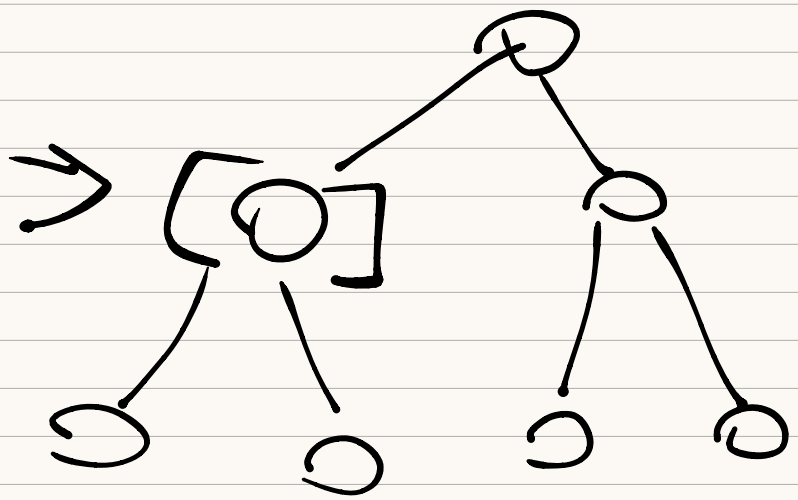
→ BILANCIAMENTO BST

LEFT − RIGHT ≤ 1

→

→



[k-ESIMO ELEMENTO]

(k = 2) 2-ESIMO

↓

K-TH SMALLEST (BST, K)

IF (BST == NULL)
    RETURN NULL

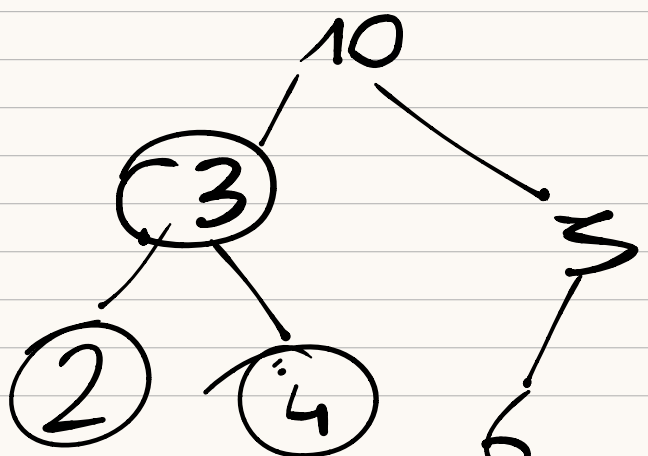
IF (KTH (BST, K) <= K)

RETURN KTH (BST -> LEFT PTR, K)

ELSE IF (KTH (RIGHT PTR, K) > K

{RETURN KTH (BST -> RIGHT PTR, K) }

RETURN PTR.



K = 3

K = 1

LEFT, K+1

If (ROOT) K+1

RETURN

RIGHT (K+1)

[10 ; 50]

/

If (LOW >=

RANGE_SUM

(ROOT → LEFTPTR,

LOW, HIGH)

SUM - LEFT += ROOT → VALUE

If (HIGH <

RANGE_SUM

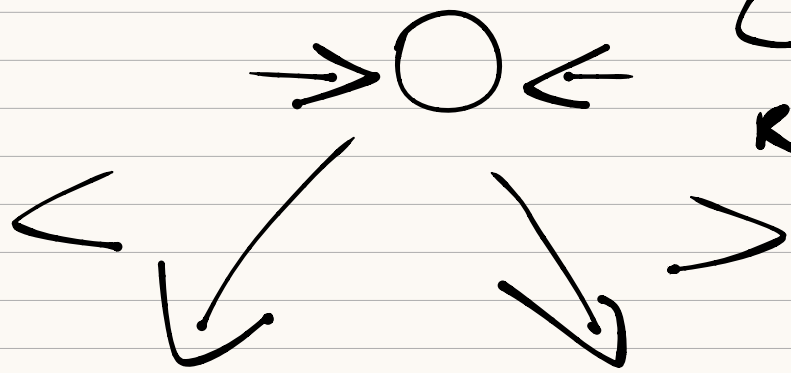
(ROOT → RIGHTPTR

LOW, HIGH)

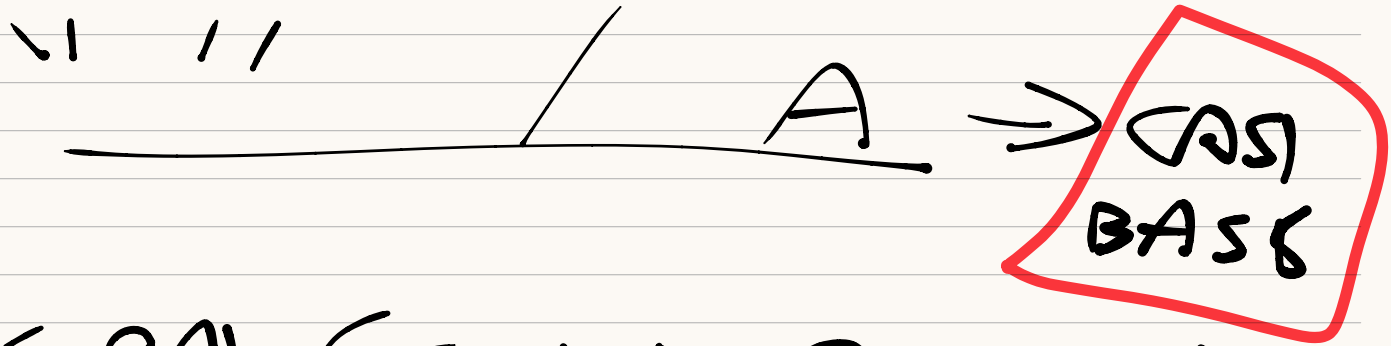
SUM - RIGHT += ROOT → VALUE

RETURN SUM_LEFT
+
SUM_RIGHT

SEARCH [IF(Root)
RSG
LEFT(<)
RIGHT(>)



[ANA/NA]



IS PAL(C+1, PAR-1)

YES / NO

ORD_INSERT(LISTA
= &PTRPTR, INTVAL)

!= NULL, PUNTATORE
AD UNA LISTA != 0.
VAL != NULL

POST →

PTRPTR

∀ i ∈ ℕ (4)

(L[1] ≤ L[2] ≤ L[3]

ORDINATI.

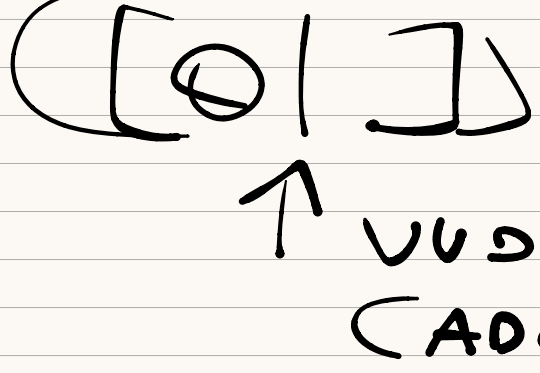
ORD_INSERT

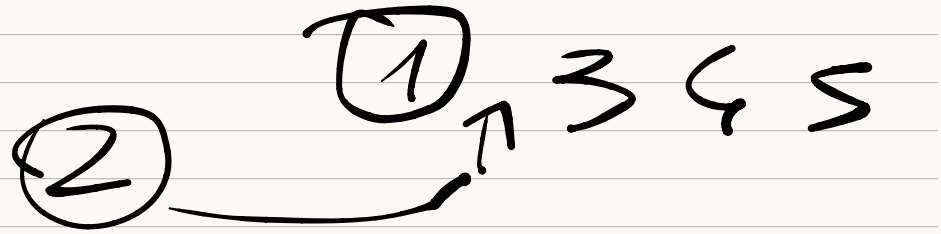
LISTA * NODO =

MALLOC (SIZEOF (LISTA))

NODO -> VALORE = VAL;

NODO -> NEXT = NULL;





WHILE (PTR->NEXT != NULL

&& VAL > *PTRPTR

→ VALORE)

{ NODO → NEXT = PTRPTR

PTRPTR = &NODO;

// PRE-INSERT
(NUOVO SPAZIO)

STRINGA DI 8 C.

→ CONST CHAR *

ELIMINA_PALINDR.(&S)

INT IMED = 0;

INT FINE = S.SIZE();

IF (S[INIZIO] != S[FINE])

{ // INIZIO++, FINE--

INIZIO++;

FINE--;

ELSE {

CHAR I = S[INIZIO]

I = S[INIZIO+1]

INIZIO++

CHAR F = S[FINE]

F = S[FINE-1]

FINE--

}

FOR (INT I=0; I<S.SIZE;

I++)

// pre

S, stringa
con "k" caratt.,

$\forall x, y$ car pal.

$x = y, x + 1 = y - 1$

post

-> S, stringa dove

$\notin$ car.
palindromi

$(\forall c \in S,$
ogni caratt. in stringa,

$(x, y) \rightarrow x != y, \forall i$

p. 05 (8 hours)

PARTNERS