DIZIONARIO

INSIEME DI COPPIE (CHIAVE, ELEMENTO)

INSERT (K, V) → AGGIUNGE AL DIZIONARIO LA COPPIA (K, V)

DELETE (K) → RIMUOVE LA COPPIA CON CHIAVE K

SEARCH (K) → RESTITUISCE L'ELEM. CON CHIAVE K, ALTRIMENTI NULL

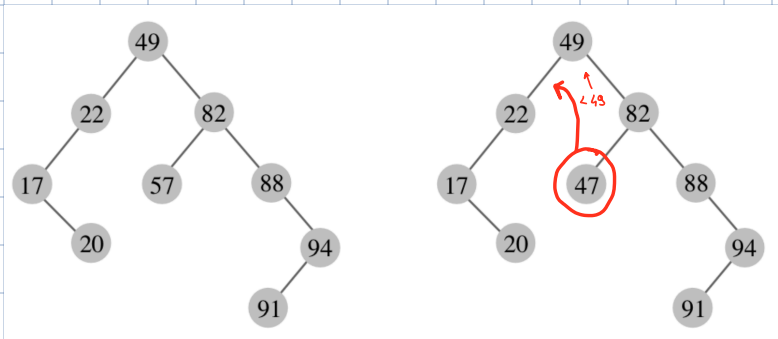
POSSIAMO GARANTIRE CHE TUTTE LE OPERAZIONI SU UN DIZ. DI M ELEMENTI ABBIANO TEMPO $O(\log m)$, PER FARE CIÓ ABBIAMO 2 POSSIBILITÀ:

- DEFINIRE UN ALBERO BINARIO T.C. OGNI OPERAZIONE RICHIEDE TEMPO $O(\text{ALTEZZA ALBERO})$
  ↳ ALBERI BINARI DI RICERCA
- FARE IN MODO CHE L'ALTEZZA DELL'ALBERO SIA SEMPRE $O(\log m)$
  ↳ ALBERI AVL

ALBERO BINARIO

PER OGNI NODO V VALE CHE:

- OGNI NODO HA LA CHIAVE E L'ELEMENTO, ORDINIAMO IN BASE ALLA CHIAVE.
- LE CHIAVI NEL SOTTOALBERO SINISTRO DI V SONO $\leq$ CHIAVE (V)
- LE CHIAVI NEL SOTTOALBERO DESTRO DI V SONO $>$ CHIAVE (V)

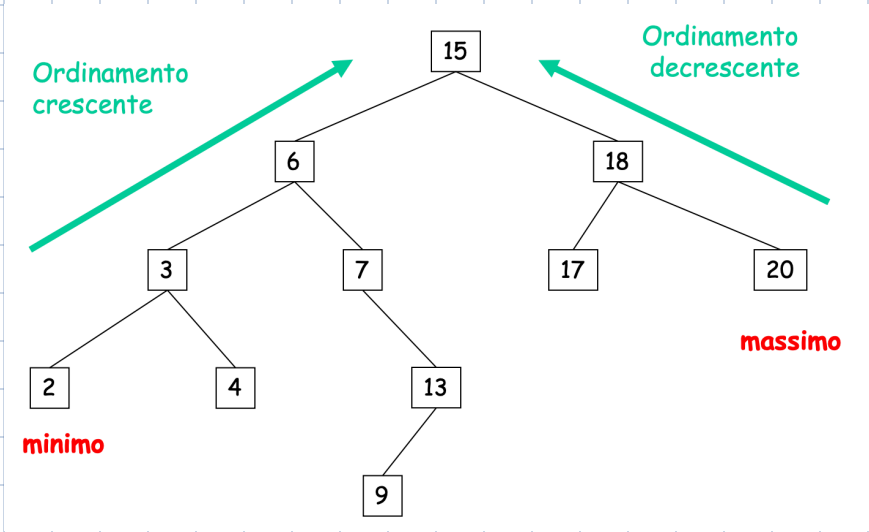


ALBERO BINARIO DI RICERCA

ALBERO BINARIO NON DI RICERCA

POSSIAMO NOTARE COME L'ELEMENTO PIÚ PICCOLO SIA QUELLO PIÚ A SINISTRA NEL SOTTOALBERO SINISTRO, MENTRE QUELLO PIÚ GRANDE

SIA QUELLO PIÙ A DESTRA NEL SOTTOALBERO DESTRO

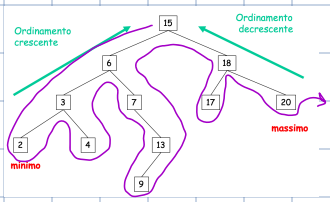


VISITANDO L'ALBERO IN ORDINE SIMMETRICO NOTIAMO CHE I NODI VERRANNO VISITATI IN ORDINE CRESCENTE.

```python
def visitaSimmetrica(nodo):

    if nodo != None:
        visitaSimmetrica(nodo.sinistro)
        print(nodo.valore)
        visitaSimmetrica(nodo.destro)
```

OUTPUT: 2 3 4 6 7 9 13 15 17 18 20



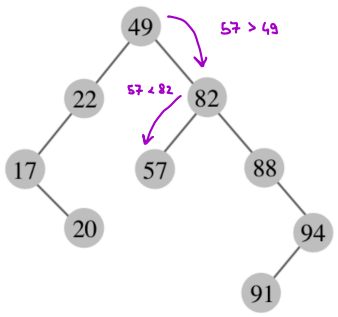
SEARCH (CHIAVE K) → ELEM E

DATE LE PROPRIETÀ DELL'ALBERO, SE DOBBIAMO CERCARE UNA CHIAVE POSSIAMO ORIENTARCI NELL'ALBERO SE LA CHIAVE È A SX O DX DEL NODO IN CUI SONO, COME NELLA RICERCA BINARIA.

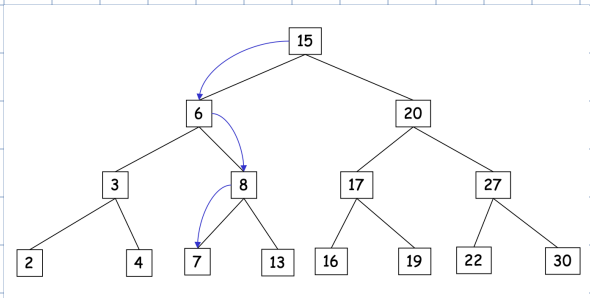
algoritmo `search`$(chiave\ k) \rightarrow elem$

1. $v \leftarrow$ radice di T
2. **while** $(v \neq$ `null`$)$ **do**
3. **if** $(k = chiave(v))$ **then return** $elem(v)$
4. **else if** $(k < chiave(v))$ **then** $v \leftarrow$ figlio sinistro di v
5. **else** $v \leftarrow$ figlio destro di v
6. **return** `null`

SEARCH (57)

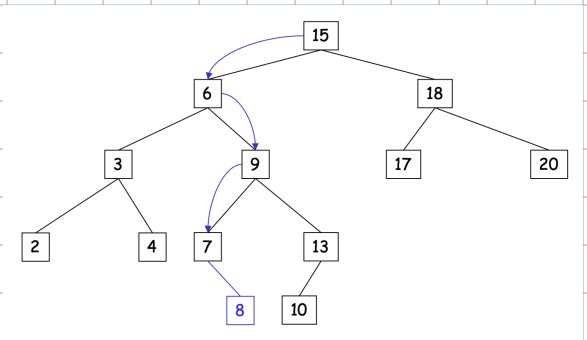


SEARCH (7)



INSERT (ELEM E, CHIAVE K)

AGGIUNGIAMO IL NODO COME FOGLIA, SIMULIAMO UNA RICERCA PER CAPIRE DOVE INSERIRLA.



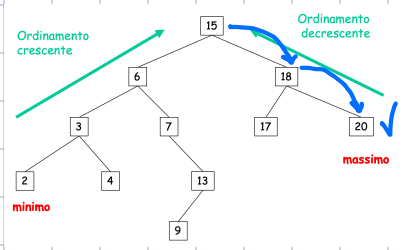
RICERCA DEL MASSIMO E DEL MINIMO

PER LE PROPRIETÀ DELL'ALBERO SAPPIAMO CHE IL MINIMO SI TROVA TUTTO A SINISTRA E IL MASSIMO TUTTO A DESTRA

algoritmo `max`$(nodo\ u) \rightarrow nodo$

1. $v \leftarrow u$
2. **while** (figlio destro di $v \neq$ `null`) **do**
3. $\quad v \leftarrow$ figlio destro di v
4. **return** v

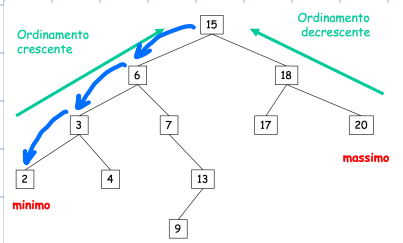
FINCHÉ IL FIGLIO DX NON È NULL VADO AL PROSSIMO FIGLIO DX



algoritmo MIN$(nodo\ u) \rightarrow nodo$

1. $v \leftarrow u$
2. **while** (figlio SINISTRO di $v \neq$ `null`) **do**
3. $\quad v \leftarrow$ figlio SINISTRO di v
4. **return** v

FINCHÉ IL FIGLIO SX NON È NULL VADO AL PROSSIMO FIGLIO SX

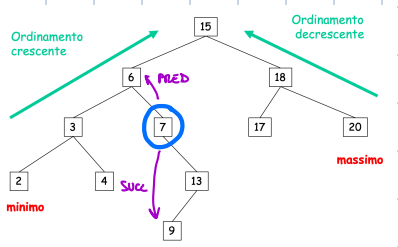


PREDESSORE & SUCCESSORE

PREDECESSORE → NODO V NELL'ALBERO CON MASSIMA CHIAVE ≤ CHIAVE DI U

SUCCESSORE → NODO V NELL'ALBERO CON MINIMA CHIAVE ≥ CHIAVE DI U

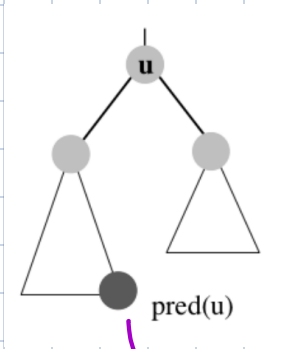
NODO DEL QUALE STIAMO CERCANDO IL PREDECESSORE



RICERCA DEL PREDECESSORE

DOBBIAMO DISTINGUERE 2 CASI, SIA V IL NODO DI CUI VOGLIAMO CERCARE IL PREDECESSORE:

- **V HA FIGLIO SINISTRO** ⟶ SE V HA FIGLIO SX ALLORA IL PREDECESSORE SARÀ IL MASSIMO TRA I NODI DEL SOTTOALBERO SINISTRO, E SARÀ IL NODO PIÚ A DX.

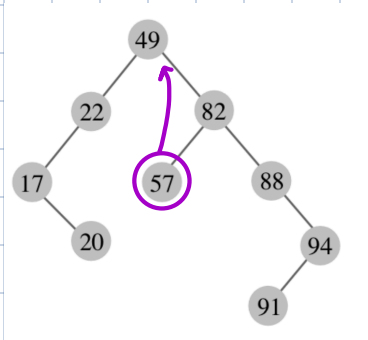


MAX SOTTOALBERO SX DI V

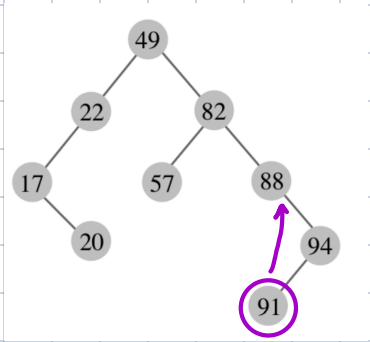
PERCHÈ ?

VEDI DEF. ANALOGA SOTTO

- **V NON HA FIGLIO SINISTRO** ⟶ SE V NON HA FIGLIO SINISTRO ALLORA IL SUO PREDEC SARÀ IL PRIMO ANTENATO CHE HA V NEL SUO SOTTOALBERO DESTRO.



IL NODO 49 È IL PRIMO AD AVERE 57 NEL SUO SOTTOALBERO DX



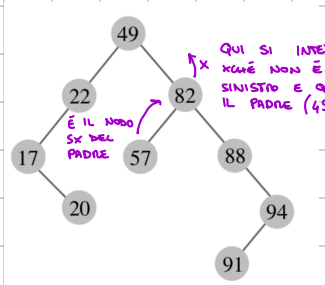
algoritmo `pred`$(nodo\ u) \rightarrow nodo$

1. **if** (u ha figlio sinistro $sin(u)$) **then**
2. **return** $\max(sin(u))$
3. **while** ($parent(u) \neq null$ e u è figlio sinistro di suo padre) **do**
4. $u \leftarrow parent(u)$
5. **return** $parent(u)$

SE U HA FIGLIO SINISTRO ALLORA IL PRED. SARÀ IL MAX DEL SOTTOALB SINISTRO

FINCHÈ C'È UN PARENTE E U È IL NODO SINISTRO ALLORA SALI FACENDO DIVENTARE U IL PADRE

U = 57



RICERCA DEL SUCCESSORE

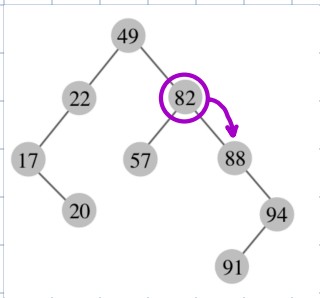
DOBBIAMO DISTINGUERE 2 CASI, SIA V IL NODO DI CUI VOGLIAMO CERCARE IL PREDECESSORE:

- **V HA FIGLIO DESTRO** → SE V HA FIGLIO DX PER LE PROPRIETÀ DELL' ALBERO SAPPIAMO CHE GLI ELEMENTI MAGGIORI DI UNA CERTA CHIAVE SONO NEL SOTTOALBERO DESTRO DI QUELLA CHIAVE.

  SAPPIAMO INOLTRE (SEMPRE x LE PROPRIETÀ) CHE IL MINIMO DI UN ALBERO SI TROVA TUTTO A SINISTRA.

  QUINDI: PER TROVARE IL SUCCESSORE:

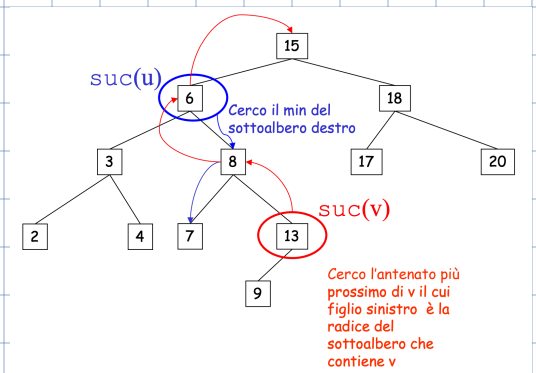
  - PRENDO IL SOTTOALBERO DESTRO (COSÍ DA PRENDERE I SUCCESSORI)
  - PRENDO IL MINIMO DEI SUCCESSORI (COSÍ DA PRENDERE IL PRIMO SUCCESSORE)



- **V NON HA FIGLIO DESTRO** → SE V NON HA FIGLIO DESTRO ALLORA IL SUO PREDEC SARÀ IL PRIMO ANTENATO CHE HA V NEL SUO SOTTOALBERO SINISTRO.

  PERCHÉ

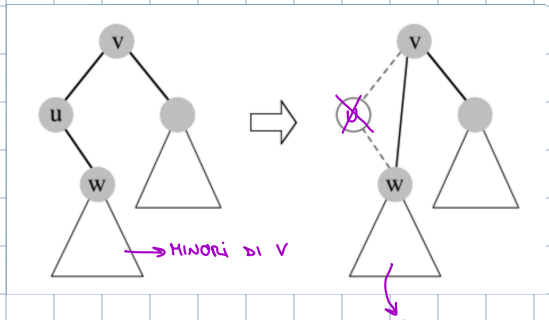
  SALENDO DA DESTRA IO RISALGO TUTTI I NODI DEI QUALI SONO PIÚ GRANDE, CIOÉ CHE SONO PIÚ PICCOLI. PRENDENDO IL PRIMO NODO DEL QUALE SONO IL FIGLIO SINISTRO SIGNIFICA CHE PRENDO IL PRIMO NODO DEL QUALE SONO MINORE



DELETE (CHIAVE K)

POSSIAMO DISTINGUERE 3 CASI PRINCIPALI:

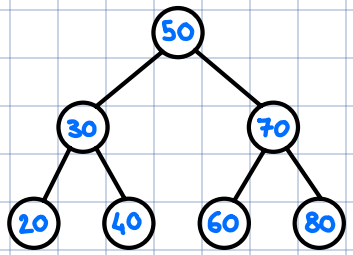
- U È UNA FOGLIA ⟶ RIMUOVILA
- U HA UN SOLO FIGLIO ⟶ COLLEGA L'UNICO FIGLIO AL PADRE DEL NODO DA RIMUOVERE



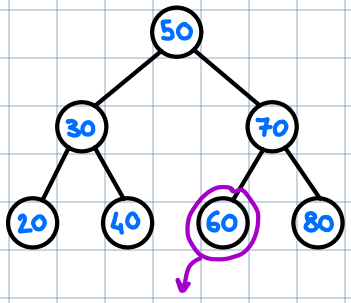
SEMPRE MINORI DI V
INFATTI SONO A SX DI V

DATE LE PROP. DELL'ALBERO ABBIAMO CHE $U \leq W \leq V$
RIMUOVENDO U E COLLEGANDOLO A V MANTERREMO COMUNQUE LE PROPRIETÀ DELL'ALBERO BST.
INFATTI TUTTI GLI ELEMENTI DEL SOTTOALBERO SX/DX DI U SONO MINORI DI V, LE PROPRIETÀ SONO QUINDI RISPETTATE

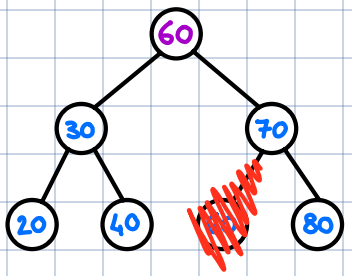
- U HA 2 FIGLI ⟶ SOSTITUISCO IL NODO DA RIMUOVERE CON IL PREDECESSORE OPPURE IL SUCCESSORE



U = 50



SUCCESSORE = 60



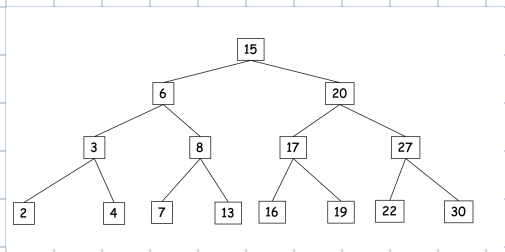
SOSTITUISCO ED ELIMINO IL PREDECESSORE/SUCCESSORE

COSTI DELLE OPERAZIONI

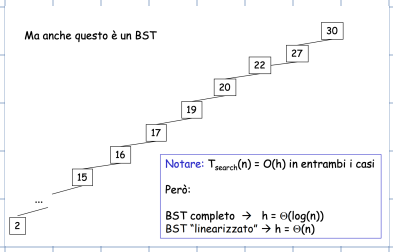
TUTTE LE OPERAZIONI HANNO COSTO $O(h)$ CON h ALTEZZA DELL'ALBERO

h ⟶ IN CASO DI ALBERI BILANCIATI $h = O(\log m)$

h ⟶ IN CASO DI ALBERI SBILANCIATI NEL CASO PEGGIORE $h = O(m)$



$\rightarrow O(\log m)$



$\rightarrow O(m)$

DIZIONARIO

- OGNI OPERAZIONE HA COSTO $O(\log m)$, ABBIAMO QUINDI 2 POSSIBILITÀ:
  - ALBERO BINARIO
  - ALBERO AVL

ALBERO BINARIO (BST)

- OGNI NODO HA NEL SOTTOALBERO SX I NODI CON CHIAVE $\leq$ DEL NODO, NEL SOTTOALBERO DX GLI ELEMENTI CON CHIAVE MAGGIORE DEL NODO.
  L'ALBERO PUÒ ANCHE ESSERE SBILANCIATO.

SEARCH NEI BST

- IN BASE ALLA CHIAVE DA CERCARE CI ORIENTIAMO SPOSTANDOCI O SUL FIGLIO SX O SUL FIGLIO DX, COME NELLA RICERCA BINARIA.
  $O(\log m)$ SE L'ALBERO È BILANCIATO
  $O(m)$ SE L'ALBERO È SBILANCIATO

INSERT NEI BST

- SIMULIAMO UNA RICERCA PER CAPIRE DOVE INSERIRE IL NODO E APPENDIAMO LA FOGLIA.
  $O(\log m)$ SE L'ALBERO È BILANCIATO
  $O(m)$ SE L'ALBERO È SBILANCIATO

DELETE NEI BST

- SE IL NODO DA RIMUOVERE È UNA FOGLIA: LA RIMUOVO
  SE HA 1 FIGLIO: COLLEGO IL FIGLIO AL PADRE DEL NODO DA RIMUOVERE
  SE HA 2 FIGLI: SOSTITUISCO IL NODO DA RIMUOVERE CON IL PREDECESSORE O SUCCESSORE.
  $O(\log m)$ SE L'ALBERO È BILANCIATO
  $O(m)$ SE L'ALBERO È SBILANCIATO