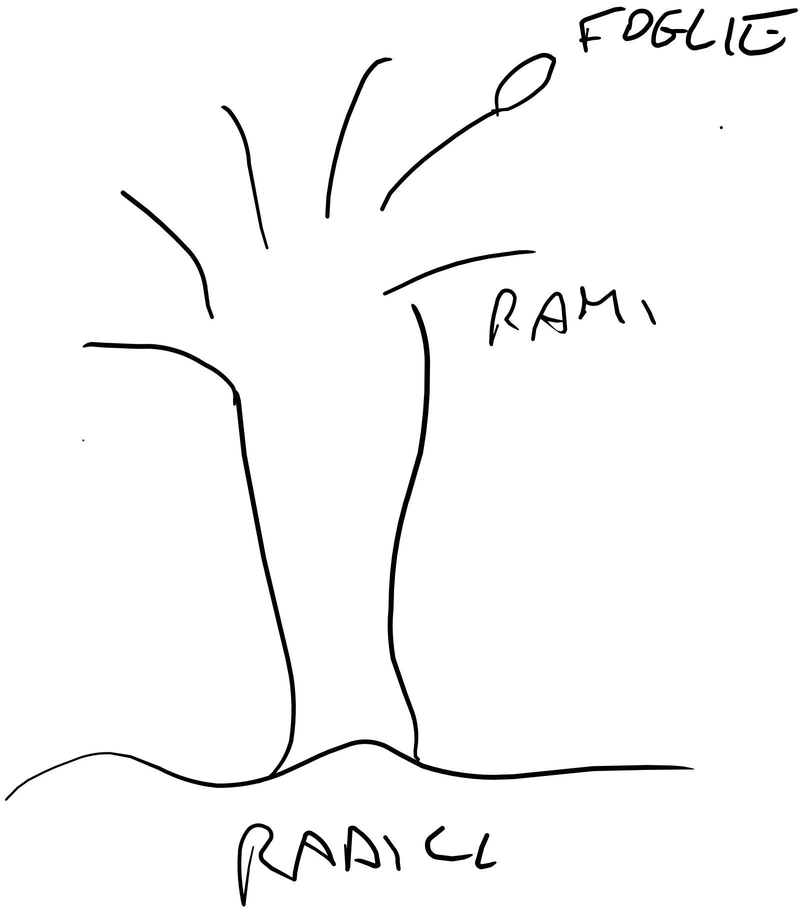
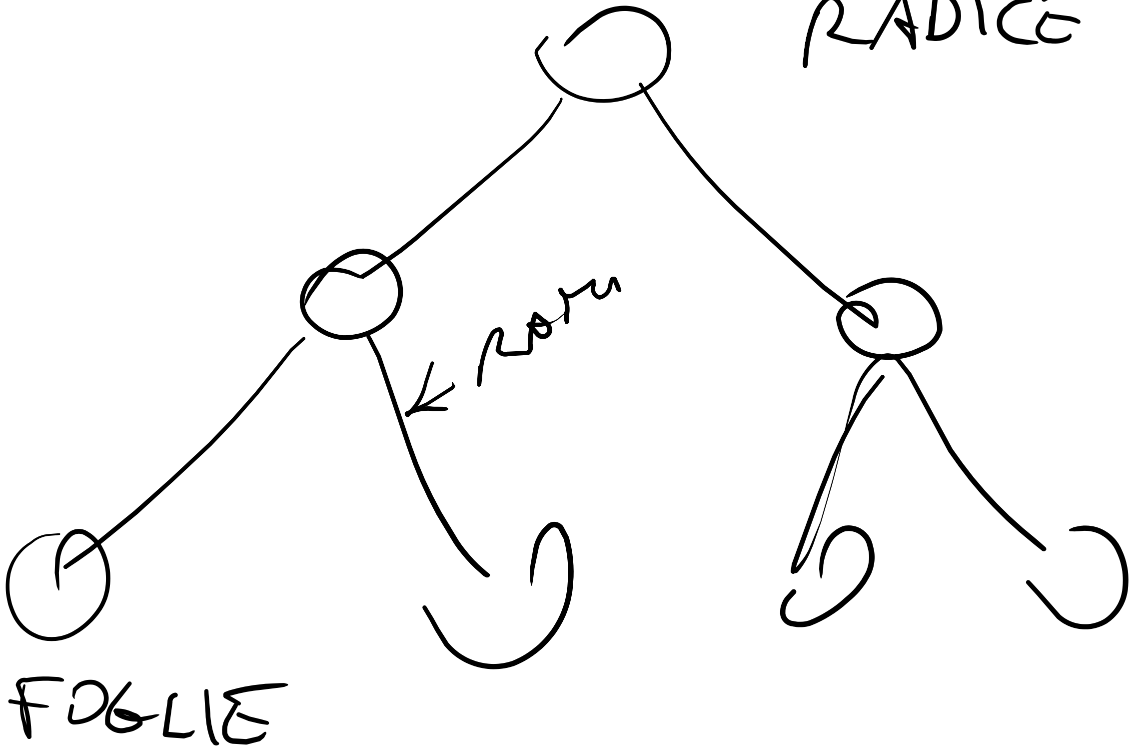
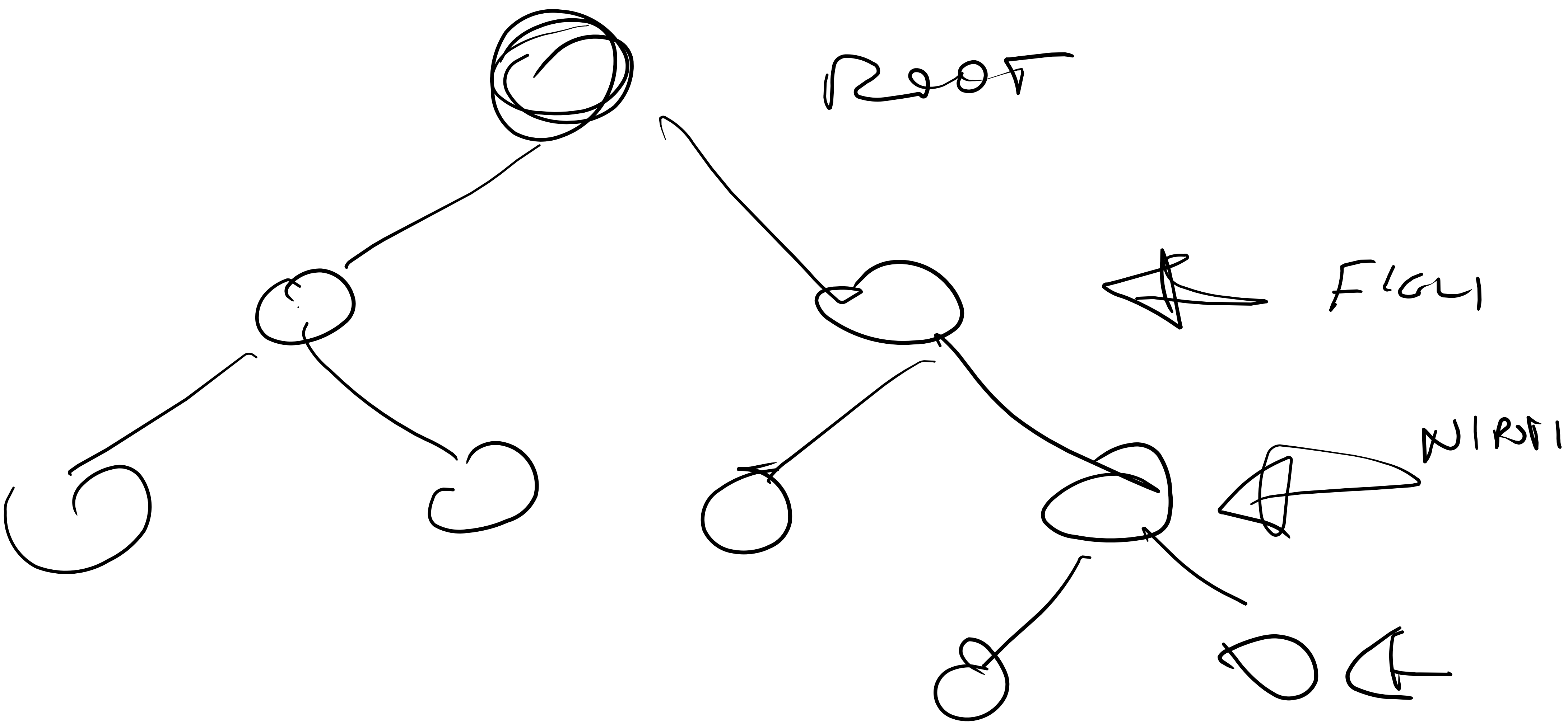
# alberi binari di ricerca

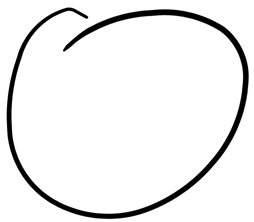



NATURA




INFORMATICA

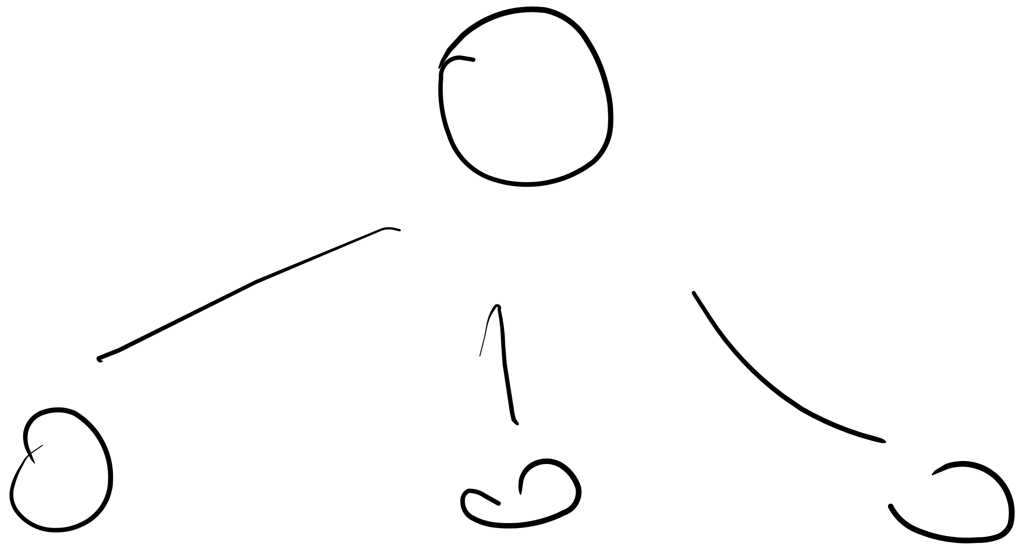

ROOT
FIGLI
NIPOTI

 NODO

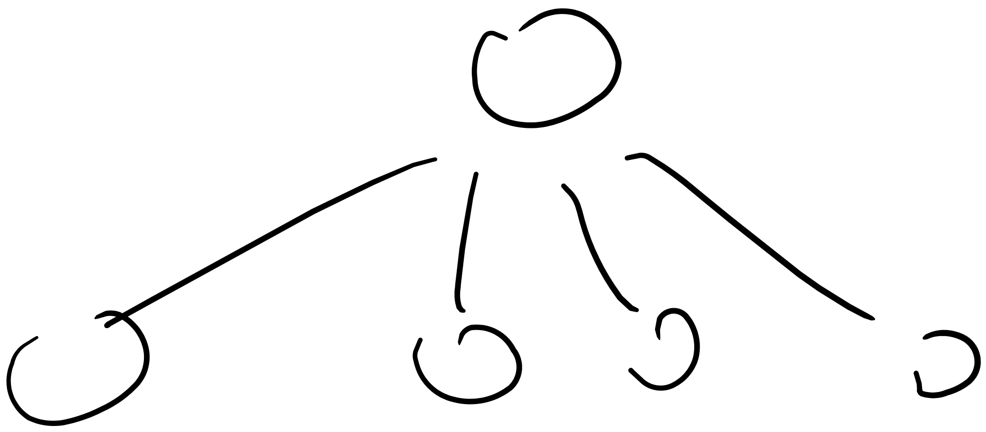
**ARIETÀ** = NUMERO DI FIGLI POSSIBILI

FIGLI → O . . . O ELENCO DI NODI

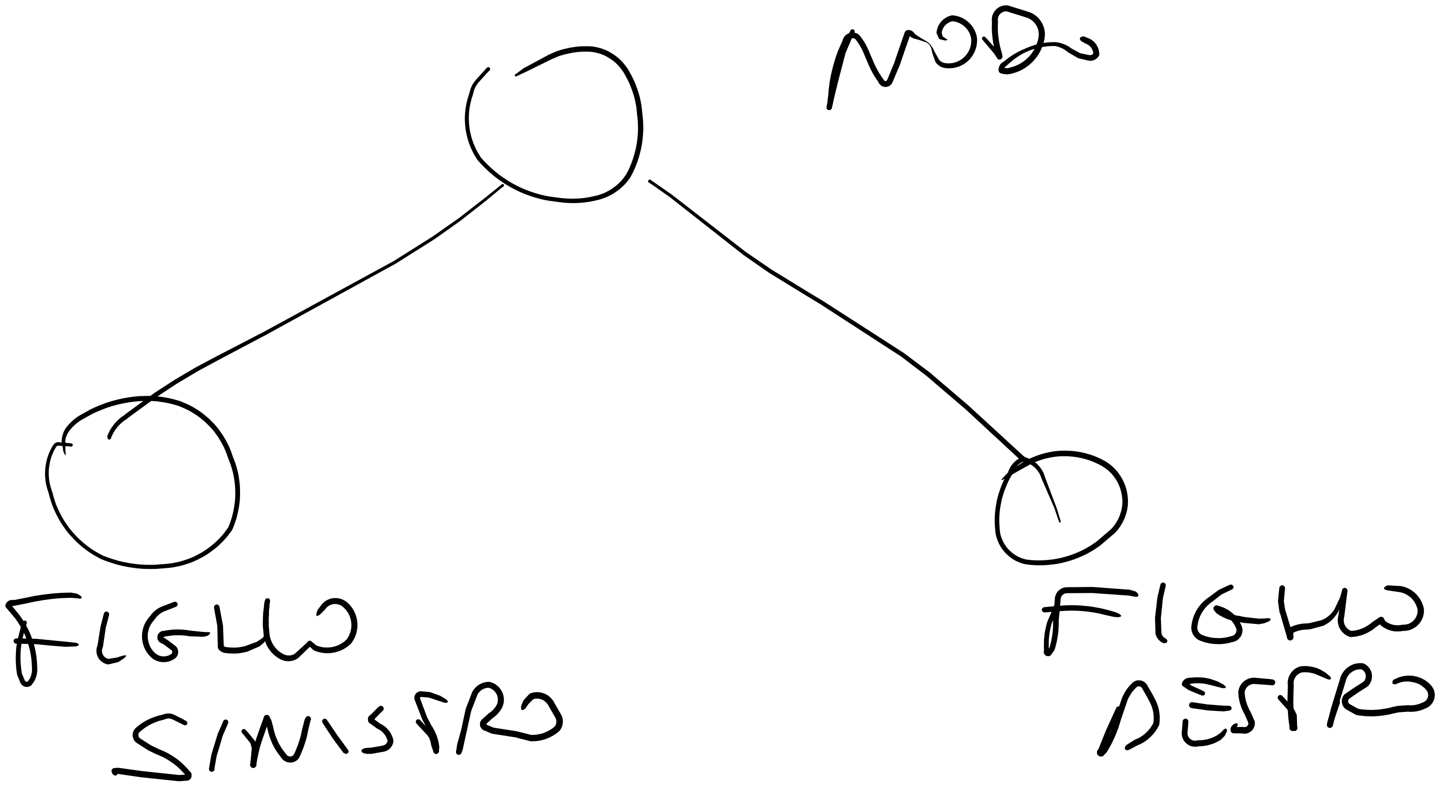
PARENT → O NODO



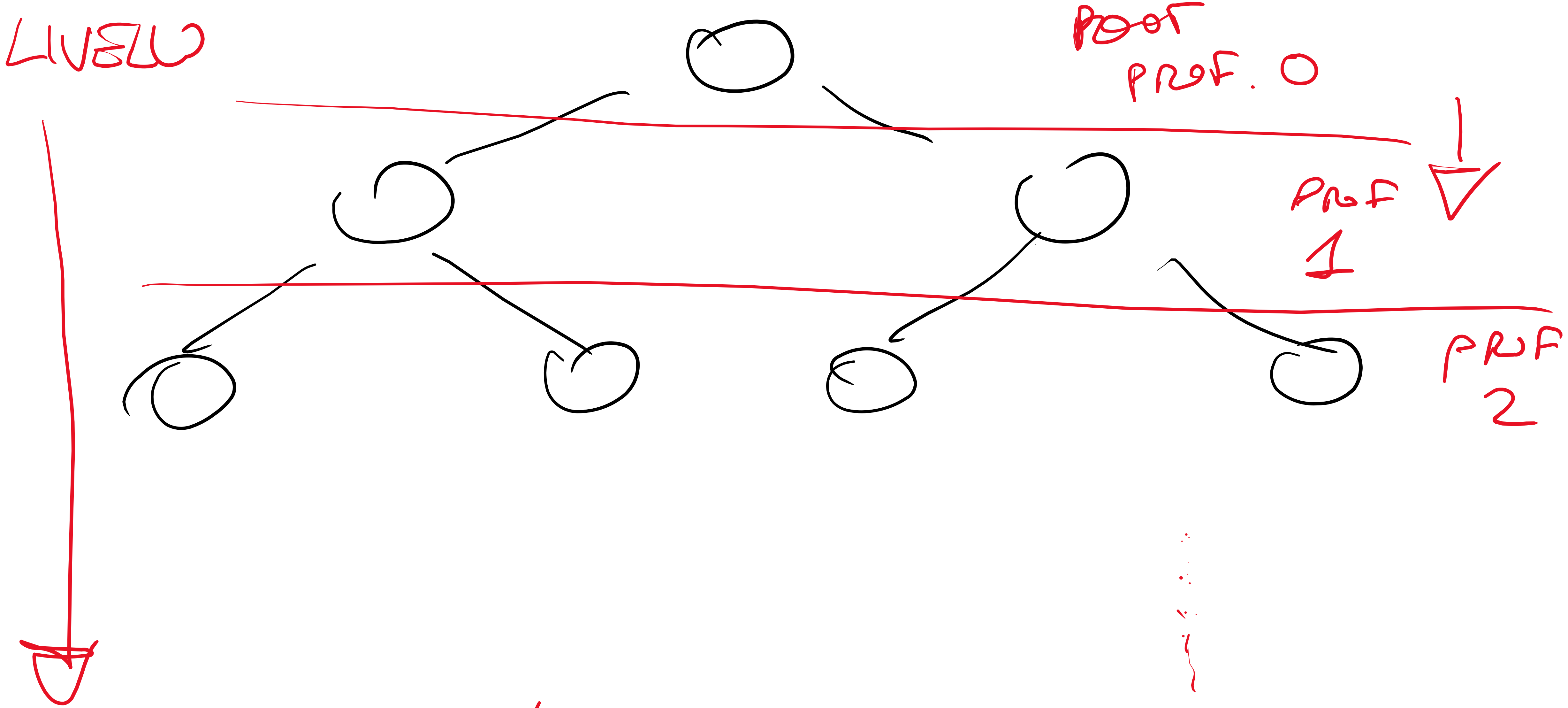
TERNARIO



QUATERNARIO

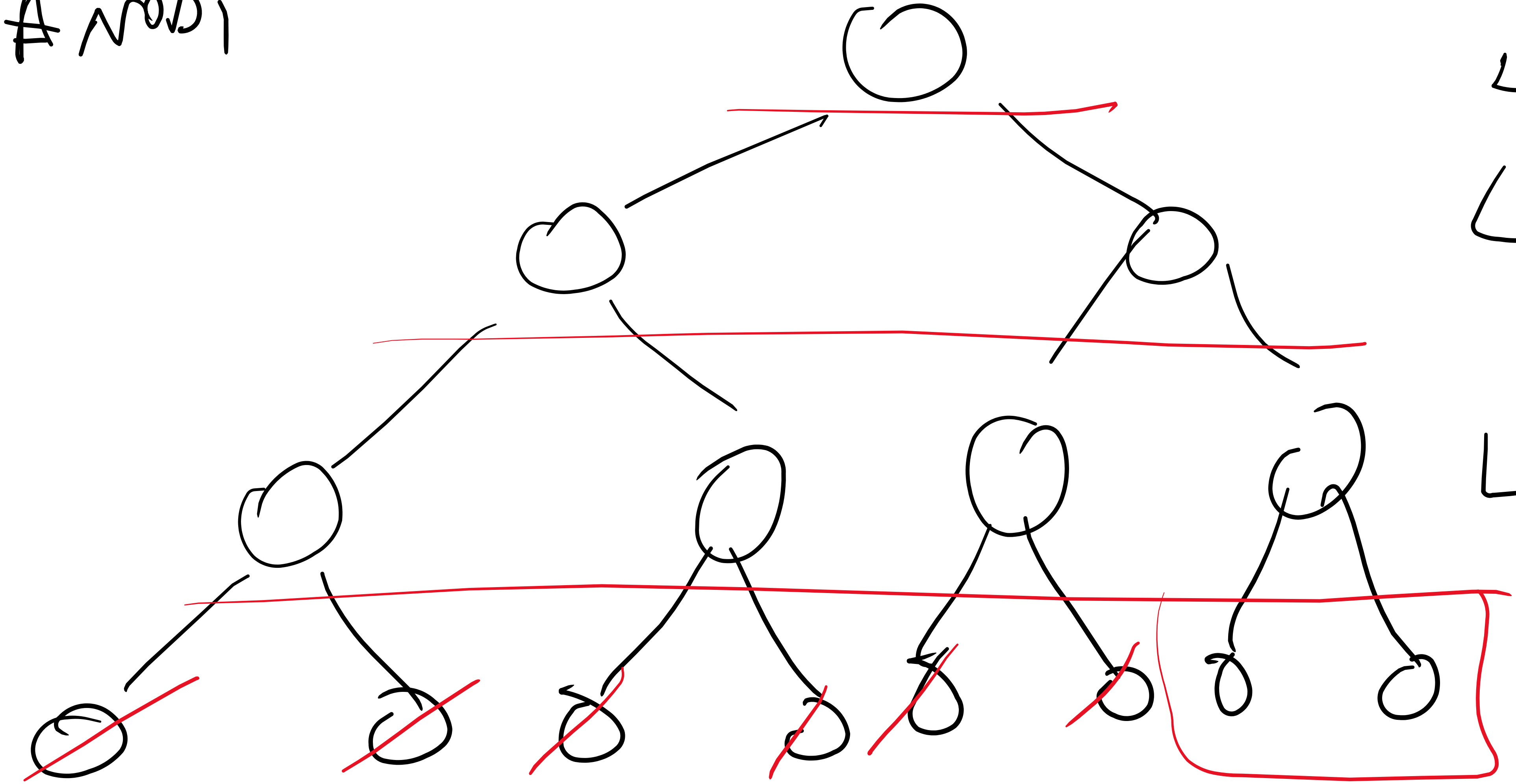



BINARIO




ALTEZZA DELL'ALBERO = # DI LIVELLI

# NODI



LIV 0: 1 $2^0$

LIV 1: 2 $2^1$

LIV 2: 4 $2^2$

LIV 3: 8 $2^3$

LIV 4: 16 $2^4$

# numero di nodi $= 2^h - 1 \rightarrow$

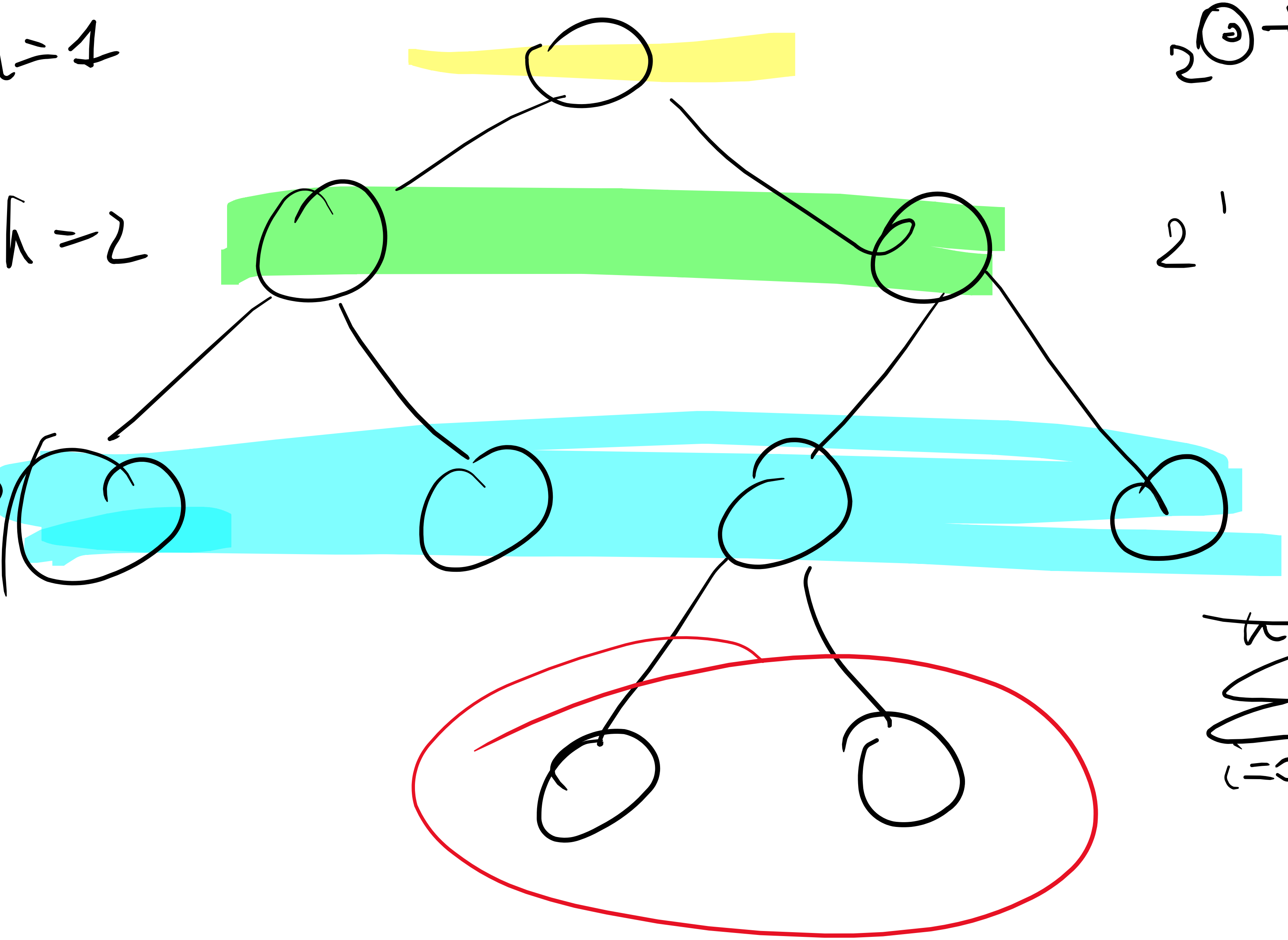
# nodi $\leq 2^h - 1 \rightarrow n \leq 2^h - 1 \rightarrow h = \lceil \log_2 n \rceil$

UN ALBERO BINARIO È COMPLETO SE E SOLO SE HA ESATTAMENTE $2^h$ NODI, DOVE $h$ è l'altezza dell'albero

$h=1$

$h=2$

$h=3$



$2^0 \rightarrow$ LIVELLO

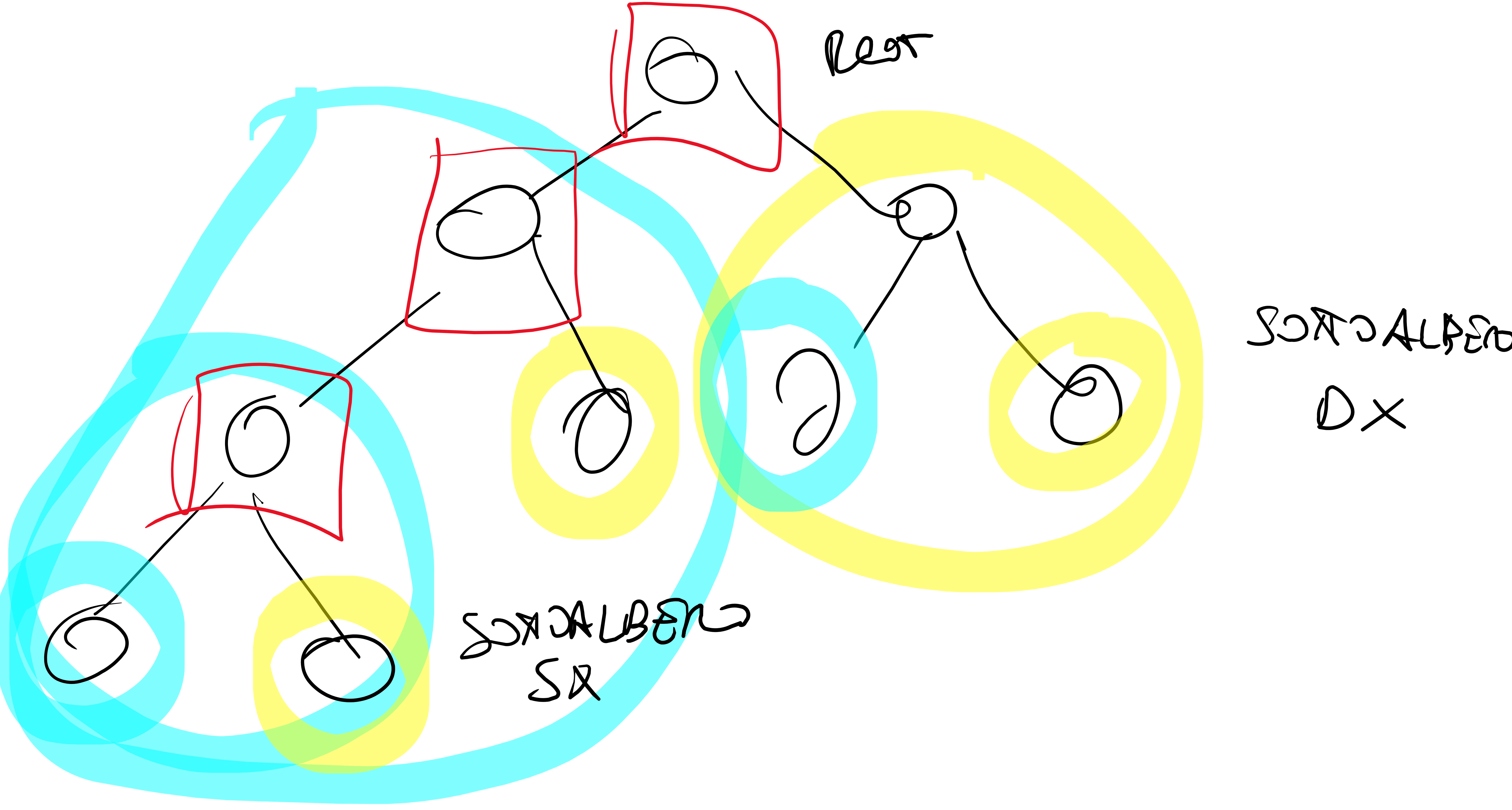
Altezza : 4

# nodi : 15

$2^1$

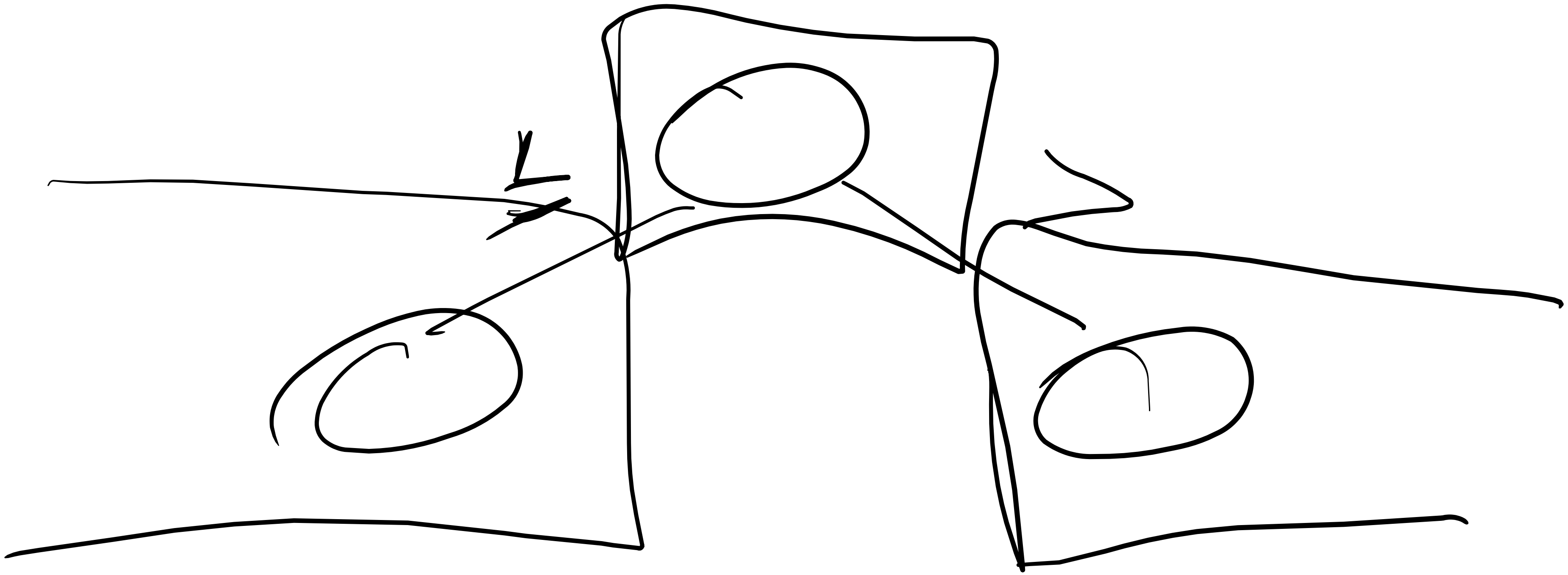
$2^2$

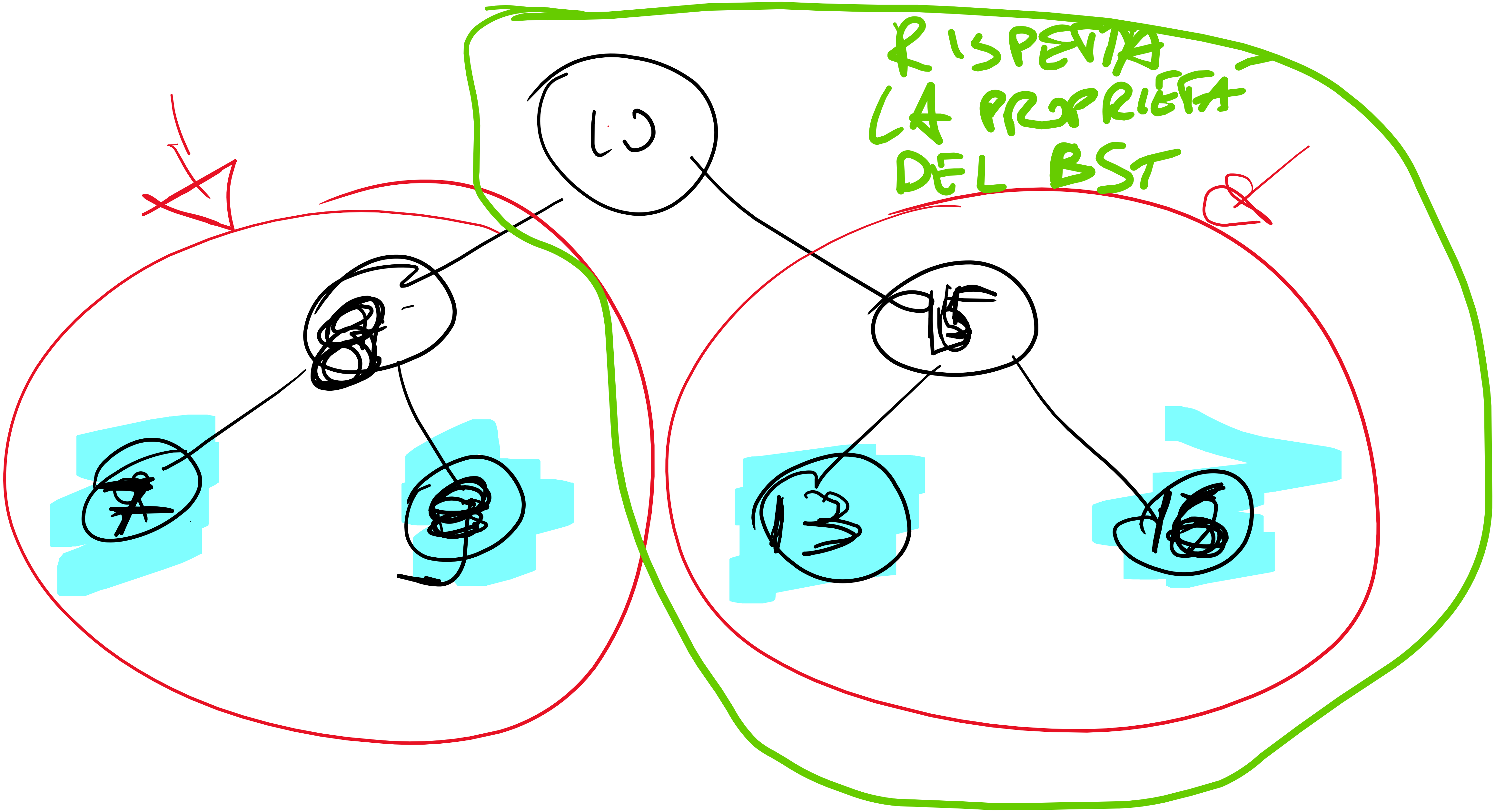
$$\sum_{i=0}^{h} 2^i = 2^h - 1$$

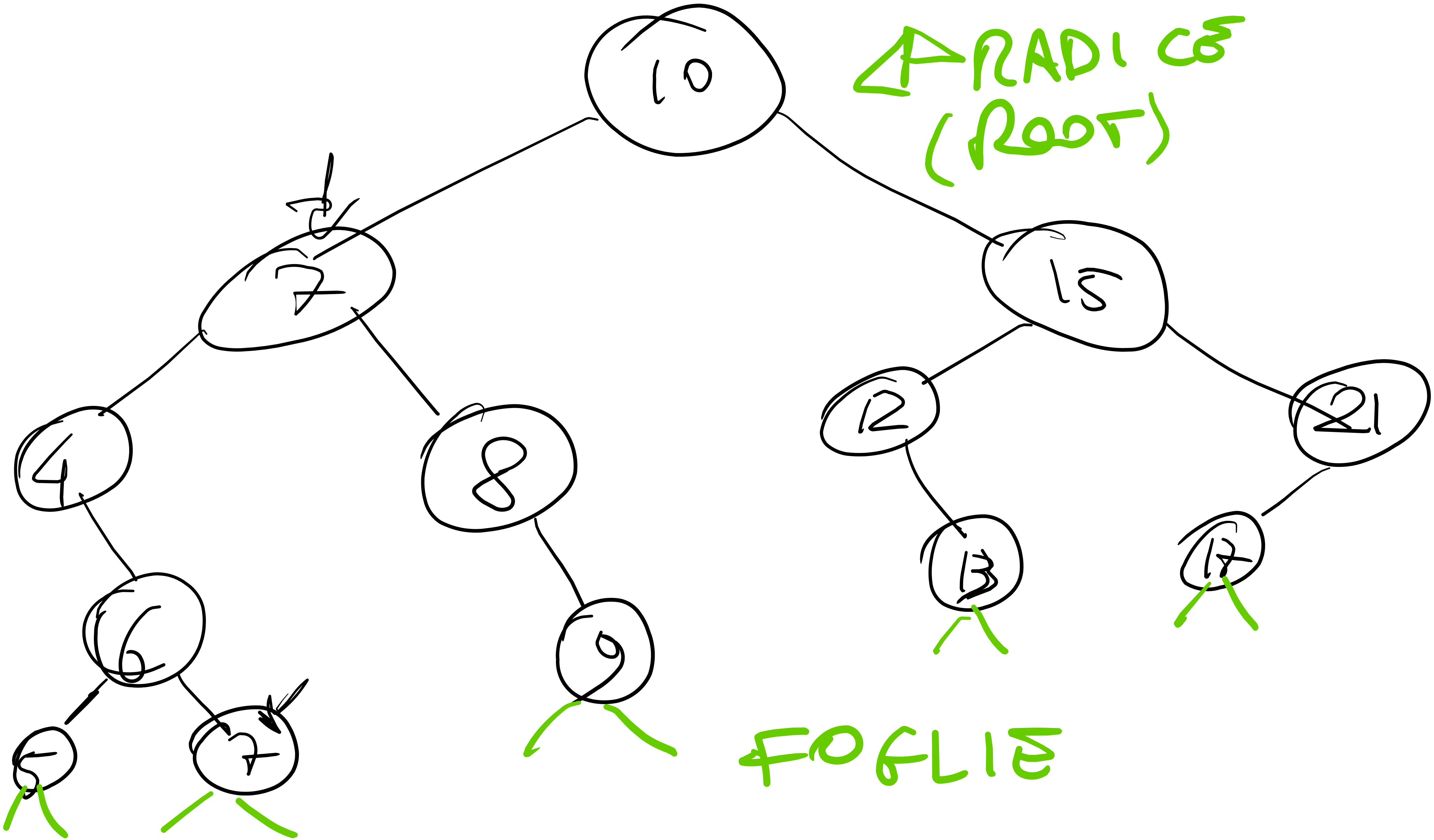

ROOT
SOTTOALBERO SX
SOTTOALBERO DX

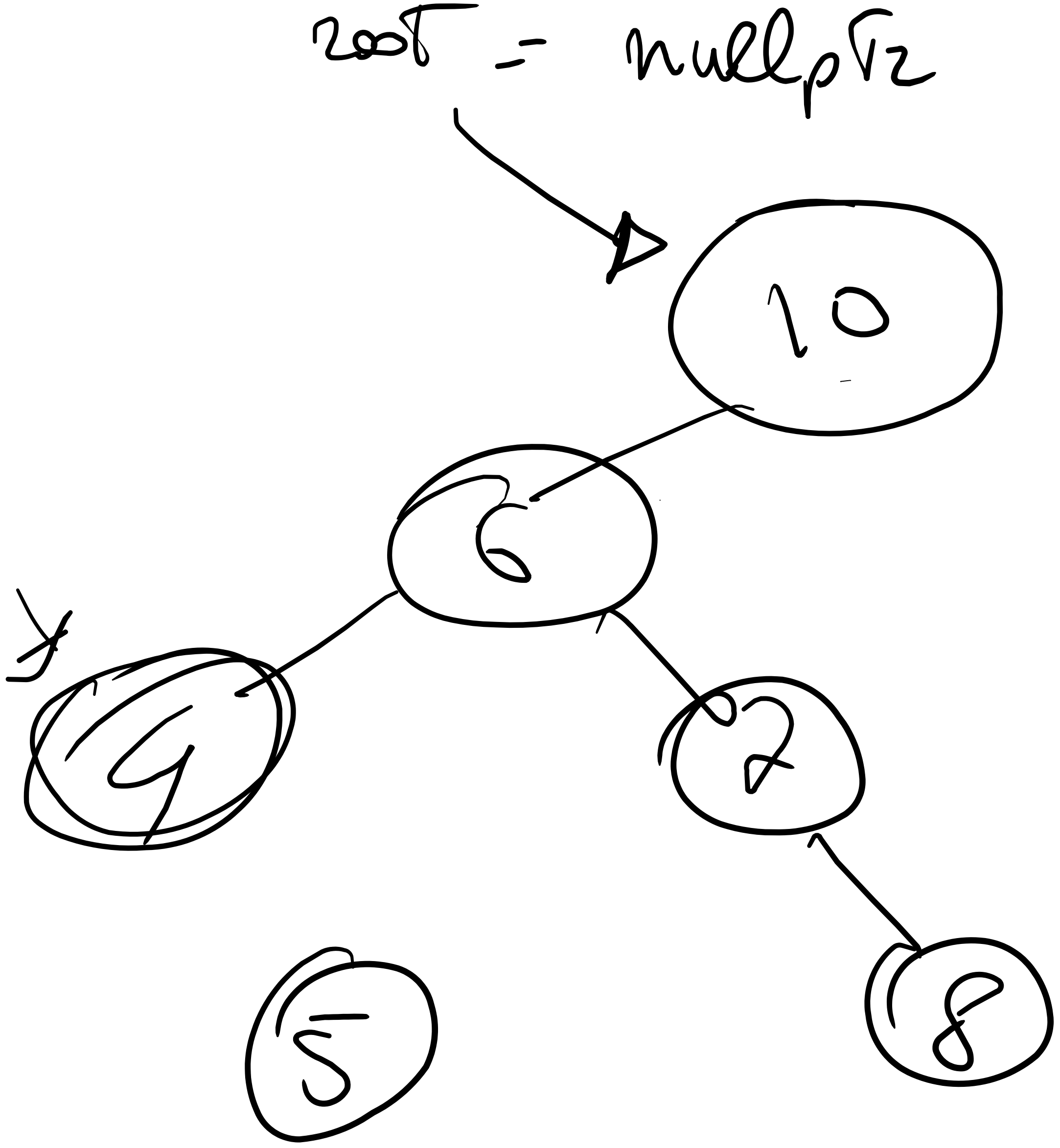
# ALBERO BINARIO DI RICERCA

(BST - Binary Search Tree)




RISPETTA
LA PROPRIETÀ
DEL BST
10
8
7
9
15
13
16


10
RADICE
(ROOT)
7
15
4
8
12
21
6
9
13
17
5
7
FOGLIE




insert (10)

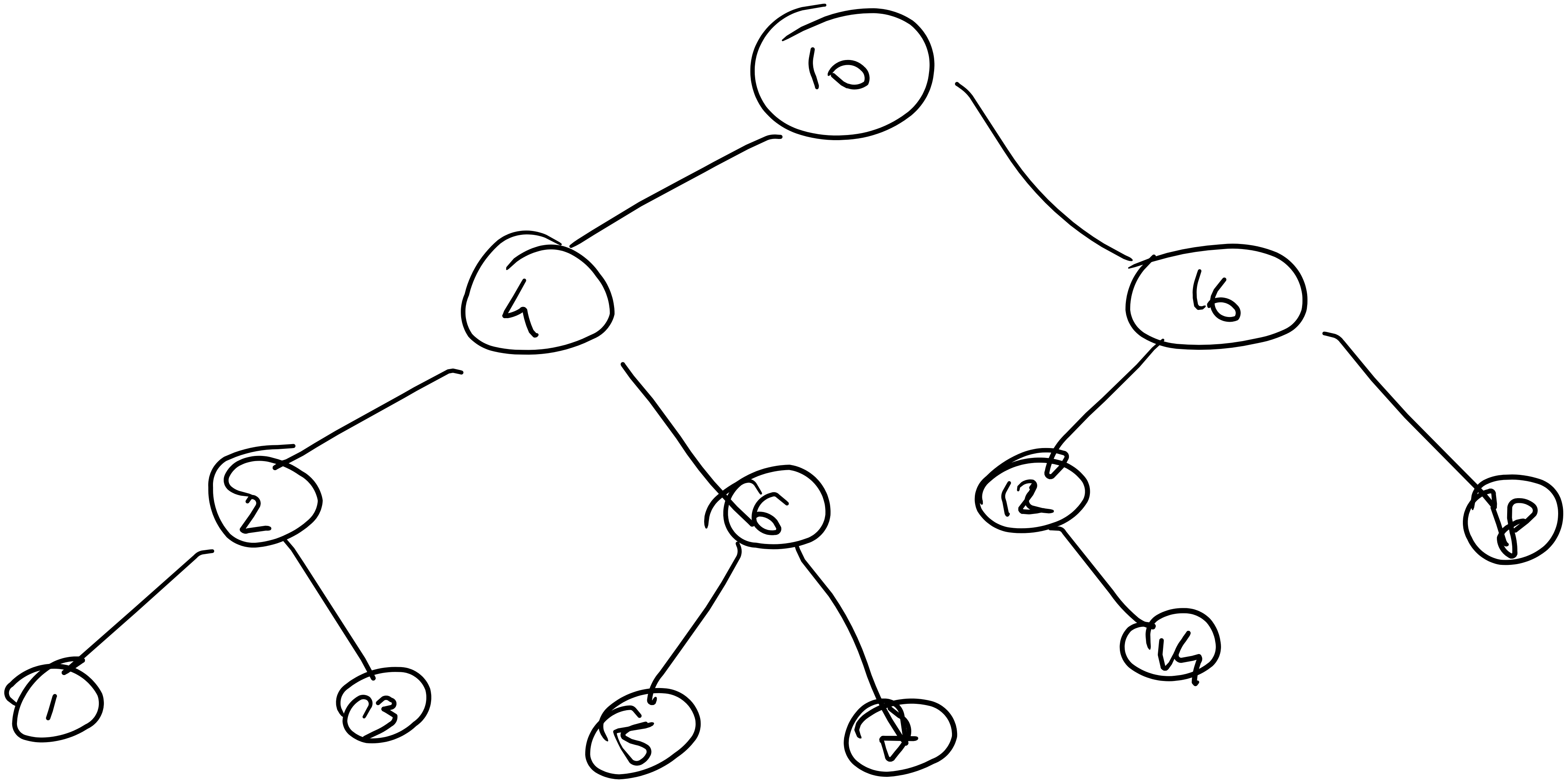
insert (6)

insert (4)

insert (2)

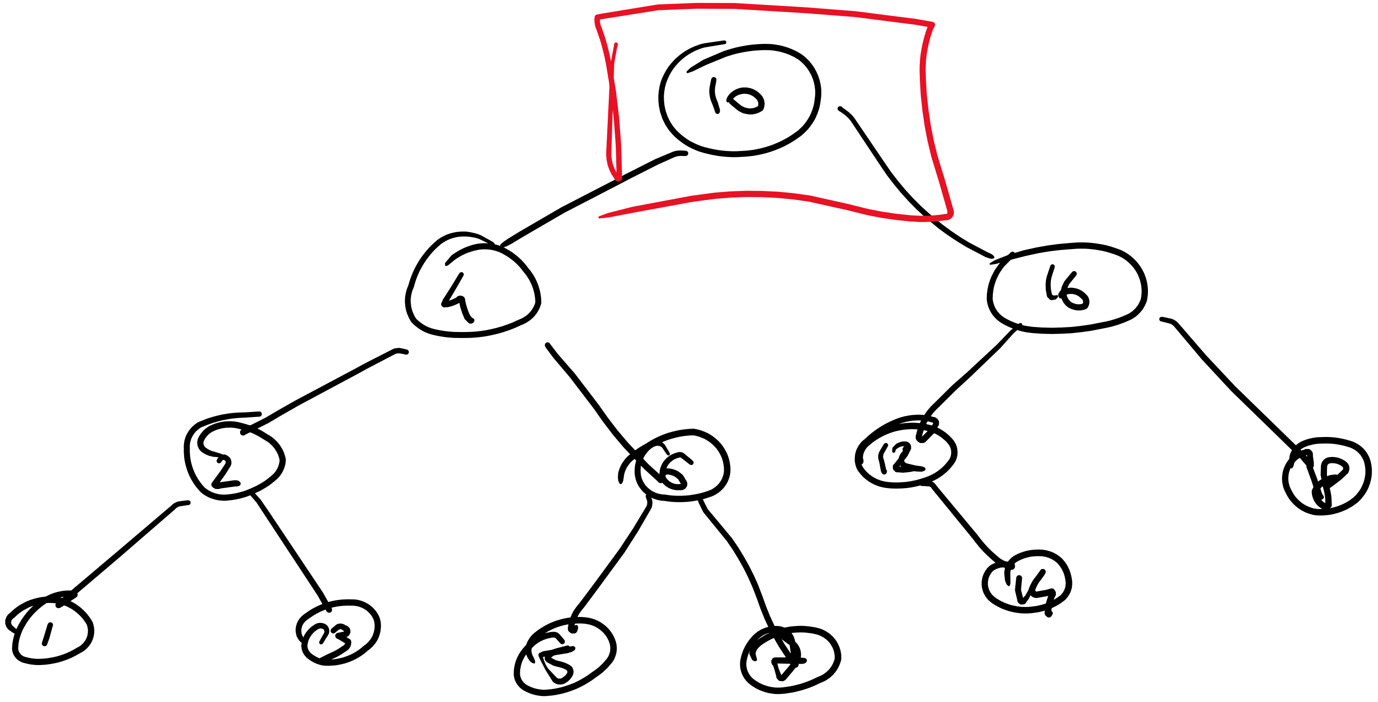
insert (8)

insert (5)



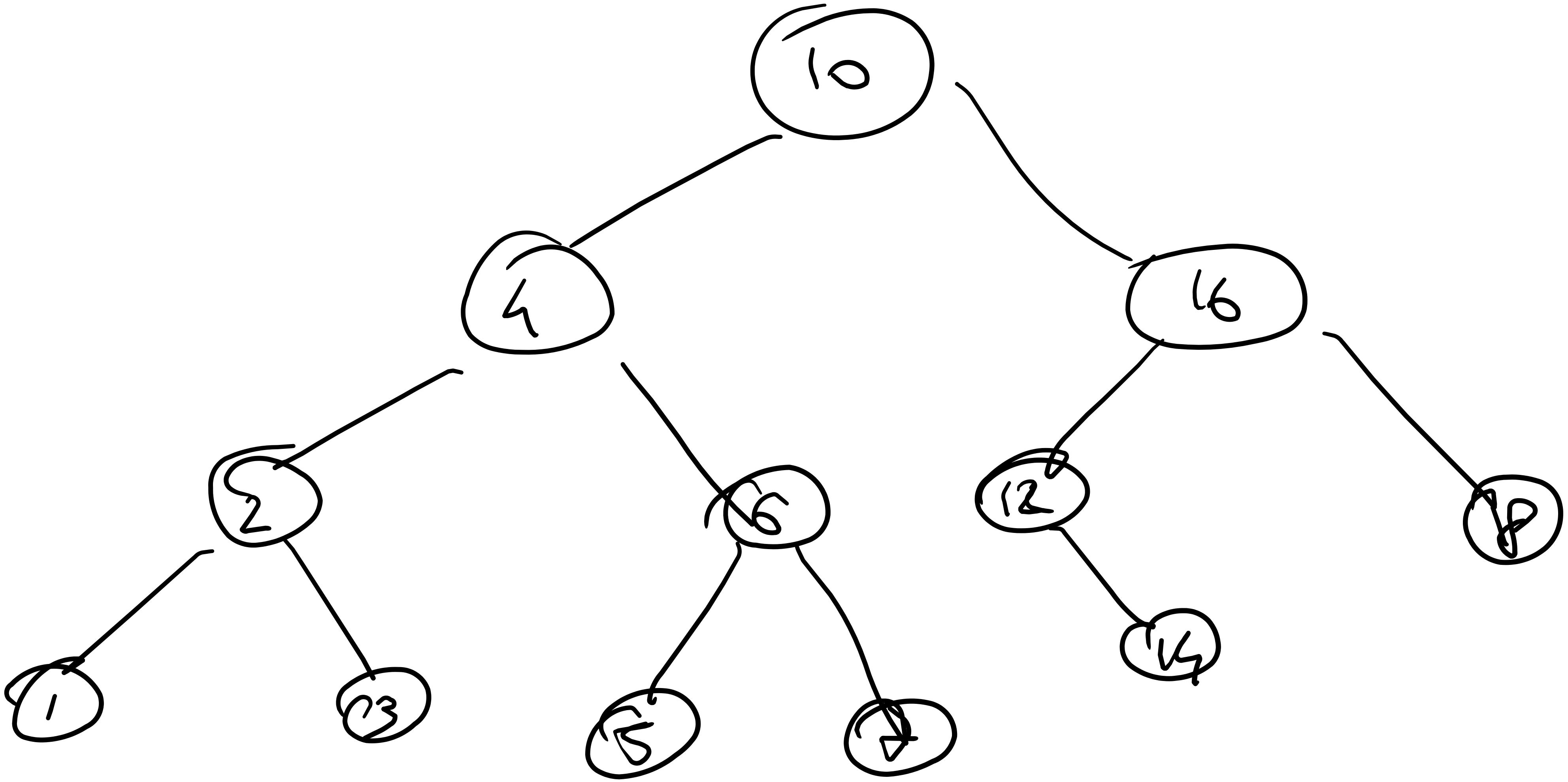

VISITO PRIMA LA RADICE, POI IL SOTT. SX
E INFINE IL SOTT. DX

## VISITA PREORDER



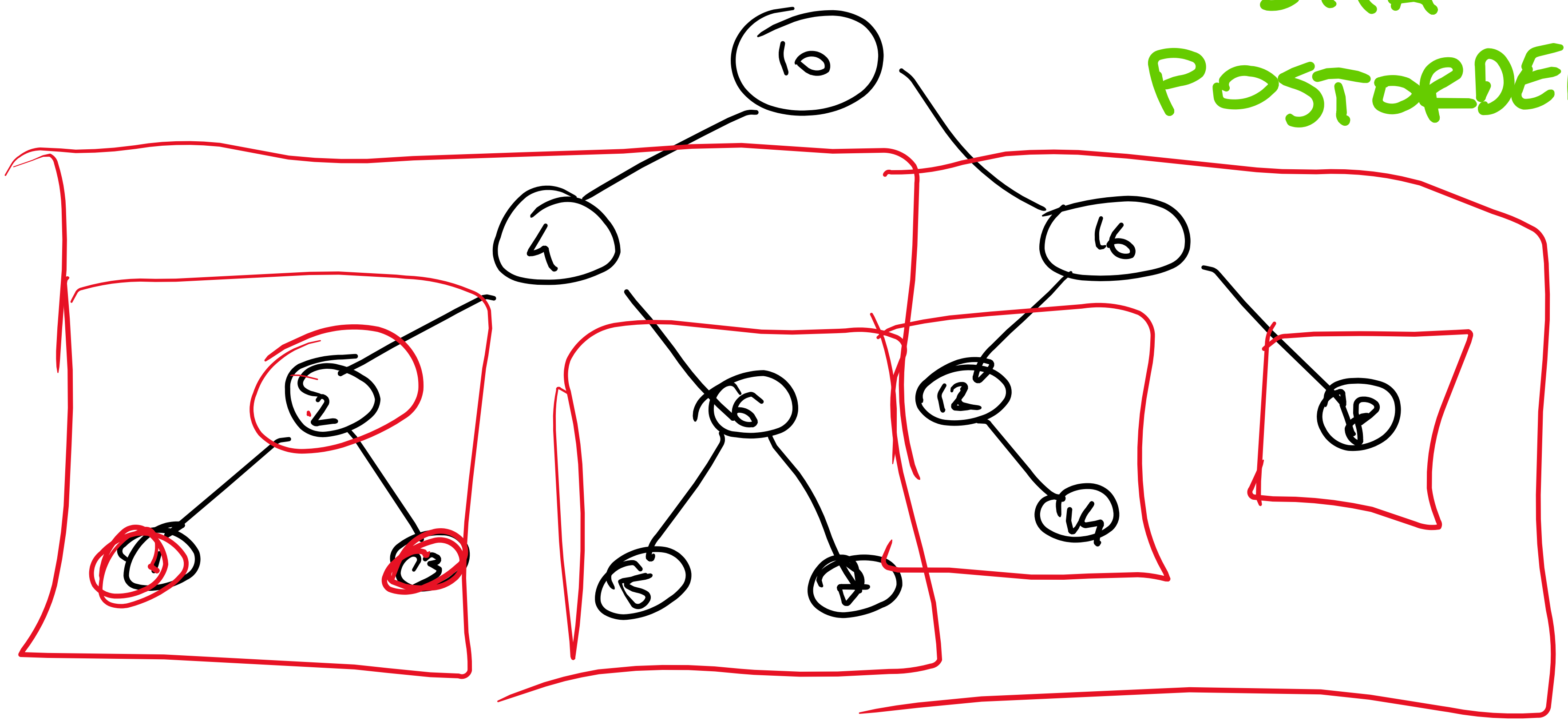

10 - 4 - 2 - 1 - 3 - 6 - 5 - 7 - 16 - 12 - 14 - 18

VISITO PRIMA LA RADICE, POI IL SOTT. SX
E INFINE IL SOTT. DX



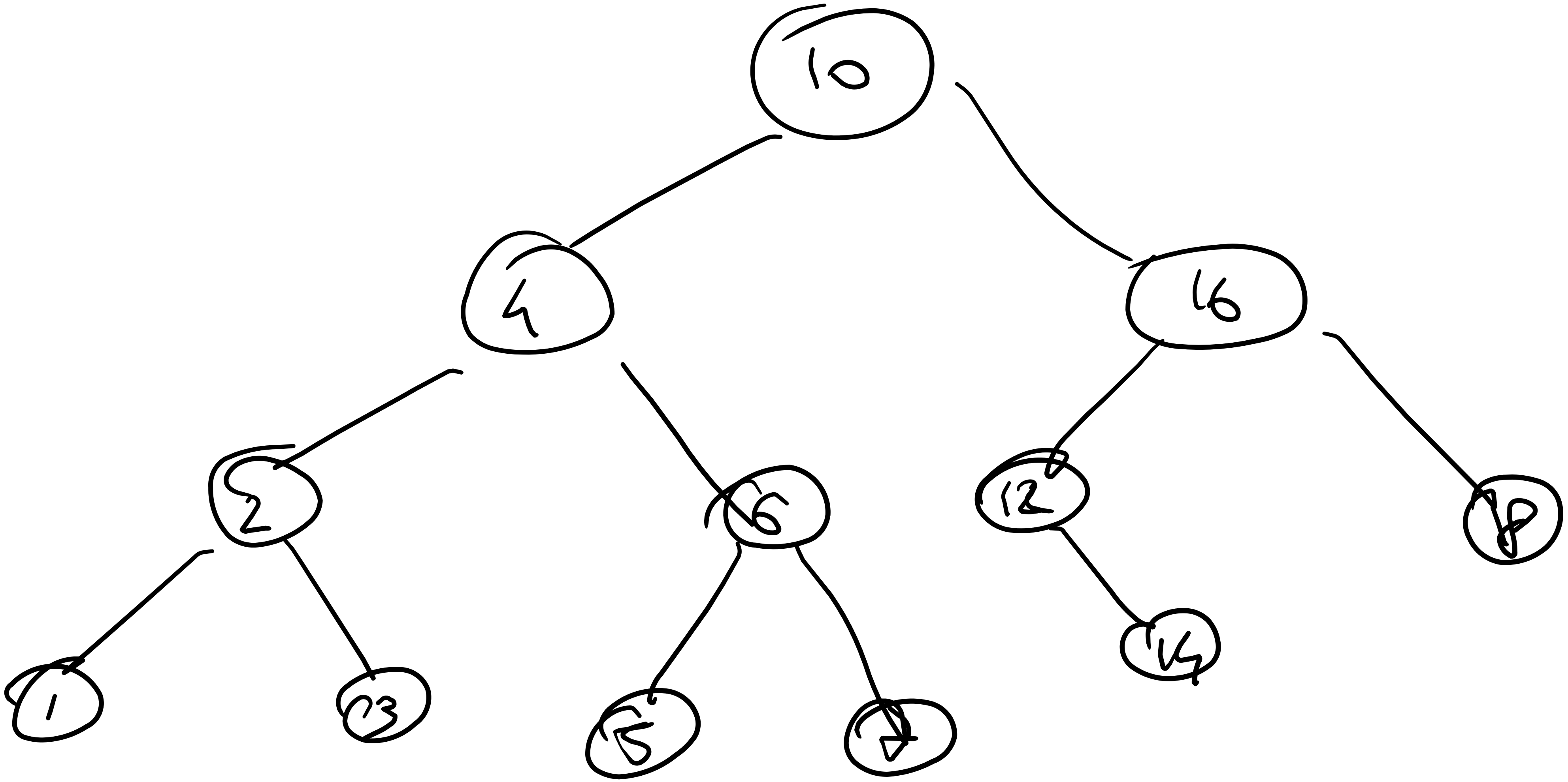

VISITO RICORSIVAMENTE IL SOTT. SX, POI, IL SOTT. DX E INFINE LA RADICE

## VISITA POSTORDER



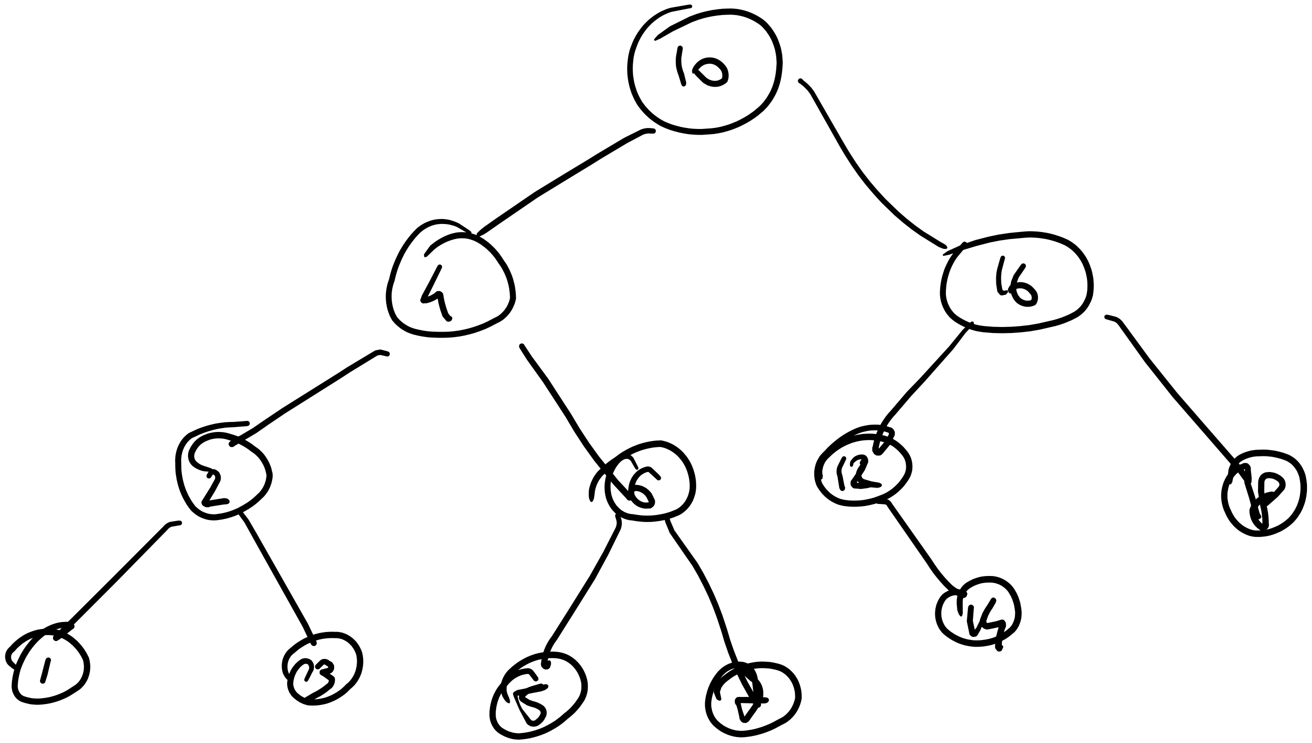
1 - 3 - 2 - 5 - 8 - 6 - 4 - 14 - 12 - 18 - 16 - 10

VISITO RICORSIVAMENTE IL SOTT. SX, POI, IL SOTT. DX E INFINE LA RADICE




VISITO IL SOTT. SX, POI LA RADICE, E
INFINE IL SOTT. DX

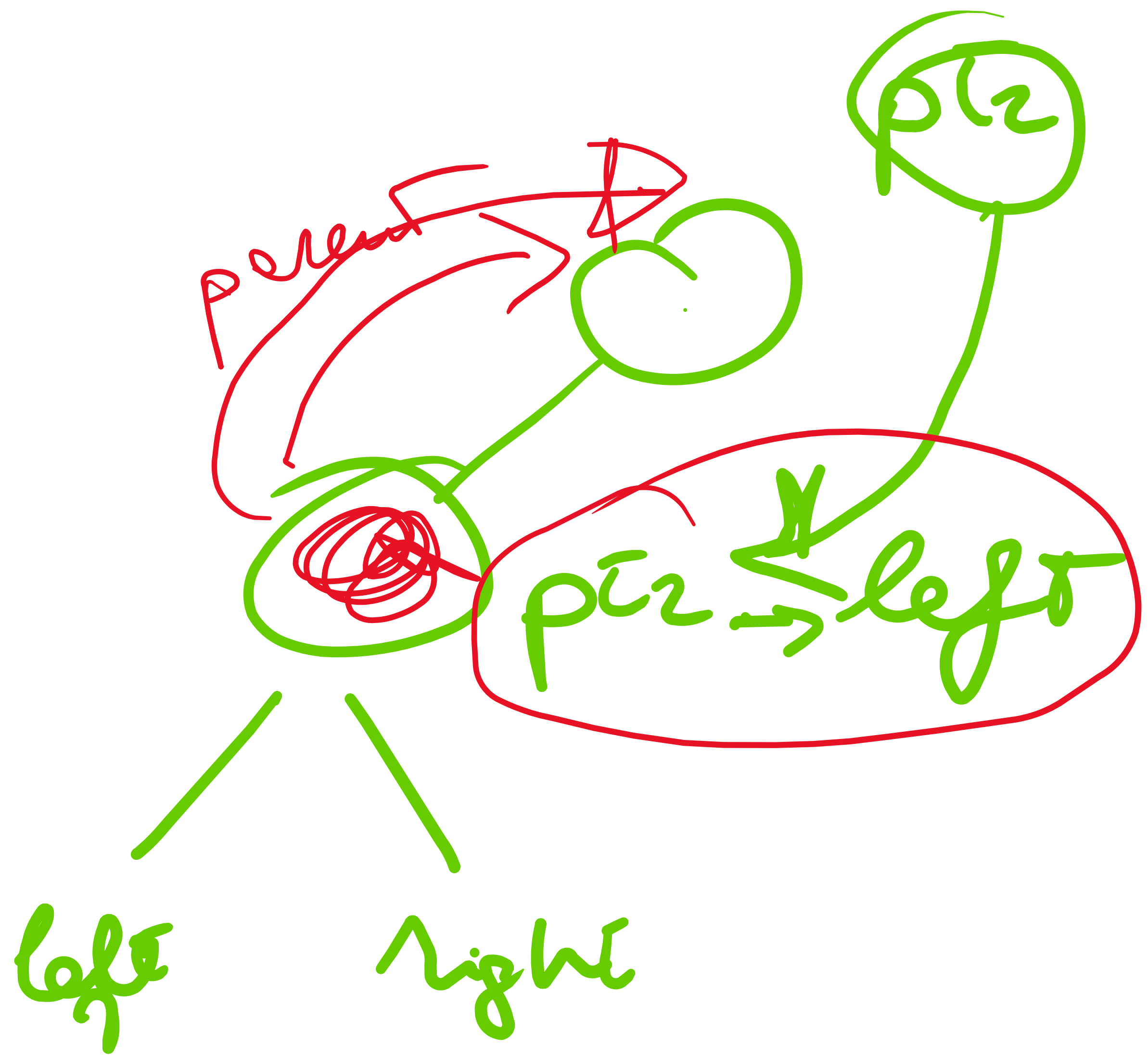
# VISITA INORDER



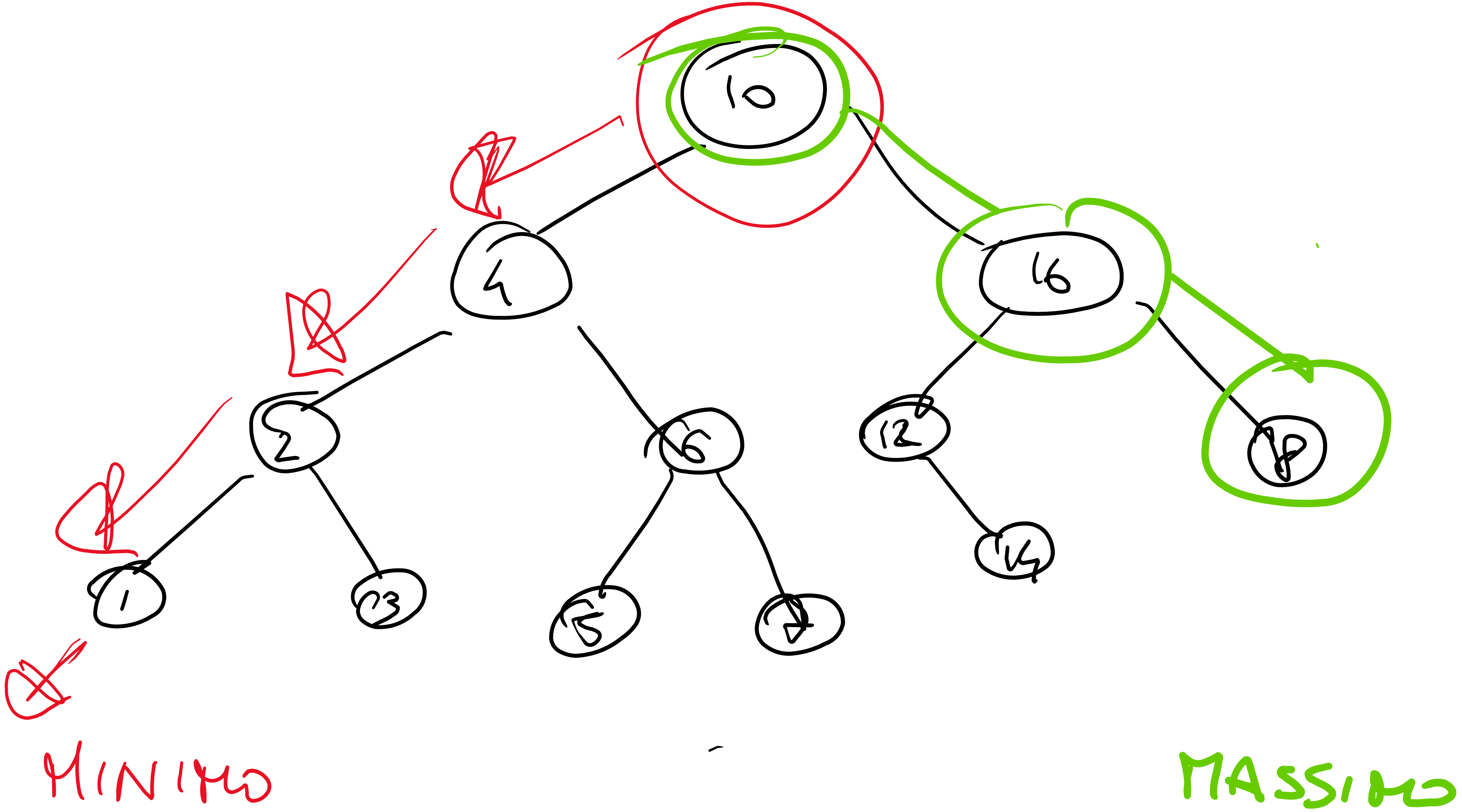

~~12~~
~~16~~
~~6~~
~~1~~
~~2~~
~~4~~
~~10~~

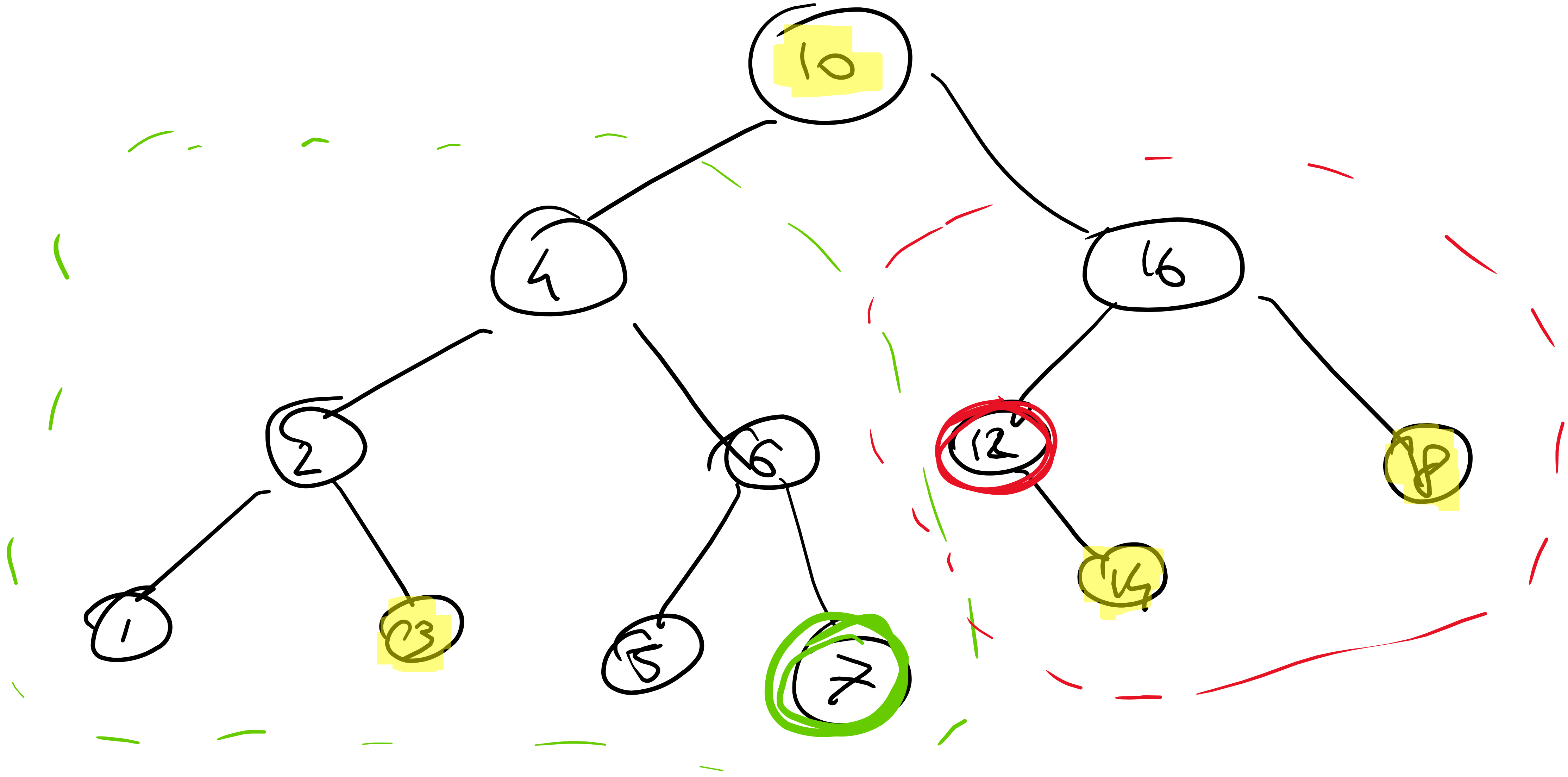
1 - 2 - 3 - 4 - 5 - 6 - 7 - 10 - 12 - 14 - 16 - 18

VISITO IL SOTT. SX, POI LA RADICE, E INFINE IL SOTT. DX

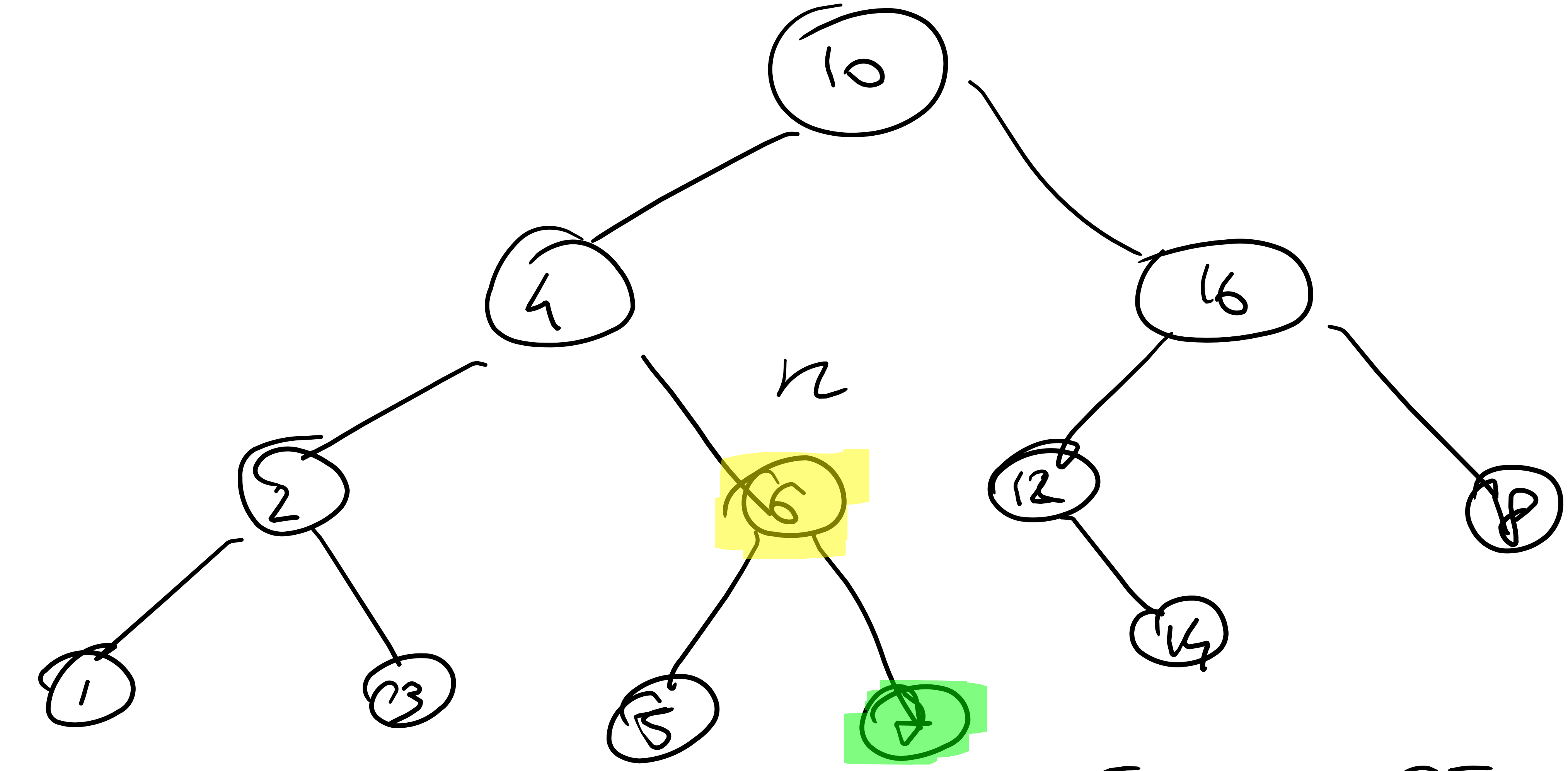


parent
pi2
pi2 → left
left
right

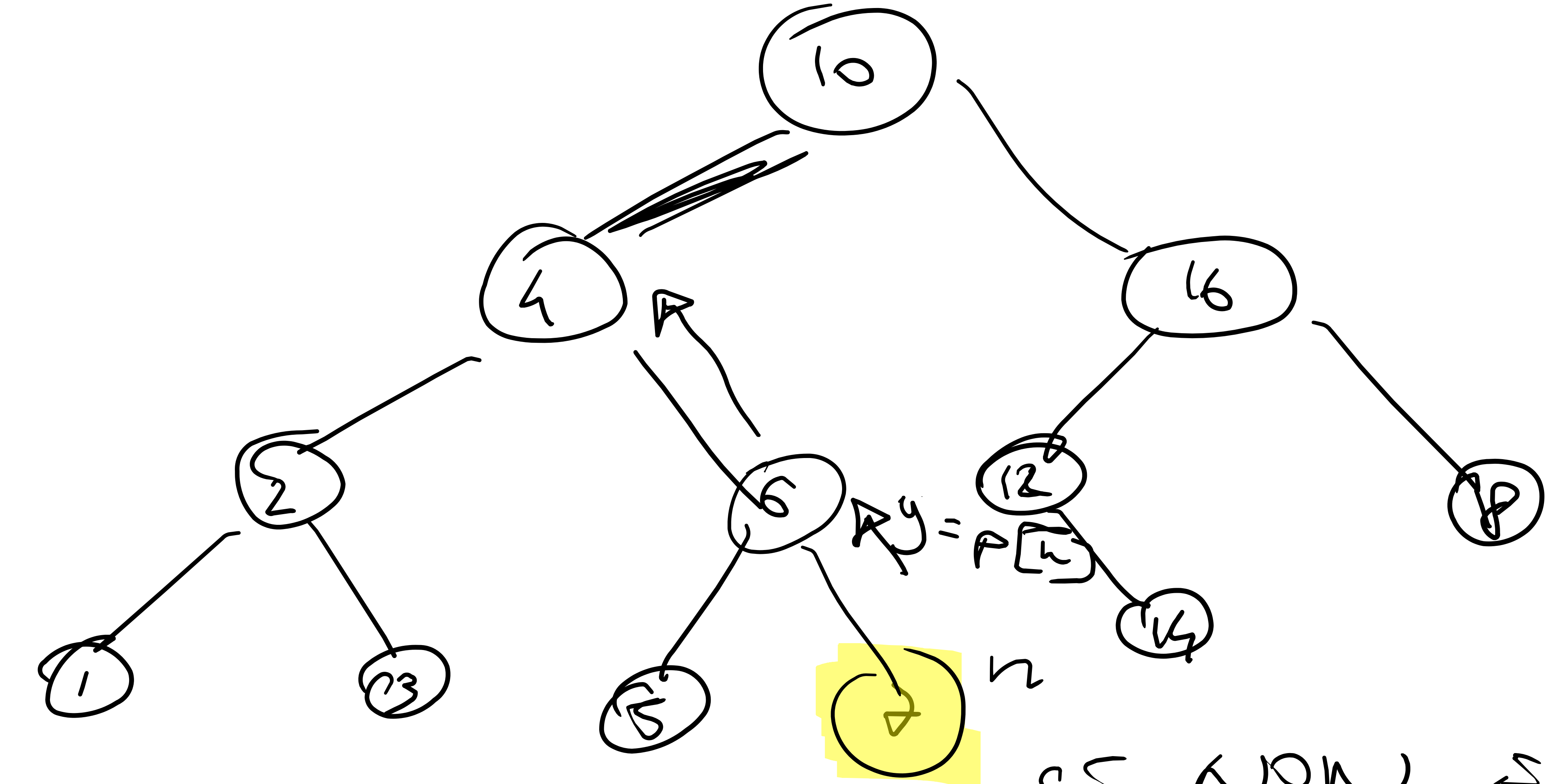

10
4
16
2
6
12
18
1
3
5
7
14
MINIMO
MASSIMO




SUCCESSORE - PREDECESSORE

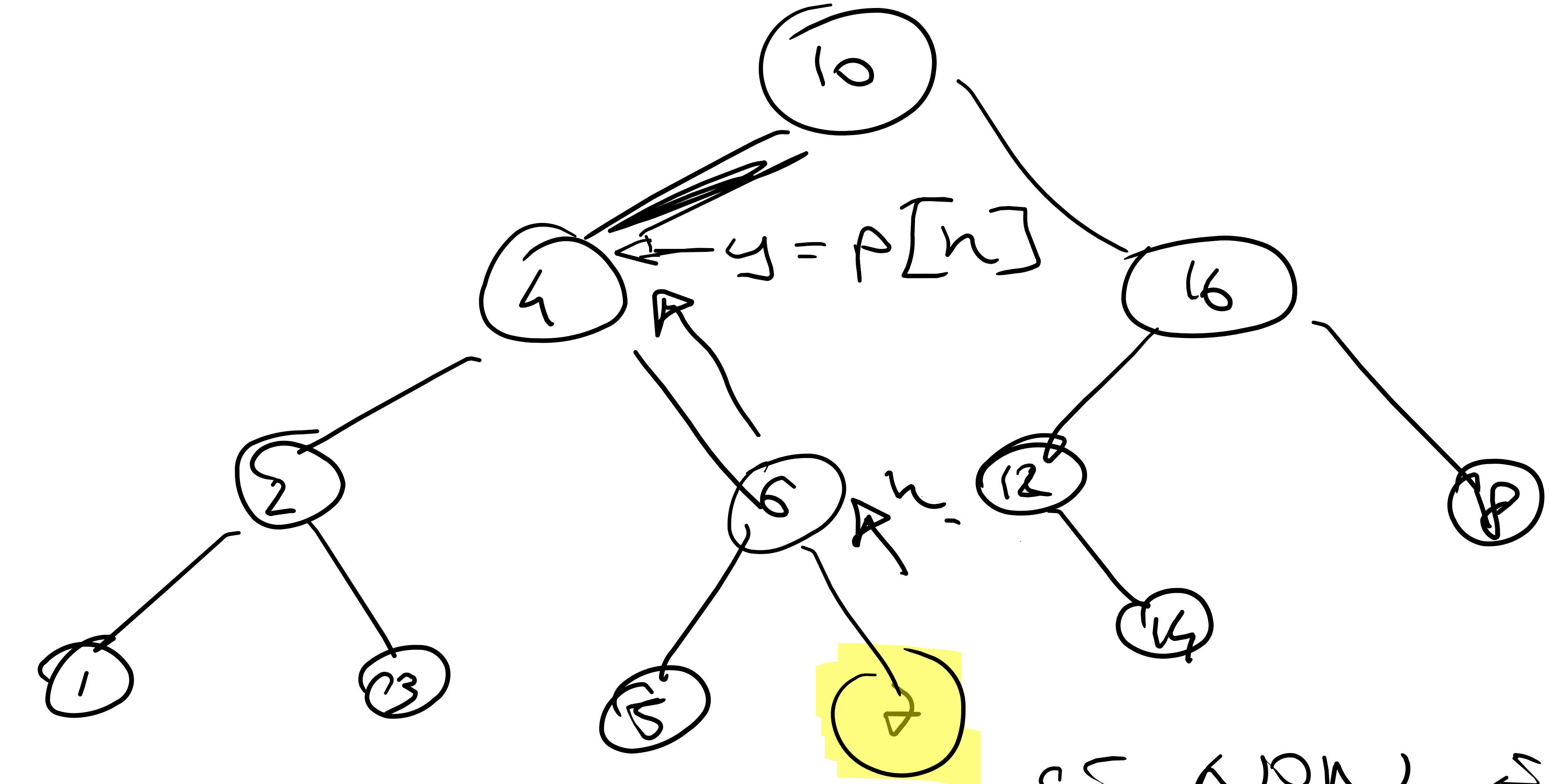



SUCCESSORE (x) → SE ESISTE UN SOTT. DX → IL MIN DEL SOTT. DX

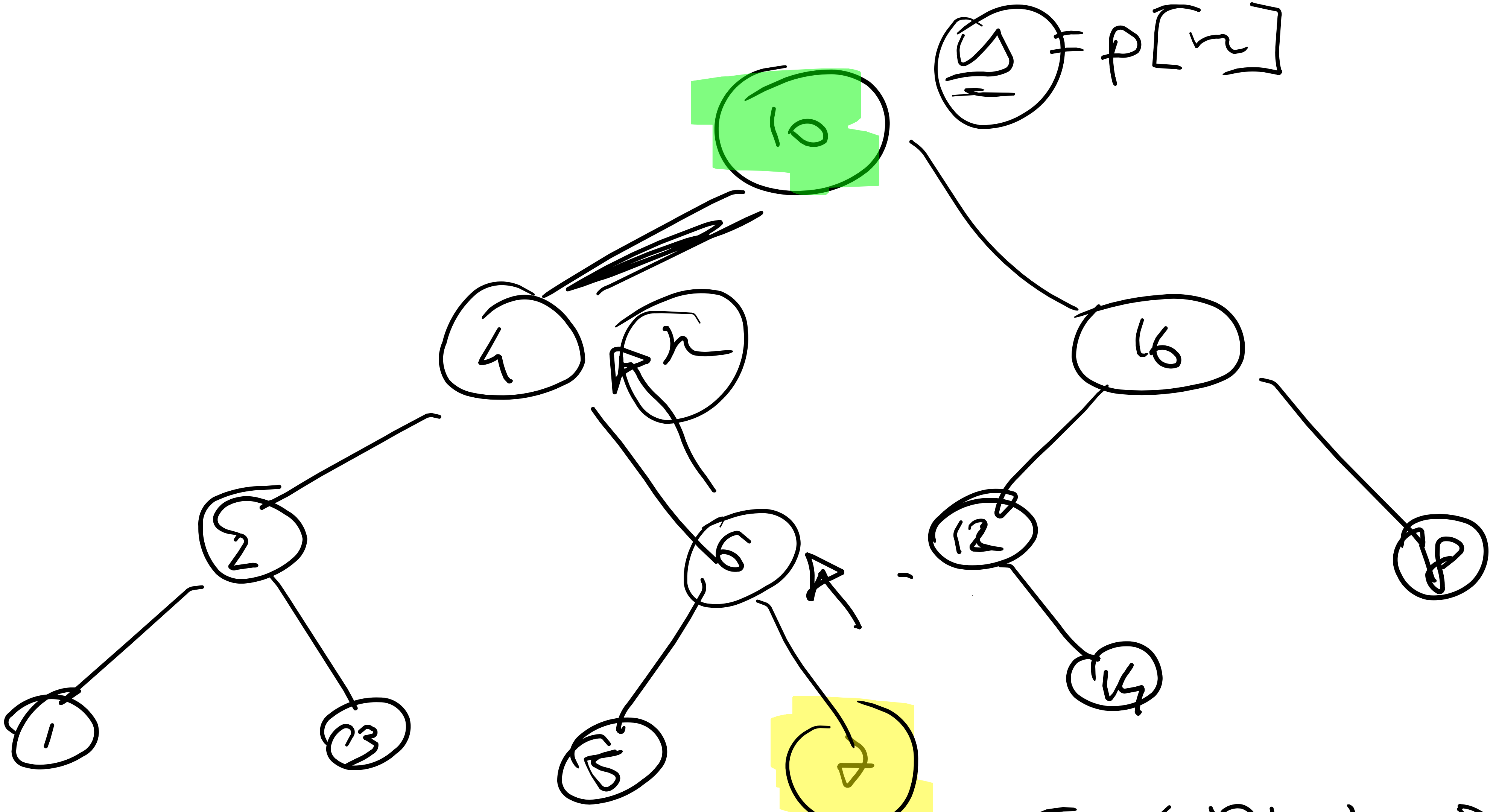



SE NON ESISTE SOTT. DX →

SUCCESSORE (x) →



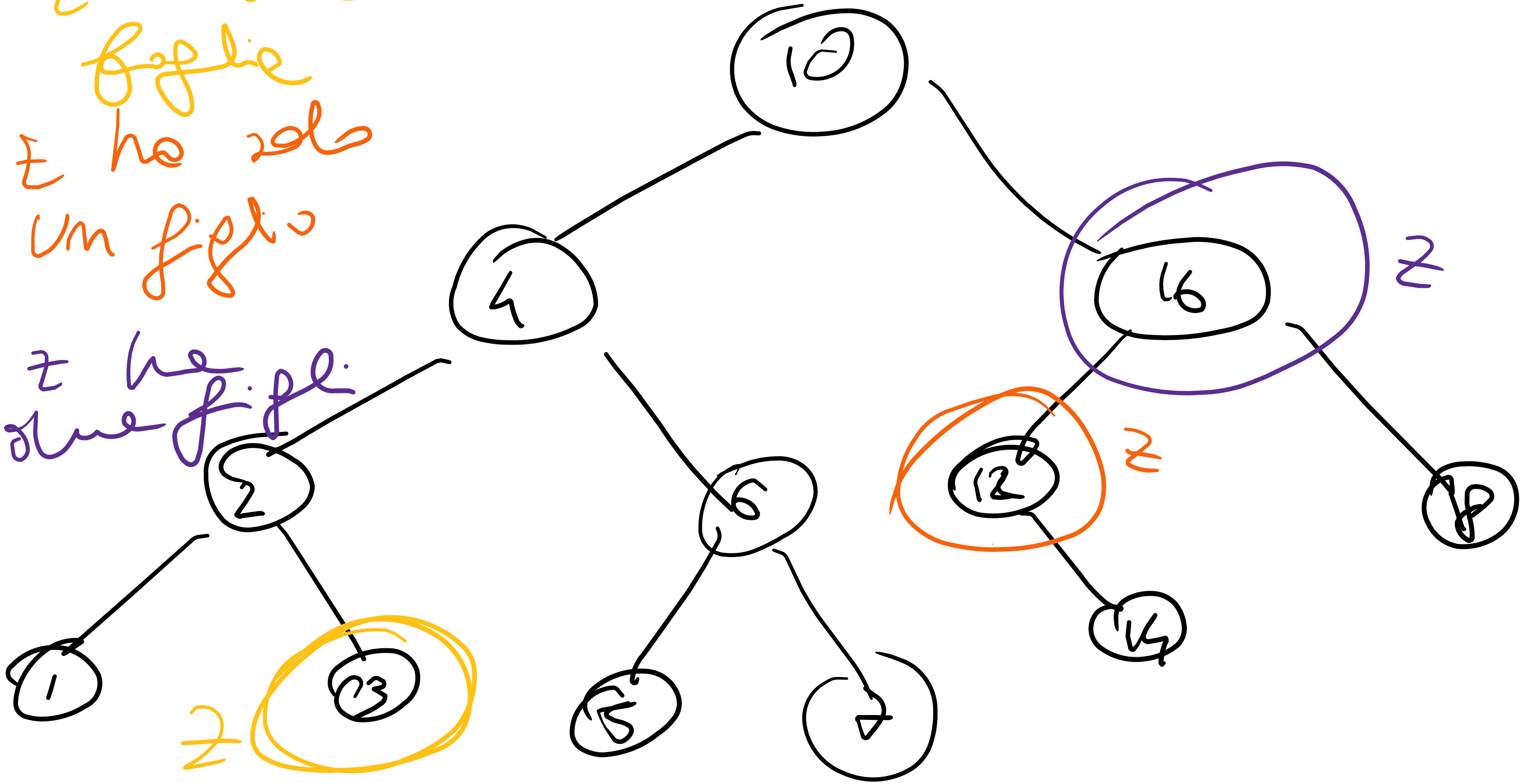

SUCCESSORE(n) → SE NON ESISTE SOTT. DX →



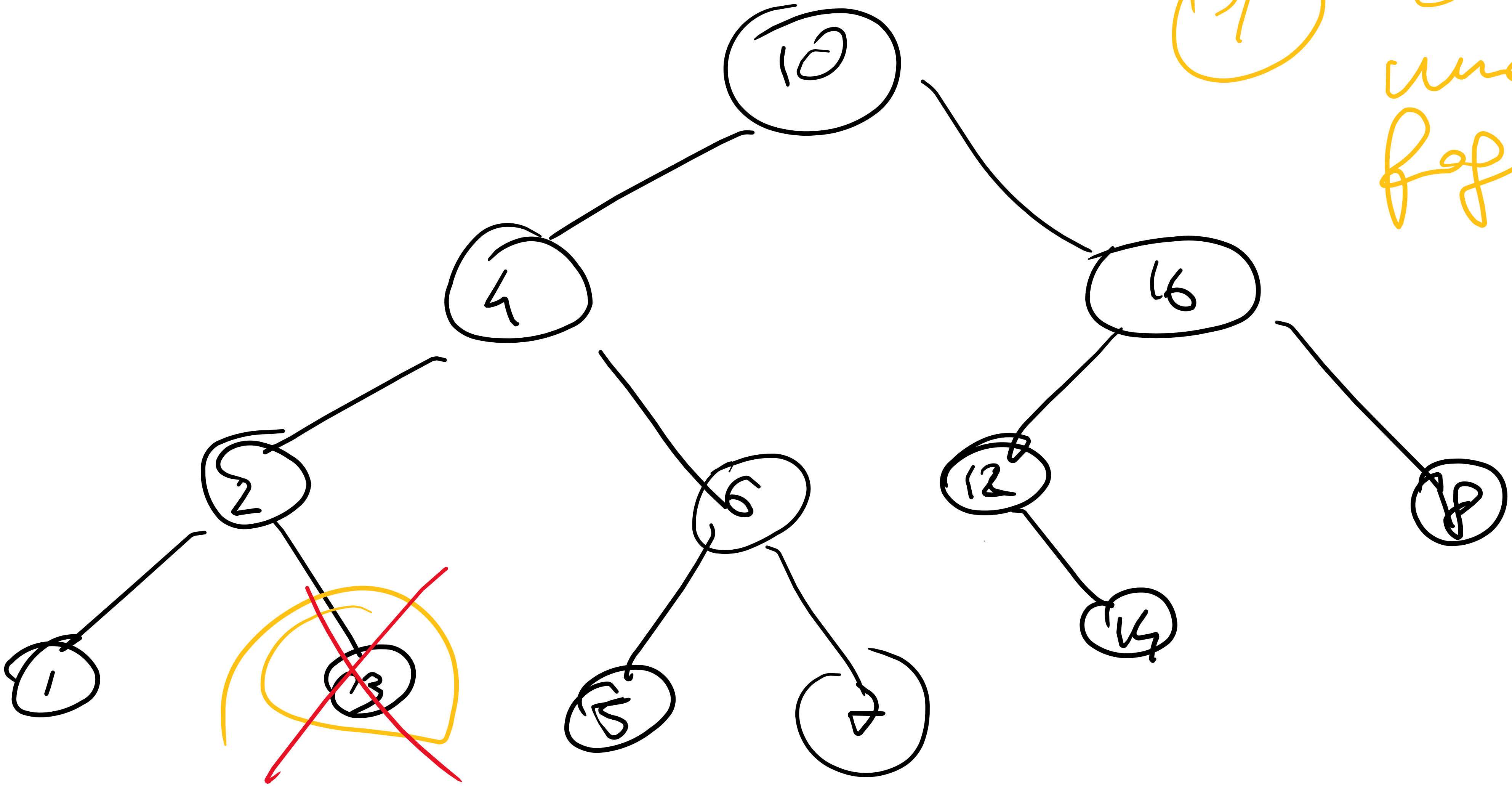

SUCCESSORE (x) → SE NON ESISTE SOTT. DX →

# CANCELLAZIONE (T, z)

1. z è una foglia
2. z ha solo un figlio
3. z ha due figli

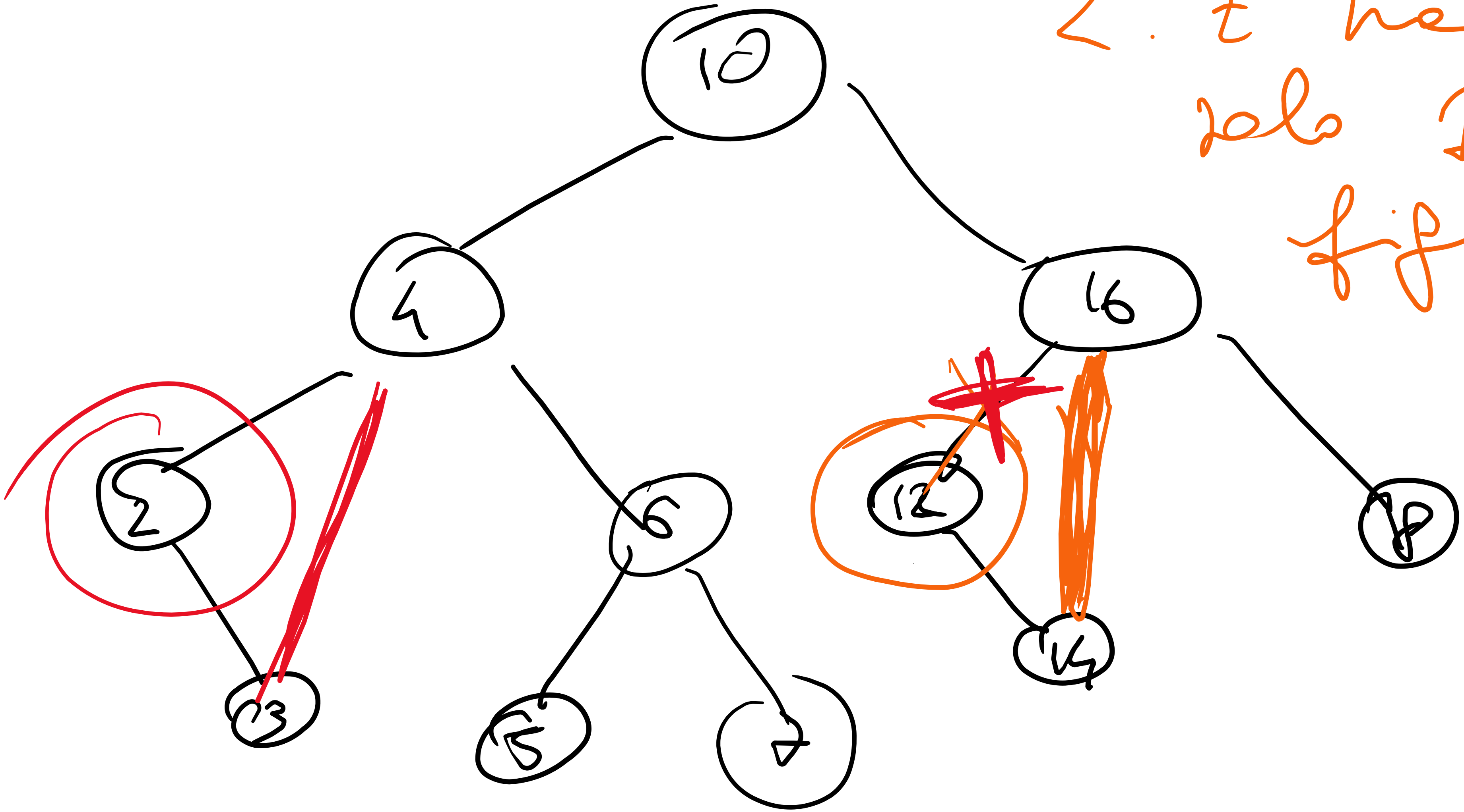


# CANCELLAZIONE




(1) t è una foglia

# CANCELLAZIONE




2. t ha solo 1 figlio

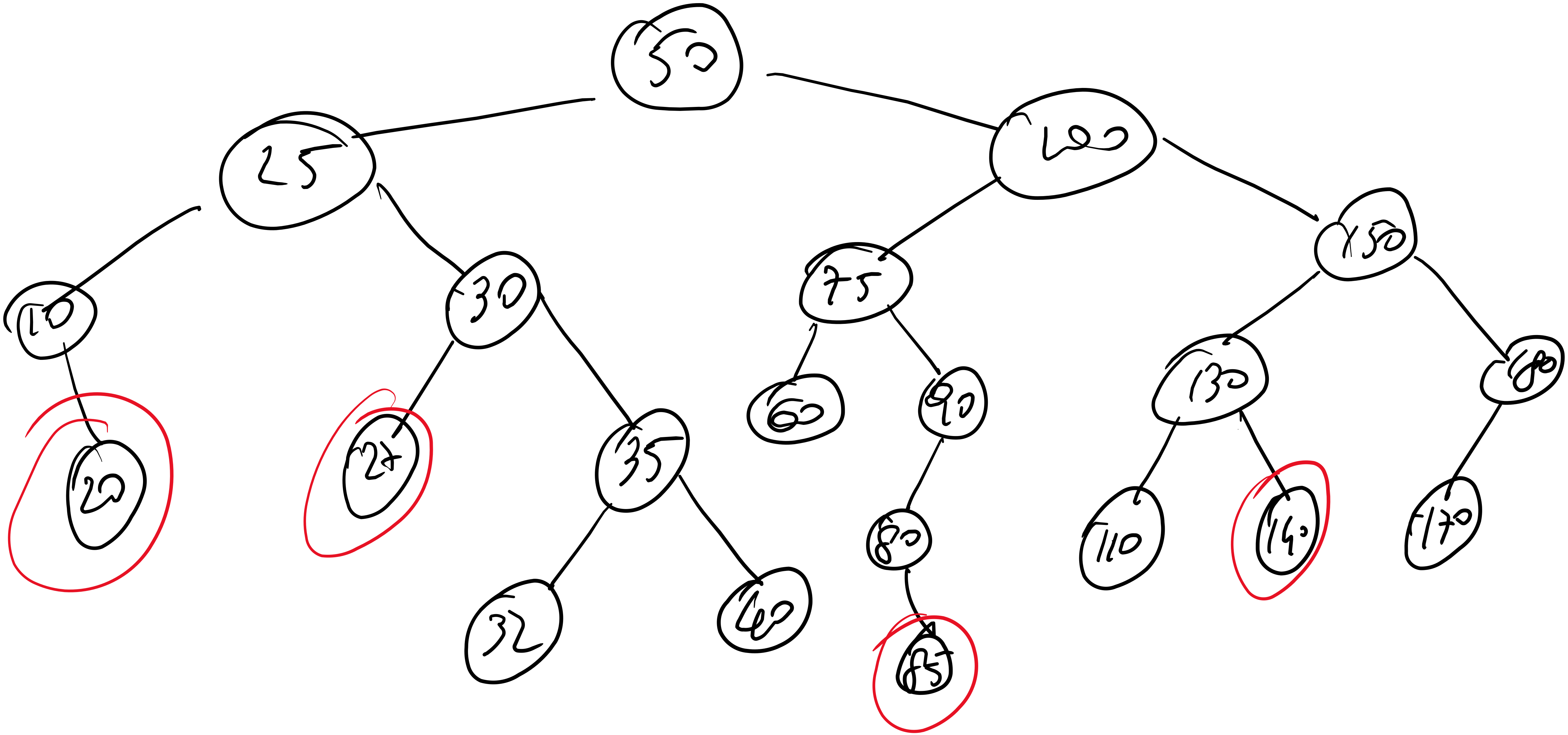
# CANCELLAZIONE

1. La chiave da cancellare si trova in una foglia

2. La chiave da cancellare si trova in un nodo con un solo figlio

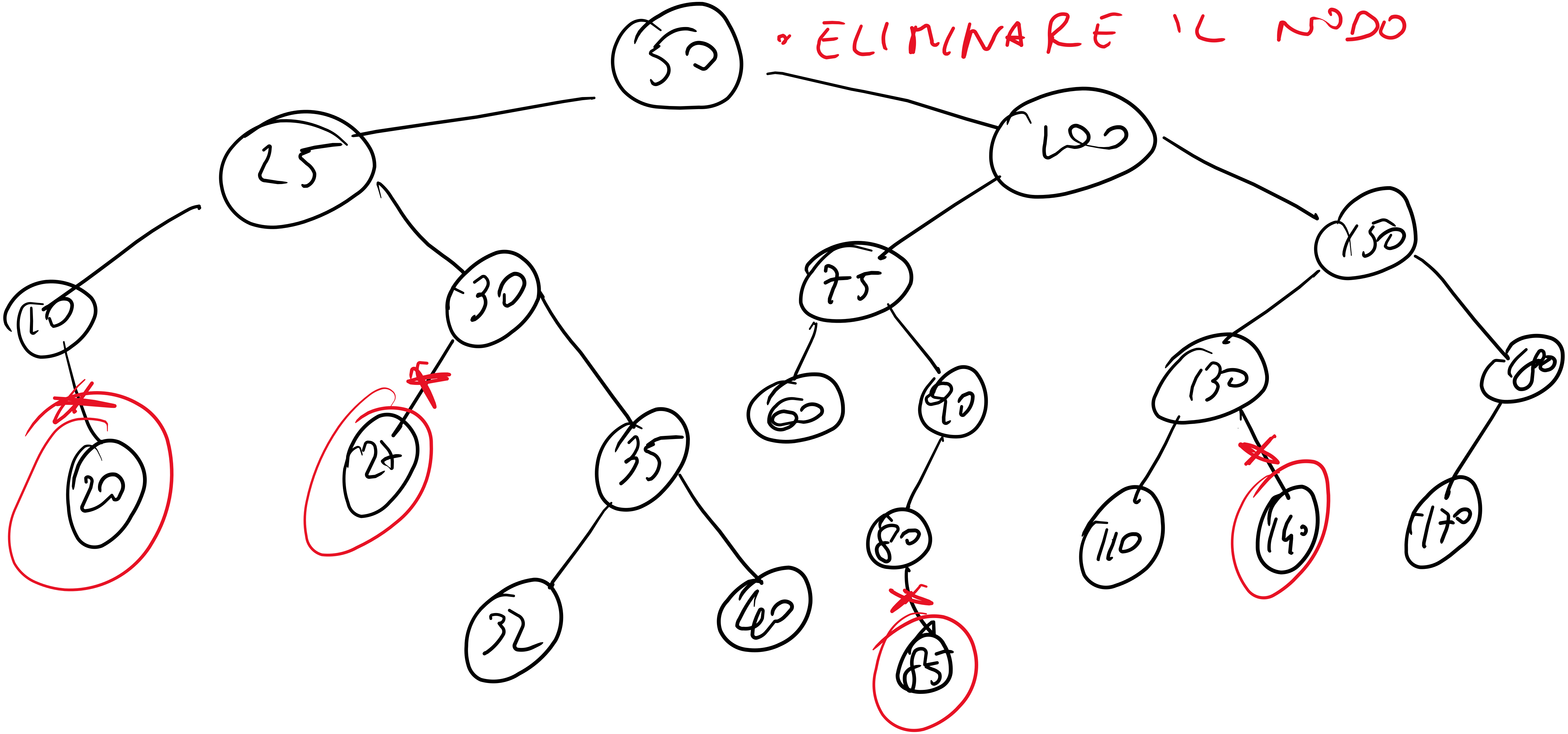
3. La chiave da cancellare si trova in un nodo con due figli

# CASO 1



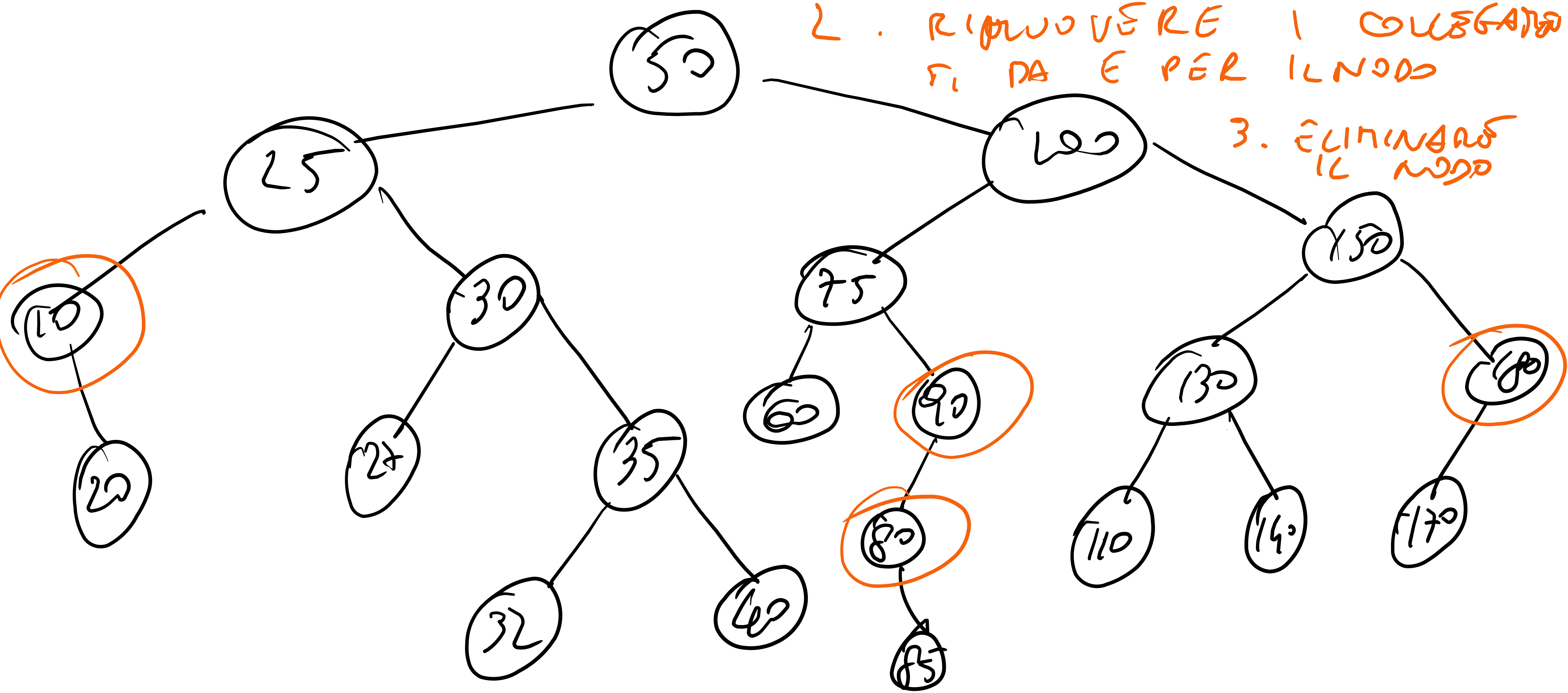
CASO 1

- RIMUOVERE IL COLLEGAMENTO TRA IL GENITORE E LA FOGLIA
- ELIMINARE IL NODO

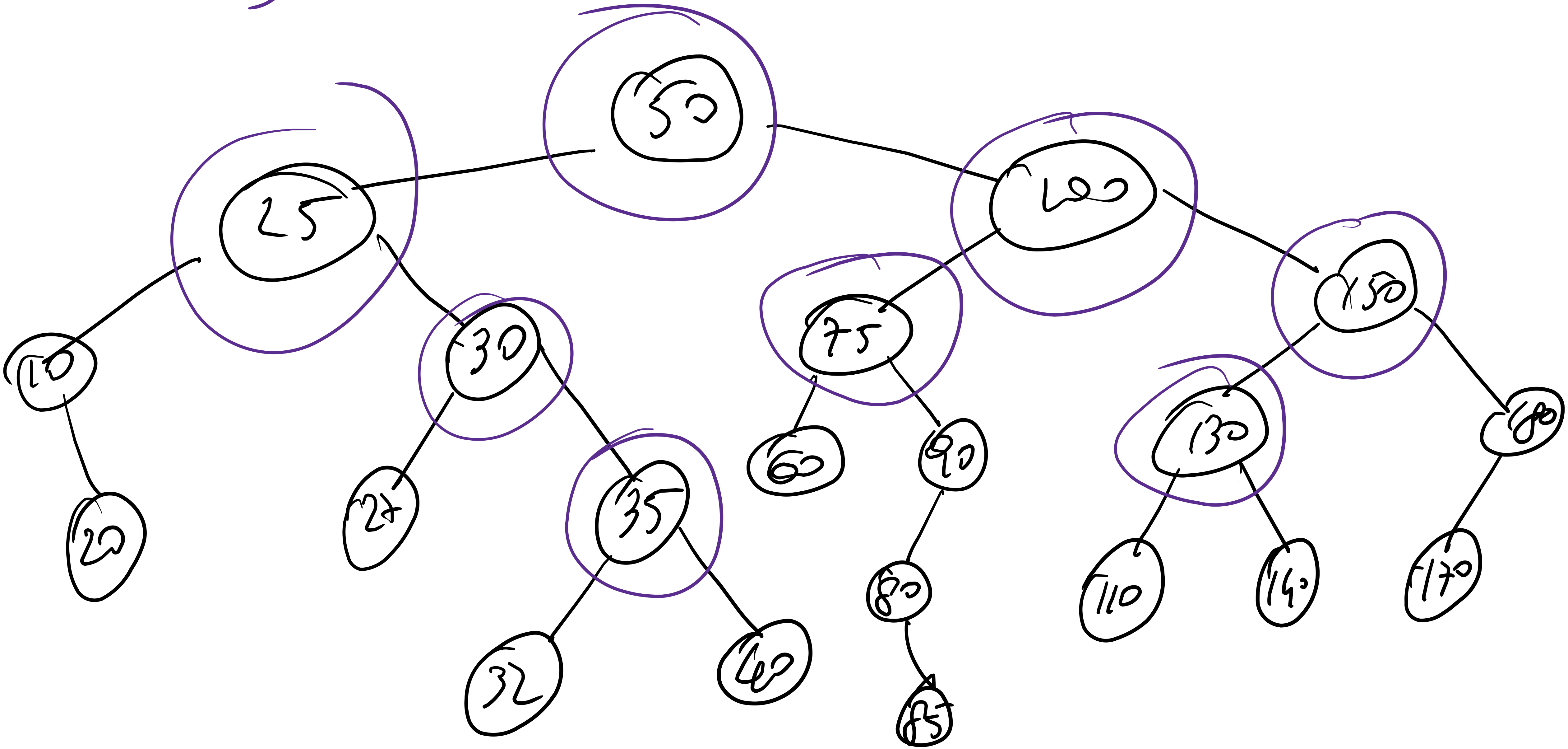


# CASO 2

1. COLLEGARE L'UNICO FIGLIO AL GENITORE
2. RIMUOVERE I COLLEGAMENTI DA E PER IL NODO
3. ELIMINARE IL NODO

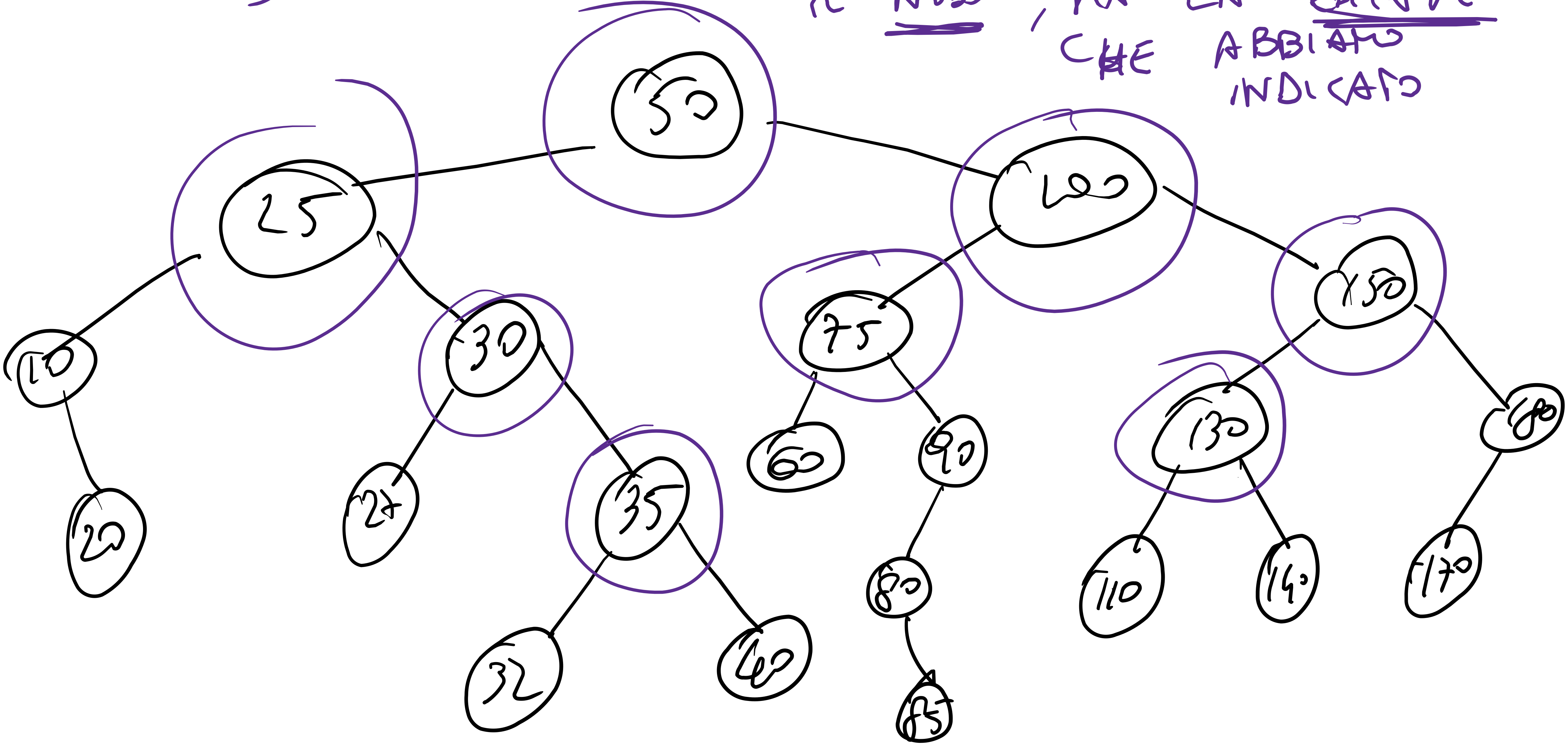


# CASO 3

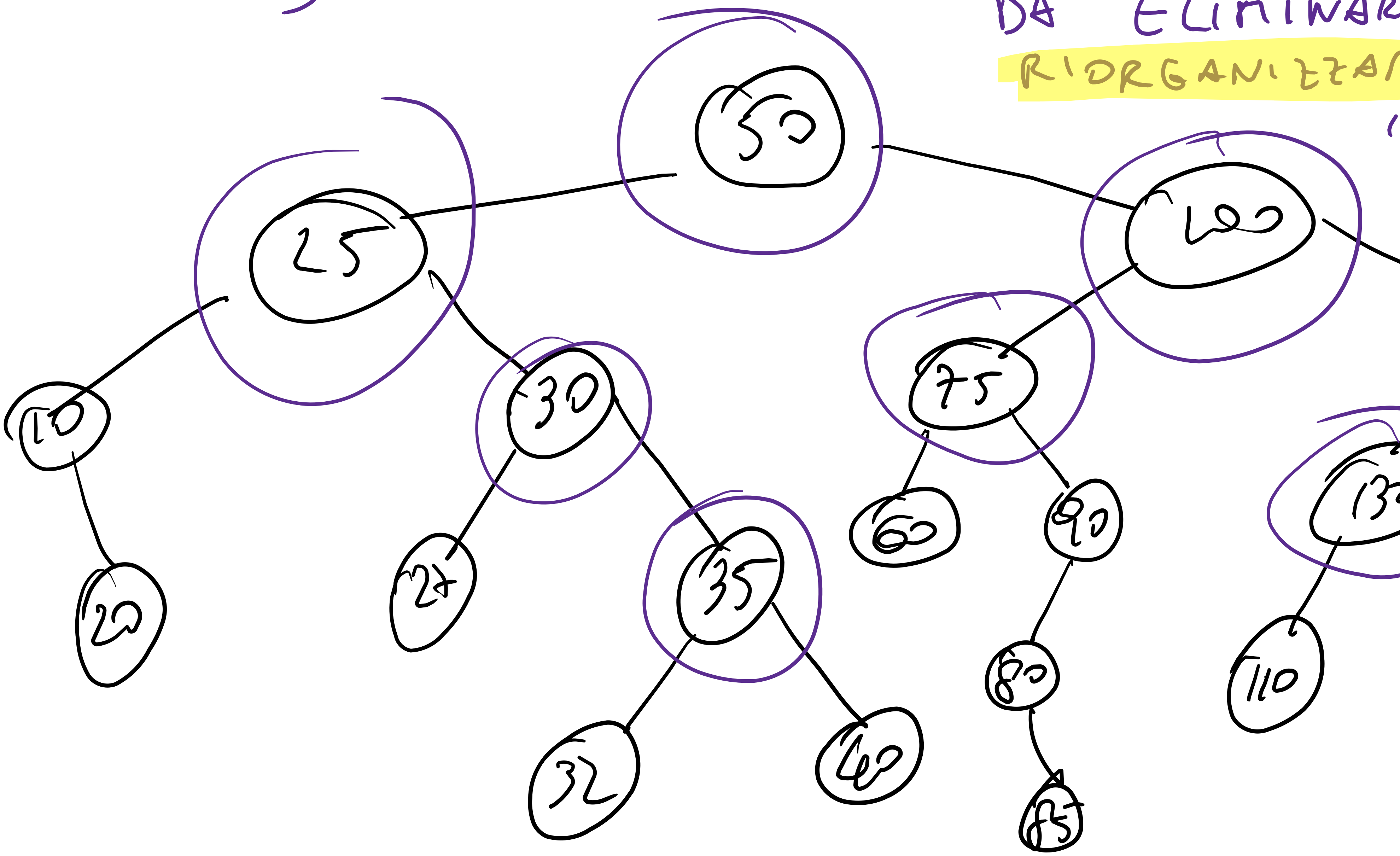


CASO 3

- NON VOGLIAMO RIMUOVERE IL NODO, MA LA CHIAVE CHE ABBIAMO INDICATO



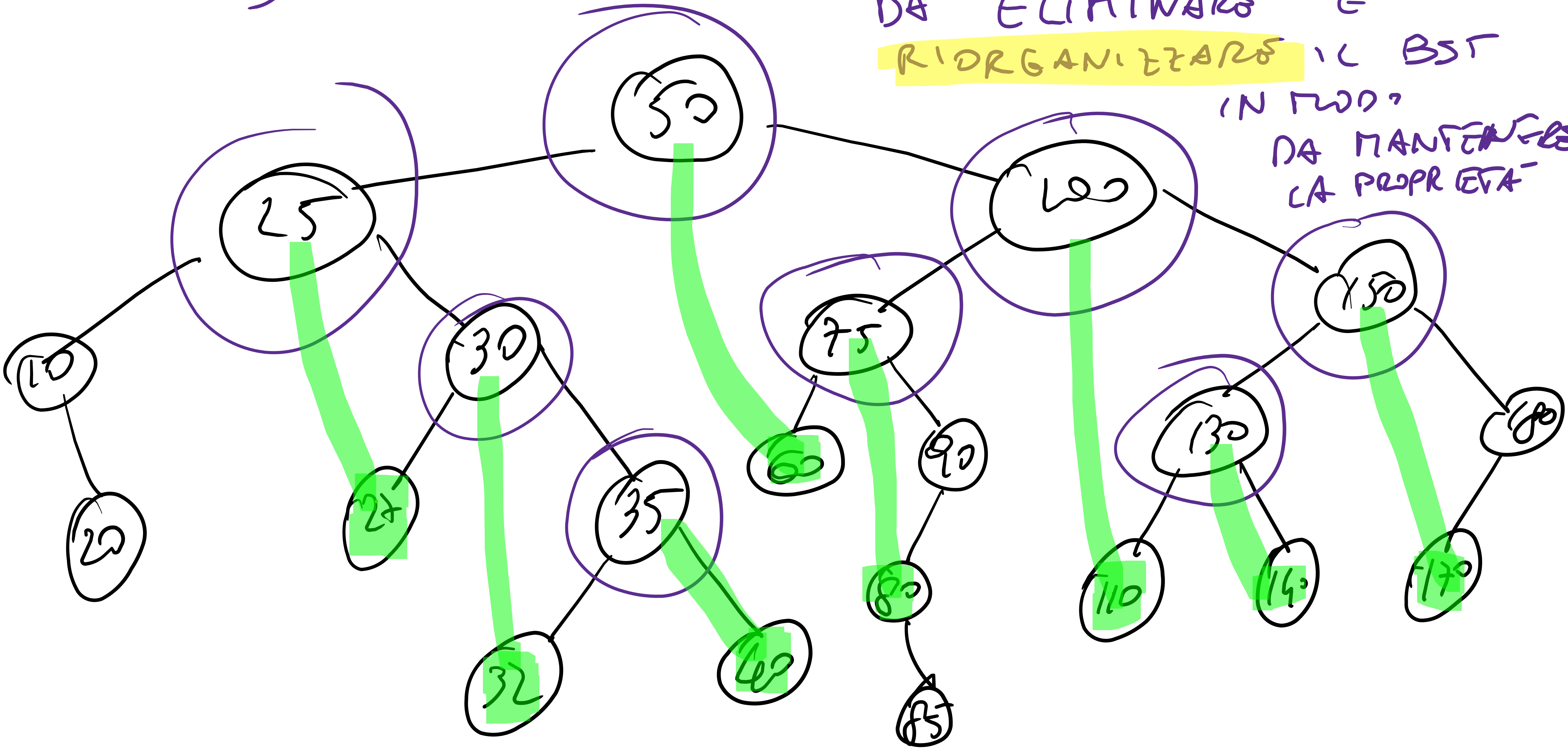
CASO 3




- SOSTITUIRE IL VALORE DA ELIMINARE E RIORGANIZZARE IL BST IN MODO DA MANTENERE LA PROPRIETÀ

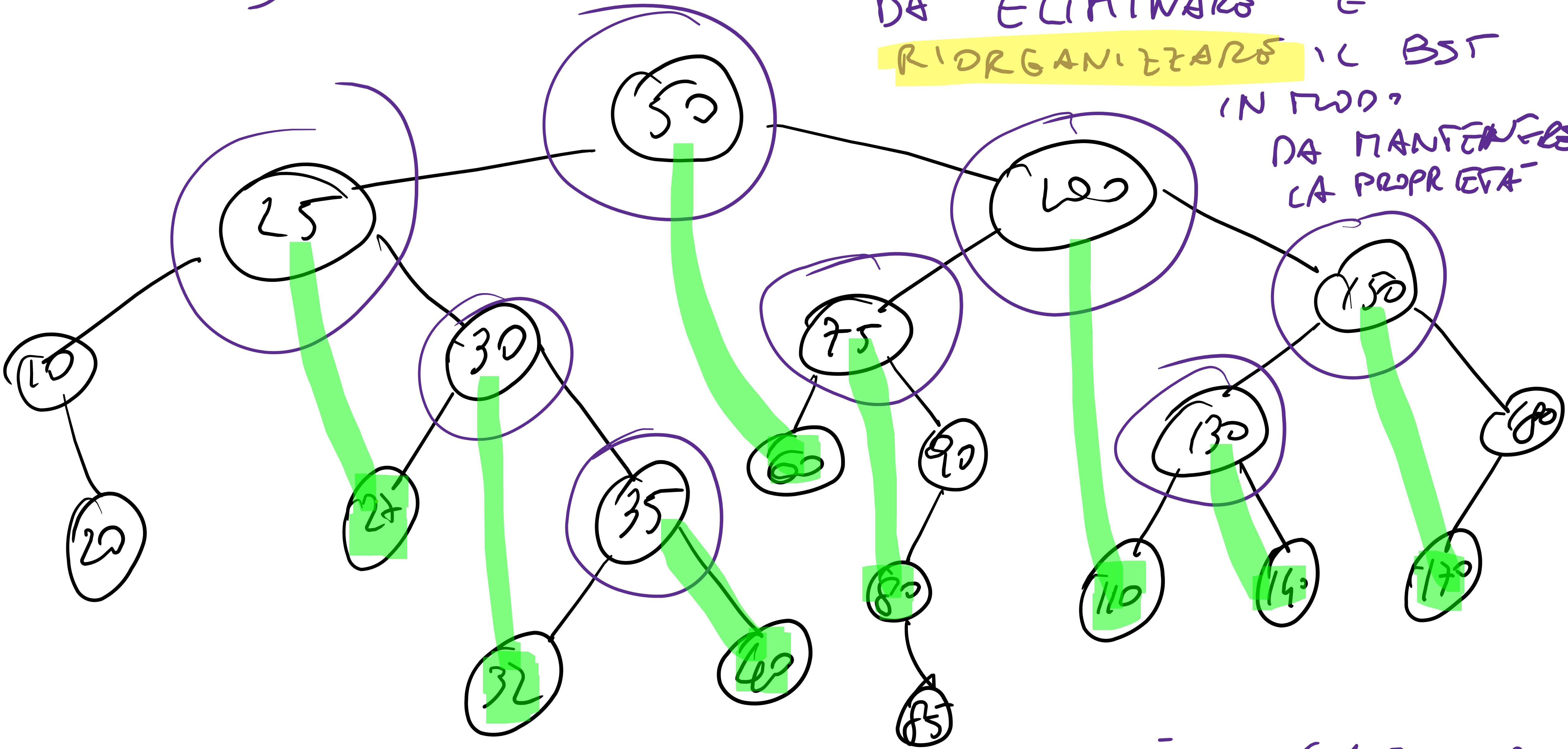
# CASO 3

• SOSTITUIRE IL VALORE DA ELIMINARE E RIORGANIZZARE IL BST IN MODO DA MANTENERE LA PROPRIETÀ



CASO 3

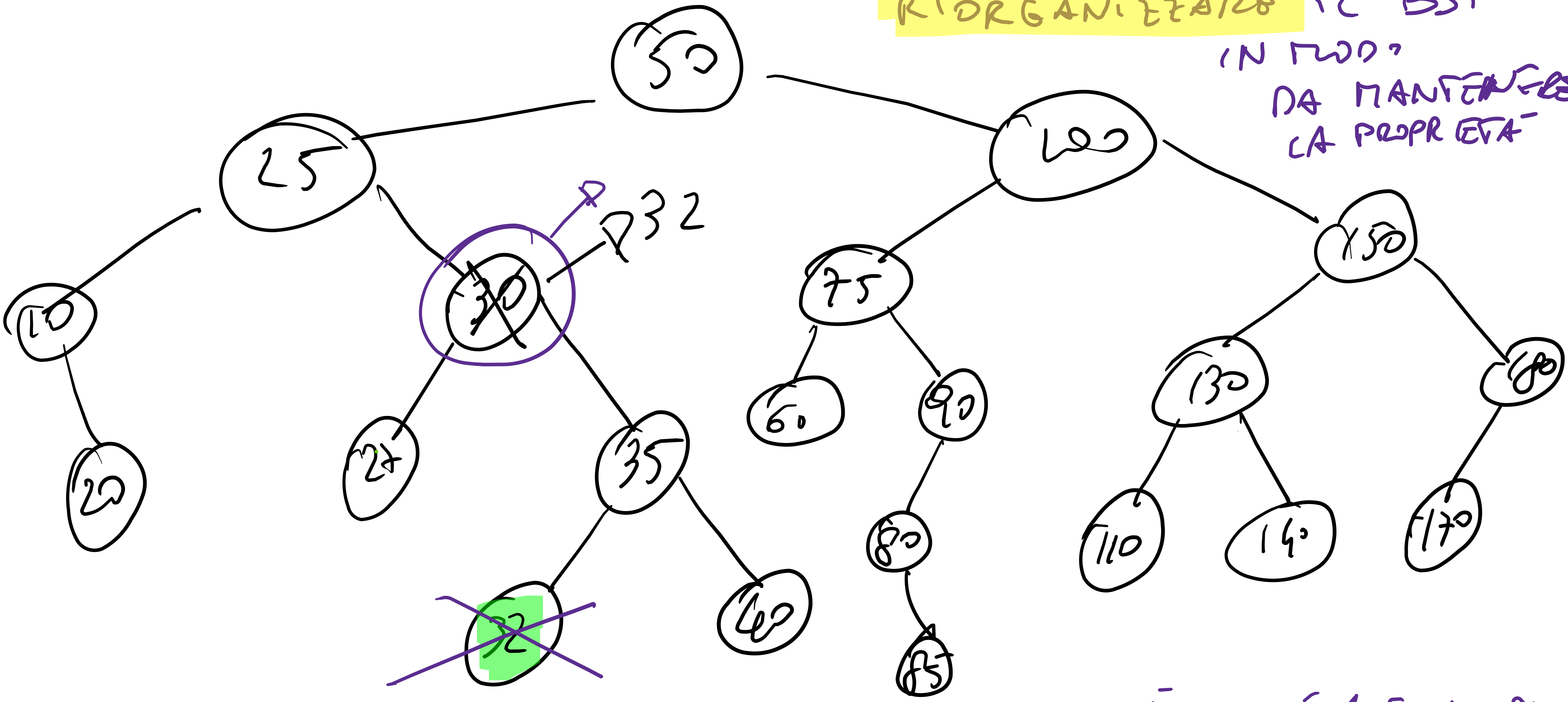
· SOSTITUIRE IL VALORE DA ELIMINARE E RIORGANIZZARE IL BST IN MODO DA MANTENERE LA PROPRIETÀ




IL SUCCESSORE DI UN NODO CON 2 FIGLI AVRÀ AL PIÙ 1 FIGLIO DX

CASO 3

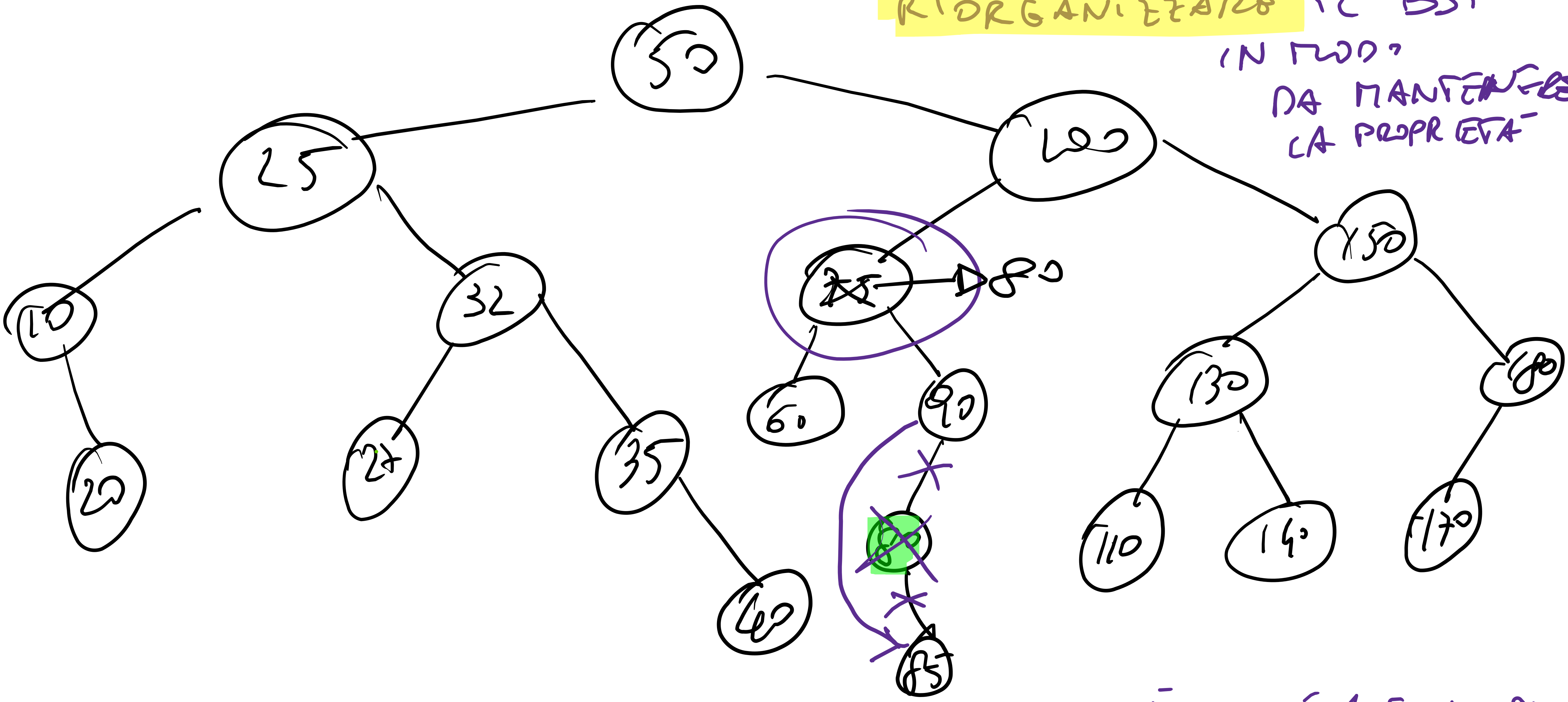
• SOSTITUIRE IL VALORE DA ELIMINARE E RIORGANIZZARE IL BST IN MODO DA MANTENERE LA PROPRIETÀ



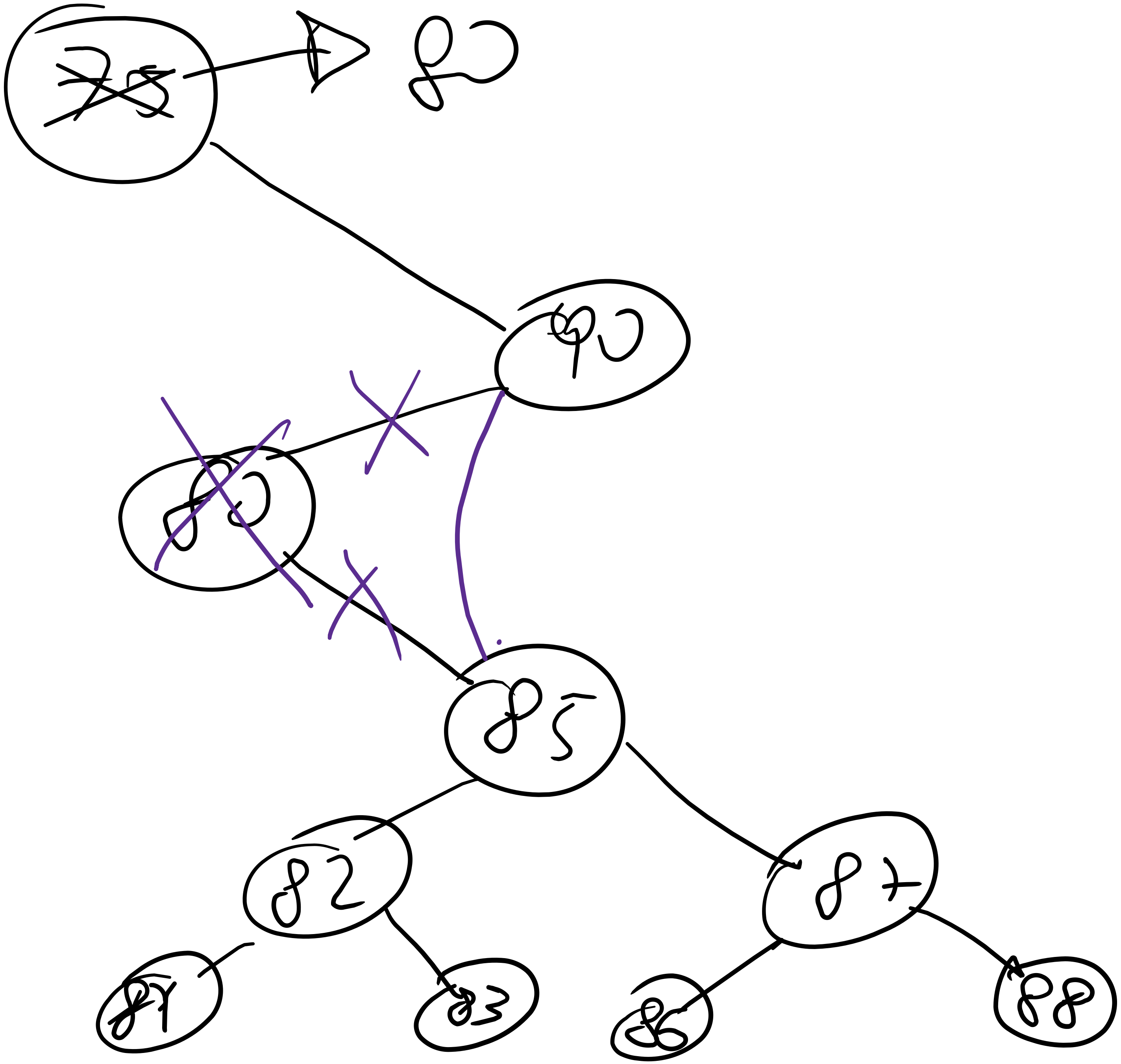

IL SUCCESSORE DI UN NODO CON 2 FIGLI AVRÀ AL PIÙ 1 FIGLIO DX

# CASO 3

· SOSTITUIRE IL VALORE DA ELIMINARE E RIORGANIZZARE IL BST IN MODO DA MANTENERE LA PROPRIETÀ



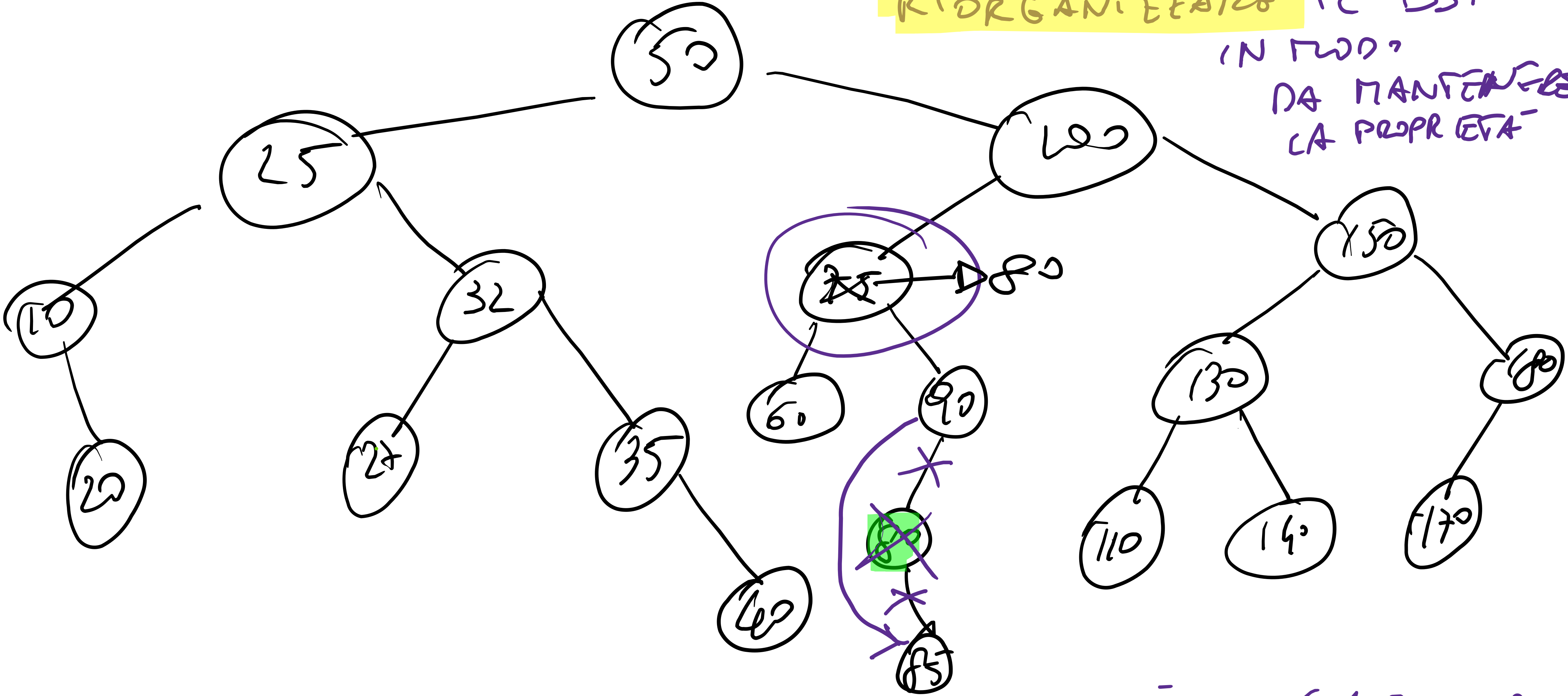

IL SUCCESSORE DI UN NODO CON 2 FIGLI AVRÀ AL PIÙ 1 FIGLIO DX



75
80
90
80
85
82
87
81
83
86
88

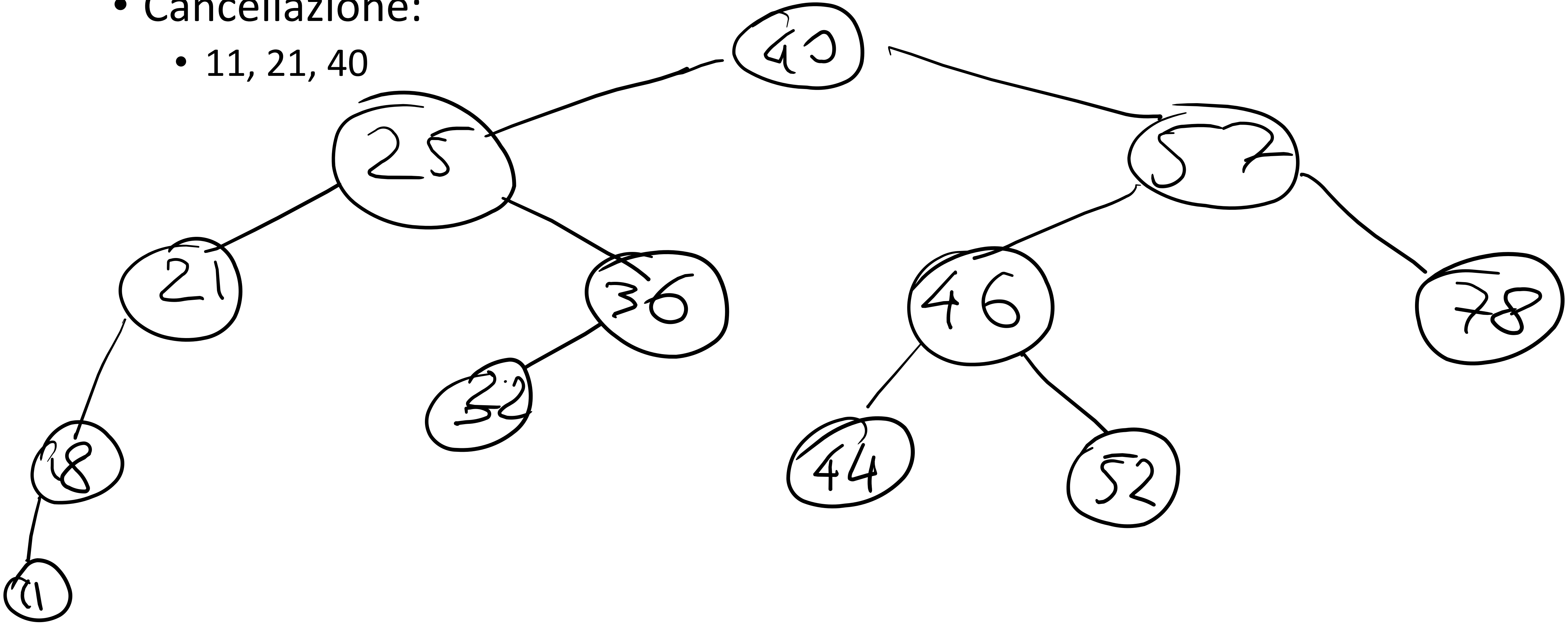
CASO 3

· SOSTITUIRE IL VALORE DA ELIMINARE E RIORGANIZZARE IL BST IN MODO DA MANTENERE LA PROPRIETÀ

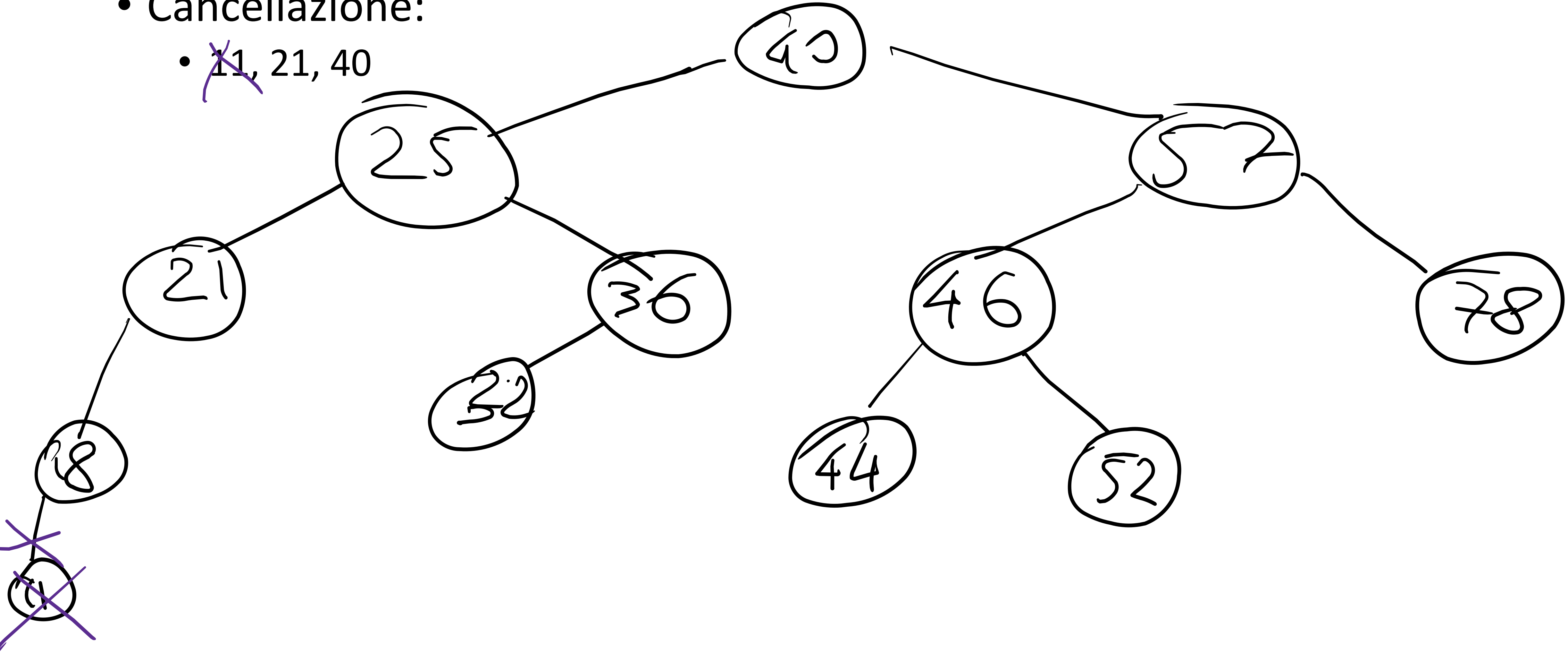



IL SUCCESSORE DI UN NODO CON 2 FIGLI AVRÀ AL PIÙ 1 FIGLIO DX

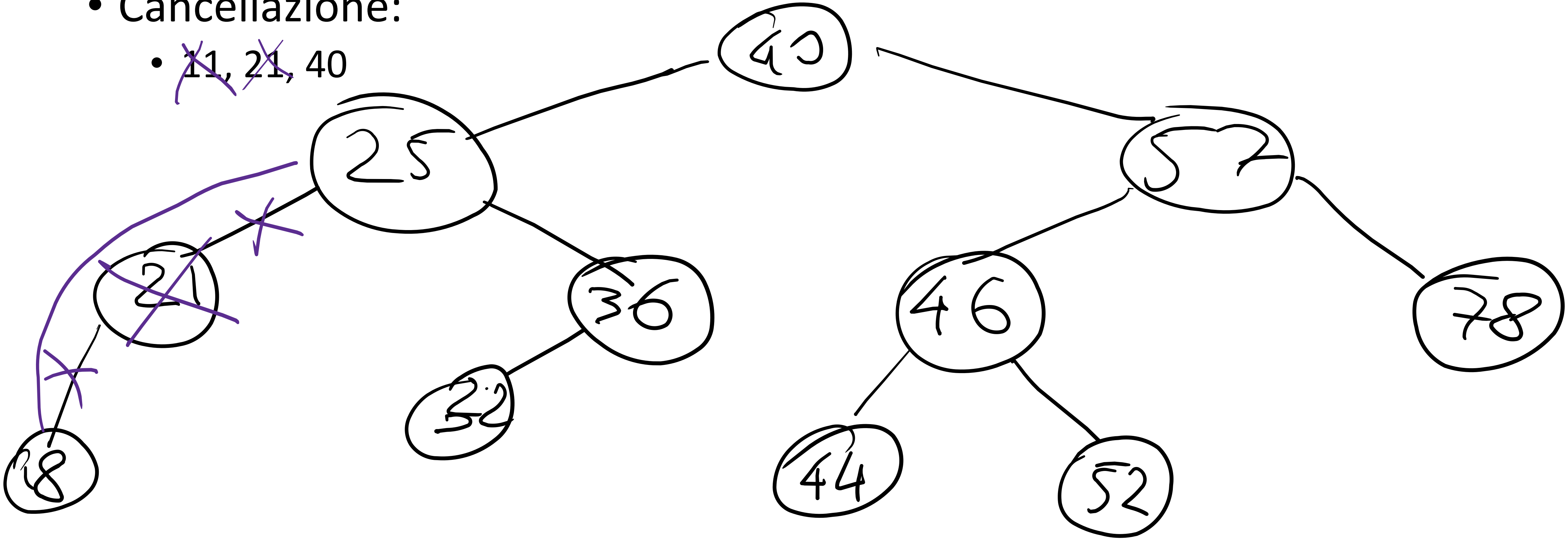
- Inserimento:
  - 40, 25, 21, 18, 57, 36, 46, 32, 78, 52, 11, 44
- Cancellazione:
  - 11, 21, 40



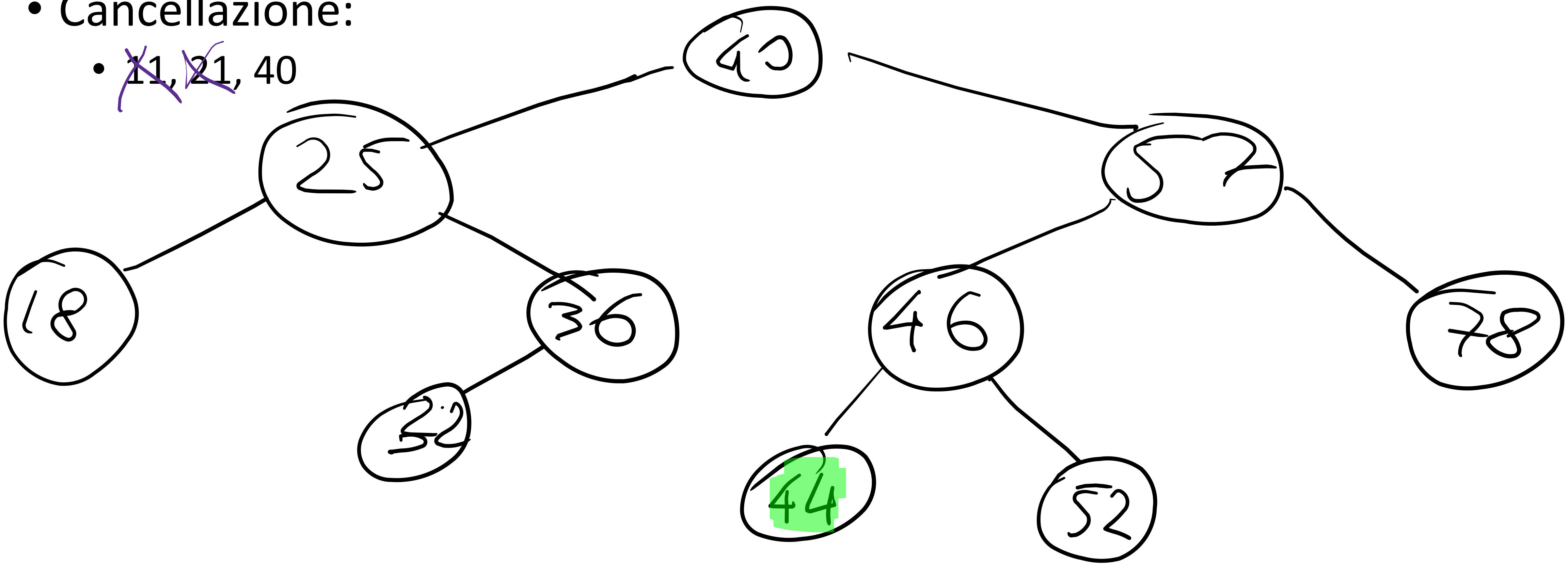
- Inserimento:
  - 40, 25, 21, 18, 57, 36, 46, 32, 78, 52, 11, 44
- Cancellazione:
  - ~~11~~, 21, 40



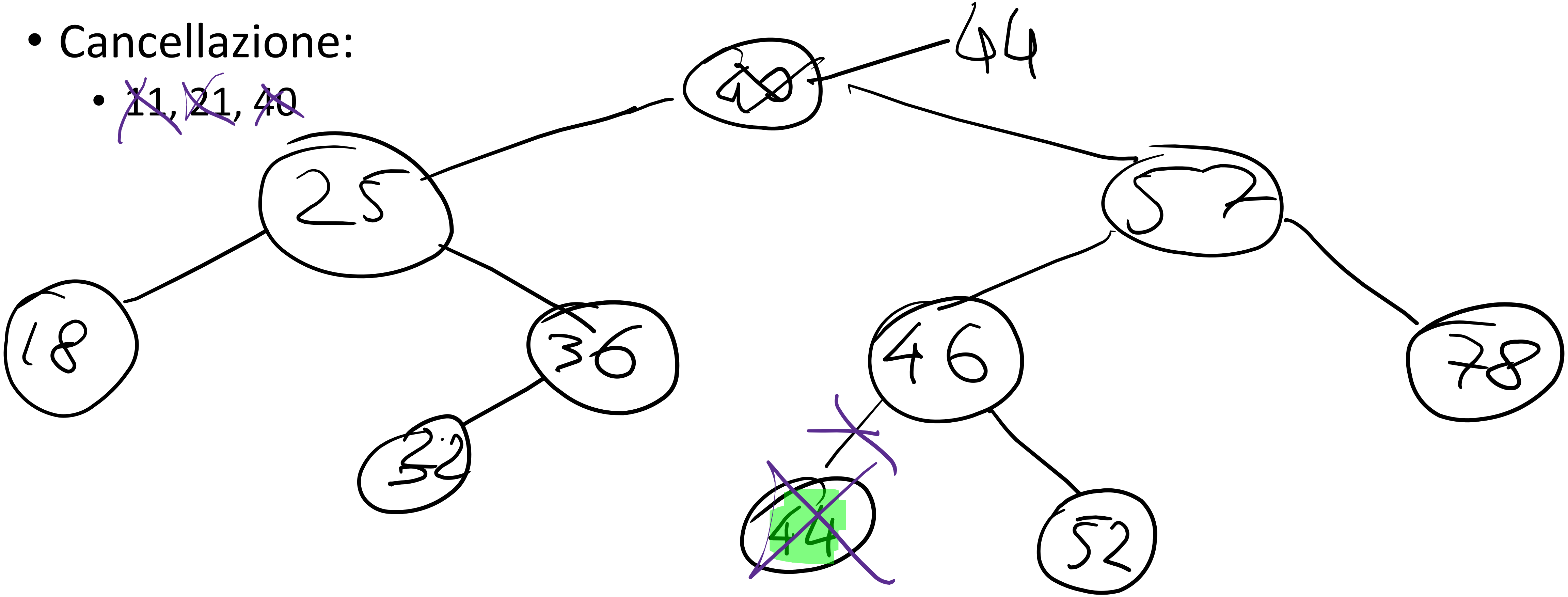
- Inserimento:
  - 40, 25, 21, 18, 57, 36, 46, 32, 78, 52, 11, 44
- Cancellazione:
  - ~~11~~, ~~21~~, 40



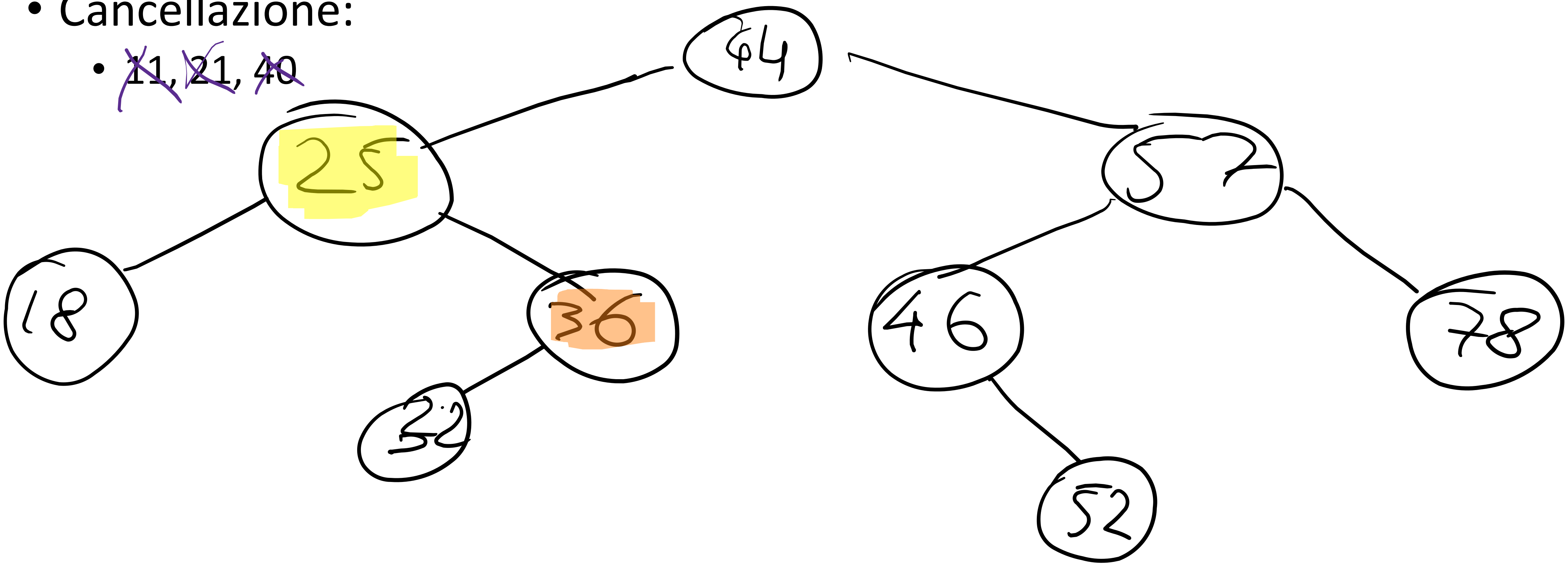
- Inserimento:
    - 40, 25, 21, 18, 57, 36, 46, 32, 78, 52, 11, 44
- Cancellazione:
    - ~~11~~, ~~21~~, 40

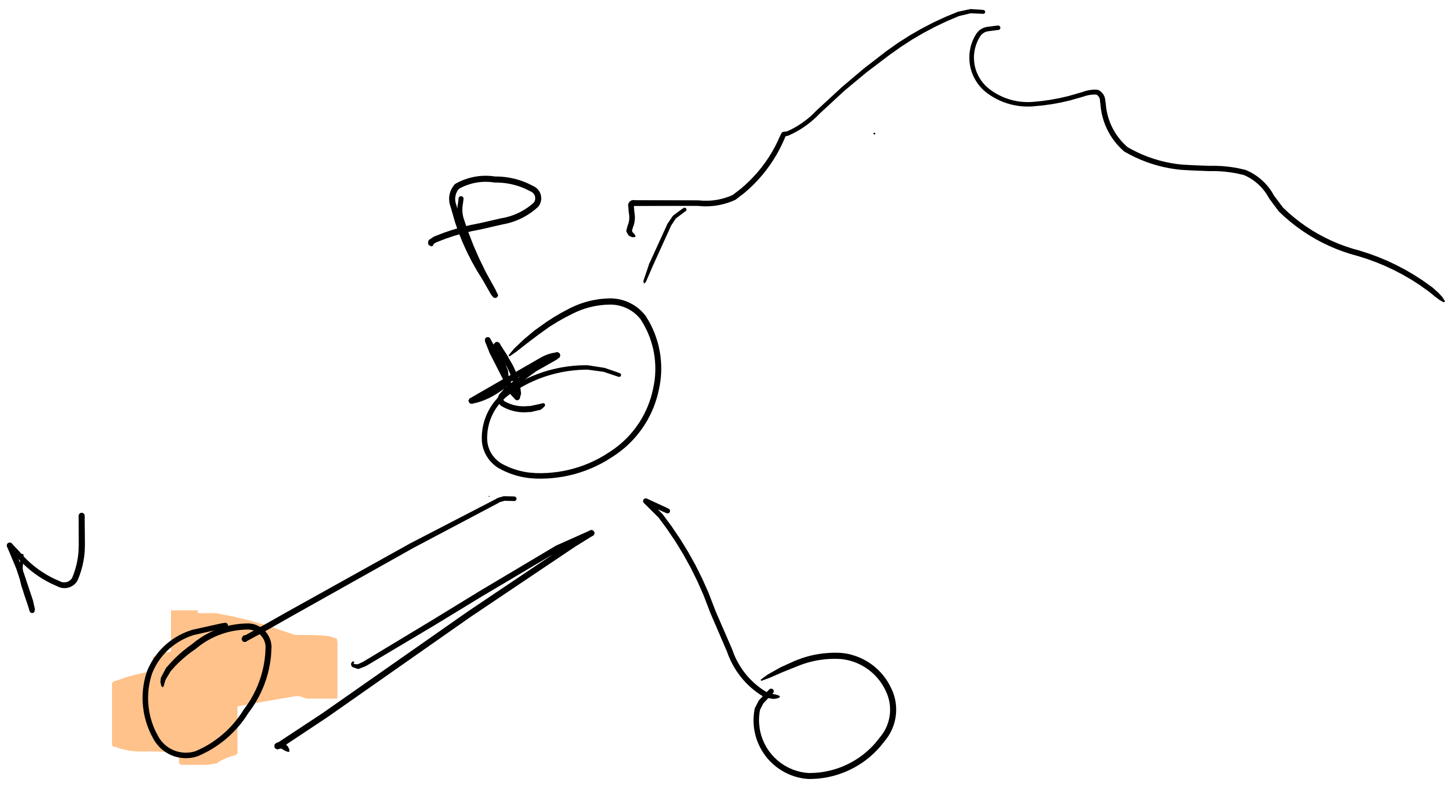


- Inserimento:
  - 40, 25, 21, 18, 57, 36, 46, 32, 78, 52, 11, 44
- Cancellazione:
  - ~~11~~, ~~21~~, ~~40~~



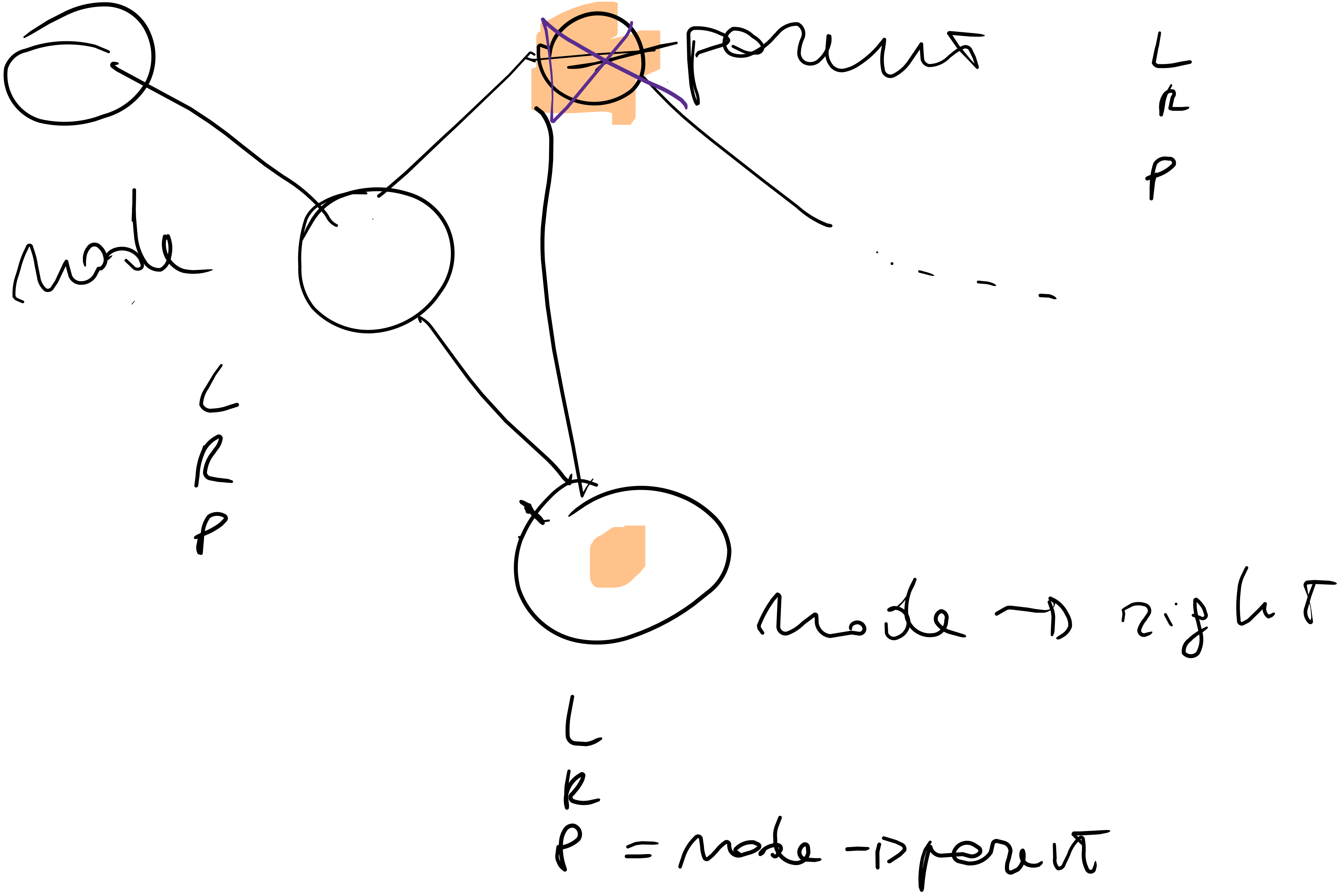
- Inserimento:
  - 40, 25, 21, 18, 57, 36, 46, 32, 78, 52, 11, 44
- Cancellazione:
  - ~~11~~, ~~21~~, ~~40~~



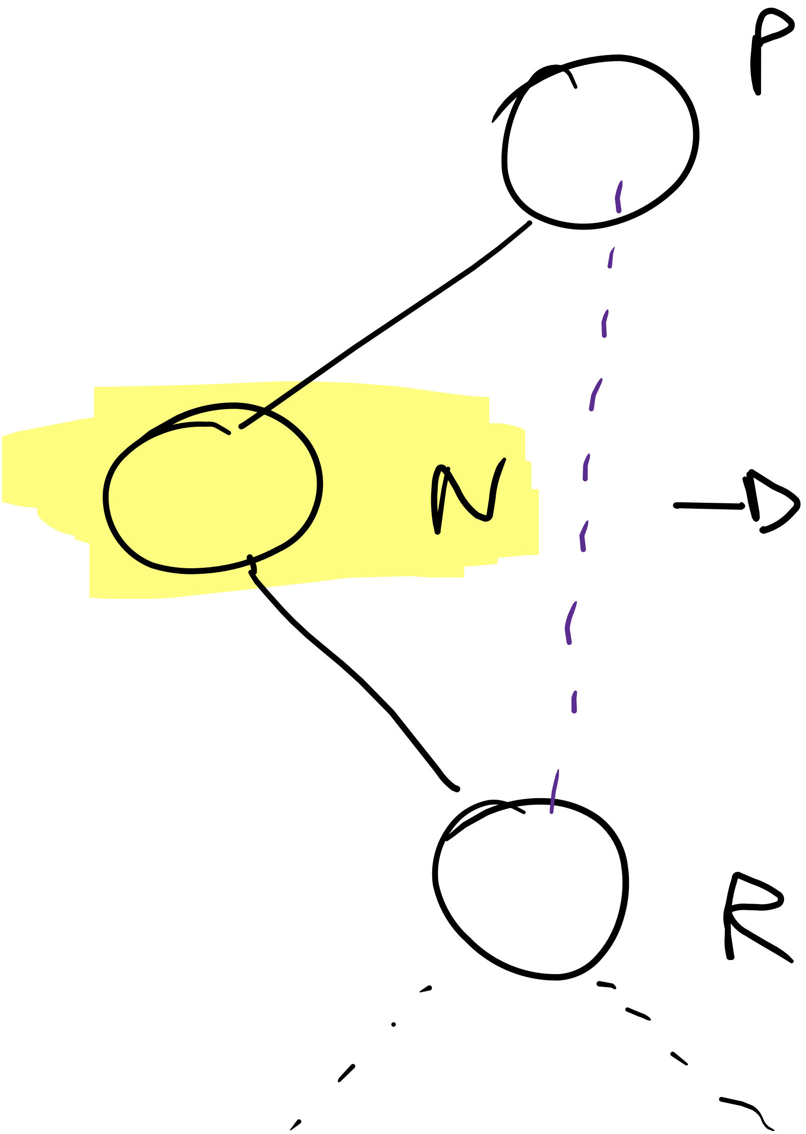



left = NULL
right = NULL
parent = P

node -> P -> left = NULL



parent
L
R
P
node
L
R
P
node -> right
L
R
P = node -> parent

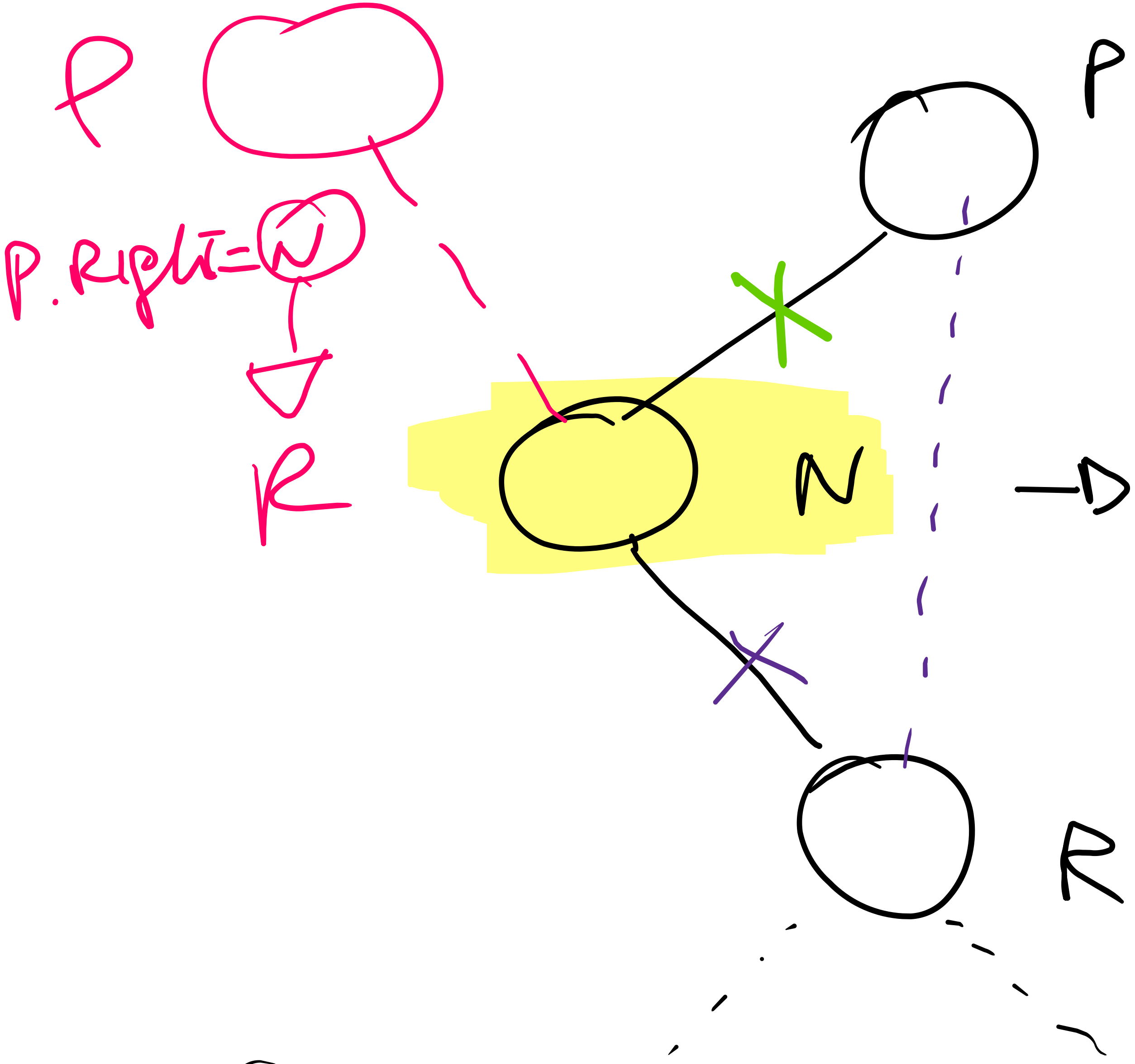



P → P.left
P.right

N → N.left
N.right = R
N.parent = P

R → R.parent = N

1. R parent = P → N.right.parent = P
N.right.parent = N.parent




P → P.left = N → R

P.right

N → N.left

N.right = R

N.parent = P

R → R.parent = N

1. R parent = P → N.right.parent = P

N.right.parent = N.parent