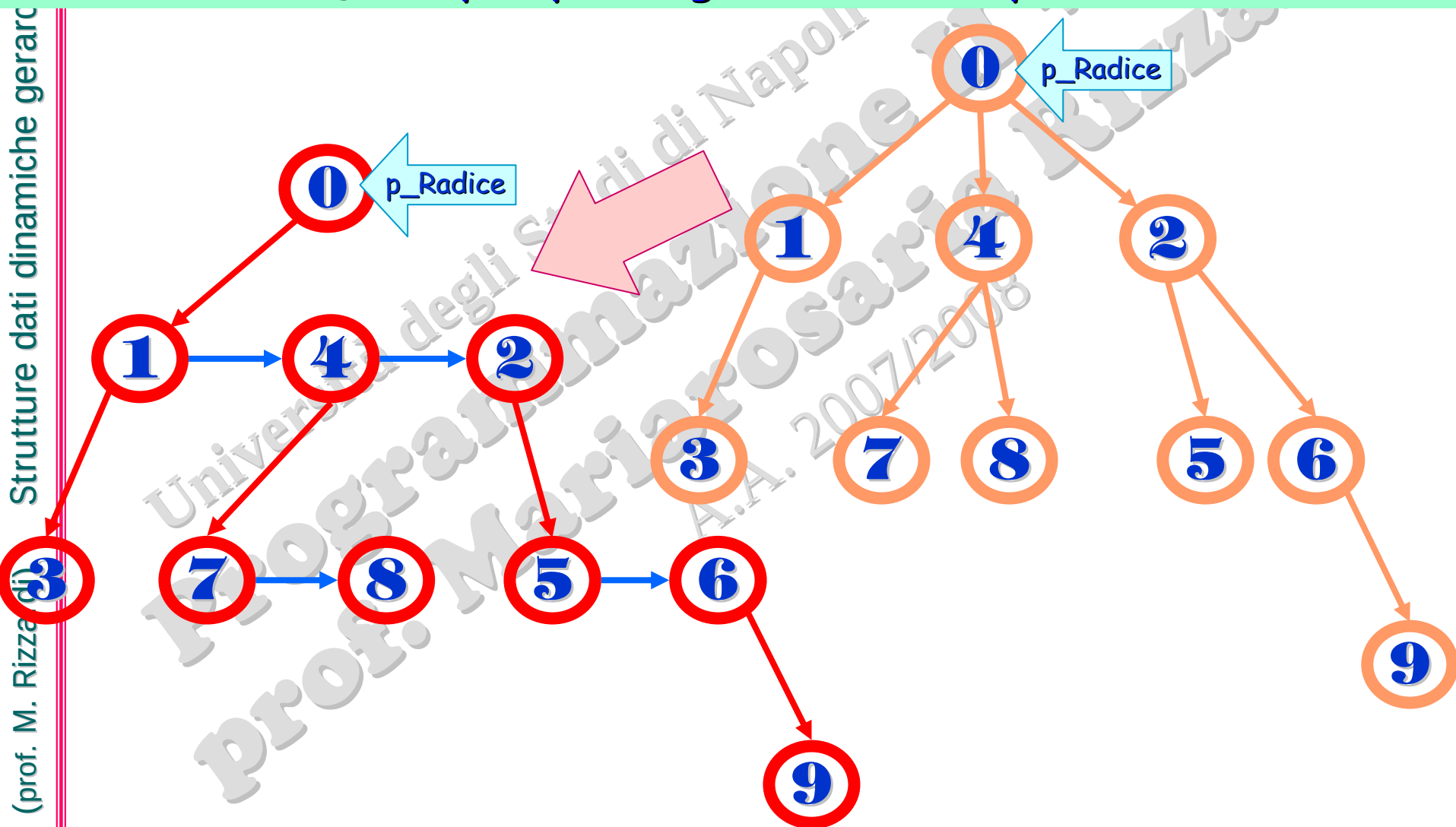
Argomenti trattati:

- ✓ Struttura di un nodo di un albero tramato
- ✓ Rappresentazione di un albero generico tramato
- ✓ Rappresentazione di un albero binario tramato
- ✓ Algoritmi di visita di un albero tramato

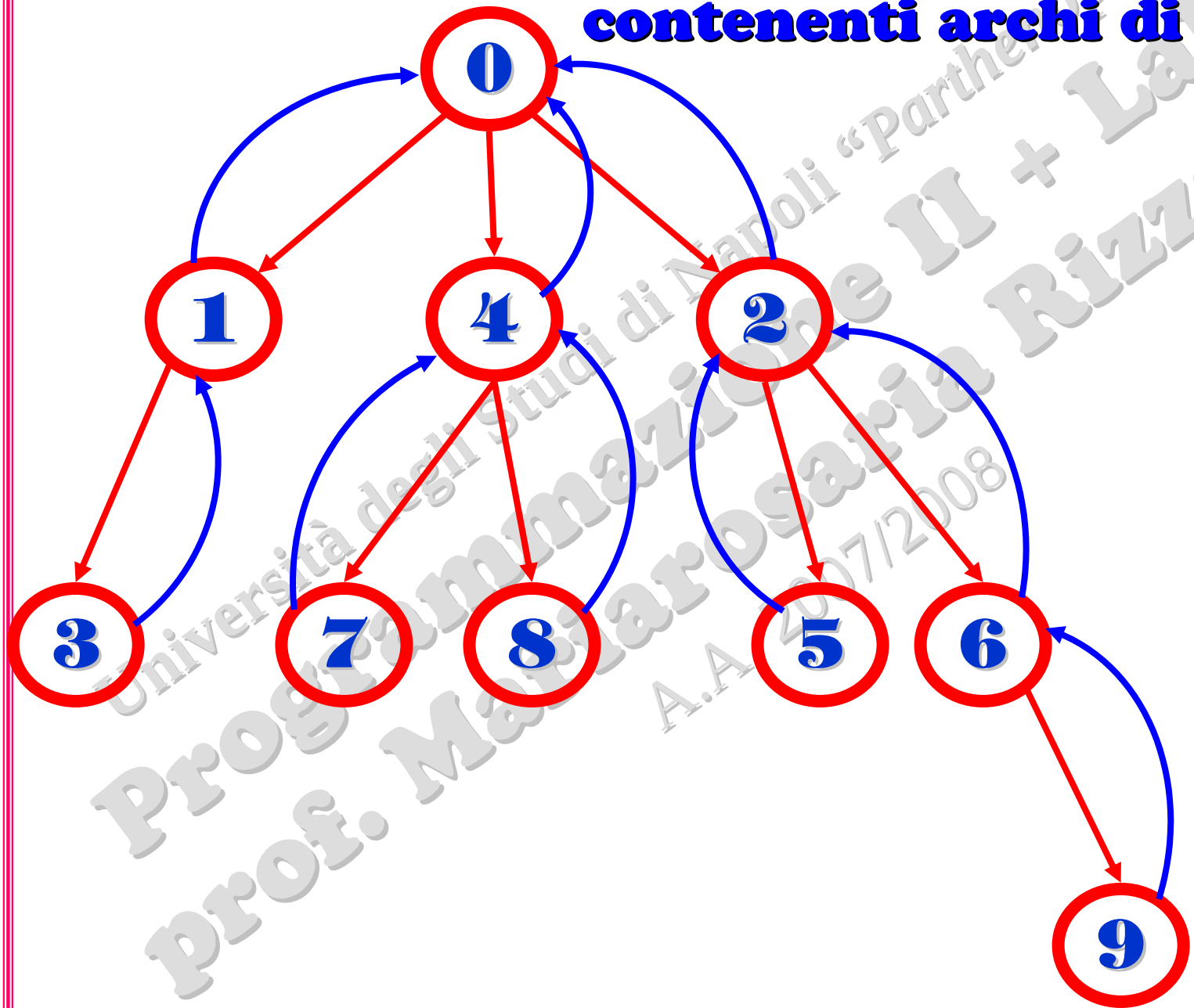
Prerequisiti richiesti: alberi generici, alberi binari

L'albero è una struttura dati gerarchica: se si vuole alterare l'ordine naturale di visita dei nodi (padre→figlio) si devono modificare i link esistenti ed inserirne eventualmente di nuovi

Esempio: per privilegiare la visita per fratelli

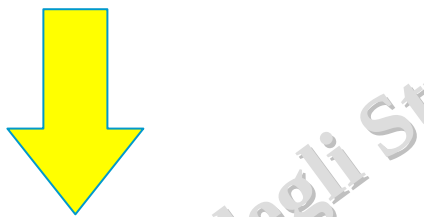


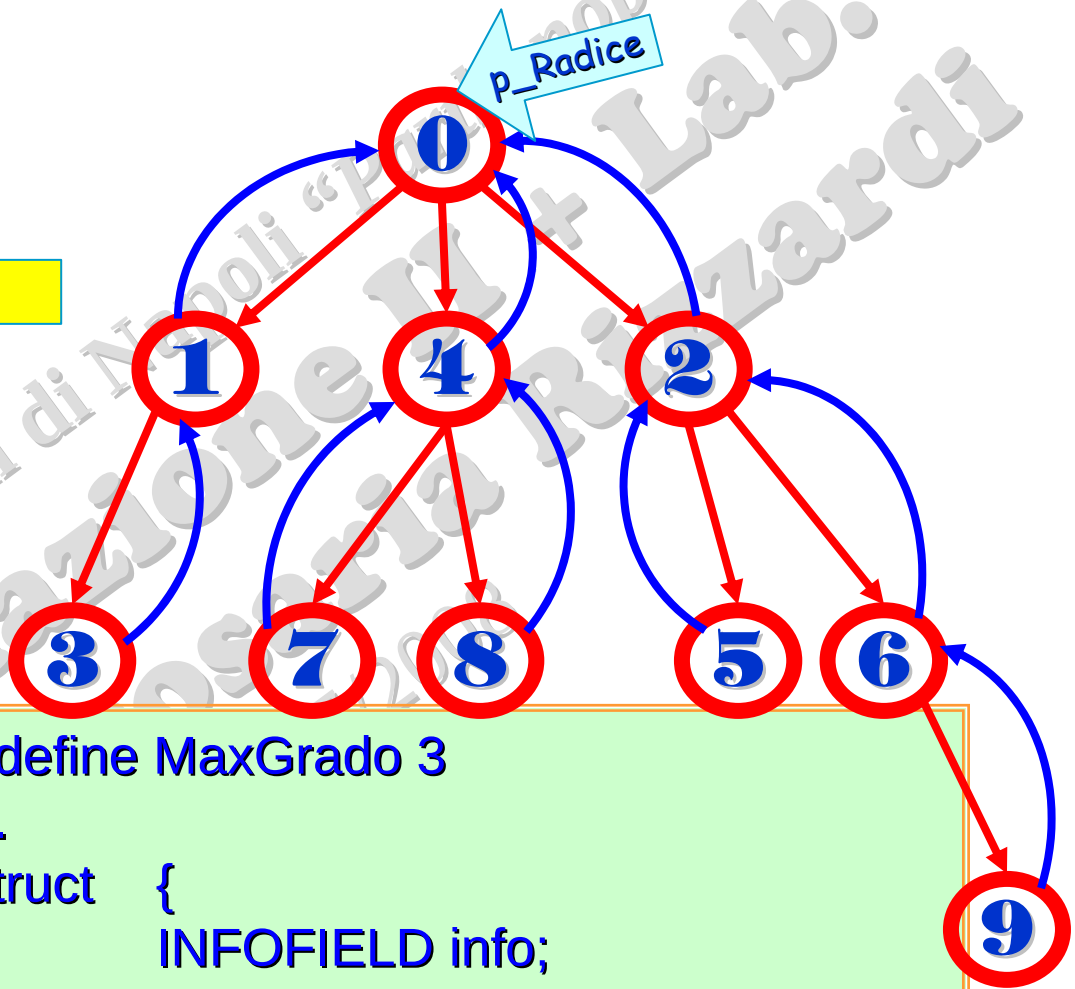
Alberi tramati (threaded trees): alberi contenenti archi di ritorno

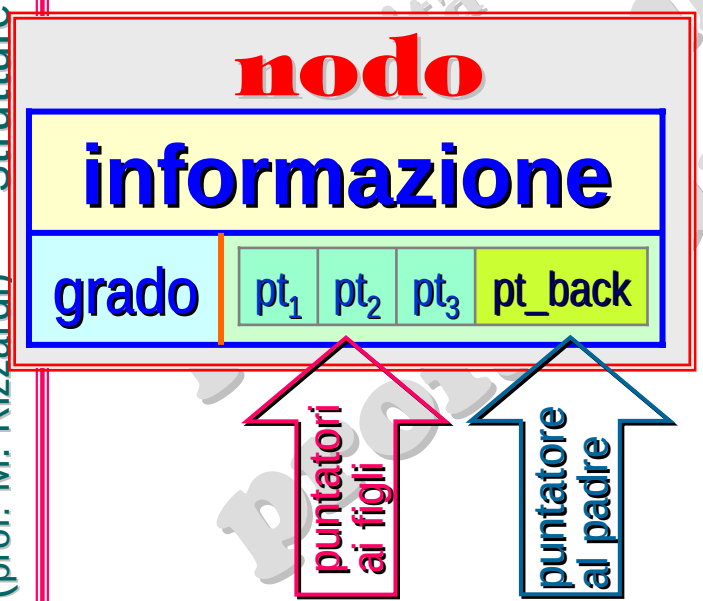


Esempio: rappresentazione dei dati

$$3 = \max_{1 \le i \le n} \text{grado}(\text{nodo}_i)$$

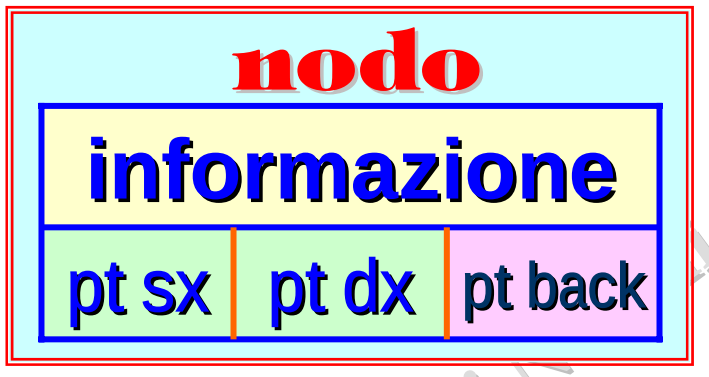


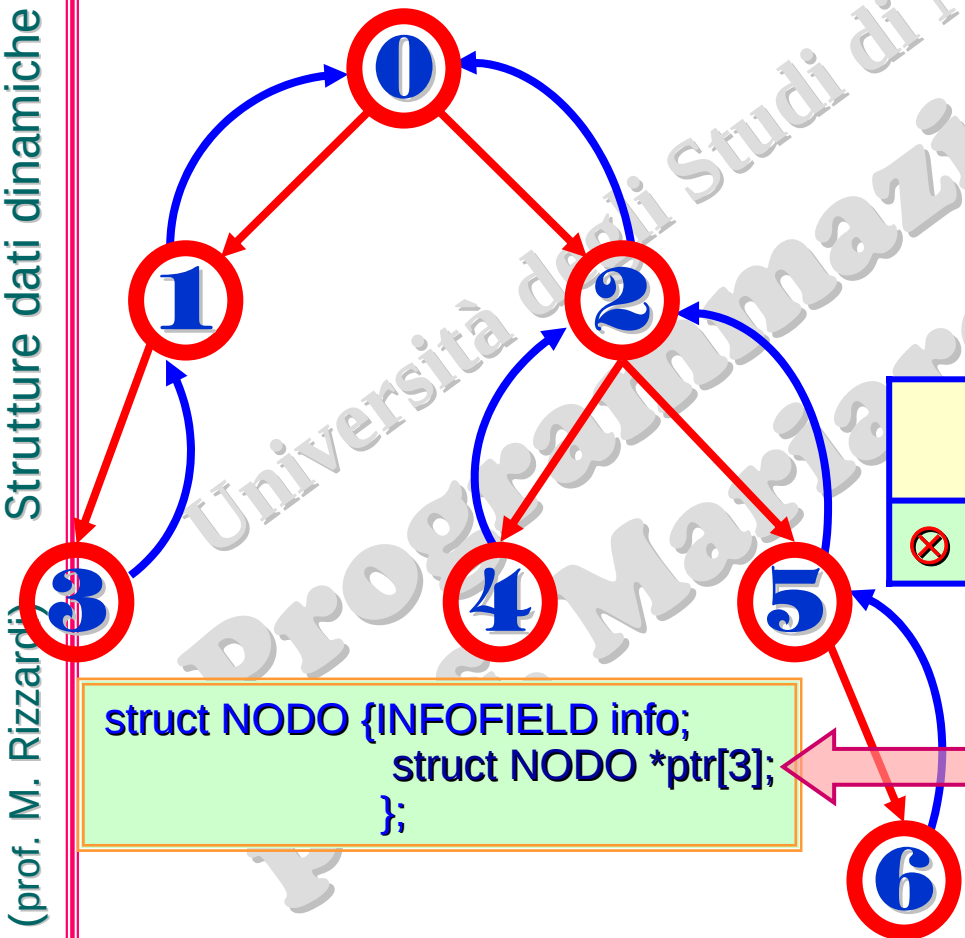


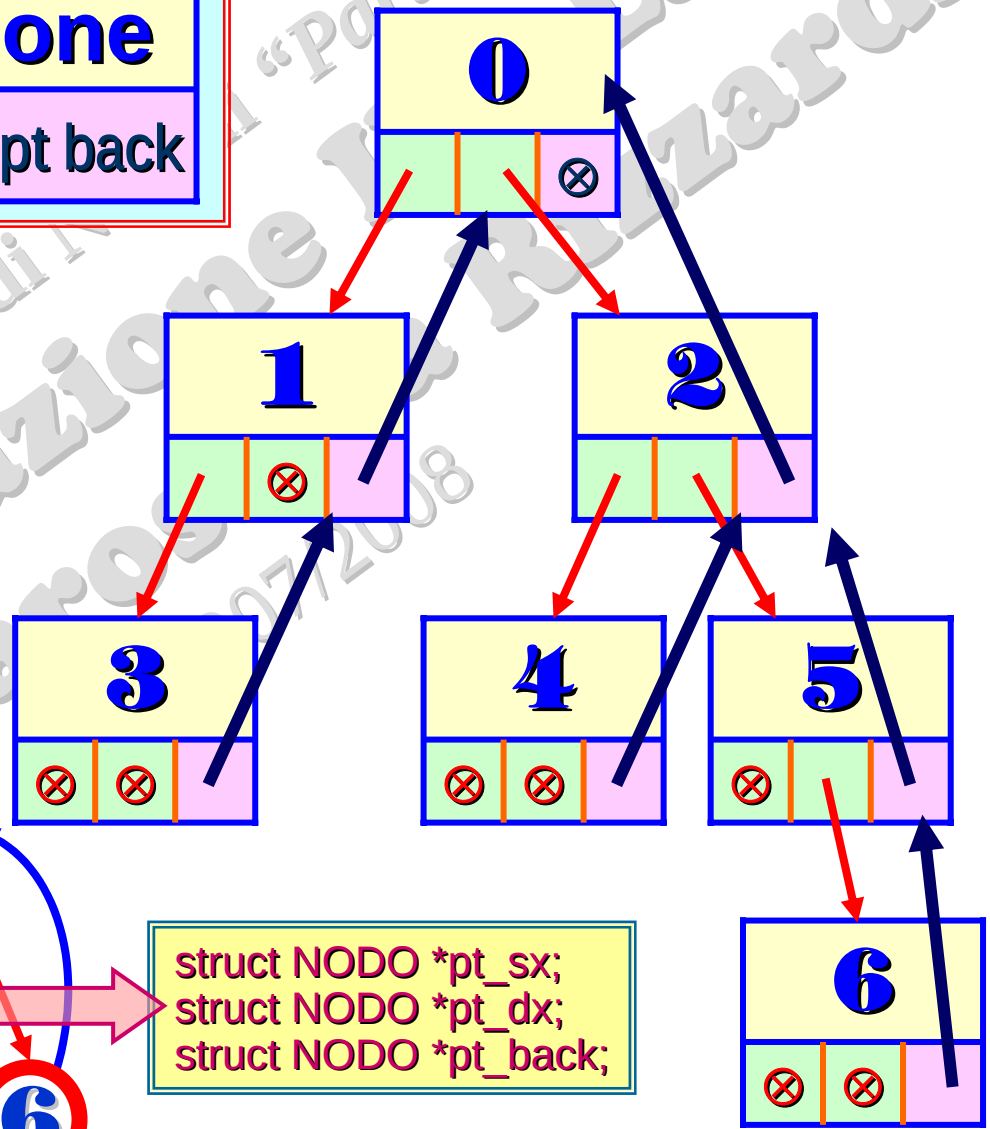



```
#define MaxGrado 3
...
struct    {
          INFOFIELD info;
          short grado;
          struct NODO *pt_figli[MaxGrado];
          struct NODO *pt_back;
          } NODO;
struct NODO *p_Radice;
```

Esempio: Albero binario tramato



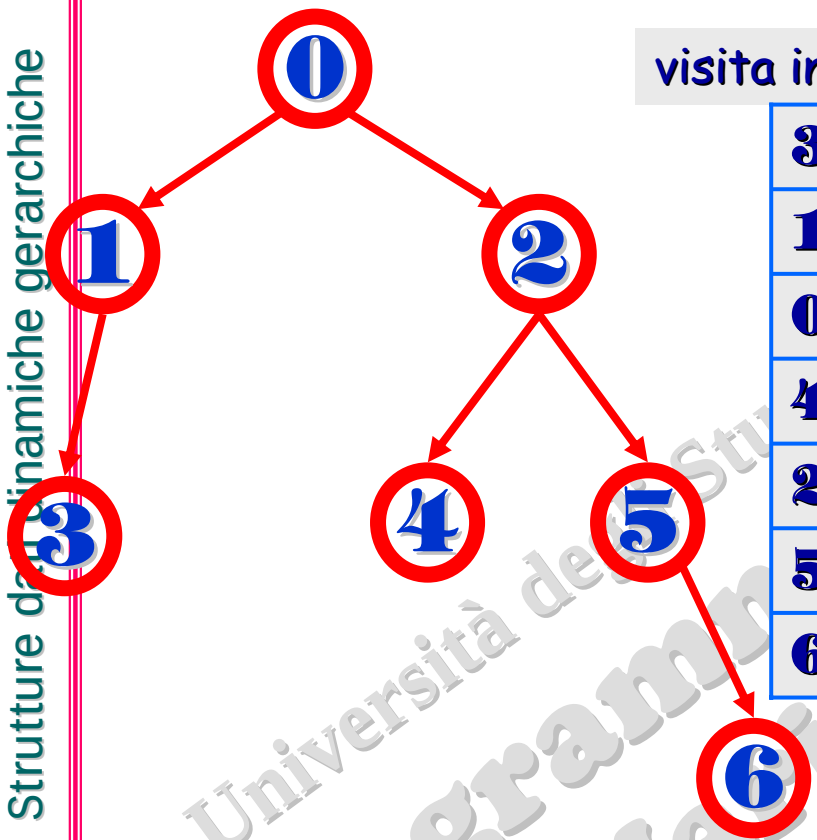





```
struct NODO {INFOFIELD info;
             struct NODO *ptr[3];
            };
```

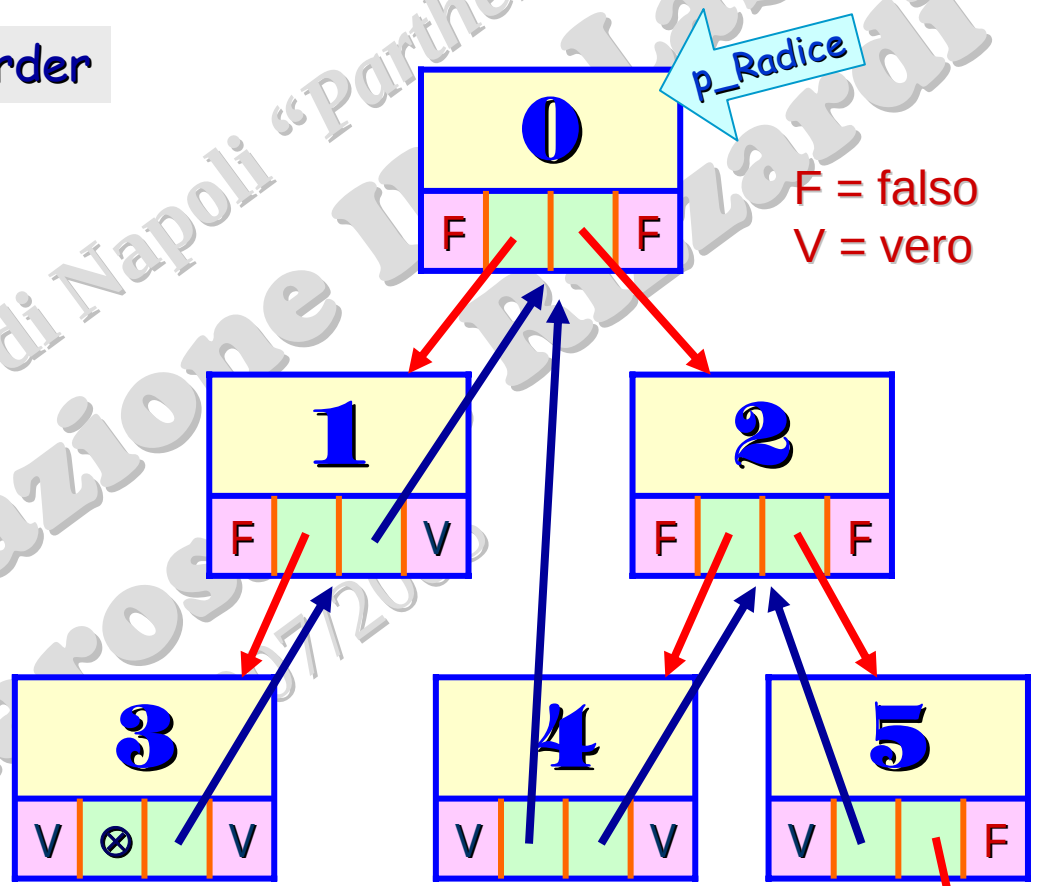
```
struct NODO *pt_sx;
struct NODO *pt_dx;
struct NODO *pt_back;
```

Esempio: albero binario tramato per facilitare l'attraversamento INORDER senza link aggiuntivi

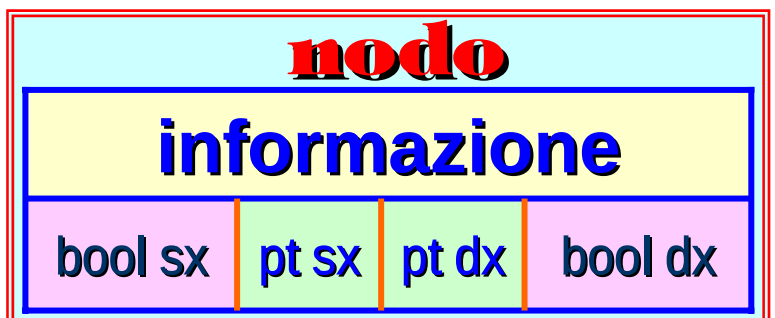


visita inorder

3
1
0
4
2
5
6




```
struct   {INFOFIELD info;
          char bool_sx;
          struct NODO *pt_sx;
          struct NODO *pt_dx;
          char bool_dx;
          } NODO;
```

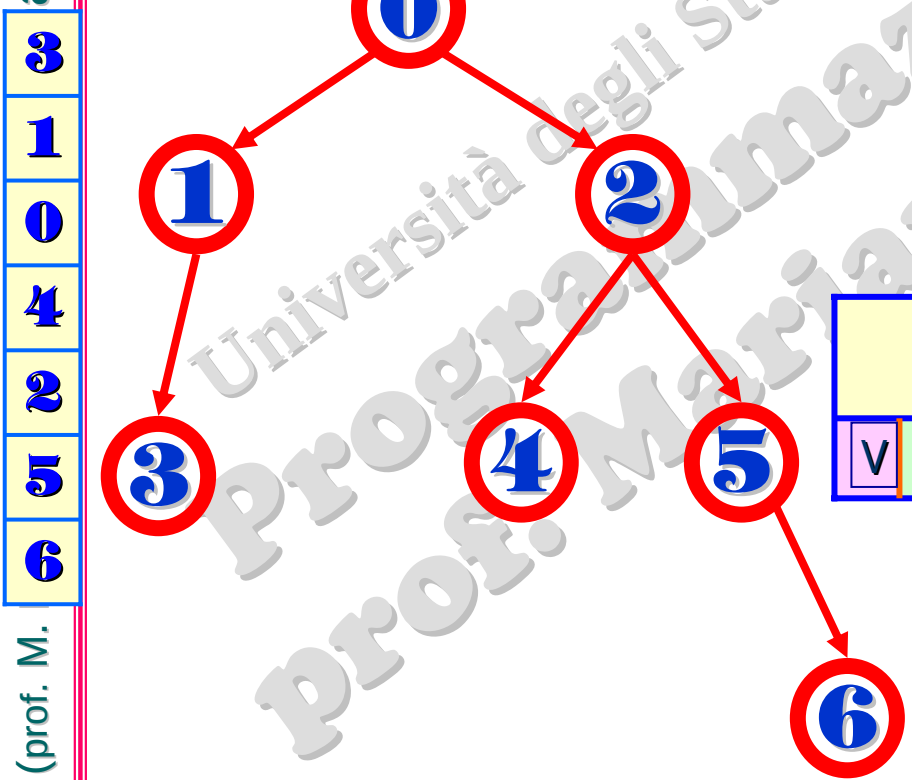


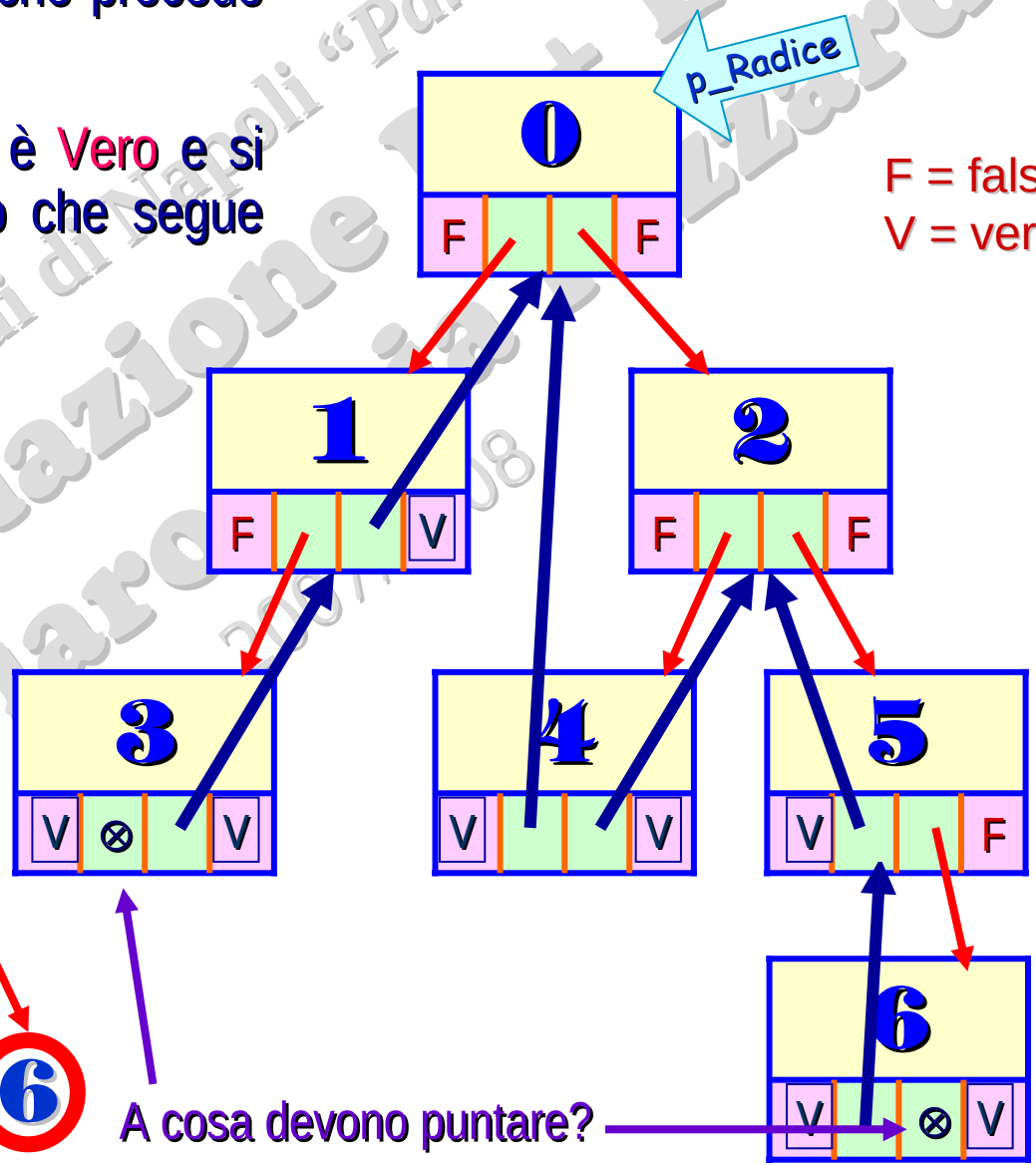
Regole di costruzione:

Se pt_sx è NULL, allora bool_sx è Vero e si ridefinisce pt_sx a puntare al nodo che precede quello corrente nella visita inorder

Se pt_dx è NULL, allora bool_dx è Vero e si ridefinisce pt_dx a puntare al nodo che segue quello corrente nella visita inorder

Quando bool_sx [risp. bool_dx] è Falso allora pt_sx [risp. pt_dx] punta al figlio sinistro [risp. destro]



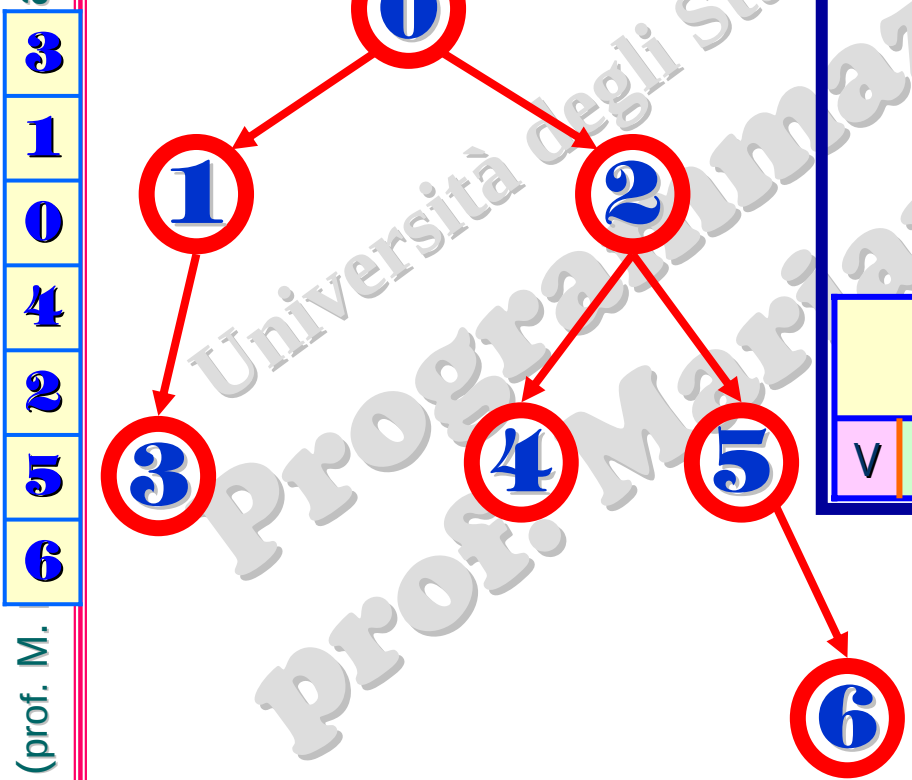


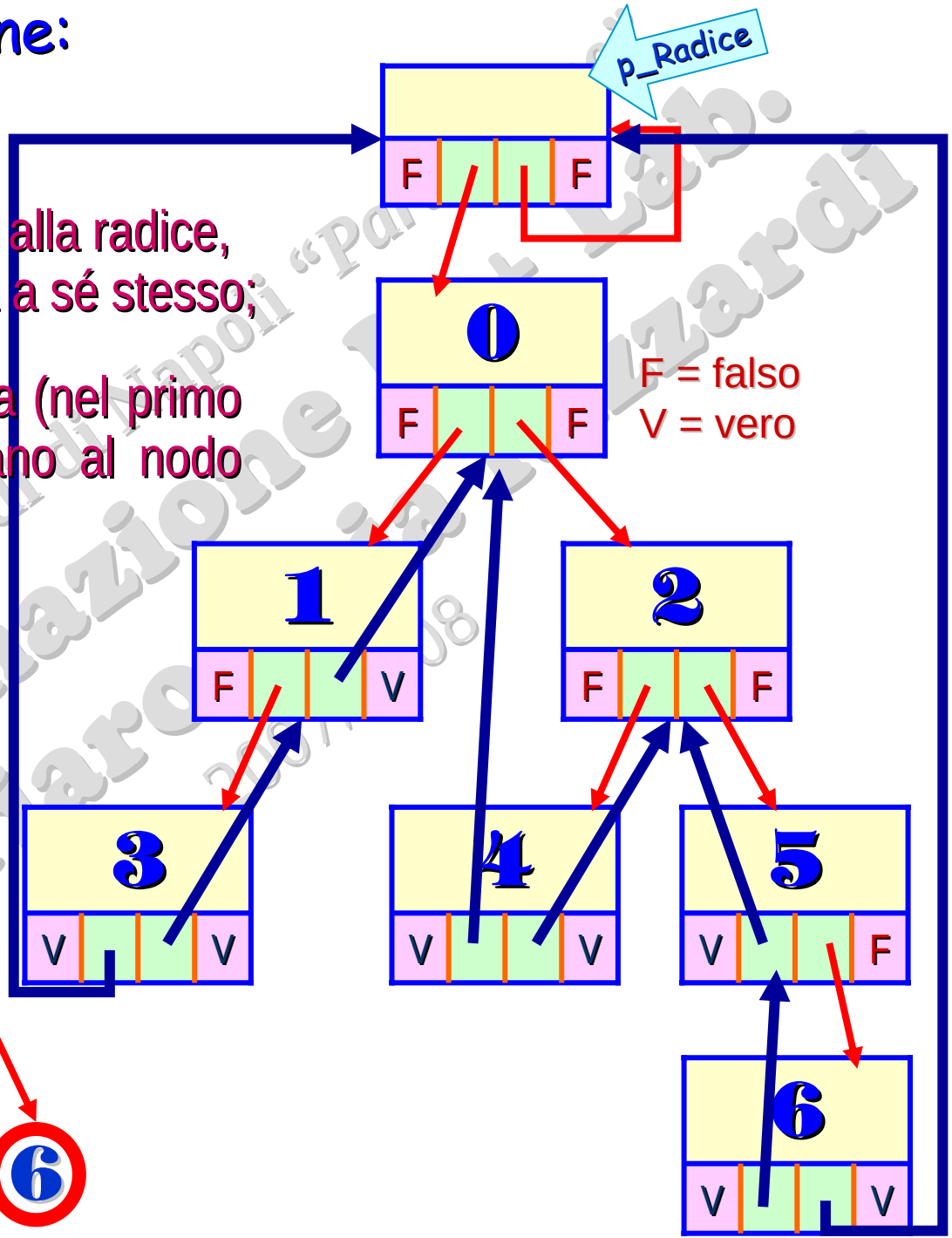
Regole di costruzione:

si aggiunge il nodo sentinella;

il pt_sx del nodo sentinella punta alla radice,
il pt_dx del nodo sentinella punta a sé stesso;

i due link che non puntano a nulla (nel primo e nell'ultimo nodo visitato) puntano al nodo sentinella;



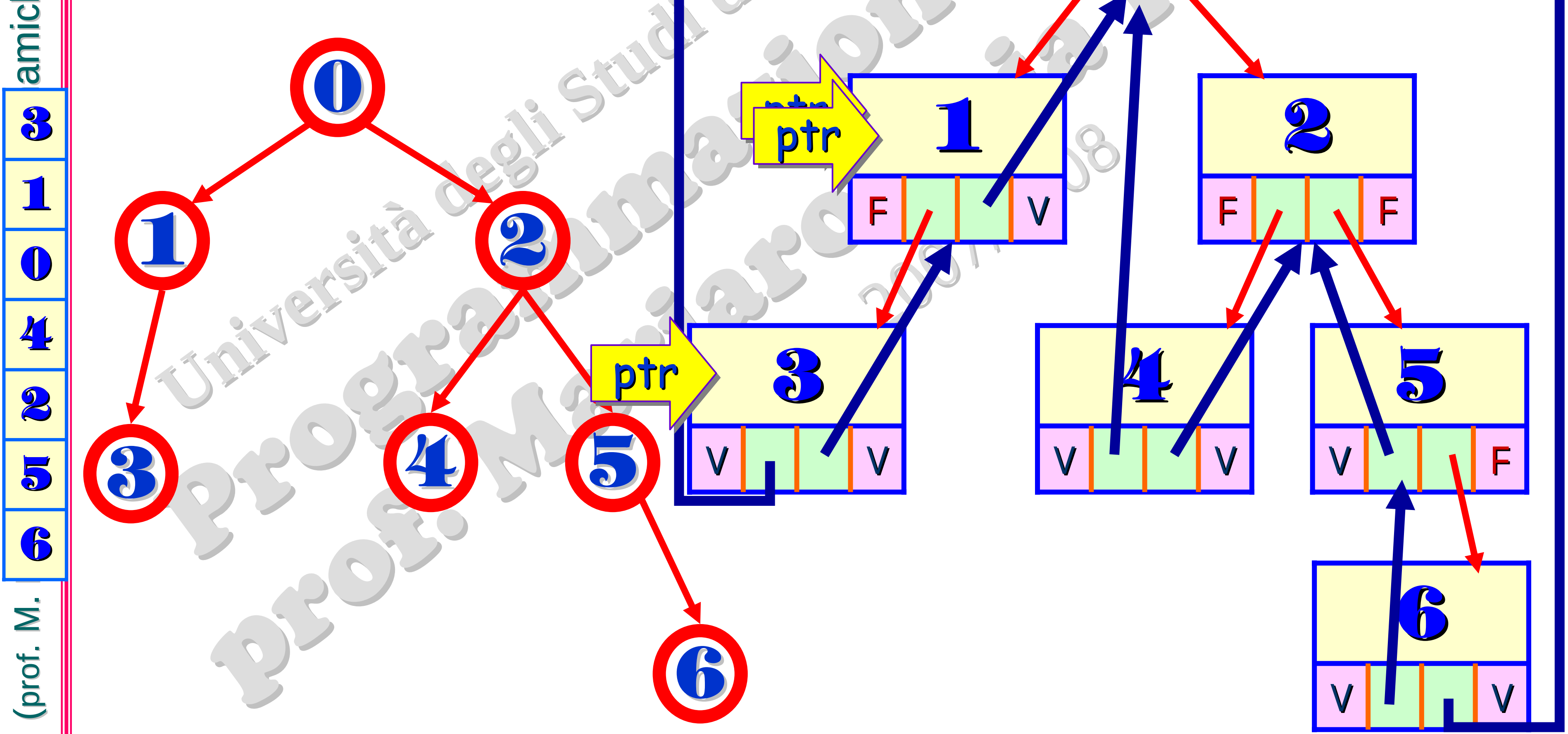


Regole di visita: successore del nodo corrente

ptr punta al nodo corrente;

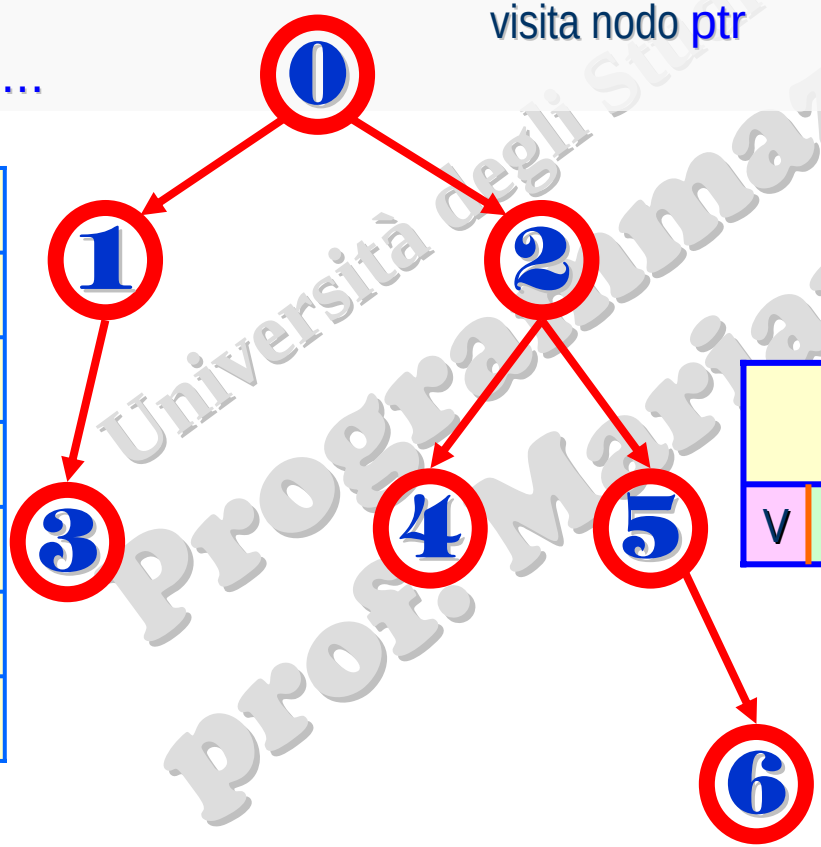
se ptr->bool_dx è Vero, il nodo successore è quello puntato da ptr->pt_dx;

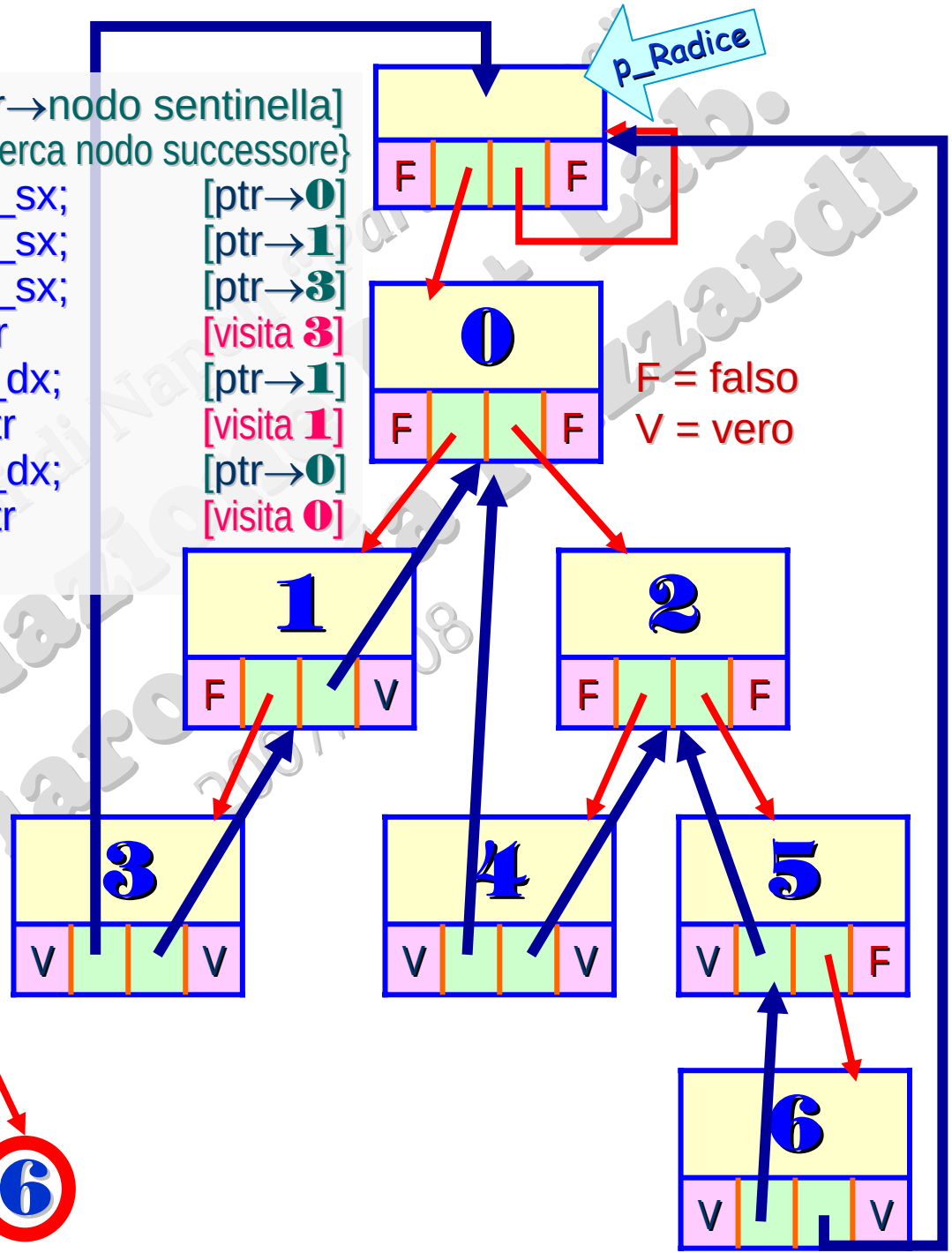
se ptr->bool_dx è Falso, allora finché ptr->bool_sx è Falso scende nel figlio sinistro;



Visita inorder: passi

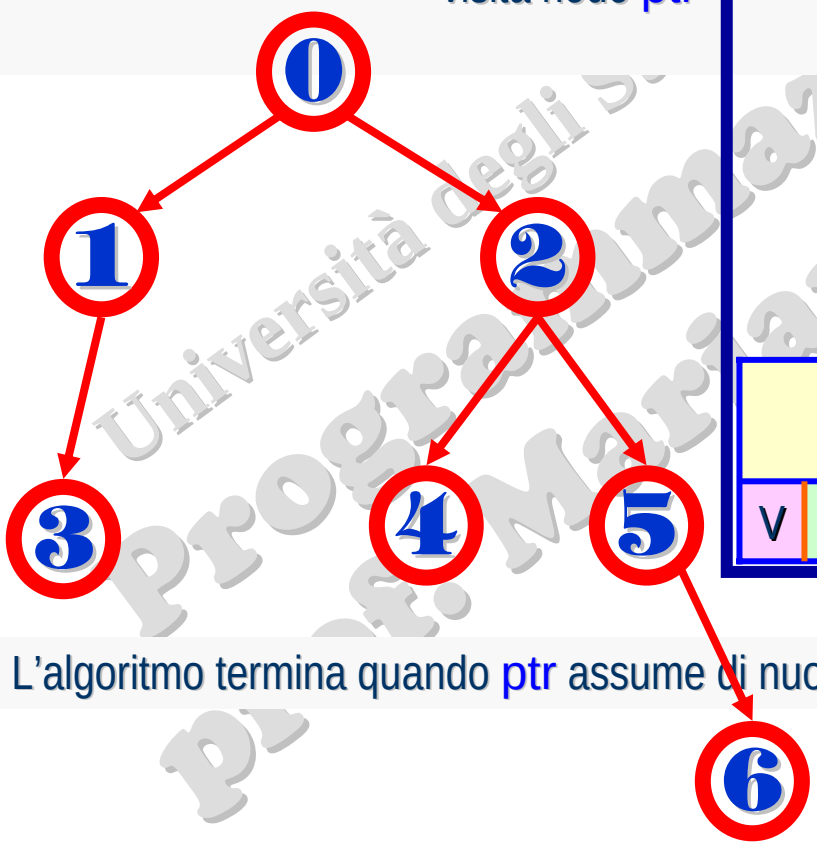
ptr=p_Radice; ptr=ptr->pt_dx; [ptr→nodo sentinella]
ptr->bool_dx è Falso quindi {cerca nodo successore}
ptr->bool_sx è Falso allora ptr=ptr->pt_sx; [ptr→0]
ptr->bool_sx è Falso allora ptr=ptr->pt_sx; [ptr→1]
ptr->bool_sx è Falso allora ptr=ptr->pt_sx; [ptr→3]
ptr->bool_sx è Vero allora visita nodo ptr [visita 3]
ptr->bool_dx è Vero allora ptr=ptr->pt_dx; [ptr→1]
visita nodo ptr [visita 1]
ptr->bool_dx è Vero allora ptr=ptr->pt_dx; [ptr→0]
visita nodo ptr [visita 0]
...

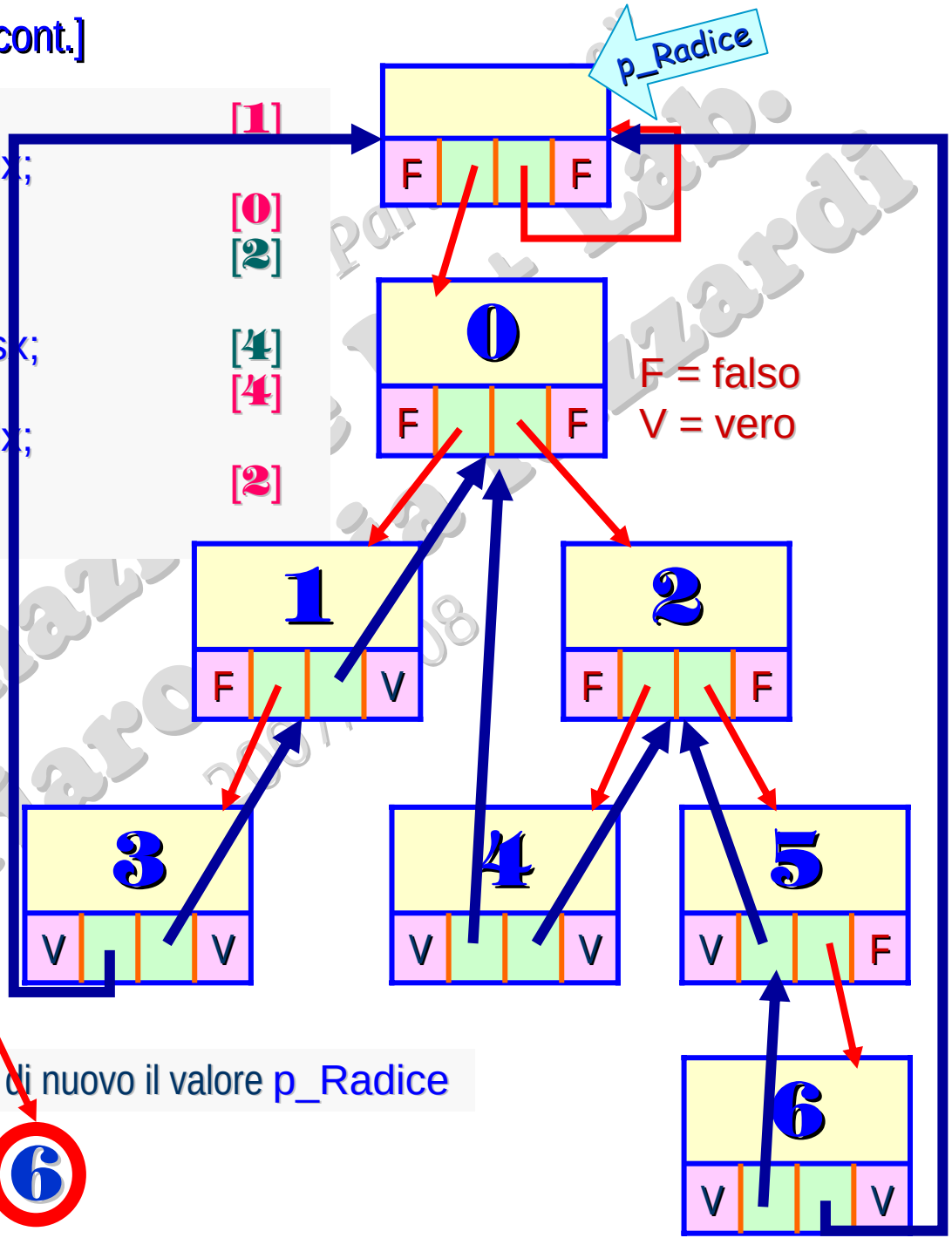




Visita inorder: passi [cont.]

visita nodo ptr [1]
ptr->bool_dx è Vero allora ptr=ptr->pt_dx;
visita nodo ptr [0]
ptr=ptr->pt_dx; [2]
ptr->bool_dx è Falso quindi
ptr->bool_sx è Falso allora ptr=ptr->pt_sx; [4]
ptr->bool_sx è Vero allora visita nodo ptr [4]
ptr->bool_dx è Vero allora ptr=ptr->pt_dx;
visita nodo ptr [2]
...





L'algoritmo termina quando ptr assume di nuovo il valore p_Radice