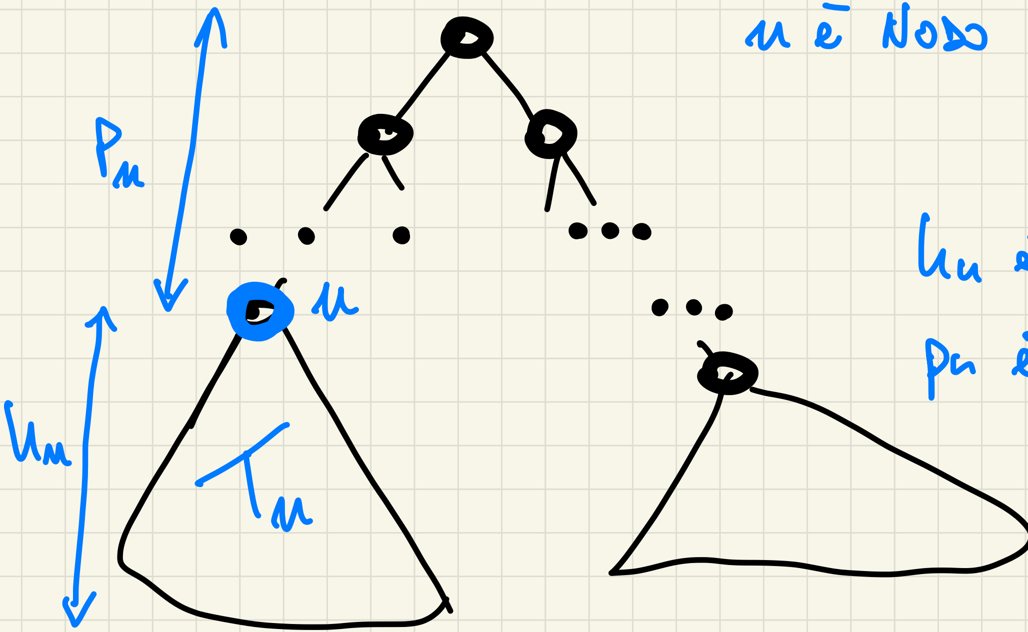
Programmazione e Algoritmica

26/2/2025

NODI CARDINE



u è NODO CARDINE $\iff p_u = h_u$

h_u è l'altezza di T_u

p_u è la profondità di u

PROBLEMA:

Dato un albero binario, stampare le chiavi di tutti i suoi nodi cardine

CARDINE(u, p) — u: nodo corrente (inizialmente è la radice); p: la sua profondità

```
CARDINE(u, p)
IF (u == nil) THEN RETURN -1;
ELSE
{ altsx = CARDINE(u.left, p+1);
  altdx = CARDINE(u.right, p+1);
altezza = 1 + max{altsx, altdx};
IF (p == altezza) THEN printf(u.key);
RETURN altezza;
```

CASO BASE $\Theta(1)$

CALL RICORSIVE

$\Theta(1)$

CALL INIZIALE CARDINE(radice, 0)

CARDINE(u, p)

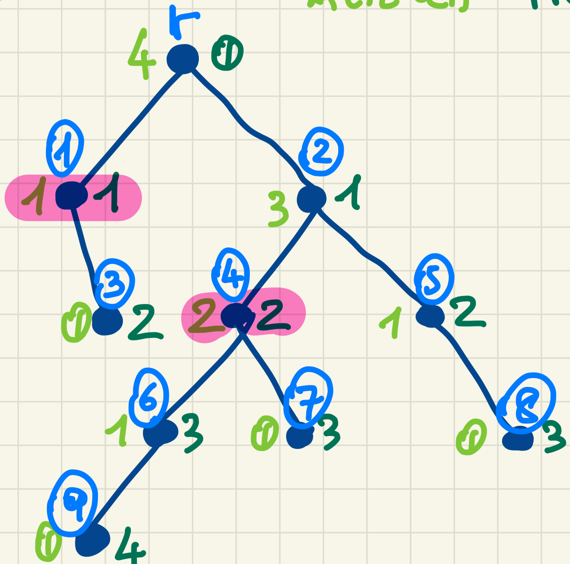
- RESTITUISCE h_u
- PRENDE IN INGRESSO p_u
- STAMPA u.key $\iff h_u = p_u$

$\Theta(n)$

COMPLESSITÀ LINEARE NEL NUMERO DI NODI di n

ESEMPIO

ALTEZZA PROFONDITÀ



CARDINE (u, p)

IF (u == nil) THEN RETURN -1;
ELSE
{ altsx = CARDINE (u.left, p+1);
altdx = CARDINE (u.right, p+1);
altezza = 1 + max { altsx, altdx };
IF (p == altezza) THEN printf(u.key);
RETURN altezza;
}

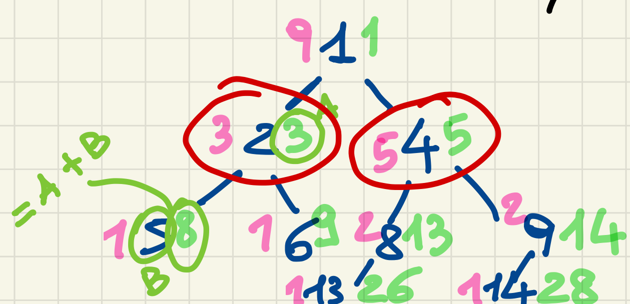
CARDINE (r , 0) → 4
CARDINE (1, 1) → 1
CARDINE (nil, 2) → -1
CARDINE (3, 2) → 0
CARDINE (nil, 3) → -1
CARDINE (nil, 3) → -1
→ STAMPO 1.key
CARDINE (2, 1) → 3
CARDINE (4, 2) → 2
CARDINE (6, 3) → 1
CARDINE (9, 4) → 0
CARDINE (nil, 5) → -1
CARDINE (nil, 5) → -1
CARDINE (nil, 4) → -1
⊗ ↓ CARDINE (7, 3) → 0
→ STAMPA 4.key

⊗ ↓ CARDINE (5, 3) → 1
CARDINE (nil, 4) → -1
CARDINE (8, 4) → 0
...

X CASA

- Un nodo u in un albero binario è detto CENTRALE se la dimensione (= numero nodi) del sottoalbero di cui è radice è uguale alla somma delle chiavi lungo il cammino dalla radice a u (inclusa)

PROGETTARE (E SCRIVERE PSEUDOCODICE) UN ALGORITMO RICORSIVO CHE, DATO UN ALBERO BINARIO, STAMPI TUTTI I SUOI NODI CENTRALI



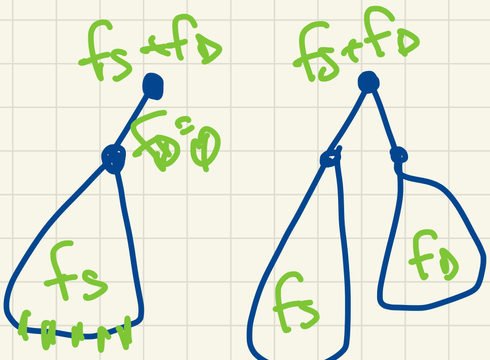
CONTAFOGLIE

Dato un albero binario, restituire il numero delle sue foglie

CASO BASE ① $n = nil$ → 0

CASO BASE ② $n \;!= nil$ e senza figli → 1

CASO GENERALE $n \;!= nil$ e ho ≥ 1 figlio → $f_S + f_D$



f_S: foglie di n.left

f_D: foglie di n.right

CONTAFOGLIE (u)

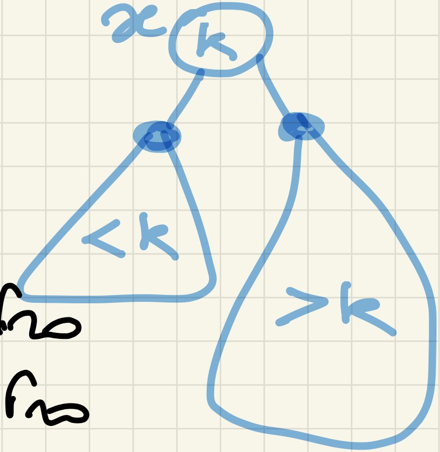
```
IF (u == nil) THEN RETURN 0;
IF ((u.left == nil) && (u.right == nil)) THEN RETURN 1;
RETURN CONTAFOGLIE(u.left) + CONTAFOGLIE(u.right);
```

$O(1)$

COMPLESSITÀ LINEARE

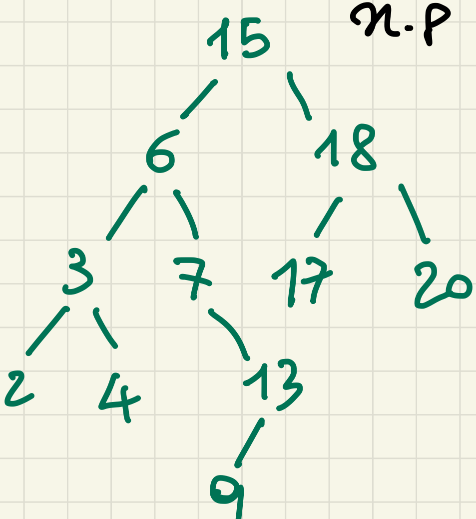
(È COME UNA VISITA)

ALBERI BINARI di RICERCA

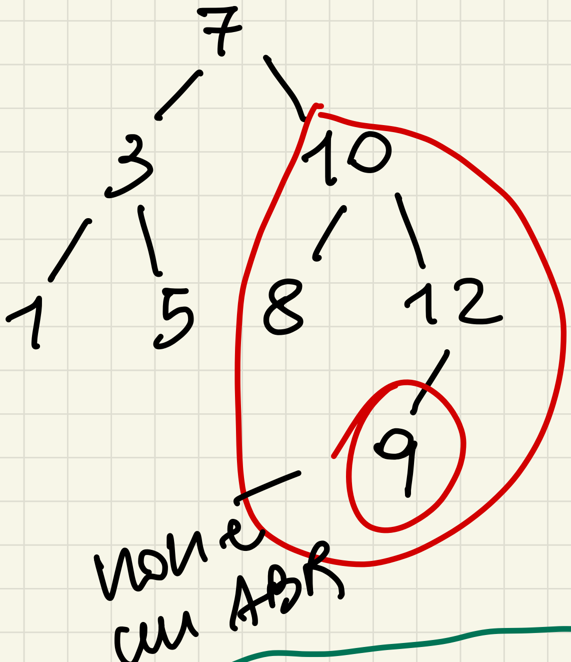


Ogni nodo x ha i campi:

- $x.key \longrightarrow$ chiave
- $x.left \longrightarrow$ sottoalbero sinistro
- $x.right \longrightarrow$ sottoalbero destro
- $x.p \longrightarrow$ genitore di x



<u>E</u>
1. tutti i nodi nel sottoalbero sinistro di x hanno chiavi $<$ di $x.key$
2. tutti i nodi nel sottoalbero destro di x hanno chiavi $> x.key$



ABR

ALBERO BINARIO RICERCA

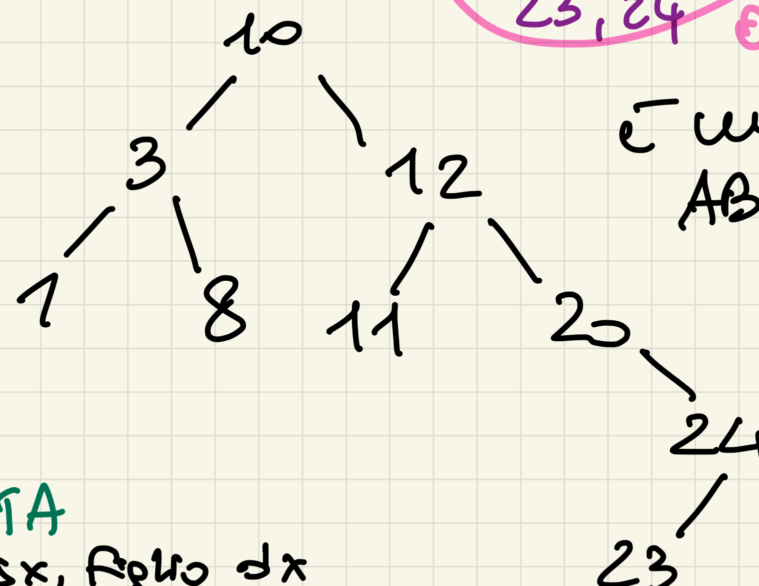
VISITA SIMMETRICA

figlio sx, radice, figlio dx

1, 3, 8, 10, 11, 12, 20, 23, 24

ORDINATA

N.B.



è un ABR

VISITA ANTICIPATA

radice, figlio sx, figlio dx

10, 3, 1, 8, 12, 11, 20, 24, 23

VISITA POSTICIPATA

figlio sx, figlio dx, radice

1, 8, 3, 11, 23, 24, 20, 12, 10

RICERCA IN ABR

$T(n)$

Ricerca di una chiave k in un ABR

CASO BASE

CASO OTTIMO

```
ABR-SEARCH(u, k)
IF ((u == nil) || (u.key == k)) THEN RETURN u;
IF (u.key > k) THEN RETURN ABR-SEARCH(u.left, k);
                ELSE RETURN ABR-SEARCH(u.right, k);
```

LA CORRETTEZZA SEGUE DALLA PROPRIETÀ DEGLI ABR

COMPLESSITÀ: CASO OTTIMO $\Theta(1)$ (lo trovo subito)

CASO PESSIMO $\Theta(h) \rightsquigarrow \Theta(n)$

h: ALTEZZA

CASO MEDIO

$\Theta(h) \simeq \Theta(\log n)$

ma $h \in O(n)$ e quindi

vado al caso pessimo