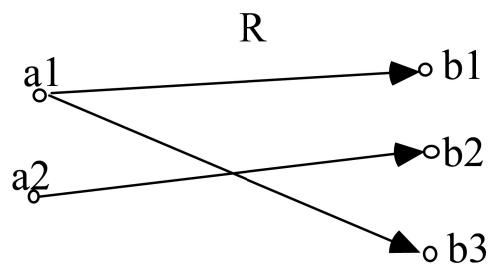


# GRAFI

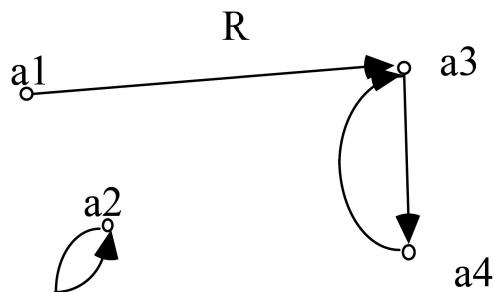
Sono strutture dati che rappresentano **relazioni binarie** su un insieme di elementi.

**Relazione** tra insiemi A, B:  $R \subseteq A \times B$



Una relazione su  $A=\{a_1, a_2\}$  e  $B=\{b_1, b_2, b_3\}$  definisce un **grafo** su A e B.

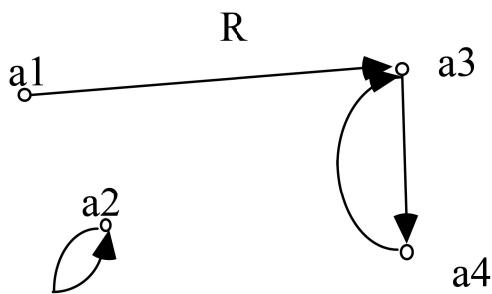
Se  $A=B$ ,  $R \subseteq A \times A$ :



Un **grafo orientato** è una struttura  $G = \langle A, R \rangle$  dove:

- A è un insieme, non vuoto, di **nodi**;
- R è un insieme di **archi orientati** tali che se  $\langle a_i, a_j \rangle \in R$  c'è un arco orientato diretto da  $a_i$  verso  $a_j$ .

# GRAFI (II)



## NOMENCLATURA

### Grado di ingresso di $a_i$

E' il numero di archi che hanno  $a_i$  come nodo finale.

### Grado di uscita di $a_i$

E' il numero di archi che hanno  $a_i$  come nodo iniziale.

### Grafo completo

Se esiste un arco che collega due elementi qualunque di A, ossia se  $\forall a_i, a_j \in A$  esiste un arco da  $a_i$  ad  $a_j$ .

Questo significa che la relazione R coincide con  $A \times A$ , ossia è una **relazione totale**.

Il grafo in figura non è un grafo completo.

I grafi sono strutture dati usate (in informatica) per rappresentare relazioni.

Le proprietà delle relazioni si riflettono nella struttura del grafo.

# GRAFI (III)

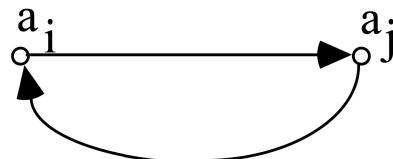
## PROPRIETÀ DELLE RELAZIONI

Una relazione  $R \subseteq A \times A$  è:

- **riflessiva** se  $\forall a \in A, \langle a, a \rangle \in R$

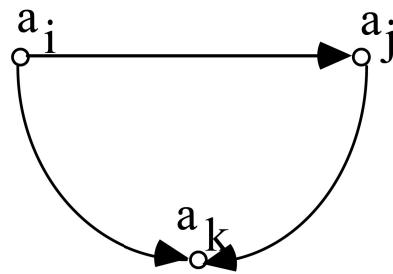


- **simmetrica**, se  $\langle a_i, a_j \rangle \in R \Rightarrow \langle a_j, a_i \rangle \in R$   
**(antisimmetrica)**, se  $\langle a_i, a_j \rangle \in R \Rightarrow \langle a_j, a_i \rangle \notin R$



NB: un grafo simmetrico implica grafo non orientato.

- **transitiva**, se  $\langle a_i, a_j \rangle \in R$  e  $\langle a_j, a_k \rangle \in R$  implica  $\langle a_i, a_k \rangle \in R$

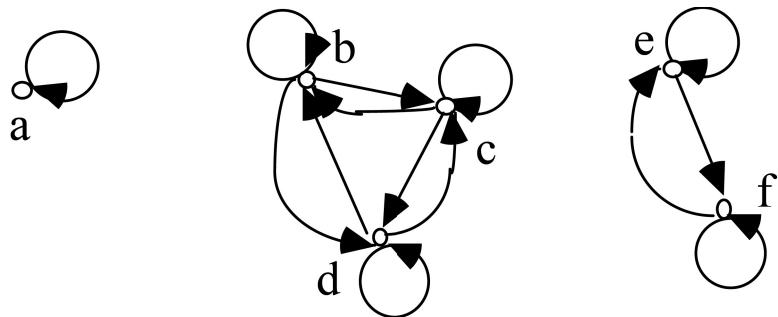


- **di equivalenza**, se è riflessiva, simmetrica, transitiva
- **d'ordine**, se è riflessiva, antisimmetrica, transitiva

## GRAFI (IV)

### ESEMPI

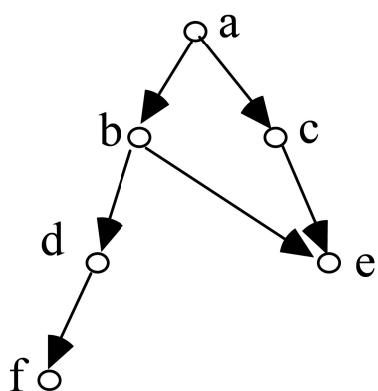
Grafo di una **relazione di equivalenza**:



Ogni “isola” rappresenta una *classe di equivalenza*.

Di solito, il primo non viene disegnato.

Grafo di una **relazione d’ordine (semplificato)**:



# GRAFI (V)

## CAMMINO IN UN GRAFO

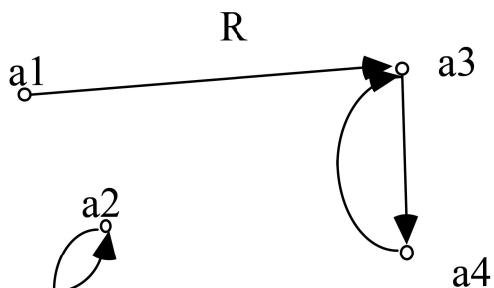
In un grafo  $G = \langle A, R \rangle$ , dati due nodi  $\alpha, \beta \in A$ , si dice **cammino** da  $\alpha$  a  $\beta$  una sequenza di nodi:

$$a_0, a_1, a_2, \dots, a_n$$

tali che:

- $n \geq 0$  (lunghezza del cammino)
- $a_0 = \alpha$
- $a_n = \beta$
- $\forall i \in [0..n-1], \langle a_i, a_{i+1} \rangle \in R$

## ESEMPIO



Cammino da  $a_1$  ad  $a_4$ :       $a_1, a_3, a_4$   
(ma anche:                           $a_1, a_3, a_4, a_3, a_4$ )

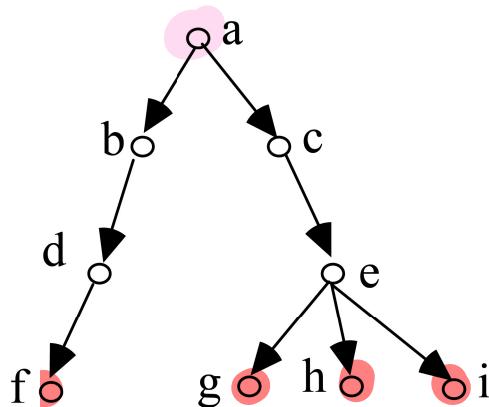
Un cammino può essere:

- **semplice**, se i nodi del cammino sono distinti
- **ciclico**, se  $\alpha = \beta$  (in tal caso il cammino è detto *ciclo*)
- **aciclico**, se non ha cicli.

# ALBERI

Un albero (finito) è un **grafo orientato aciclico** tale che:

- esiste un nodo (**radice**) con grado di ingresso 0
- ogni altro nodo ha grado di ingresso 1.



I nodi con grado di uscita 0 sono detti **foglie**.

L' **altezza** di un albero è la lunghezza del cammino più lungo dalla radice ad una foglia.

Se un arco collega il nodo  $\alpha$  al nodo  $\beta$ ,

- $\alpha$  si dice **nodo padre** di  $\beta$
- $\beta$  si dice **nodo figlio** di  $\alpha$ .

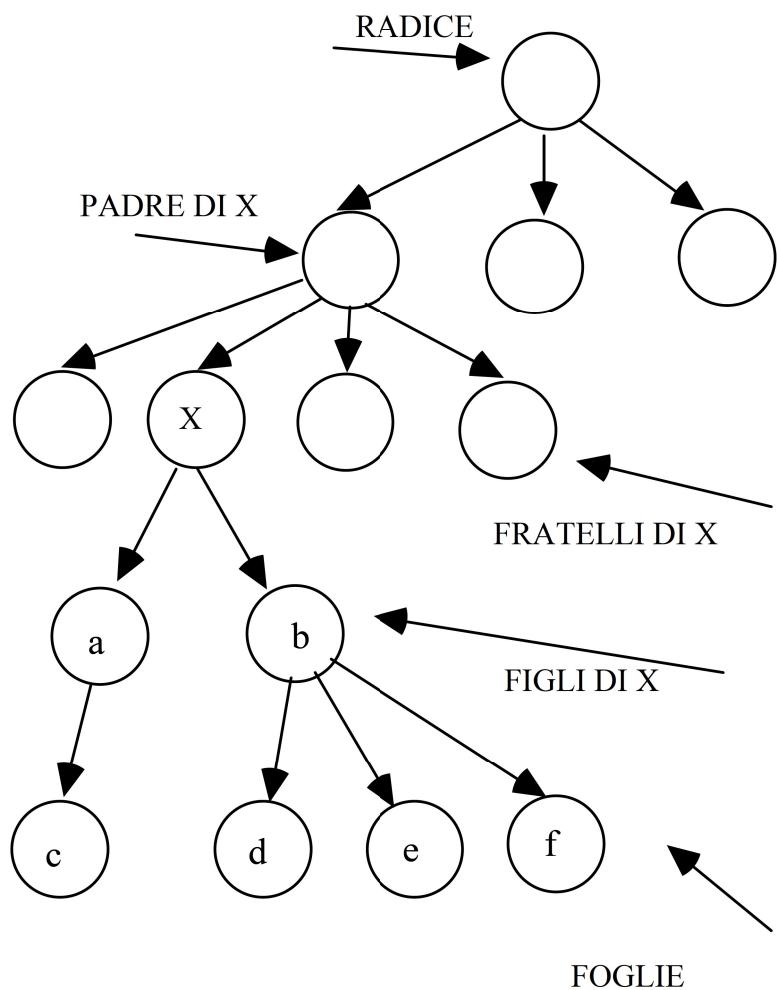
Dalla definizione segue che:

- tutti i nodi, tranne la radice, hanno esattamente un padre
- un padre può avere zero o più figli
- esiste esattamente un cammino (semplice) dalla radice a qualunque altro nodo.

## ALBERI (II)

Un albero rappresenta una *relazione d'ordine parziale*.

Gli alberi si usano per rappresentare *relazioni gerarchiche* fra oggetti.



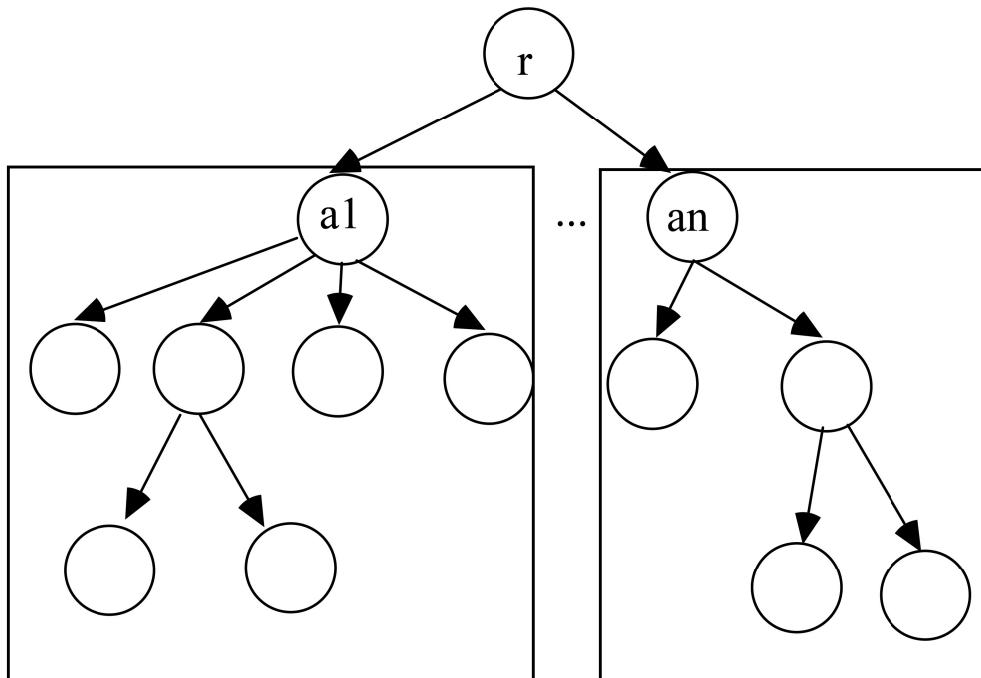
I **discendenti** di un certo nodo sono calcolati come *chiusura transitiva* della relazione “figlio di”.

$$\text{discendenti}(X) = \{a, b, c, d, e, f\}$$

# ALBERI COME STRUTTURE DATI RICORSIVE

Un albero  $T$  è un grafo tale che:

- esiste un nodo  $r$  (radice) con  $n$  successori ( $n \geq 0$ ),  $a_1, \dots, a_n$
- tutti i nodi di  $T$  (escluso  $r$ ) possono essere ripartiti in  $n$  insiemi disgiunti, ciascuno dei quali individua un albero  $T_1, \dots, T_n$  con radice, rispettivamente,  $a_1, \dots, a_n$ .

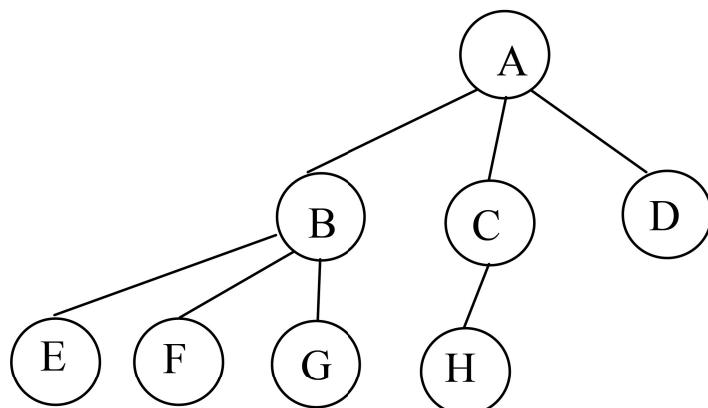


Se si impone un ordine tra i figli di ogni nodo, l'albero si dice **ordinato**.

## ALBERI N-ARI

Ciascun nodo può avere  $n$  figli (con  $n$  finito, ma non limitato)

Consideriamo il caso di alberi *ordinati* (nel senso che esiste un ordinamento fra i sottoalberi figli)



## VISITE DI ALBERI N-ARI

Visitare un albero significa percorrerlo in modo da passare una e una sola volta in ogni nodo.

Visitare un albero è necessario ogni qual volta si debbano effettuare operazioni sui suoi elementi:

- visualizzare / stampare tutti gli elementi
- cercare un elemento in un albero
- ...

Naturalmente, nel caso di una ricerca, una visita può essere interrotta anche prima del suo termine naturale, non appena l'informazione cercata venga trovata.

## VISITE DI ALBERI N-ARI

La visita di un albero  $n$ -ario (*non vuoto*) può essere fatta secondo diverse modalità:

- **visita in profondità in ordine anticipato**

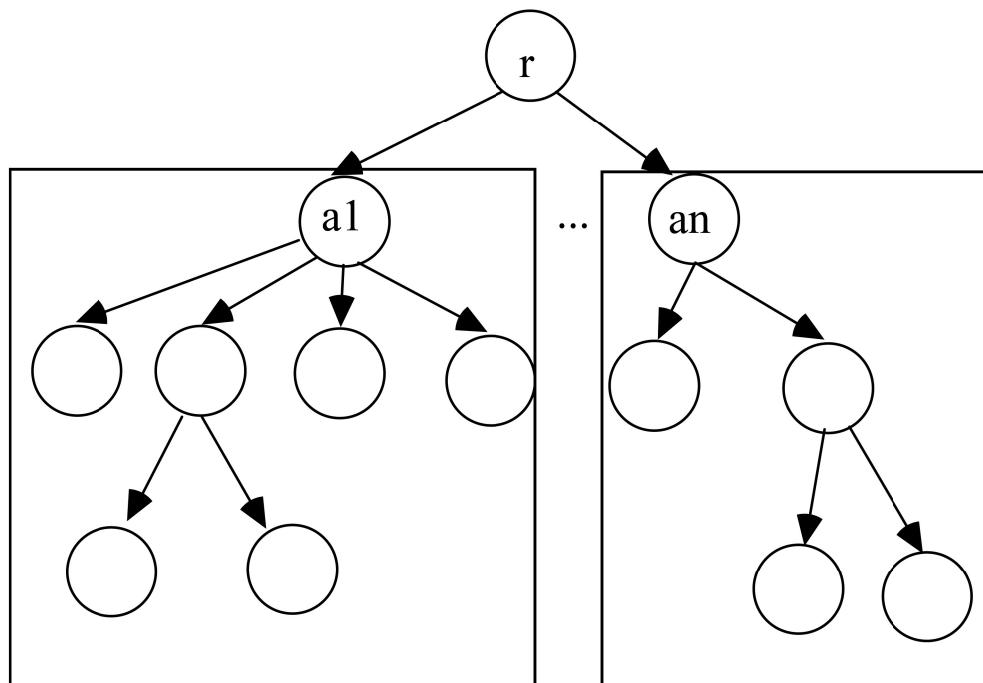
prima la radice,  
poi tutti i sottoalberi (in ordine)

- **visita in profondità in ordine posticipato**

prima tutti i sottoalberi (in ordine),  
poi la radice

- **visita in ampiezza**

prima la radice (livello 0),  
poi tutti i discendenti di livello 1,  
poi tutti i discendenti di livello 2,  
poi quelli di livello 3, etc.

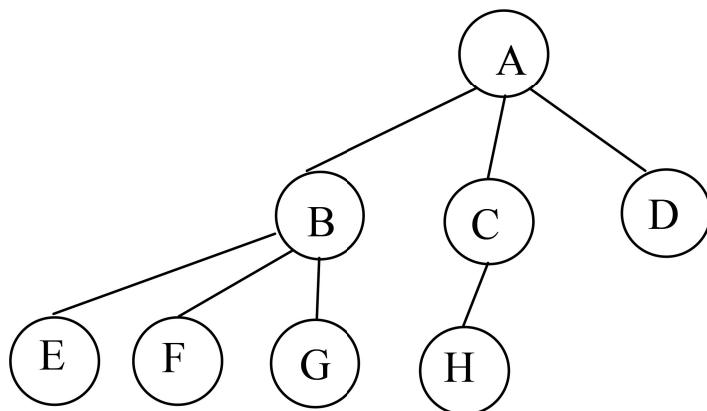


# VISITA IN ORDINE ANTICIPATO

## SPECIFICA

```
void visitaInPreOrdine(tree T) {  
    if (!empty(T)) {  
        <analizza radice di T>  
        for (i=0; i<n; i++)  
            /* per ogni sottoalbero Ti */  
            visitaInPreOrdine(Ti);  
    }  
}
```

## ESEMPIO



### Visita in ordine anticipato:

A (sottoalbero B) (sottoalbero C) (sottoalbero D)

A (B (E F G)) (sottoalbero C) (sottoalbero D)

A (B (E F G)) (C (H)) (D)

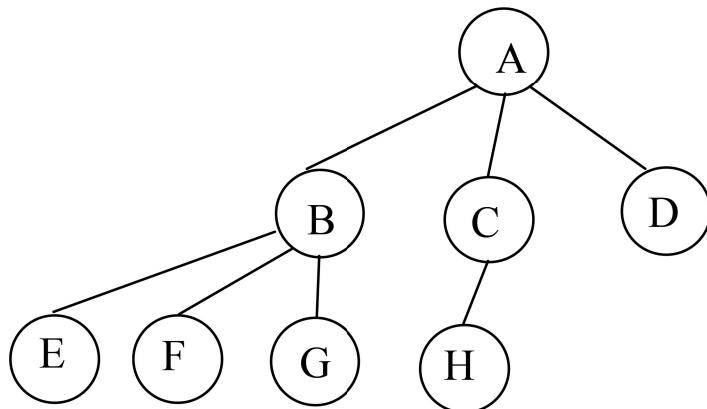
**A B E F G C H D**

# VISITA IN ORDINE POSTICIPATO

## SPECIFICA

```
void visitaInPostOrdine(tree T) {  
    if (!empty(T)) {  
        for (i=0; i<n; i++)  
            /* per ogni sottoalbero  $T_i$  */  
            visitaInPostOrdine( $T_i$ );  
        <analizza radice di  $T$ >  
    }  
}
```

## ESEMPIO



### Visita in ordine posticipato:

(sottoalbero B) (sottoalbero C) (sottoalbero D) A

((E F G) B) (sottoalbero C) (sottoalbero D) A

((E F G) B) ((H) C) (D) A

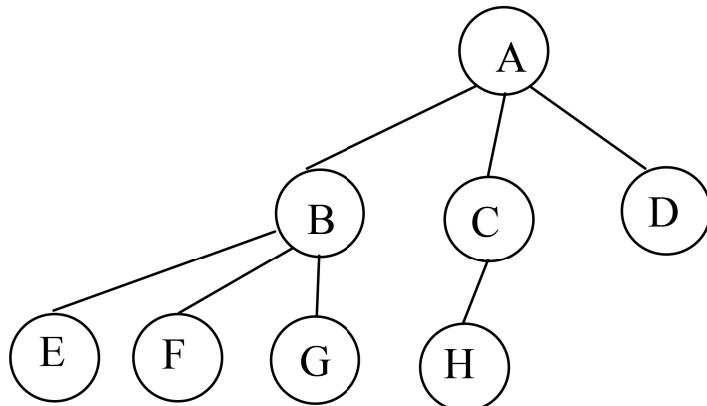
**E F G B H C D A**

# VISITA IN AMPIEZZA

## SPECIFICA

```
void visitaInAmpiezza(tree T) {  
    if (!empty(T)) {  
        <analizza radice di T>  
        visitaDiscendenti(T);  
    }  
  
void visitaDiscendenti(tree T) {  
    if (!empty(T)) {  
        for (i=0; i<n; i++)  
            <analizza radice di Ti>  
        for (i=0; i<n; i++)  
            visitaDiscendenti(Ti);  
    }  
}
```

## ESEMPIO



### Visita in ampiezza (per livelli):

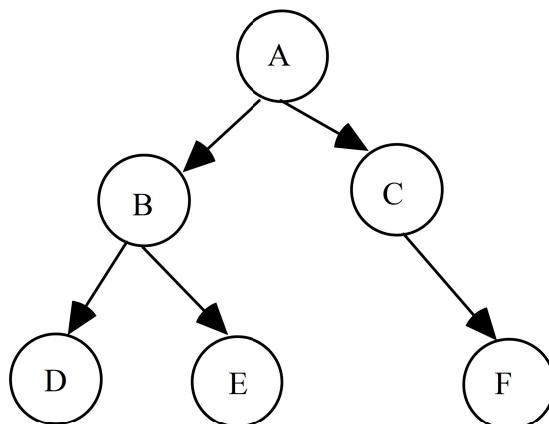
A (livello 1) (livello 2)

A (B C D) (E F G H)

A B C D E F G H

# ALBERI BINARI

Albero ordinato in cui ogni nodo ha *al massimo* due figli (**figlio sinistro** e **figlio destro**).



## Definizione induttiva

Un albero binario è un grafo orientato aciclico che

- è vuoto (cioè ha un insieme vuoto di nodi)
- oppure è formato da un nodo radice e da due sottoalberi binari (sinistro e destro).

## Visita di un albero binario

In aggiunta ai normali modi di visita, un albero binario consente una ulteriore modalità: la **visita in ordine**

prima il sottoalbero di sinistra (in ordine)  
poi la radice,  
infine il sottoalbero di destra (in ordine).

## ALBERI BINARI: PROPRIETÀ

1) Il numero di nodi di livello  $k$  ( $k \geq 0$ ) di un albero binario è al massimo  $2^k$

Si può dimostrare per *induzione*:

- per  $k=0$ , c'è un solo nodo (radice)  $\rightarrow 2^0$
- supponendo che al livello  $k$  il numero massimo di nodi sia  $2^k$ , al livello  $k+1$  saranno al più  $2^{k+1}$ , in quanto ogni nodo ha al massimo 2 figli.

2) In un albero binario di altezza  $p$ , il numero massimo di nodi è  $2^{p+1}-1$

Si dimostra semplicemente osservando che, se il numero massimo di nodi al livello  $k$  è  $2^k$ , con  $p$  livelli si hanno al massimo:

$$\sum_{(k=0..p)} 2^k = 2^{p+1} - 1$$

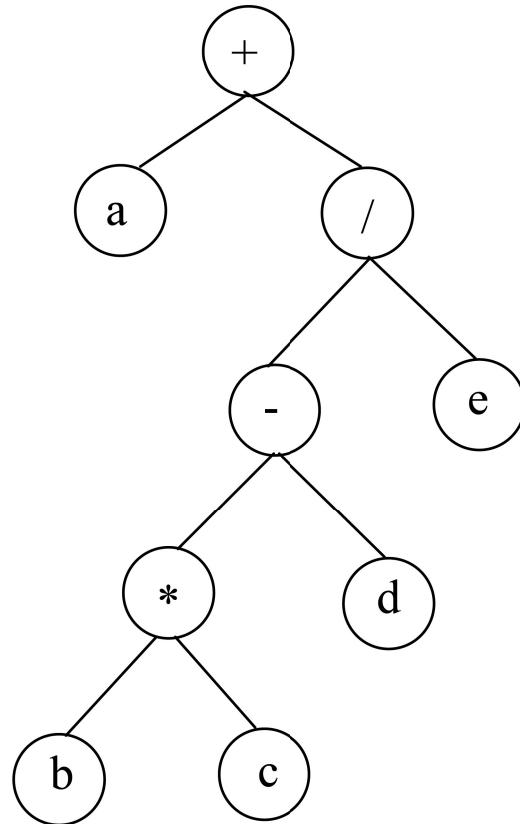
Conseguenza:

un albero binario di altezza  $p$  è **completo** se ha esattamente  $2^{p+1} - 1$  nodi.

In un albero binario completo ciascun nodo (tranne le foglie) ha grado di uscita pari a 2.

# ALBERI BINARI: ESEMPIO DI VISITE

Come esempio significativo, consideriamo l'albero sintattico corrispondente all'espressione aritmetica  $a + (b * c - d) / e$



**Visita in ordine anticipato:** + a / - \* b c d e

→ **notazione polacca prefissa**

(prima l'operatore, poi i due operandi)

**Visita in ordine anticipato:** a b c \* d - e / +

→ **notazione polacca postfissa**

(prima i due operandi, poi l'operatore)

**Visita in ordine:** a + (b \* c - d) / e

→ **notazione infissa**

(prima un operando, poi l'operatore, poi l'altro operando)

(richiede parentesi per priorità / associatività operatori)