Modulo: Strutture dati dinamiche gerarchiche [P2_09]

Unità didattica: generalità sulla struttura dati albero (tree) [1-T]

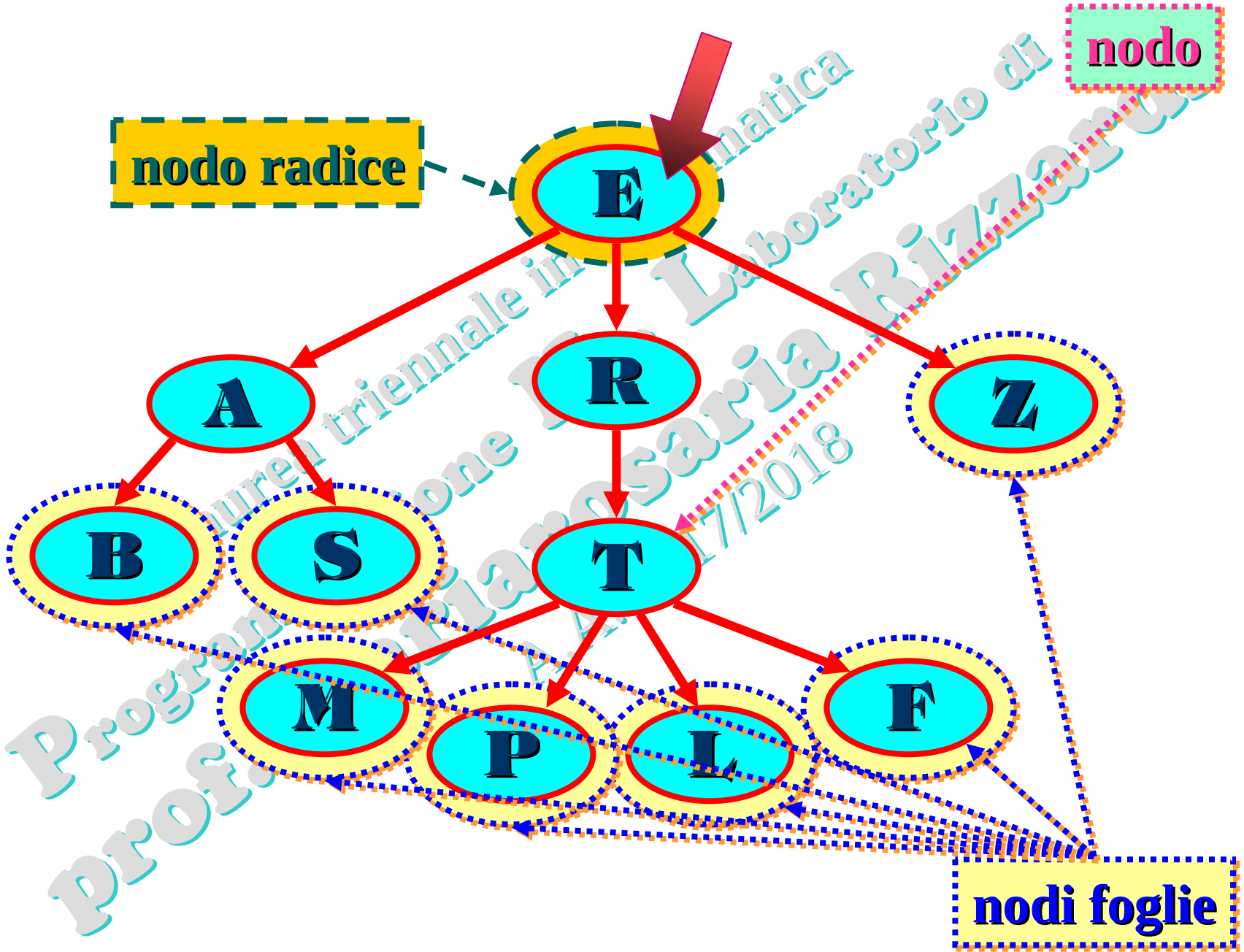
Titolo: Definizioni e proprietà

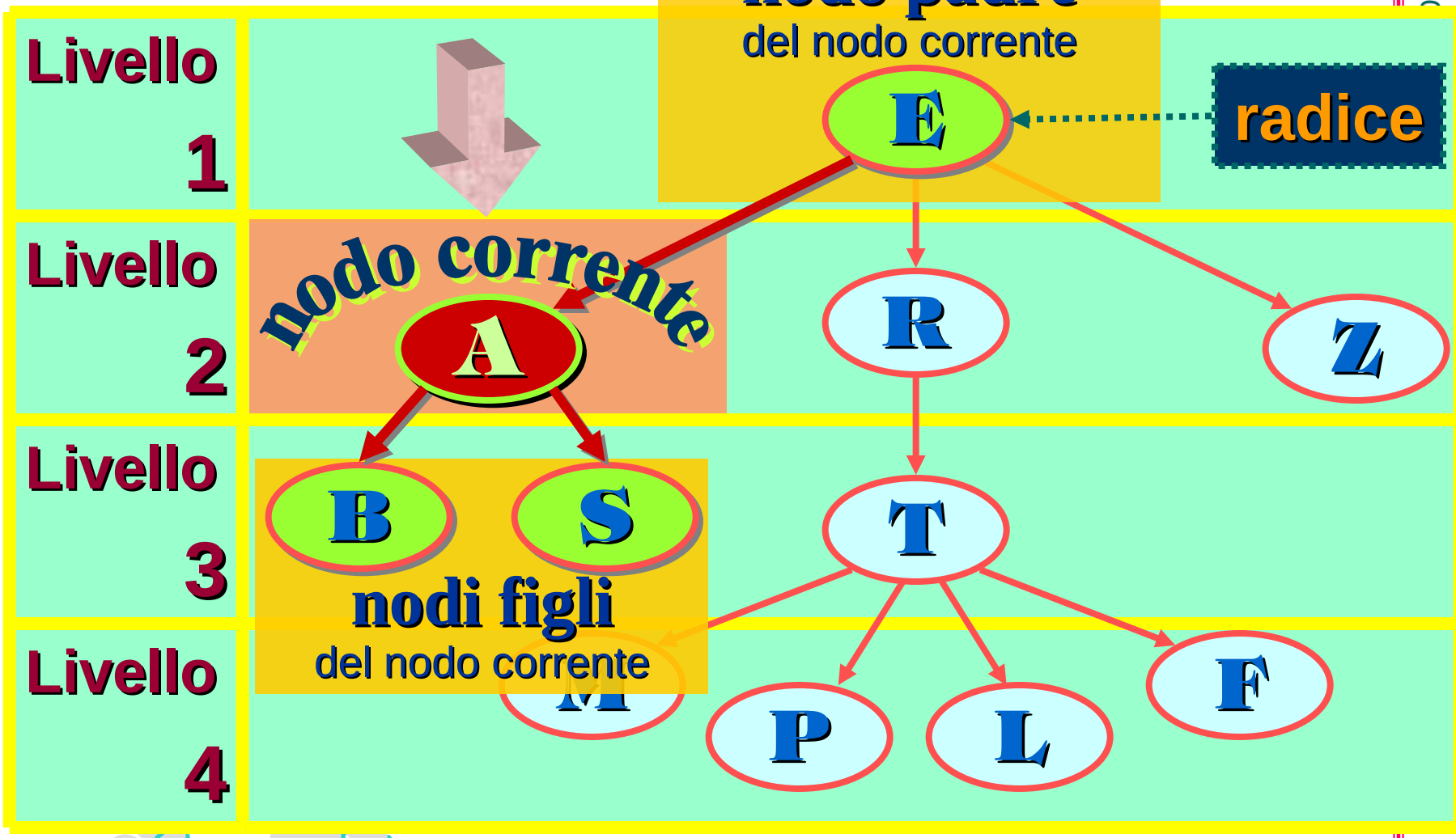
Argomenti trattati:

- ✓ Definizione di nodo radice, nodo corrente, nodo figlio, nodo padre, nodi foglie
- ✓ Grado e sottoalbero di un nodo
- ✓ Livelli di un albero ed algoritmo di visita di un albero qualsiasi per livelli

Prerequisiti richiesti: array, record, strutture dati dinamiche lineari

Struttura dati gerarchica: albero (tree)



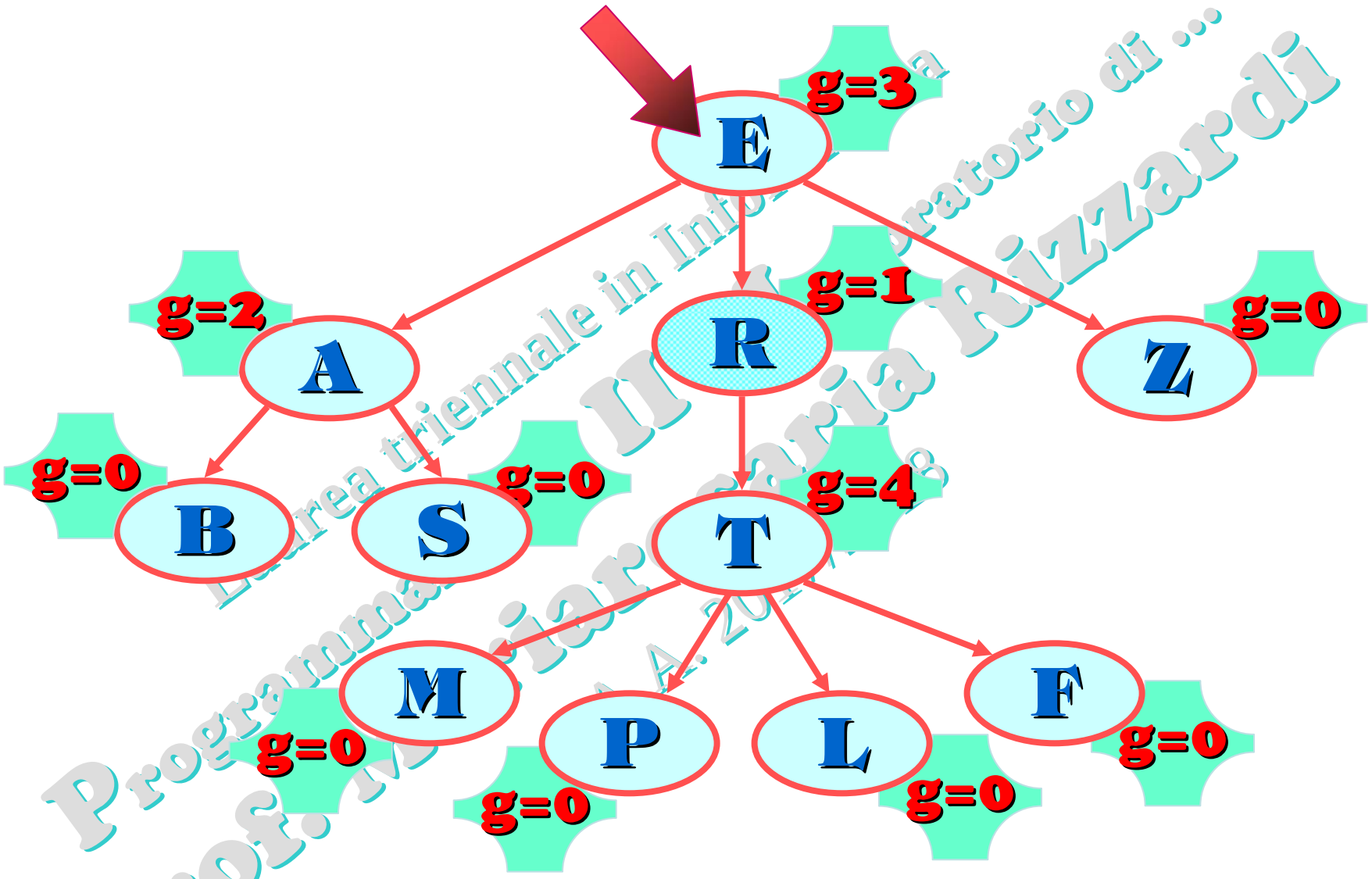


La radice è l'unico nodo che non ha padre.

Le foglie sono gli unici nodi che non hanno figli.

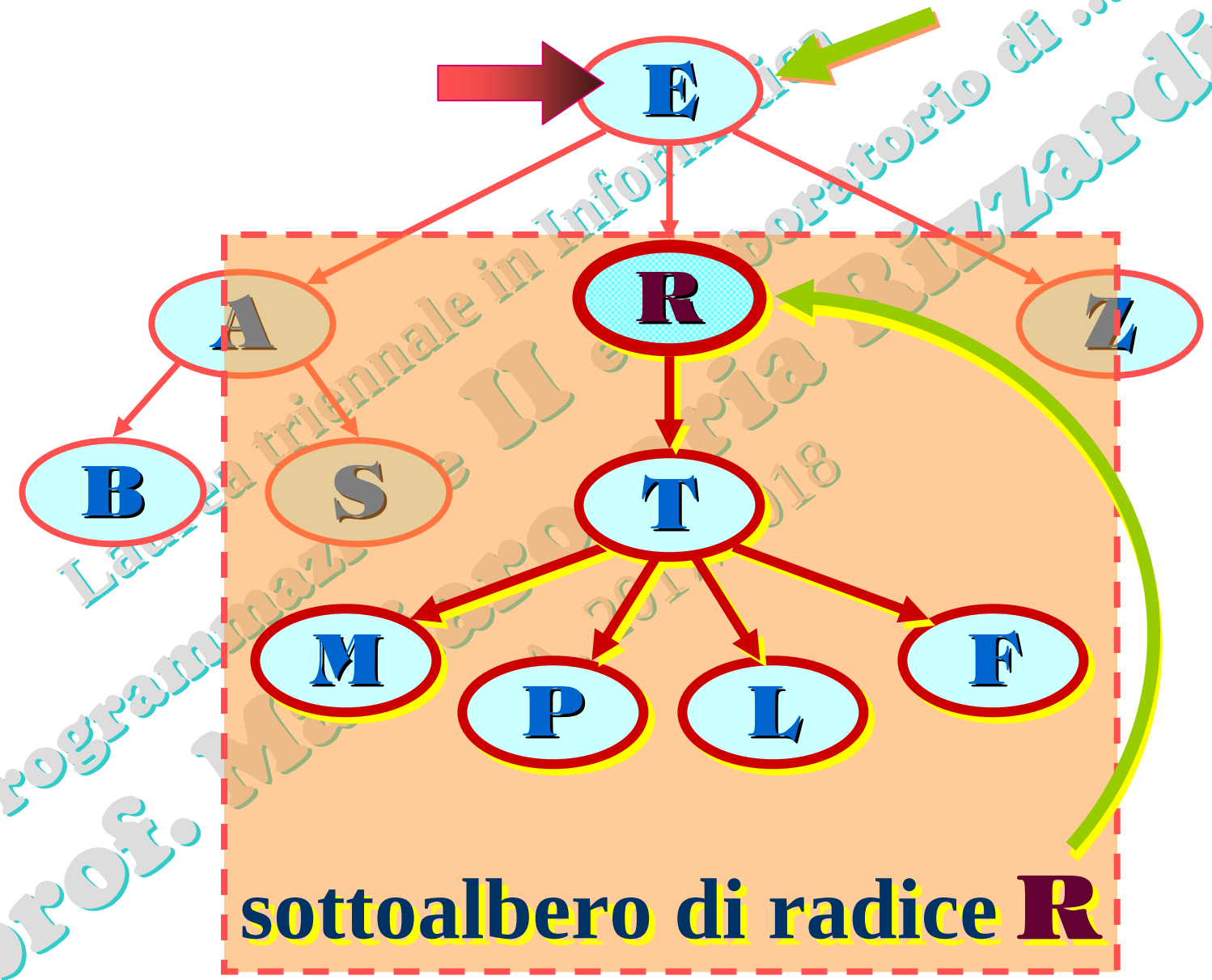
Ogni nodo (tranne la radice) ha un unico padre.

grado g di un nodo = numero di figli

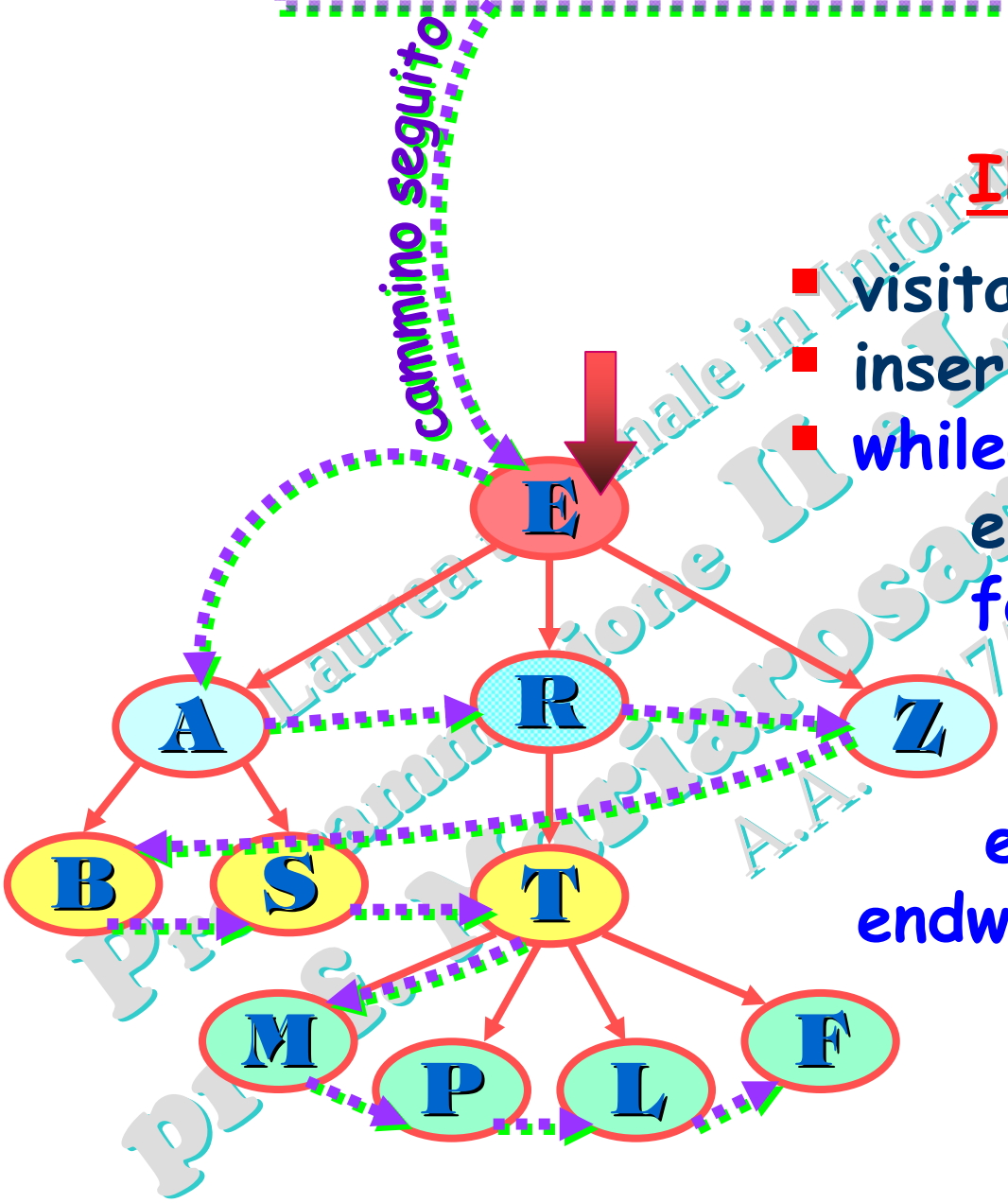


le foglie hanno sempre grado 0

albero di radice E



Visita di un albero (qualsiasi) per livelli

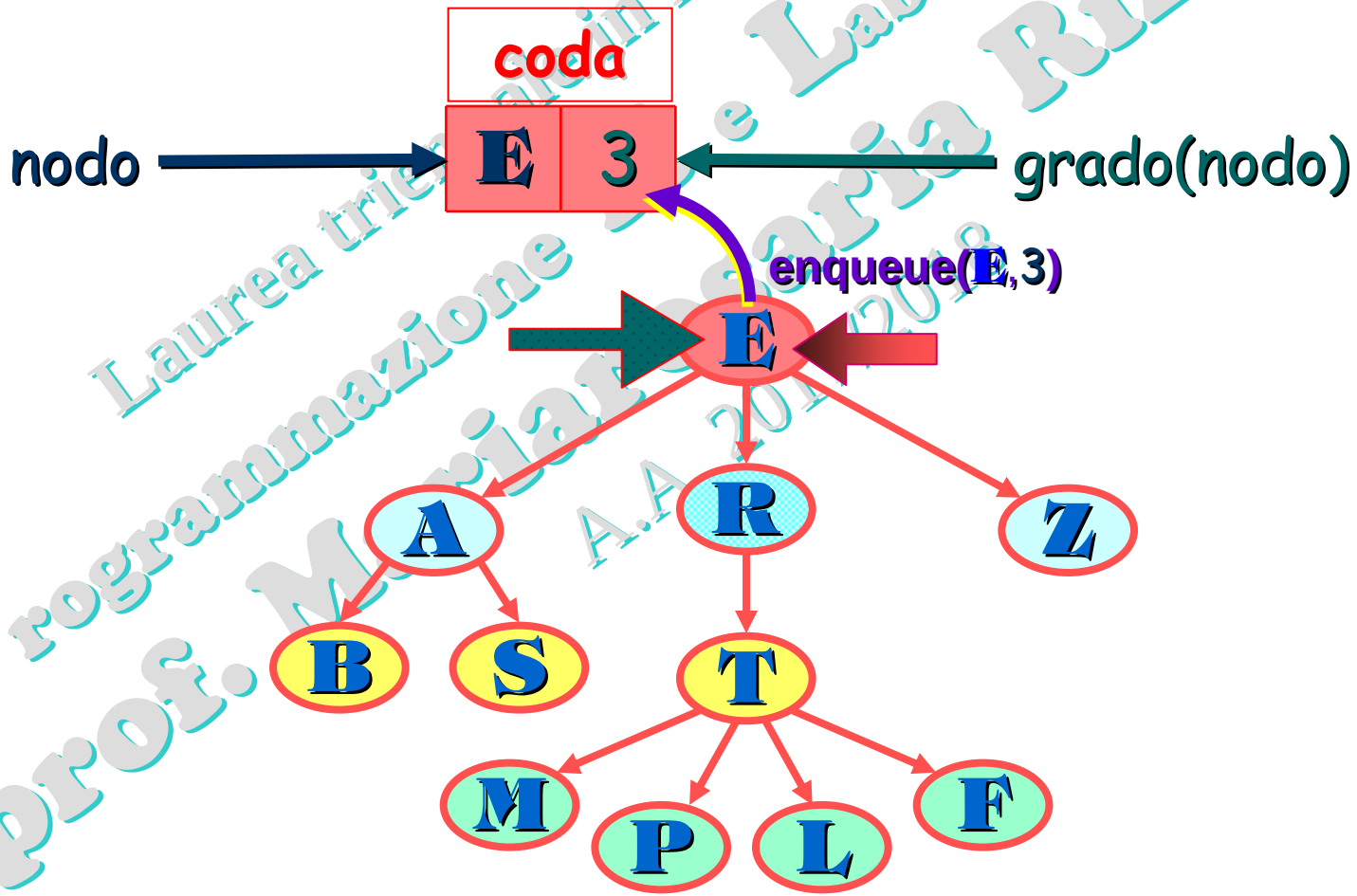


IDEA algoritmo

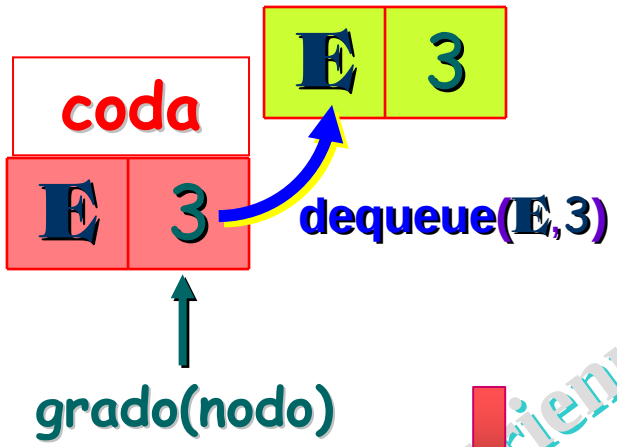
- visita la radice;
- inseriscila nella coda;
- **while** coda non vuota
  estrai nodo dalla coda;
  for k=1 **to** grado(nodo),
    visita figlio k^{simo};
    inseriscilo nella coda;
  endfor
endwhile

Visita di un albero per livelli [1]

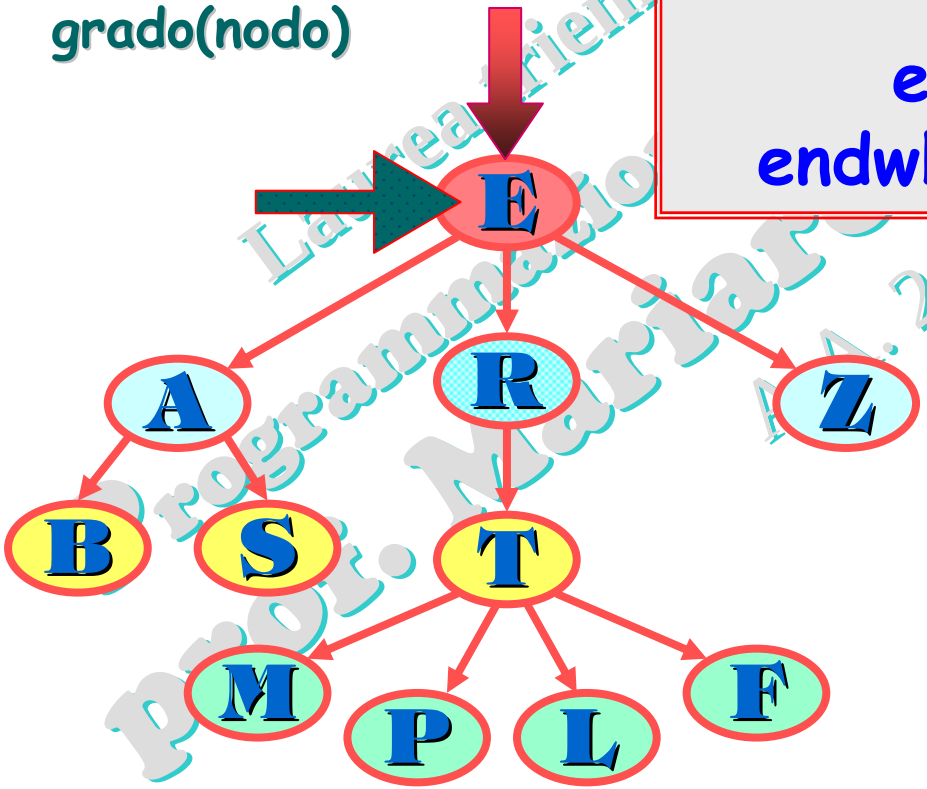
- visita la radice;
- inseriscila nella coda;

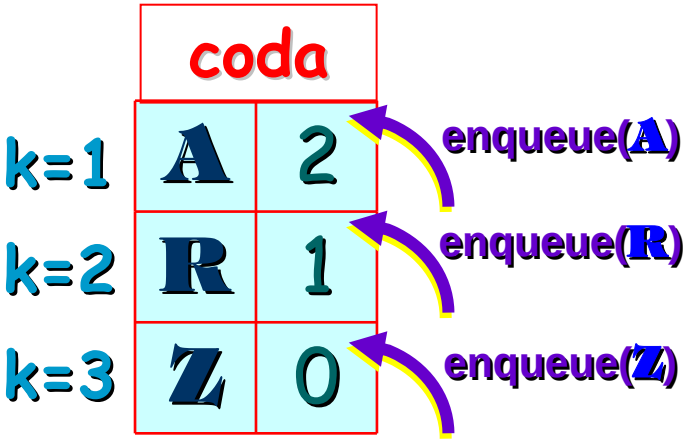


Visita di un albero per livelli [2]



- while coda non vuota
  estrai nodo dalla coda;
  for k=1 to grado(nodo),
  visita figlio k^{simo};
  inseriscilo nella coda;
  endfor
  endwhile





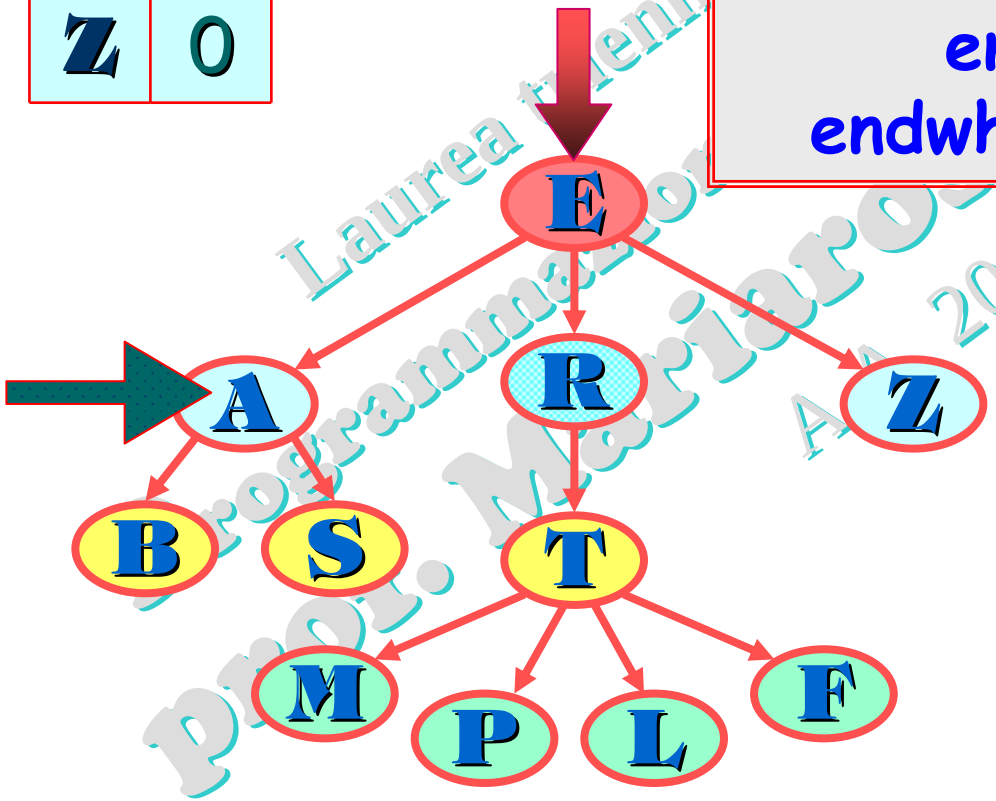
Visita di un albero per livelli [3]

A	2

coda	
A	2
R	1
Z	0

dequeue(A,2)

- while coda non vuota
  estrai nodo dalla coda;
  for k=1 to grado(nodo),
  visita figlio k^{simo};
  inseriscilo nella coda;
  endfor
  endwhile



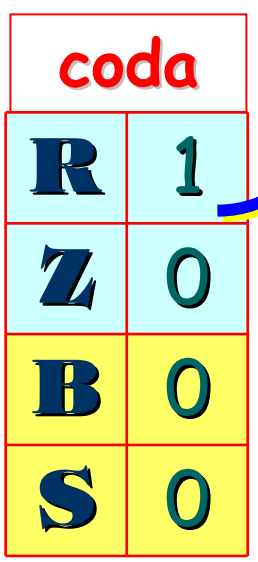
coda	
R	1
Z	0
B	0
S	0

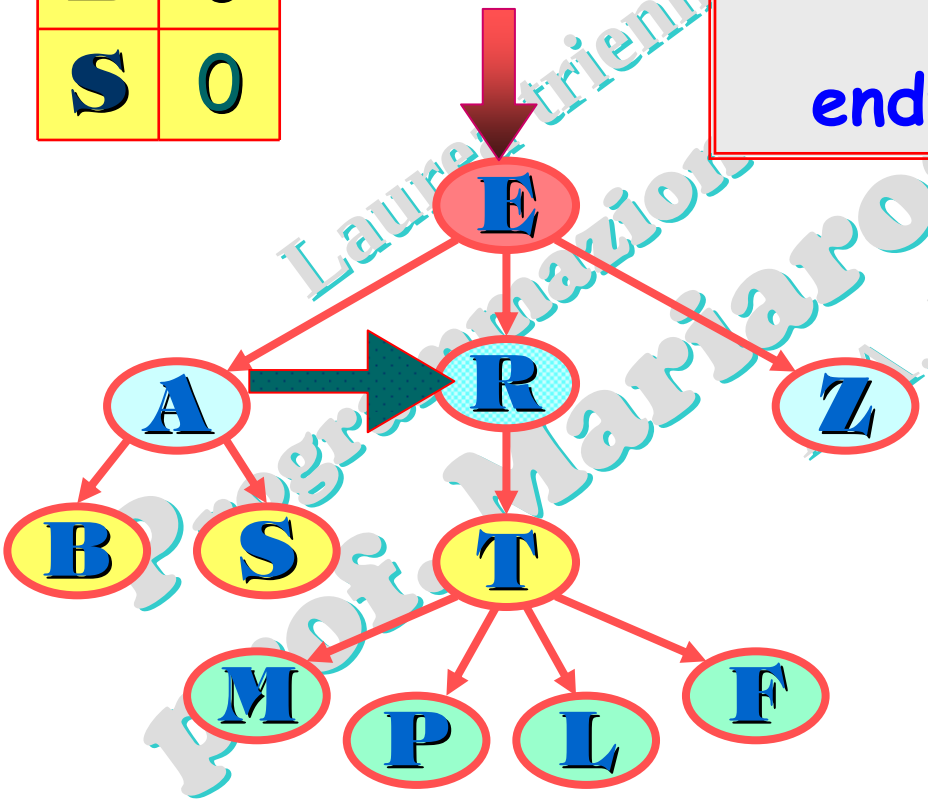
k=1 enqueue(B)

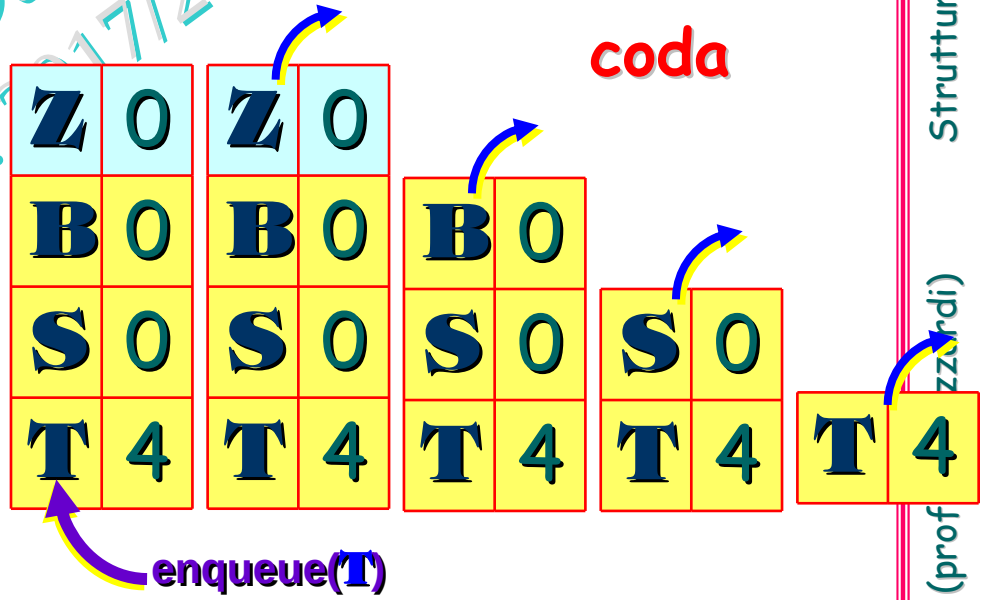
k=2 enqueue(S)

Visita di un albero per livelli [4]

- while coda non vuota
  estrai nodo dalla coda;
  for k=1 to grado(nodo),
  visita figlio k^{simo};
  inseriscilo nella coda;
  endfor
  endwhile

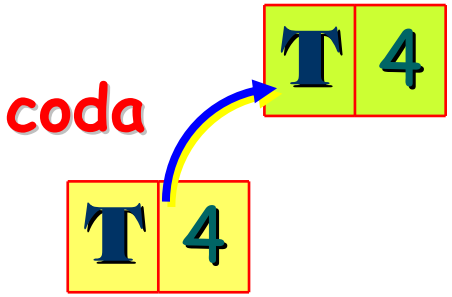


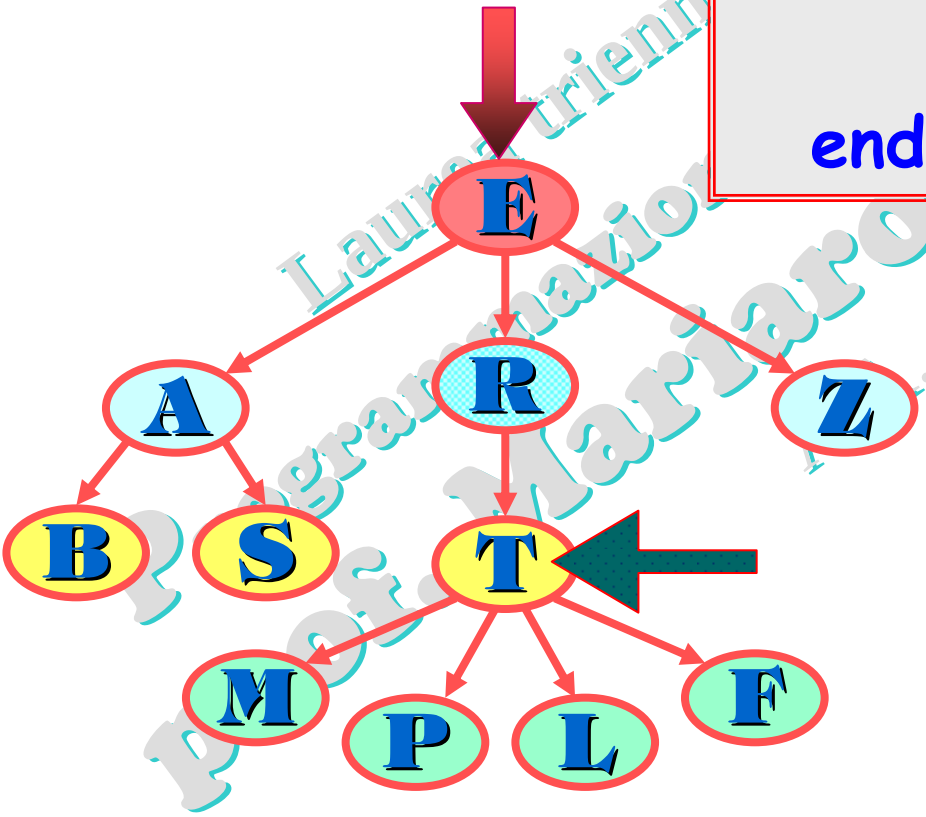


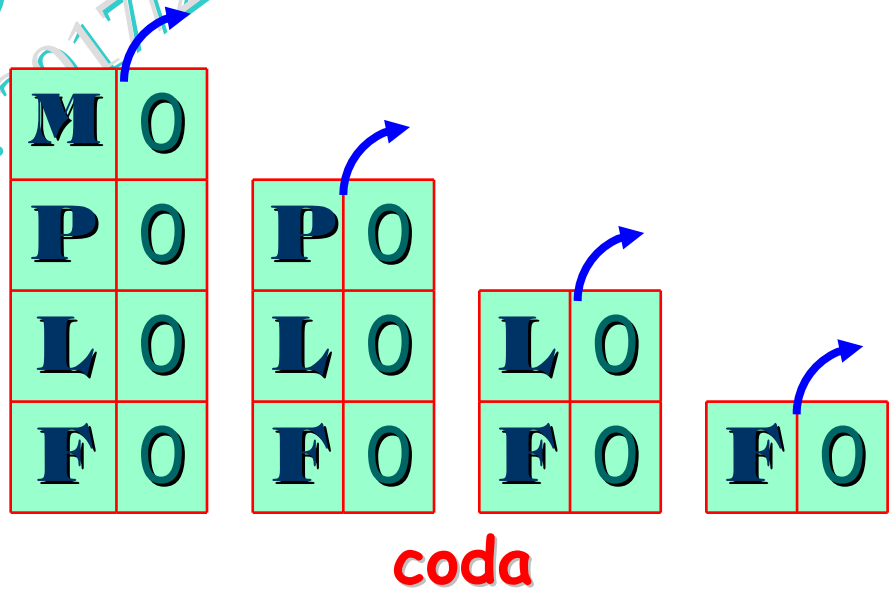


Visita di un albero per livelli [5]

- while coda non vuota
  estrai nodo dalla coda;
  for k=1 to grado(nodo),
  visita figlio k^{simo};
  inseriscilo nella coda;
  endfor
  endwhile



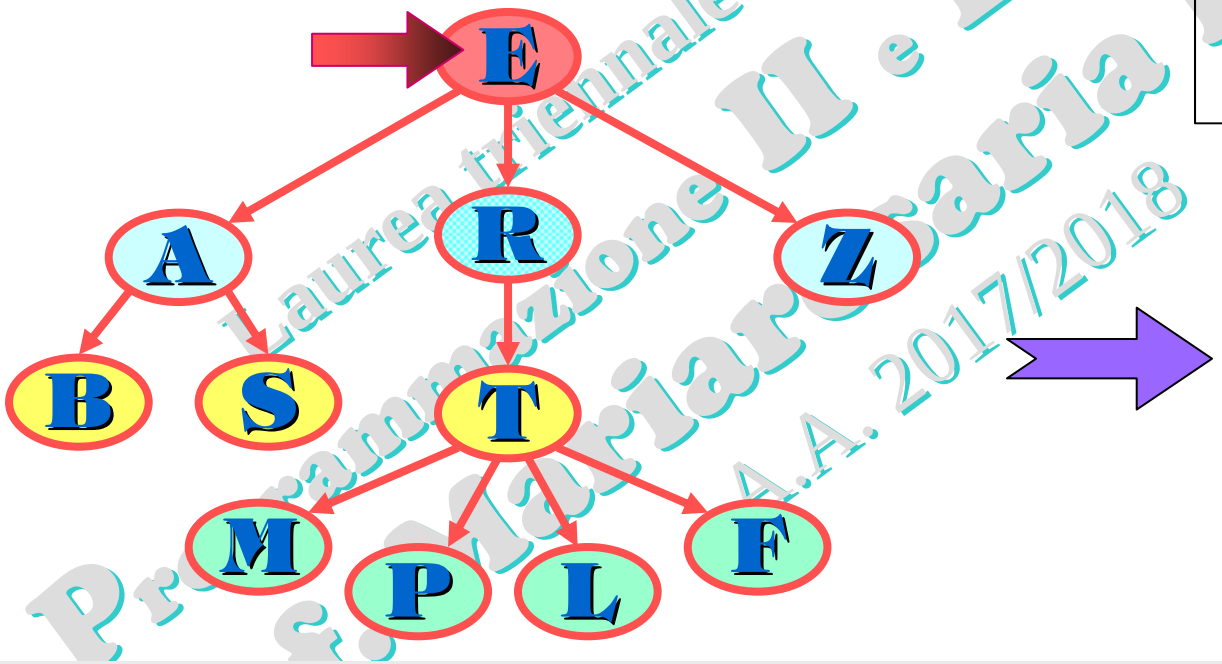


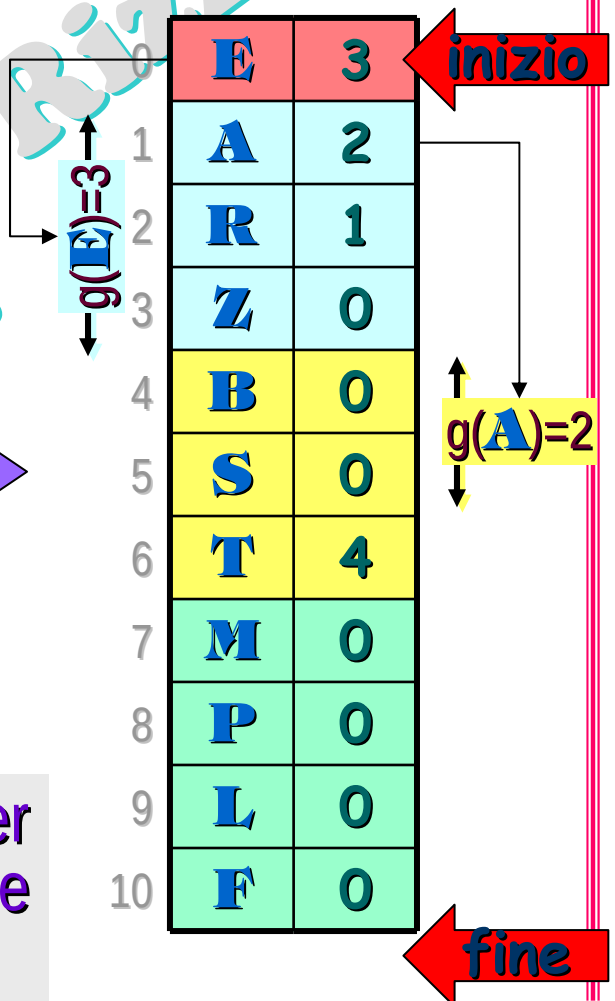


In C l'algoritmo della visita per livelli può essere utilizzato anche per la costruzione dell'albero

INPUT: i nodi, ordinati per livello, ed il relativo grado.

OUTPUT: array di struct contenente il campo informazione di un nodo ed il suo grado.



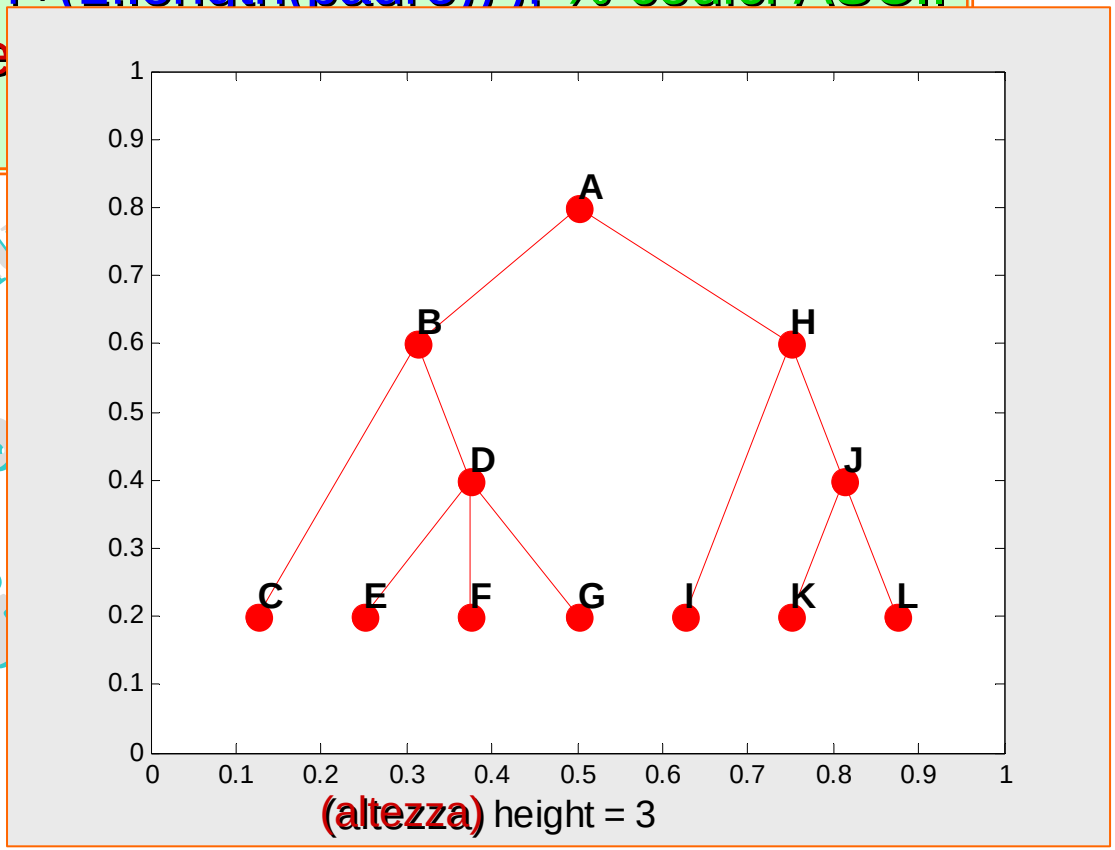


L'array può essere gestito come una coda per visitare l'albero mediante i due puntatori **inizio** e **fine**

Esempio

In MATLAB la funzione treeplot() disegna un albero qualsiasi. L'albero è rappresentato come array dove per ogni nodo è assegnato l'indice del padre.

```
padre = [0 1 2 2 4 4 4 1 8 8 10 10];
nodi = char(64+(1:length(padre))'); % codici ASCII
treeplot(padre
...
```



nodo	...di padre
1 A	0
2 B	1
3 C	2
4 D	2
5 E	4
6 F	4
7 G	4
8 H	1
9 I	8
10 J	8
11 K	10
12 L	10

Esercizio:

1

Scrivere *function C* per la costruzione e la visita per livelli di un albero qualsiasi (rappresentato mediante array).
In input si conosce per ogni nodo l'indice del padre. [liv. 2]

[suggerimento: la struct che definisce il generico nodo dell'albero deve contenere i seguenti campi: l'informazione, il suo grado ed un array di puntatori di dimensione pari al massimo grado dei nodi]

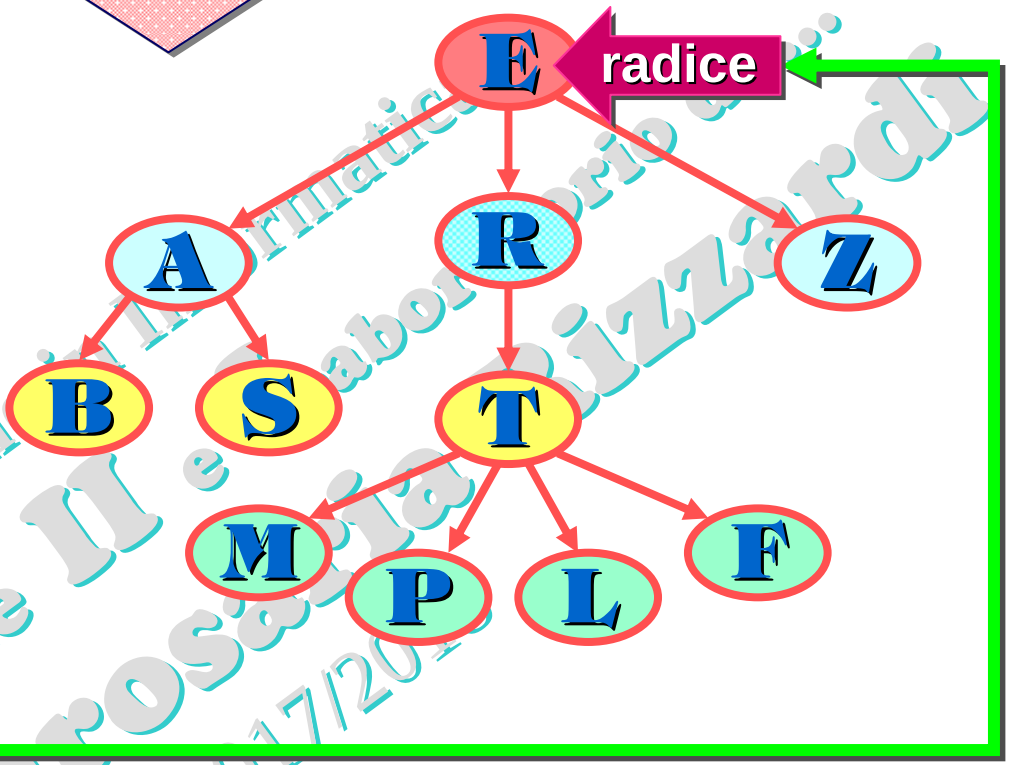
help

Dati di input:
nodi in ordine qualsiasi
- informazione del nodo,
- padre del nodo

help

0	M	2
1	A	4
2	T	10
3	F	2
4	E	–1
5	Z	4
6	L	2
7	S	1
8	B	1
9	P	2
10	R	4

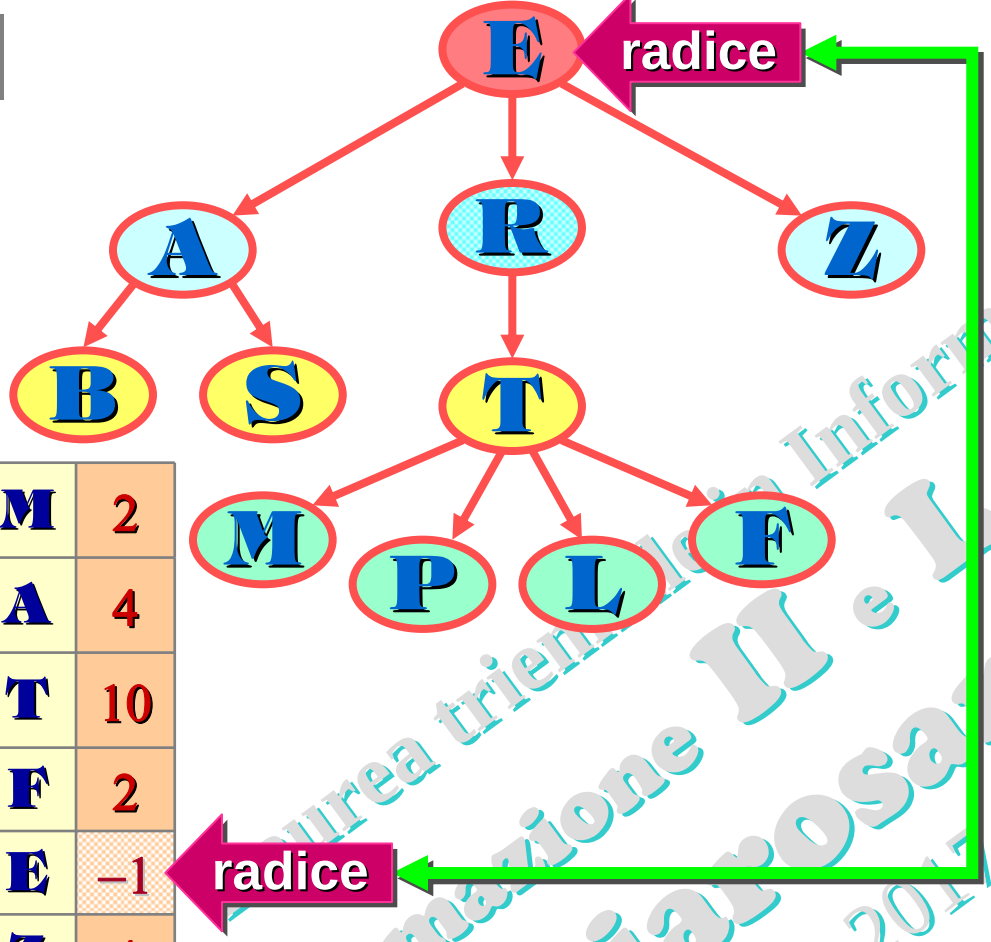
radice



nodo

informazione

grado | pt_1 | pt_2 | pt_3 | pt_4



0	M	2
1	A	4
2	T	10
3	F	2
4	E	–1
5	Z	4
6	L	2
7	S	1
8	B	1
9	P	2
10	R	4

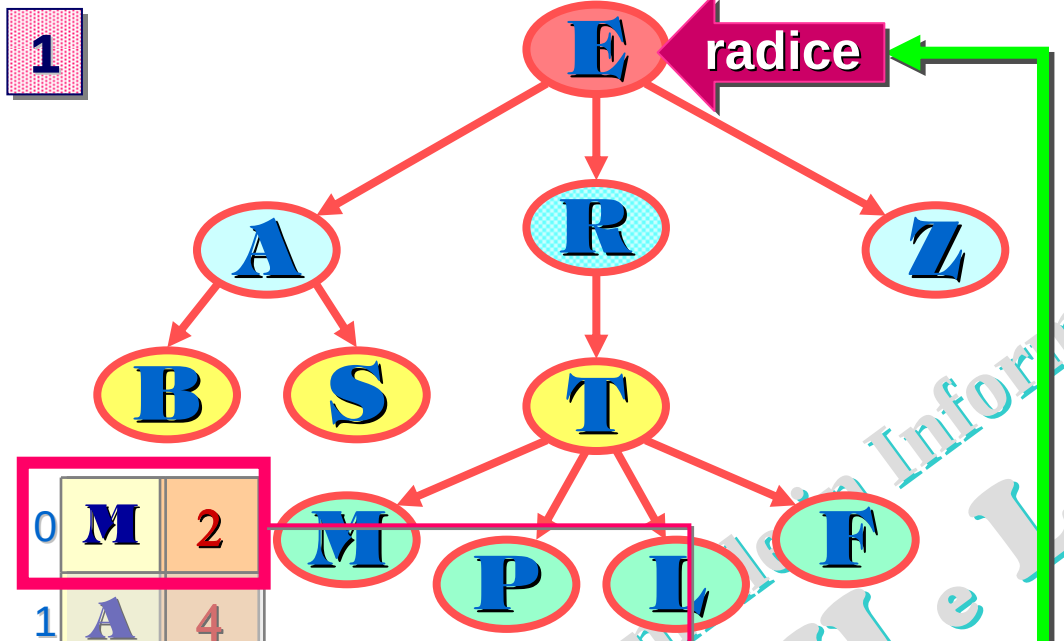
radice

nodo

informazione				
grado	pt_1	pt_2	pt_3	pt_4

	INF	GR	pt_1	pt_2	pt_3	pt_4
0		0				
1		0				
2		0				
3		0				
4		0				
5		0				
6		0				
7		0				
8		0				
9		0				
10		0				

situazione iniziale tabella



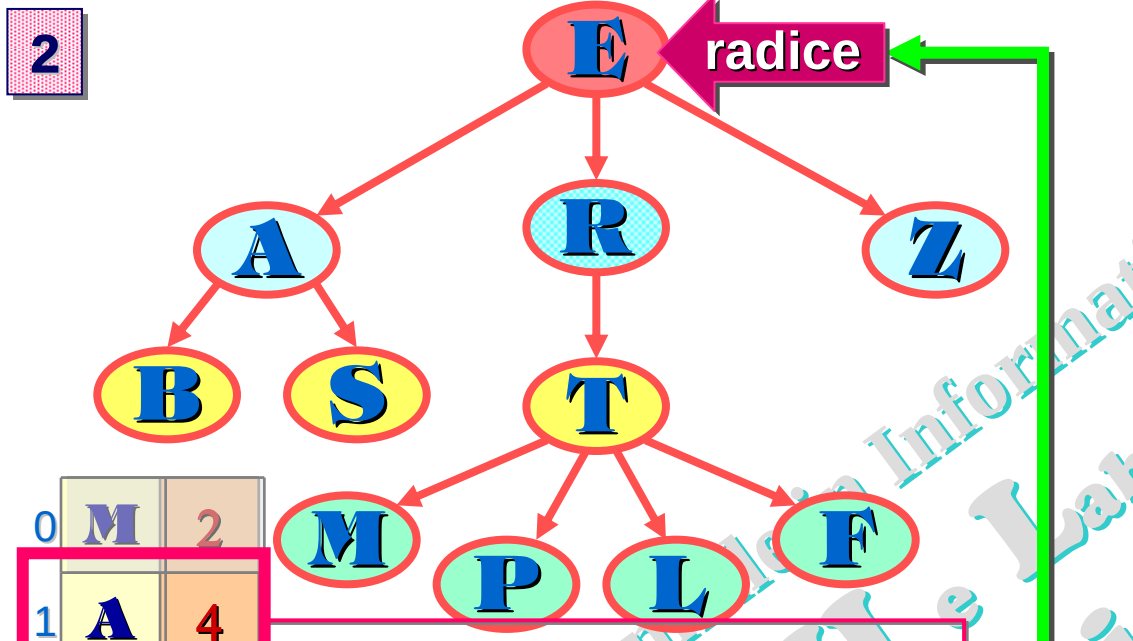
0	M	2
1	A	4
2	T	10
3	F	2
4	E	–1
5	Z	4
6	L	2
7	S	1
8	B	1
9	P	2
10	R	4

radice

nodo

informazione				
grado	pt_1	pt_2	pt_3	pt_4

	INF	GR	pt_1	pt_2	pt_3	pt_4
0	M	0				
1		0				
2		1	0			
3		0				
4		0				
5		0				
6		0				
7		0				
8		0				
9		0				
10		0				



0	M	2
1	A	4
2	T	10
3	F	2
4	E	–1
5	Z	4
6	L	2
7	S	1
8	B	1
9	P	2
10	R	4

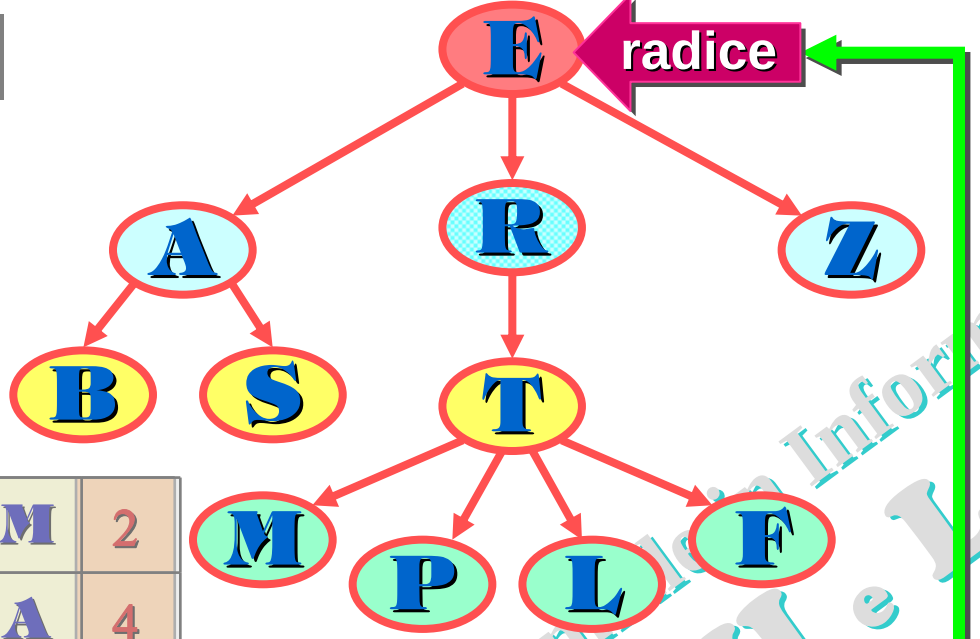
radice

nodo

informazione

grado	pt_1	pt_2	pt_3	pt_4

	INF	GR	pt_1	pt_2	pt_3	pt_4
0	M	0				
1	A	0				
2		1	0			
3		0				
4		1	1			
5		0				
6		0				
7		0				
8		0				
9		0				
10		0				



0	M	2
1	A	4
2	T	10
3	F	2
4	E	–1
5	Z	4
6	L	2
7	S	1
8	B	1
9	P	2
10	R	4

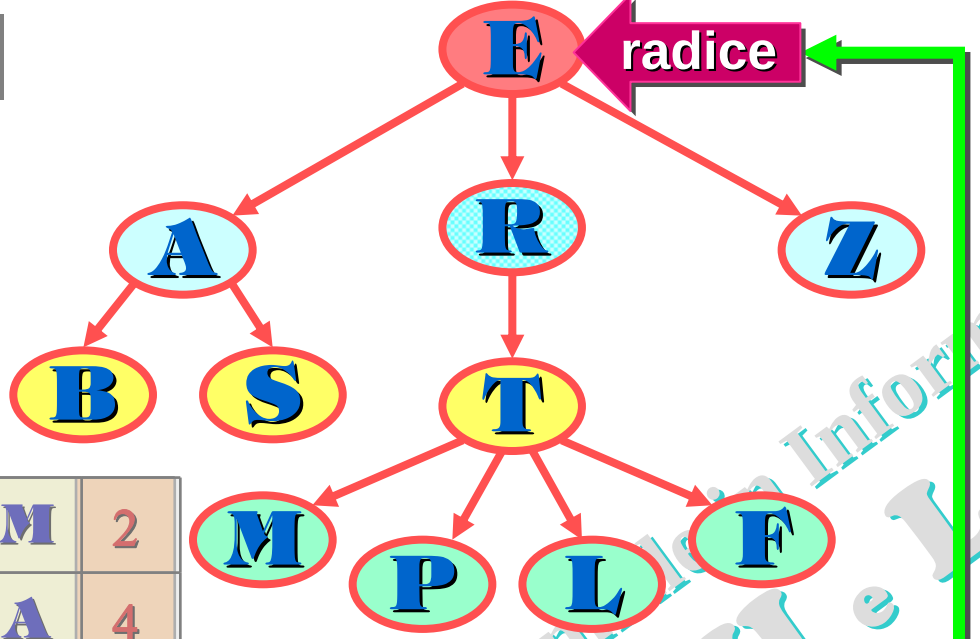
radice

nodo

informazione

grado	pt_1	pt_2	pt_3	pt_4

	INF	GR	pt_1	pt_2	pt_3	pt_4
0	M	0				
1	A	0				
2	T	~~1~~ 0				
3		0				
4		1	1			
5		0				
6		0				
7		0				
8		0				
9		0				
10		1	2			



0	M	2
1	A	4
2	T	10
3	F	2
4	E	–1
5	Z	4
6	L	2
7	S	1
8	B	1
9	P	2
10	R	4

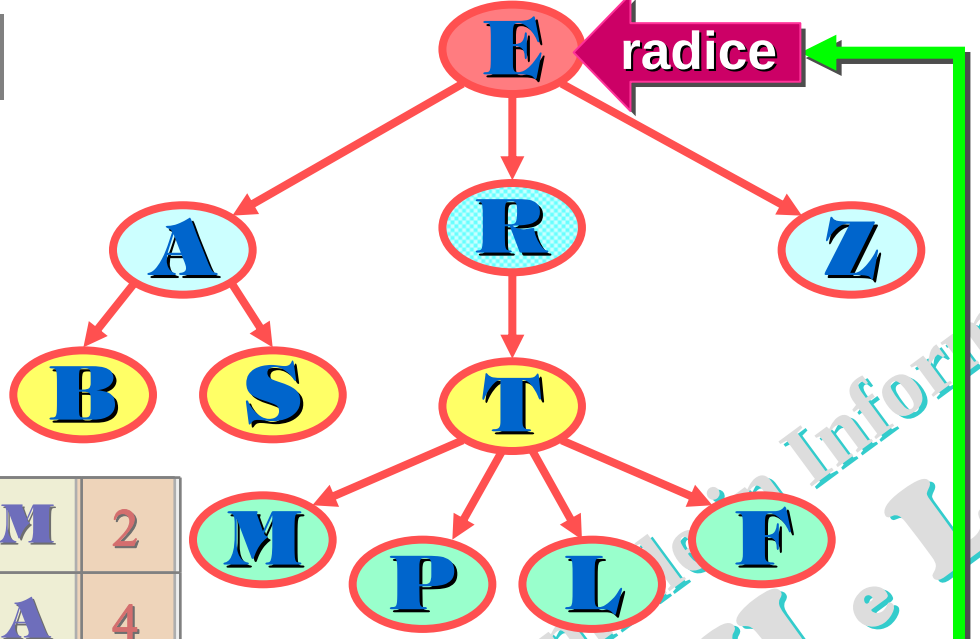
radice

nodo

informazione

grado	pt_1	pt_2	pt_3	pt_4

	INF	GR	pt_1	pt_2	pt_3	pt_4
0	M	0				
1	A	0				
2	T	2	0	3		
3	F	0				
4		1	1			
5		0				
6		0				
7		0				
8		0				
9		0				
10		1	2			



0	M	2
1	A	4
2	T	10
3	F	2
4	E	–1
5	Z	4
6	L	2
7	S	1
8	B	1
9	P	2
10	R	4

radice

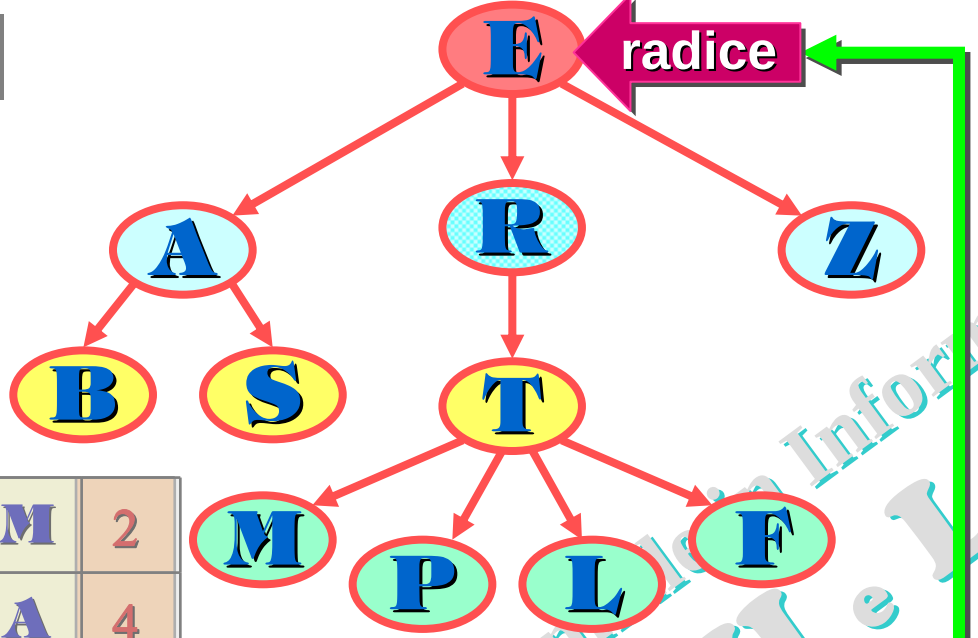
radice

nodo

informazione

grado	pt_1	pt_2	pt_3	pt_4

	INF	GR	pt_1	pt_2	pt_3	pt_4
0	M	0				
1	A	0				
2	T	2	0	3		
3	F	0				
4	E	1	1			
5		0				
6		0				
7		0				
8		0				
9		0				
10		1	2			



nodo

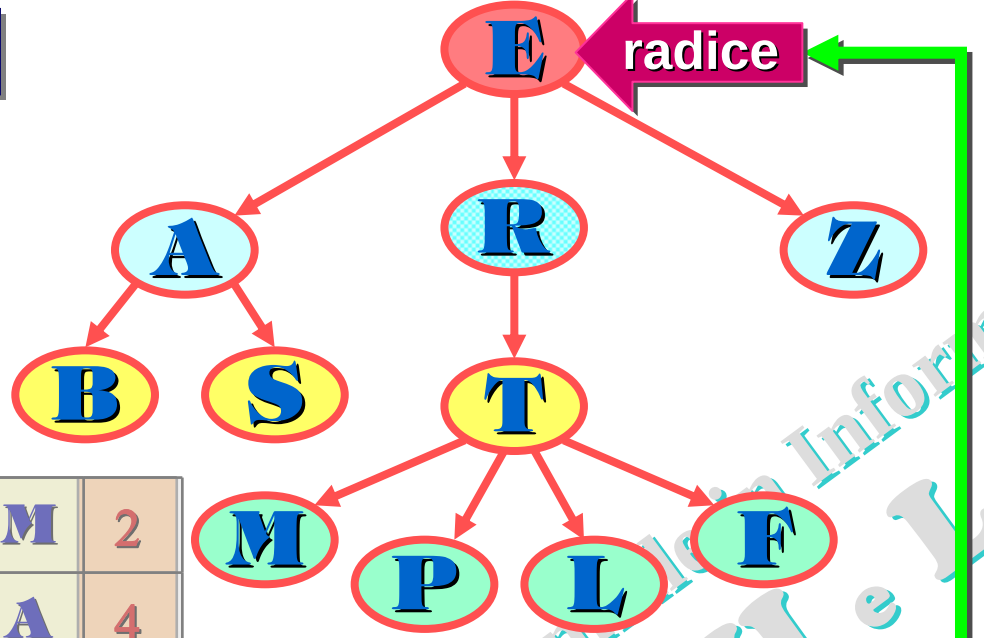
informazione				
grado	pt_1	pt_2	pt_3	pt_4

0	M	2
1	A	4
2	T	10
3	F	2
4	E	–1
5	Z	4
6	L	2
7	S	1
8	B	1
9	P	2
10	R	4

radice

radice

	INF	GR	pt_1	pt_2	pt_3	pt_4
0	M	0				
1	A	0				
2	T	2	0	3		
3	F	0				
4	E	2	1	5		
5	Z	0				
6		0				
7		0				
8		0				
9		0				
10		1	2			



0	M	2
1	A	4
2	T	10
3	F	2
4	E	–1
5	Z	4
6	L	2
7	S	1
8	B	1
9	P	2
10	R	4

radice

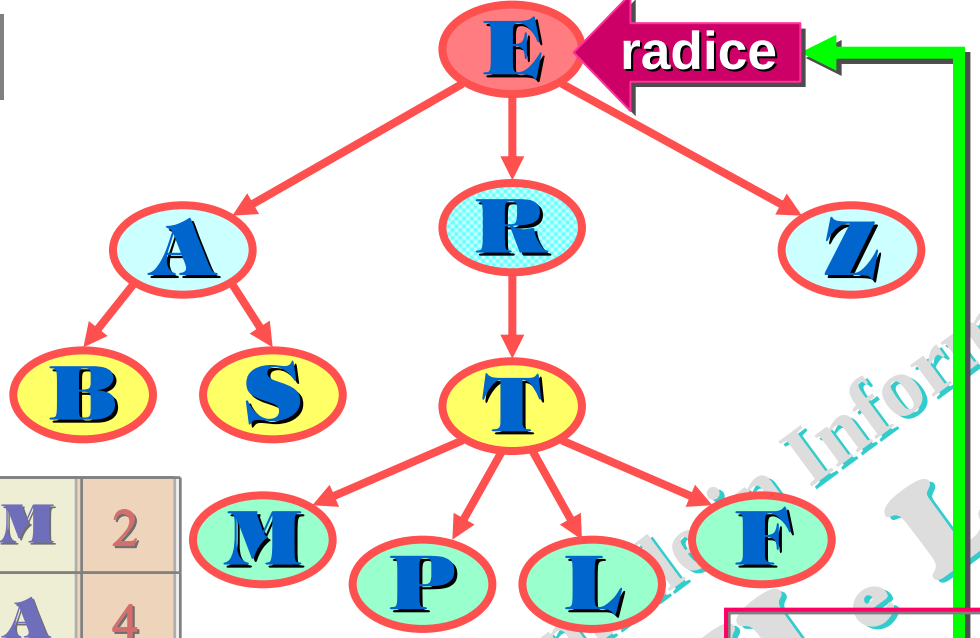
radice

nodo

informazione

grado	pt_1	pt_2	pt_3	pt_4

	INF	GR	pt_1	pt_2	pt_3	pt_4
0	M	0				
1	A	0				
2	T	3	0	3	6	
3	F	0				
4	E	2	1	5		
5	Z	0				
6	L	0				
7		0				
8		0				
9		0				
10		1	2			



0	M	2
1	A	4
2	T	10
3	F	2
4	E	–1
5	Z	4
6	L	2
7	S	1
8	B	1
9	P	2
10	R	4

radice

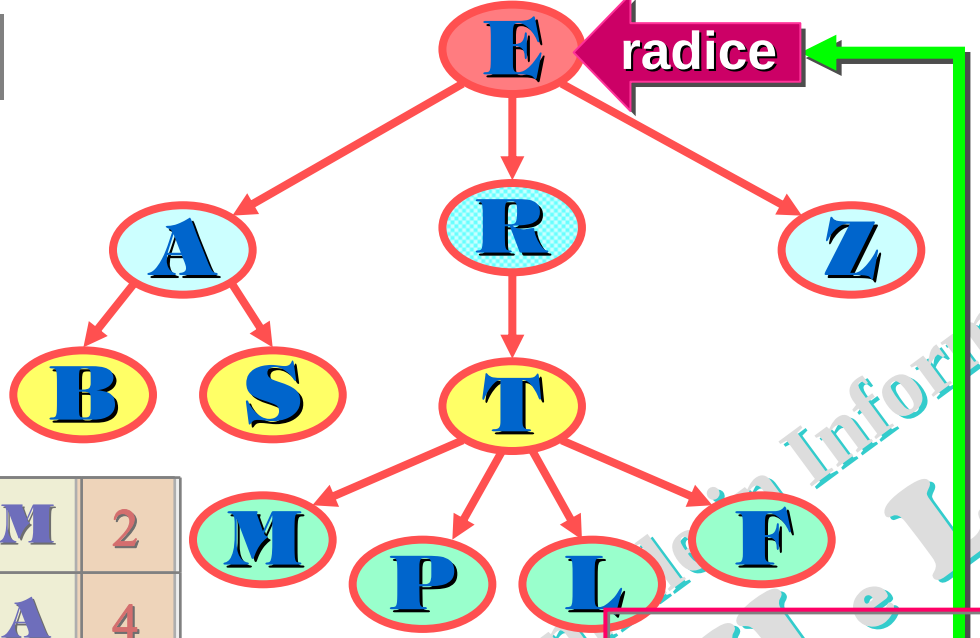
radice

nodo

informazione

grado	pt_1	pt_2	pt_3	pt_4

	INF	GR	pt_1	pt_2	pt_3	pt_4
0	M	0				
1	A	1	7			
2	T	3	0	3	6	
3	F	0				
4	E	2	1	5		
5	Z	0				
6	L	0				
7	S	0				
8		0				
9		0				
10		1	2			



0	M	2
1	A	4
2	T	10
3	F	2
4	E	–1
5	Z	4
6	L	2
7	S	1
8	B	1
9	P	2
10	R	4

radice

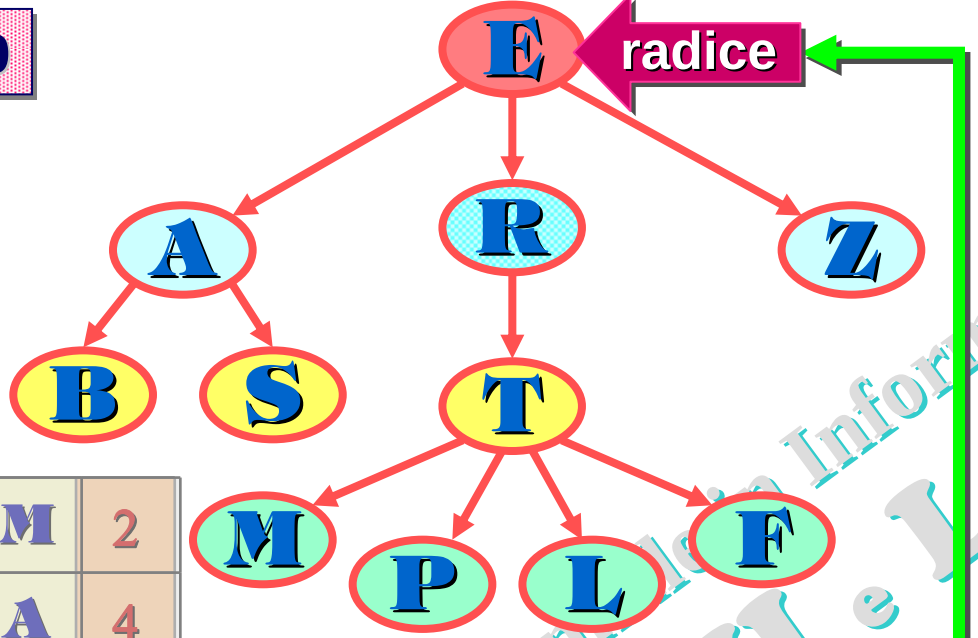
radice

nodo

informazione

grado	pt_1	pt_2	pt_3	pt_4

	INF	GR	pt_1	pt_2	pt_3	pt_4
0	M	0				
1	A	2	7	8		
2	T	3	0	3	6	
3	F	0				
4	E	2	1	5		
5	Z	0				
6	L	0				
7	S	0				
8	B	0				
9		0				
10		1	2			



0	M	2
1	A	4
2	T	10
3	F	2
4	E	–1
5	Z	4
6	L	2
7	S	1
8	B	1
9	P	2
10	R	4

radice

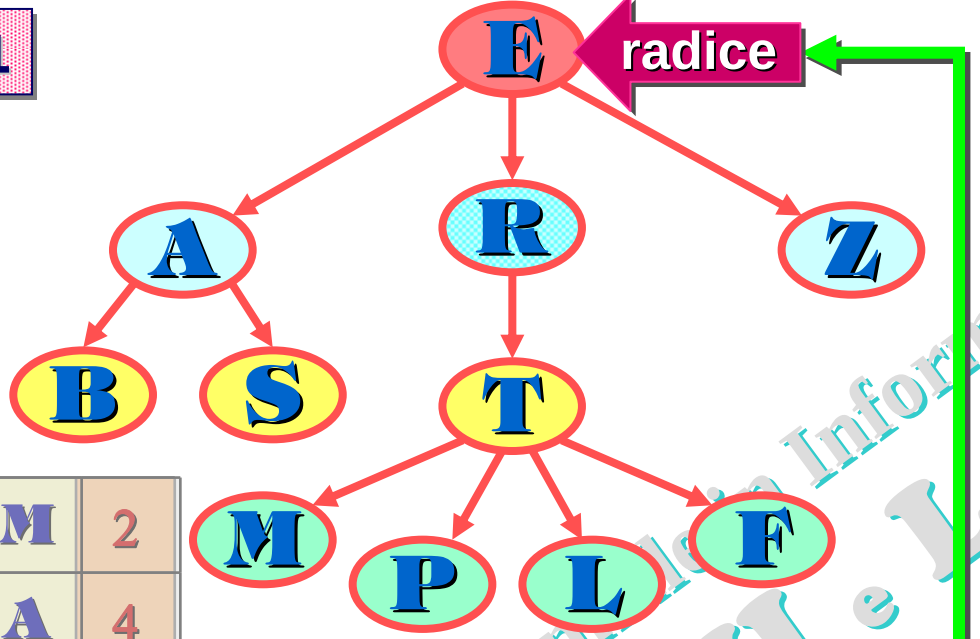
radice

nodo

informazione

grado	pt_1	pt_2	pt_3	pt_4

	INF	GR	pt_1	pt_2	pt_3	pt_4
0	M	0				
1	A	2	7	8		
2	T	4	0	3	6	9
3	F	0				
4	E	2	1	5		
5	Z	0				
6	L	0				
7	S	0				
8	B	0				
9	P	0				
10		1	2			



nodo

informazione				
grado	pt_1	pt_2	pt_3	pt_4

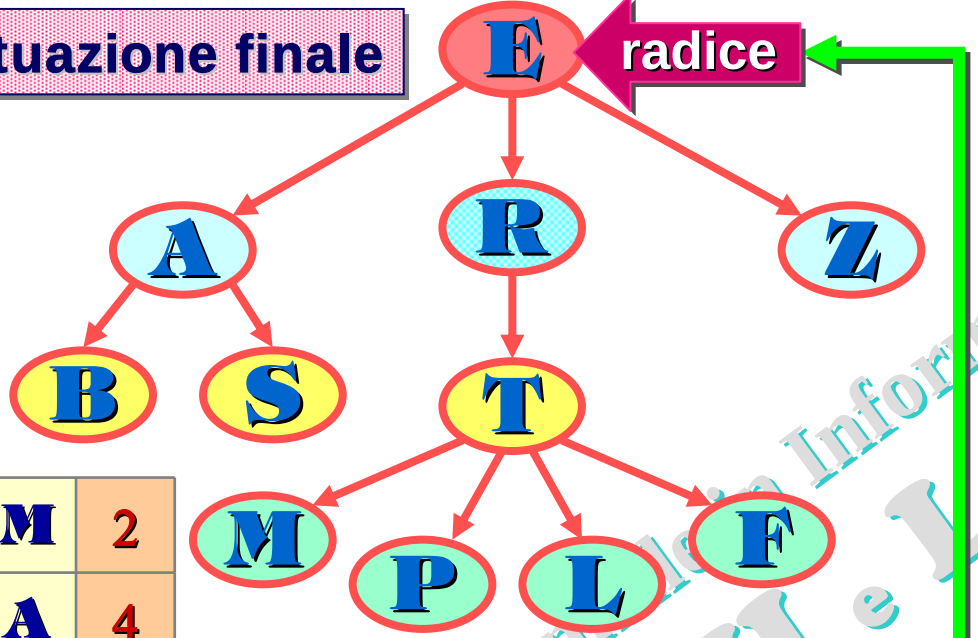
	INF	GR	pt_1	pt_2	pt_3	pt_4
0	M	0				
1	A	2	7	8		
2	T	4	0	3	6	9
3	F	0				
4	E	3	1	5	10	
5	Z	0				
6	L	0				
7	S	0				
8	B	0				
9	P	0				
10	R	–1	2			

0	M	2
1	A	4
2	T	10
3	F	2
4	E	–1
5	Z	4
6	L	2
7	S	1
8	B	1
9	P	2
10	R	4

radice

radice

situazione finale



0	M	2
1	A	4
2	T	10
3	F	2
4	E	–1
5	Z	4
6	L	2
7	S	1
8	B	1
9	P	2
10	R	4

radice

radice

nodo

informazione

grado	pt_1	pt_2	pt_3	pt_4

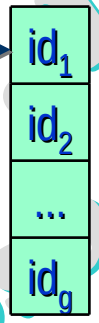
	INF	GR	pt_1	pt_2	pt_3	pt_4
0	M	0				
1	A	2	7	8		
2	T	4	0	3	6	9
3	F	0				
4	E	3	1	5	10	
5	Z	0				
6	L	0				
7	S	0				
8	B	0				
9	P	0				
10	R	1	2			

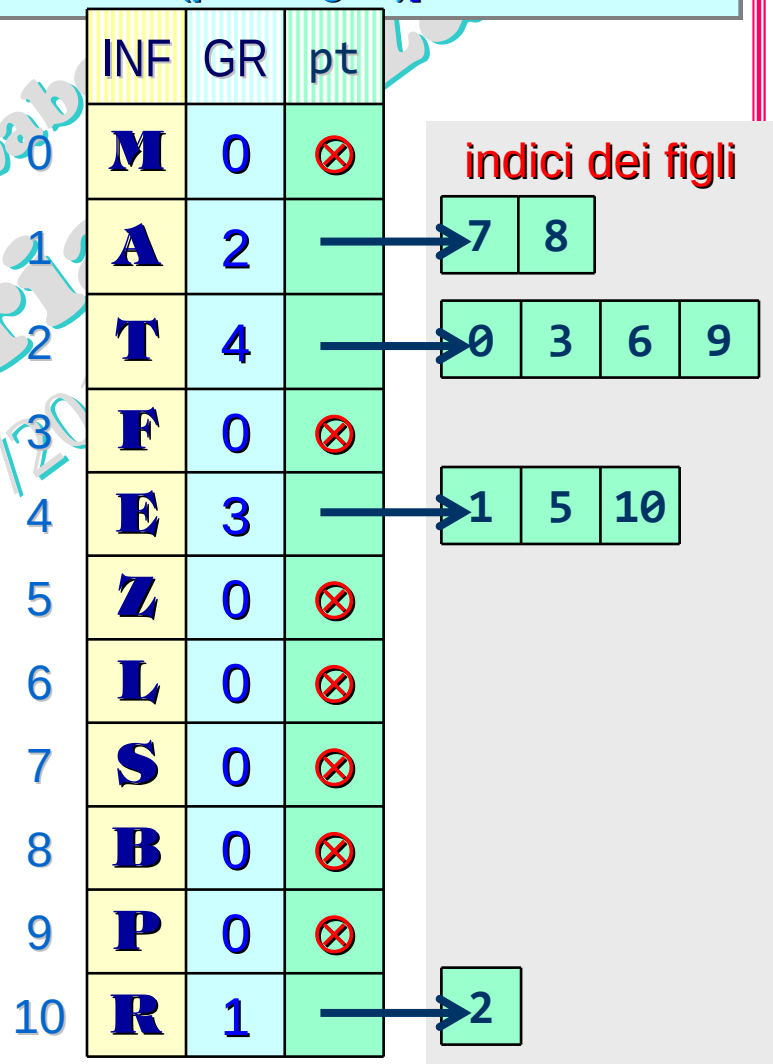
Spreca spazio!

Esercizio:

2 Scrivere *function C* per la costruzione e visita per livelli di un albero. [liv. 2]

[suggerimento: gli indici dei figli di un nodo risiedono in un array dinamicamente allocato e indirizzato dall'unico campo puntatore del nodo dell'albero (`pt_figli`)]





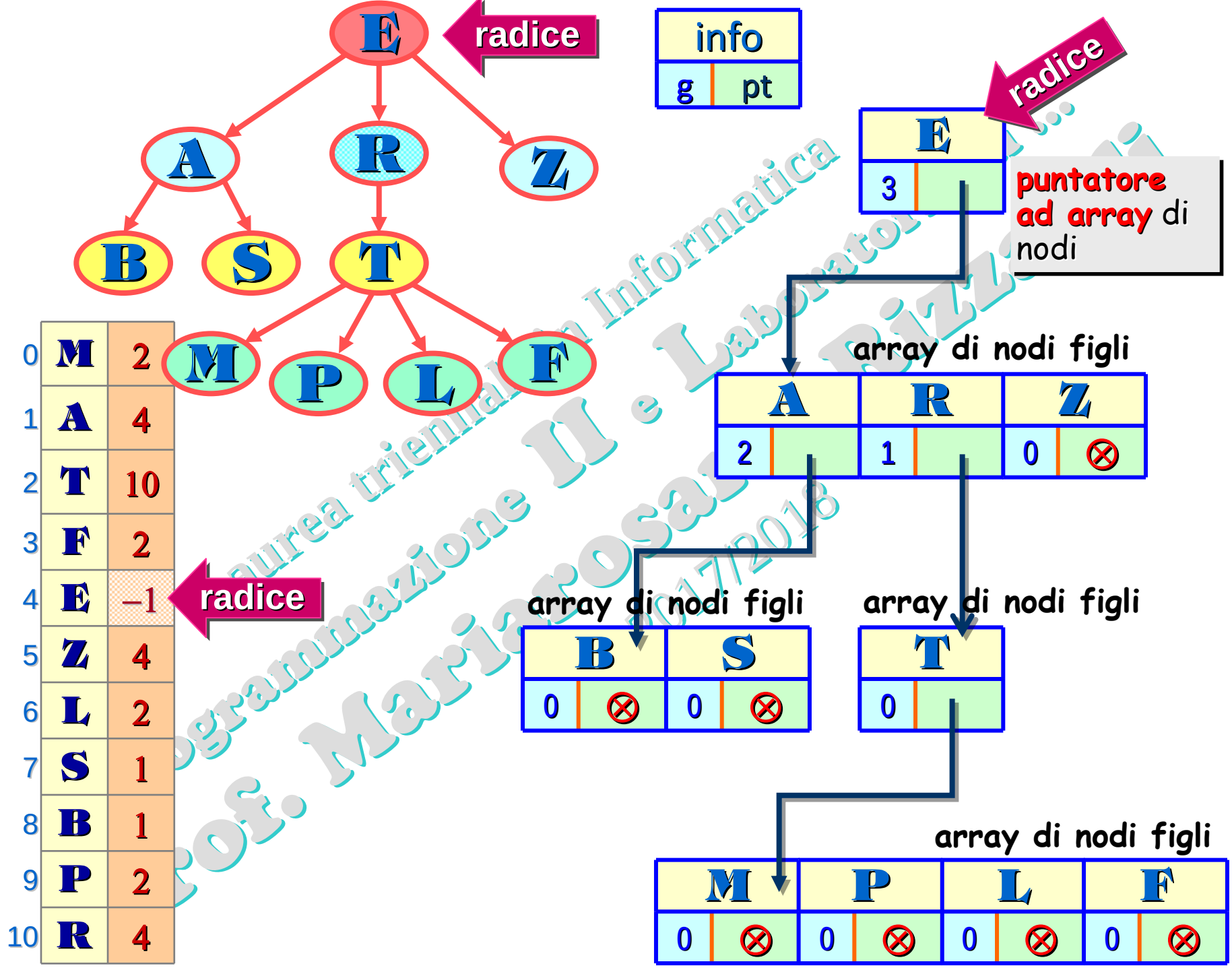
Risparmia spazio!

Esercizio:

3 Scrivere *function C* per la costruzione e visita per livelli di un albero (mediante liste multiple). [liv. 3]

[suggerimento: i puntatori ai figli di un nodo risiedono in un array dinamicamente allocato e indirizzato dall'unico campo puntatore del nodo dell'albero (`pt`)]

help


E
radice
A
R
Z
B
S
T
M
P
L
F
info
g
pt
radice
E
3
puntatore ad array di nodi
array di nodi figli
A
R
Z
2
1
0
array di nodi figli
B
S
0
0
array di nodi figli
T
0
array di nodi figli
M
P
L
F
0
0
0
0
0 M 2
1 A 4
2 T 10
3 F 2
4 E –1
5 Z 4
6 L 2
7 S 1
8 B 1
9 P 2
10 R 4
radice
Laurea triennale in Informatica
Programmazione II e Laboratorio
Prof. Mariarosaria Rizzardi
2017/2018

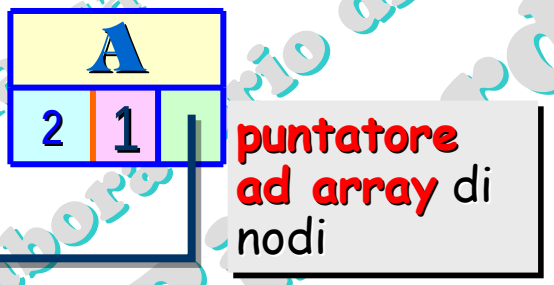
Organizzazione dei dati

```
typedef struct {
     char info;
     int padre;
} DATI;
```

array di dati

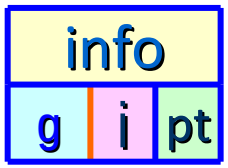
	input	
0	M	2
1	A	4
2	T	10
3	F	2
4	E	–1
5	Z	4
6	L	2
7	S	1
8	B	1
9	P	2
10	R	4

radice



array di nodi figli

struttura di un nodo




```
struct nodo {
   char info;
   int g;       /* grado */
   int indice; /* indice del nodo
                  nell'array input */
   struct nodo *pt;
};
typedef struct nodo NODO;
```

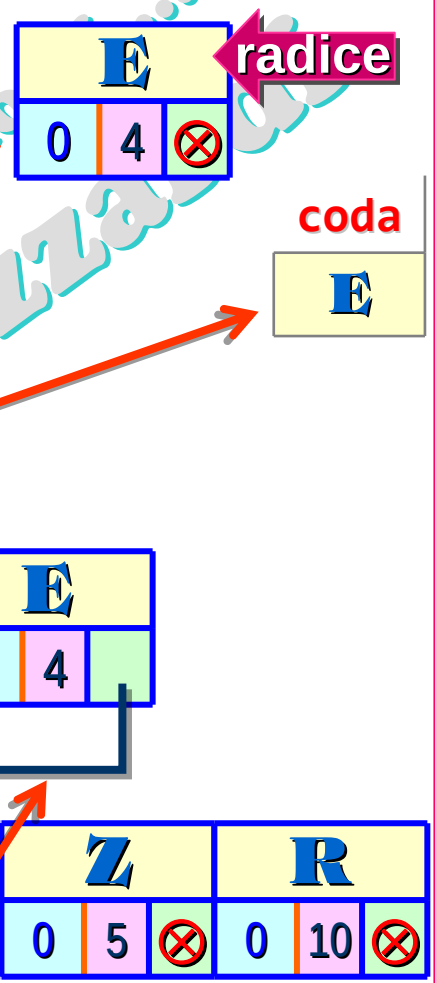
Costruzione dell'albero per livelli

	input	
0	M	2
1	A	4
2	T	10
3	F	2
4	E	–1
5	Z	4
6	L	2
7	S	1
8	B	1
9	P	2
10	R	4

radice

algoritmo

```
Cerca la radice in input
Costruisce nodo per la radice
Inserisce nodo radice nella coda
while coda non vuota
{
    Estrae nodo da coda
    Cerca i figli di nodo nell'array input
    if (esistono figli)
    {
        Aggiorna grado del nodo
        Crea array di nodi figli
        Collega l'array di nodi figli al padre
        Inserisce i nodi figli nella coda
    }
}
```



coda: A, Z, R