UNIMORE
UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

Dispense per il Laboratorio di Fondamenti di Informatica II e Lab

Federico Bolelli

Esercitazione 05: Note su Alberi BST

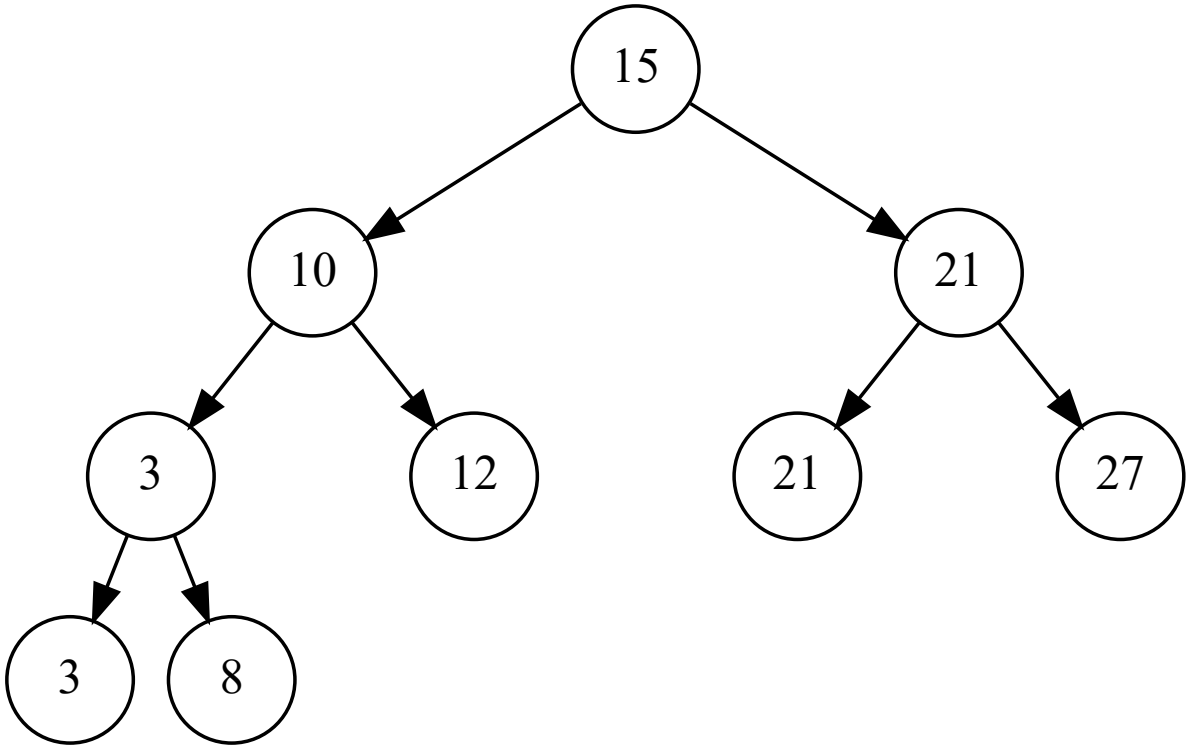
Ultimo aggiornamento: 19/05/2021

Binary Search Tree - Definizione

- Un albero binario di ricerca (dall'inglese Binary Search Tree o BST in breve) è un particolare albero binario che soddisfa le seguenti proprietà:
  1. Il sottoalbero sinistro di un nodo contiene soltanto chiavi minori (o al più uguali) della chiave del nodo stesso;
  2. Il sottoalbero destro di un nodo contiene soltanto chiavi maggiori della chiave del nodo;

- In sostanza $left \leq root < right$

Binary Search Tree - Esempio



Binary Search Tree - Definizione

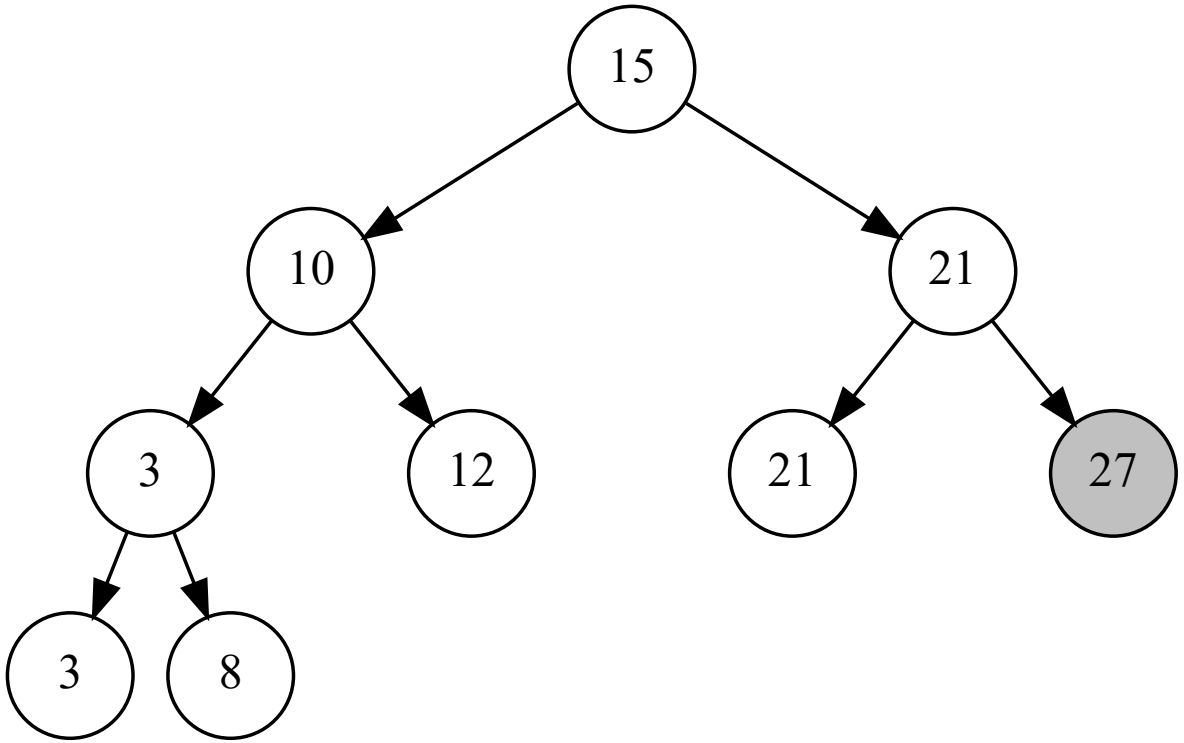
- In realtà, a seconda di dove viene posizionata la condizione di uguaglianza possiamo definire le proprietà BST in 4 modi differenti:
  1. $left \leq root < right$, le chiavi duplicate, se presenti, si trovano sempre nel sottoalbero sinistro di un nodo;
  2. $left < root \leq right$, le chiavi duplicate, se presenti, si trovano sempre nel sottoalbero destro di un nodo;
  3. $left \leq root \leq right$, le chiavi duplicate, se presenti, possono trovarsi sia nel sottoalbero destro che in quello sinistro;
  4. $left < root < right$, non sono ammesse chiavi duplicate.

Binary Search Tree – Delete Node

- È importante tenere sempre bene a mente la definizione scelta perché questa può influenzare l'implementazione di un algoritmo di manipolazione/accesso al BST.
- Prendiamo ad esempio l'algoritmo di eliminazione di un nodo dal BST. Al termine, le proprietà BST devono ancora essere verificate.
- Occorre distinguere tre casi:
  1. Il nodo da eliminare è una foglia;
  2. Il nodo da eliminare ha un solo sottoalbero (sinistro o destro);
  3. Il nodo da eliminare ha entrambi i figli.

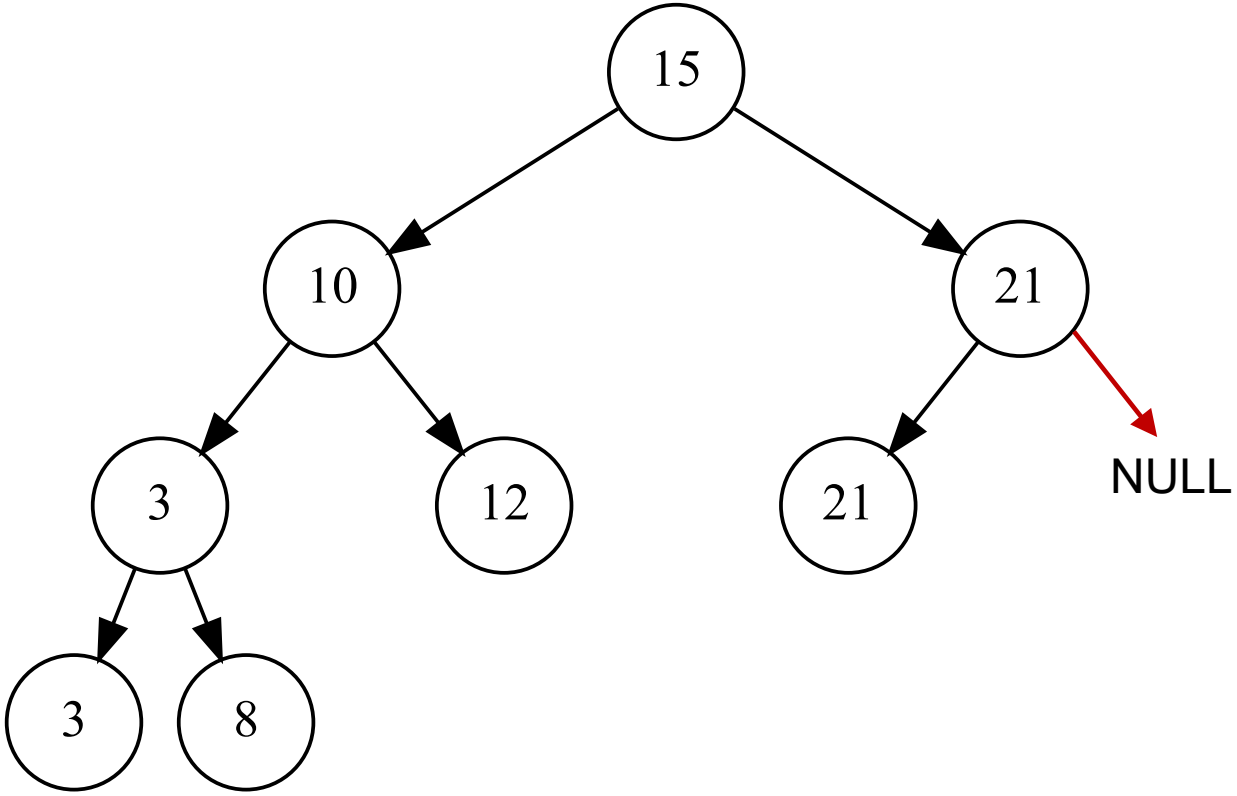
Delete Node – Caso 1

- Il nodo da eliminare è una foglia:



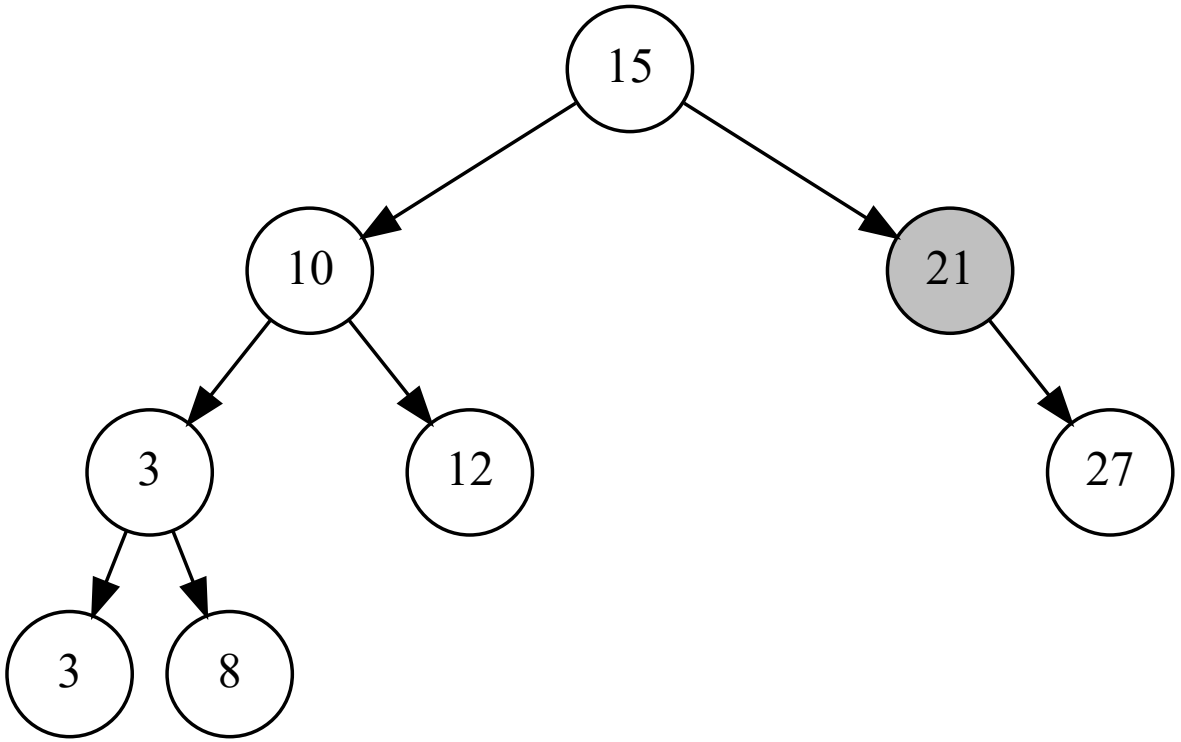
Delete Node – Caso 1

- Libero la memoria e aggiorno il padre perché punti a NULL:



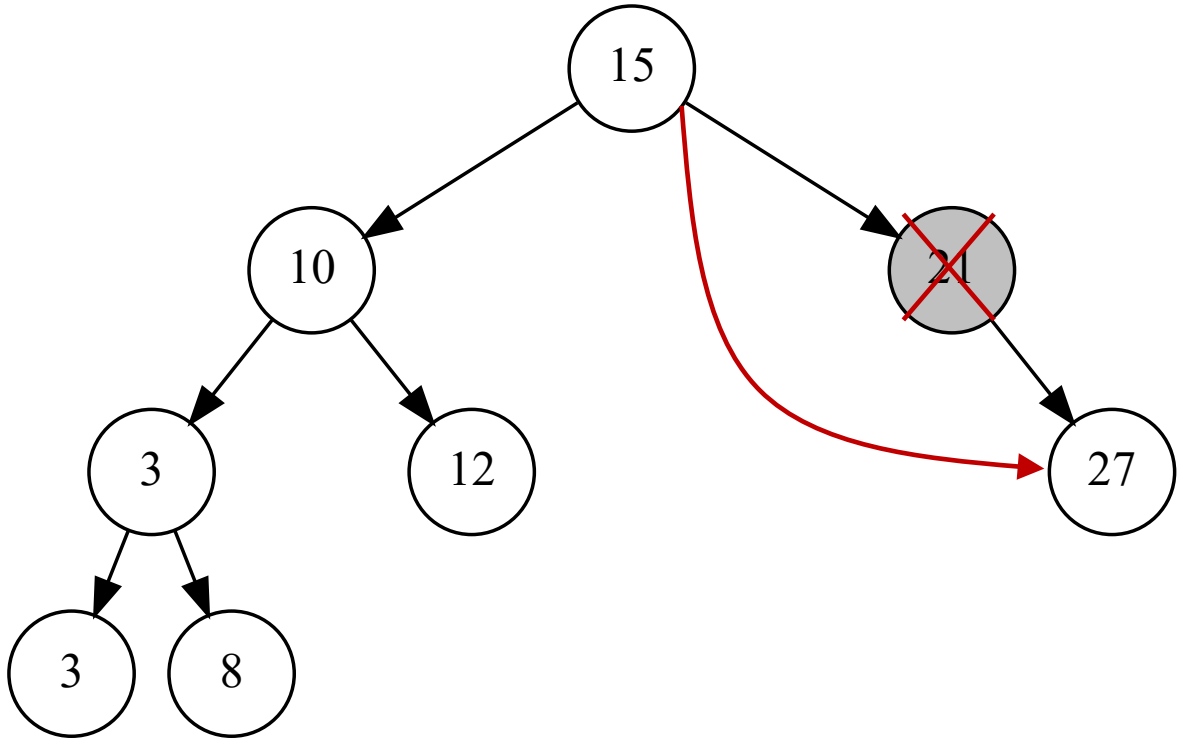
Delete Node – Caso 2

- Il nodo da eliminare ha un solo figlio:



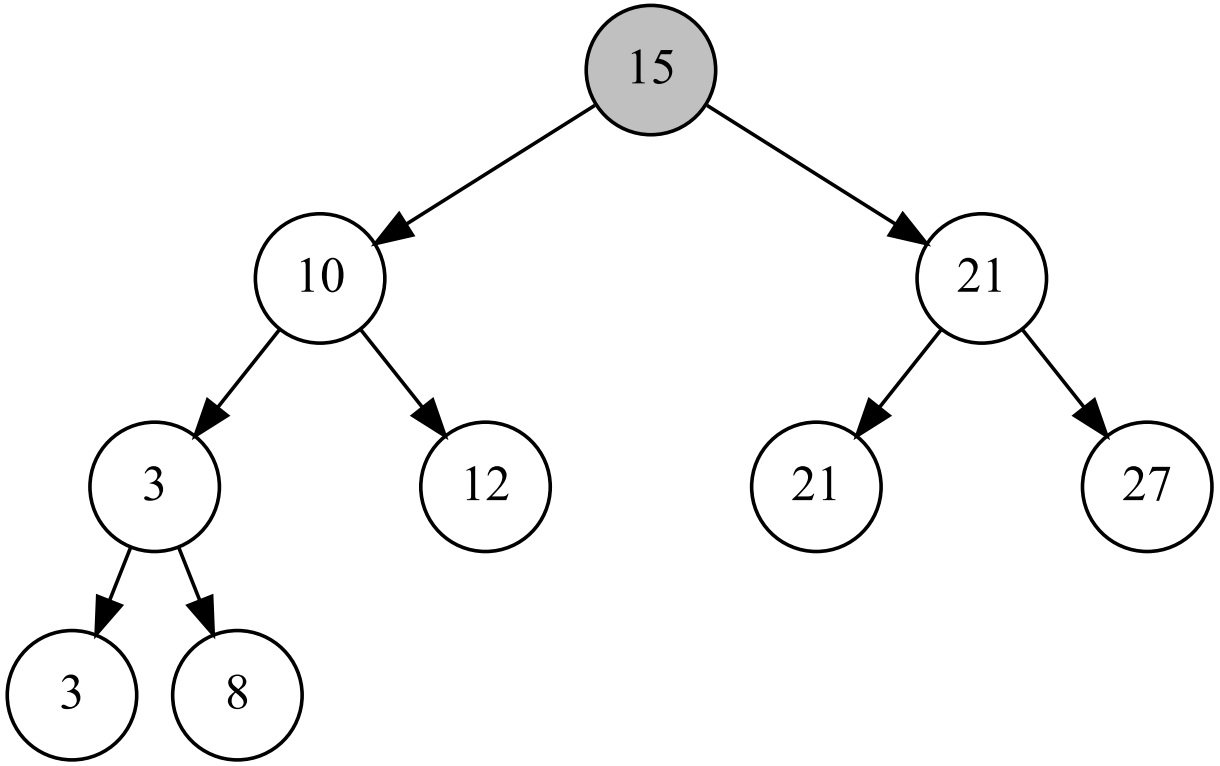
Delete Node – Caso 2

- Libero la memoria e aggiorno il padre perché punti al figlio del nodo da rimuovere:



Delete Node – Caso 3

- Il nodo da eliminare ha entrambi i figli:



Delete Node – Caso 3

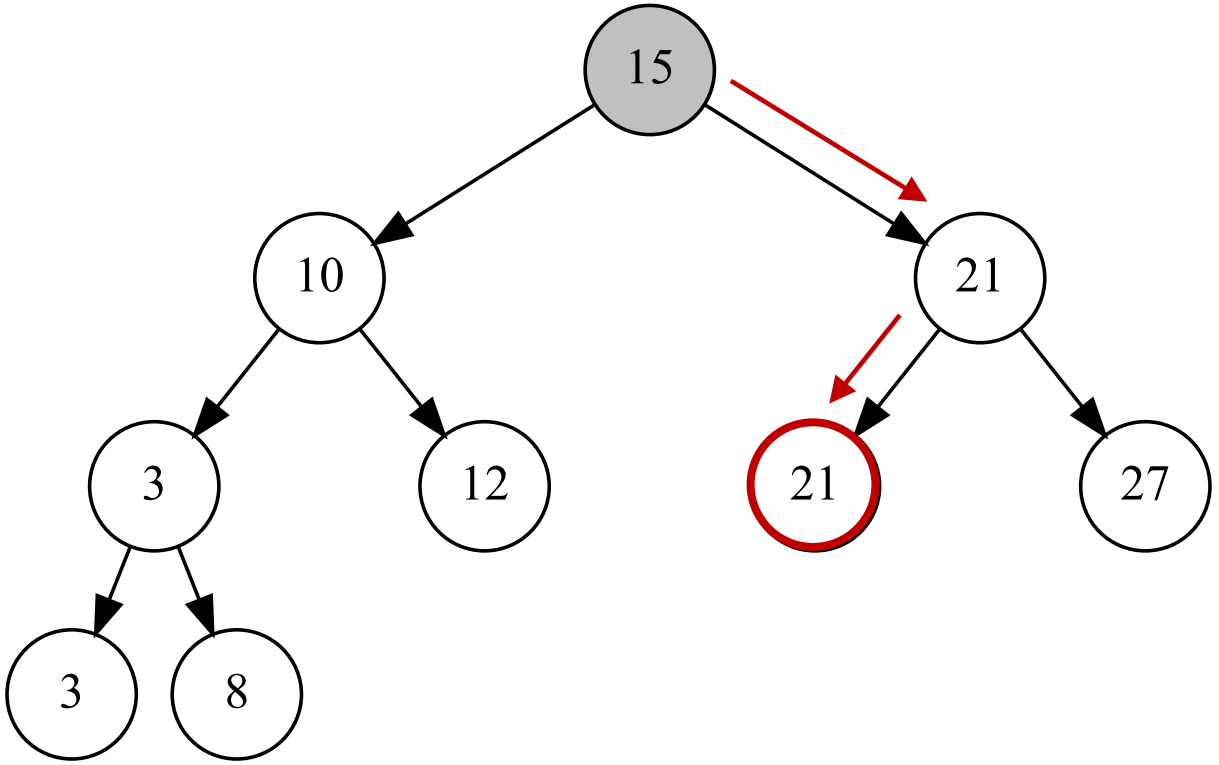
- Ci sono due possibilità:
  1. Sostituire il nodo con il suo successore;
  2. Sostituire il nodo con il suo predecessore;

Delete Node – Caso 3

- Ci sono due possibilità:
  1. **Sostituire il nodo con il suo successore;**
  2. Sostituire il nodo con il suo predecessore;

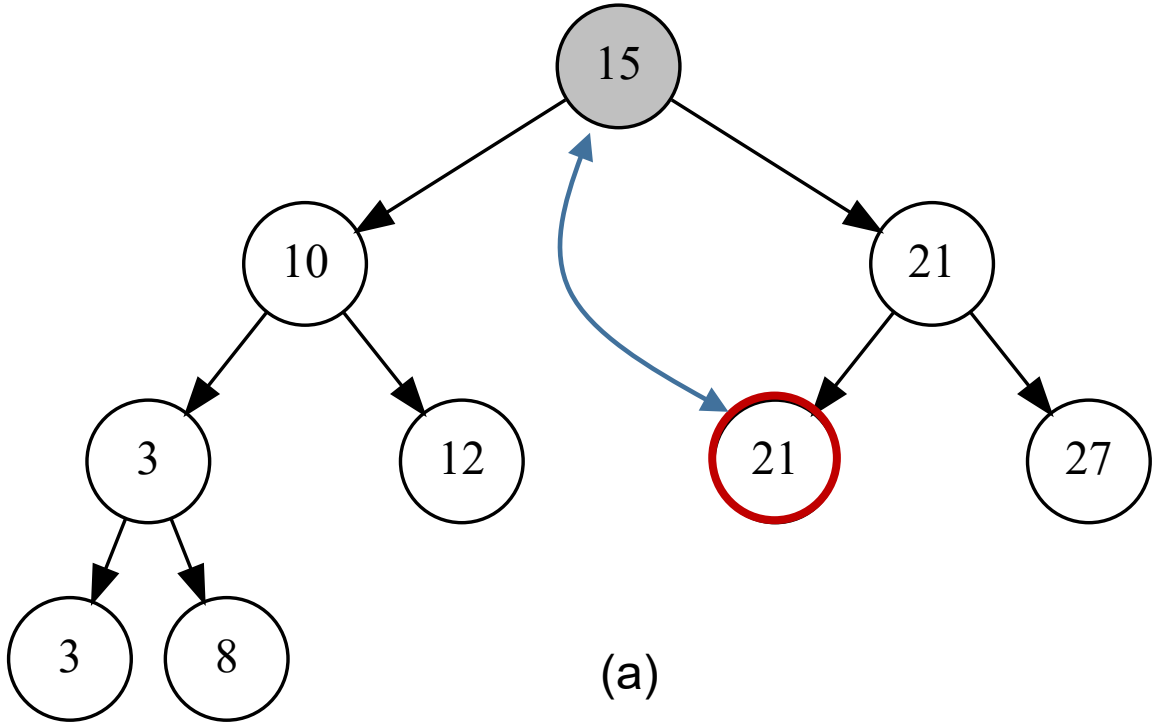
Delete Node – Caso 3

- Qual è il successore di un nodo?
- Il nodo con chiave minima nel sottoalbero destro del nodo stesso.



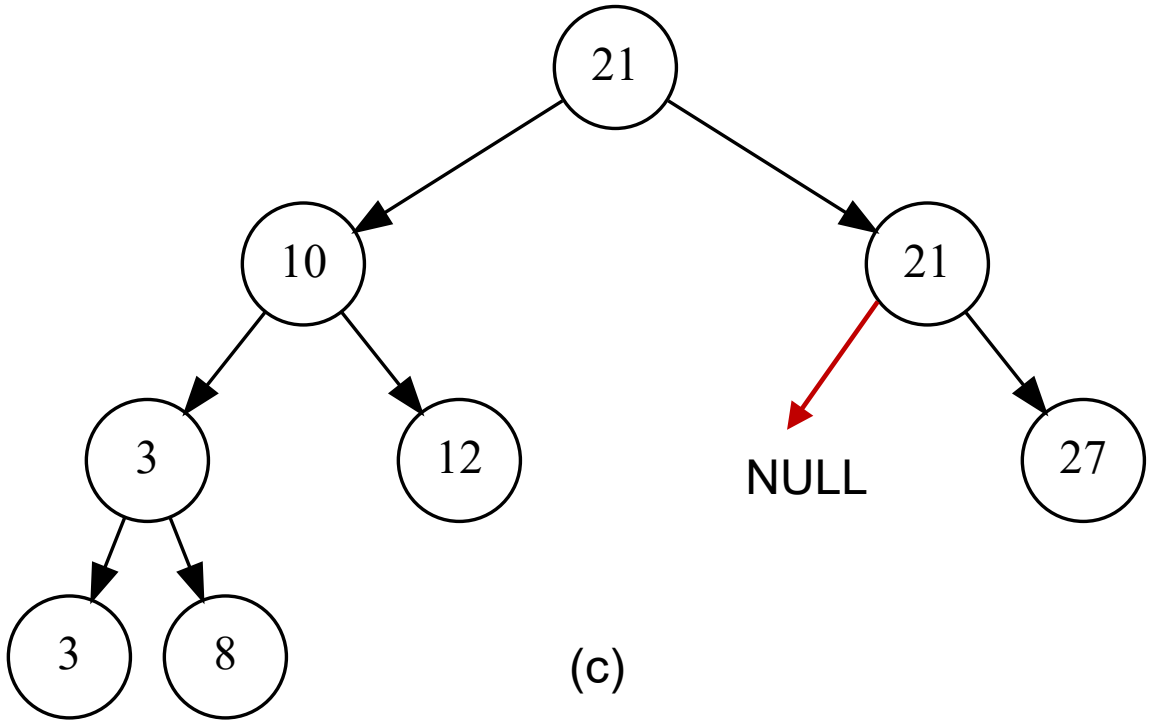
Delete Node – Caso 3

- Come procediamo?
  a) «Sostituiamo» il nodo 15 con il nodo 21;
  b) «Sostituiamo» il nodo 15 con l'eventuale figlio (non presente in questo esempio);



Delete Node – Caso 3

c) Eliminiamo il nodo



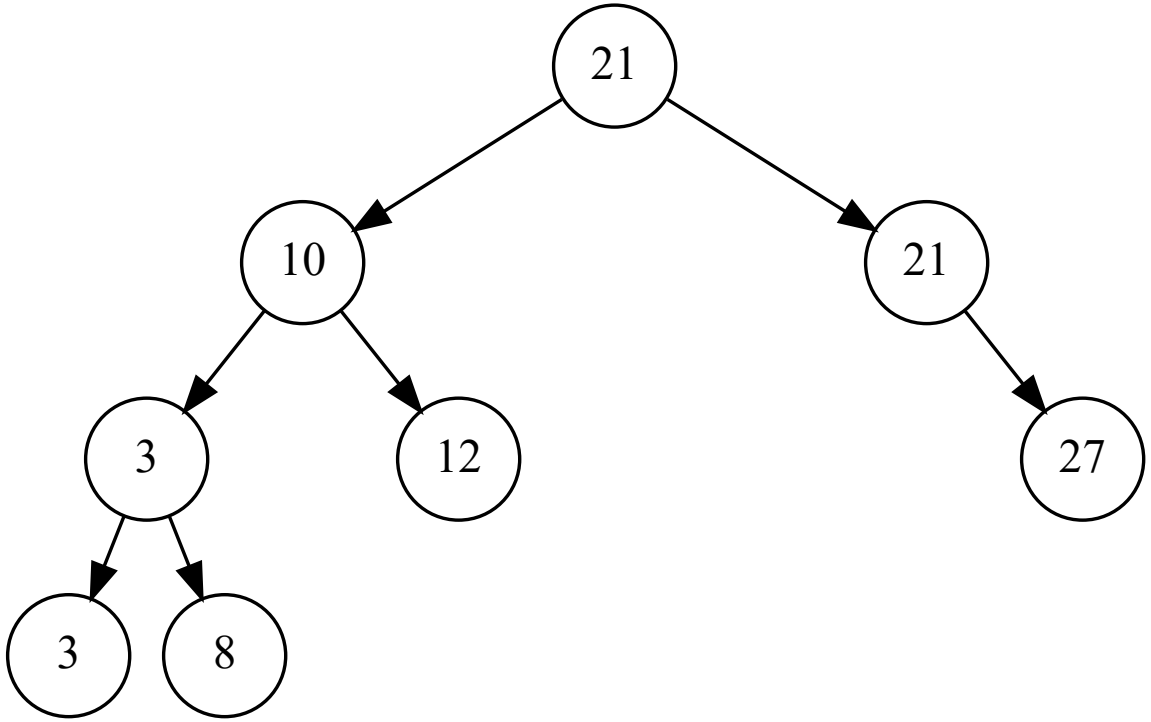
Delete Node – Caso 3

- La «sostituzione» può essere uno scambio di nodi vero e proprio o una scambio del contenuto dei nodi.
- La seconda soluzione è più semplice da implementare, ma potrebbe causare dei problemi se altri elementi del programma facevano riferimento ai nodi che sono stati sottoposti a scambio.

Delete Node – Caso 3

- Il risultato è sempre corretto?
- Dipende dalla definizione che abbiamo scelto per il BST!

1. $left \leq root < right$
2. $left < root \leq right$
3. $left \leq root \leq right$
4. $left < root < right$

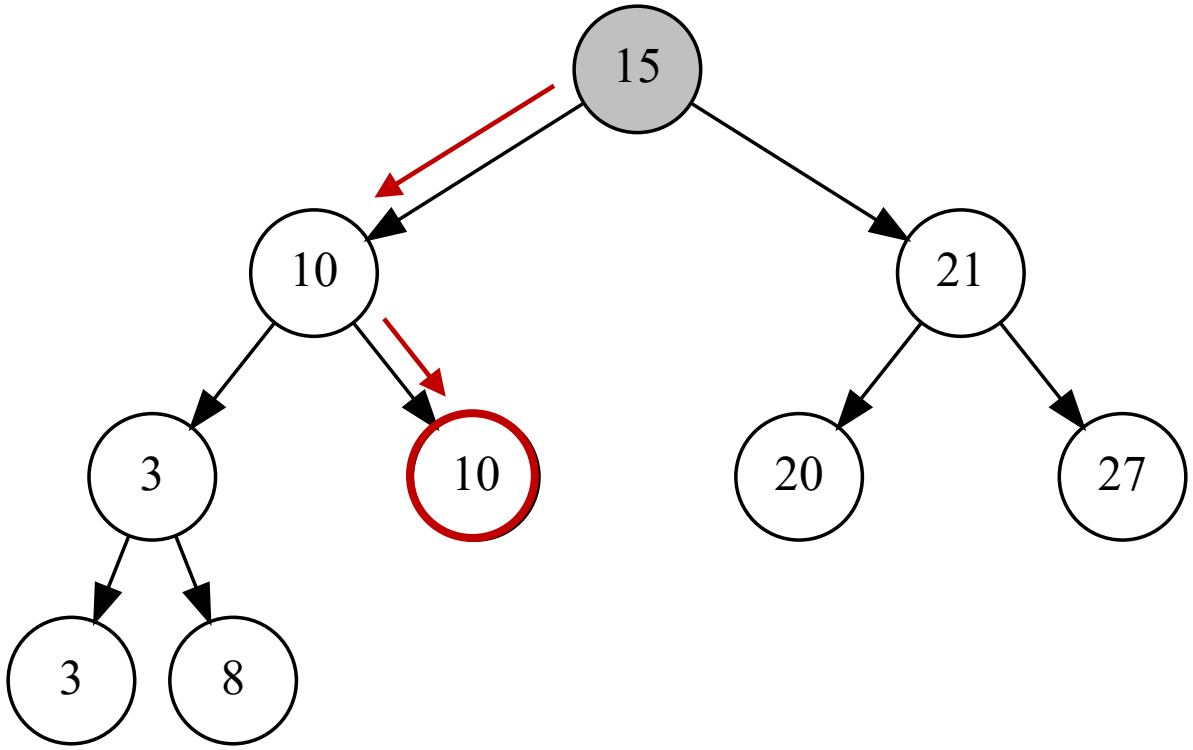


Delete Node – Caso 3

- Ci sono due possibilità:
  1. Sostituire il nodo con il suo successore;
  2. **Sostituire il nodo con il suo predecessore;**

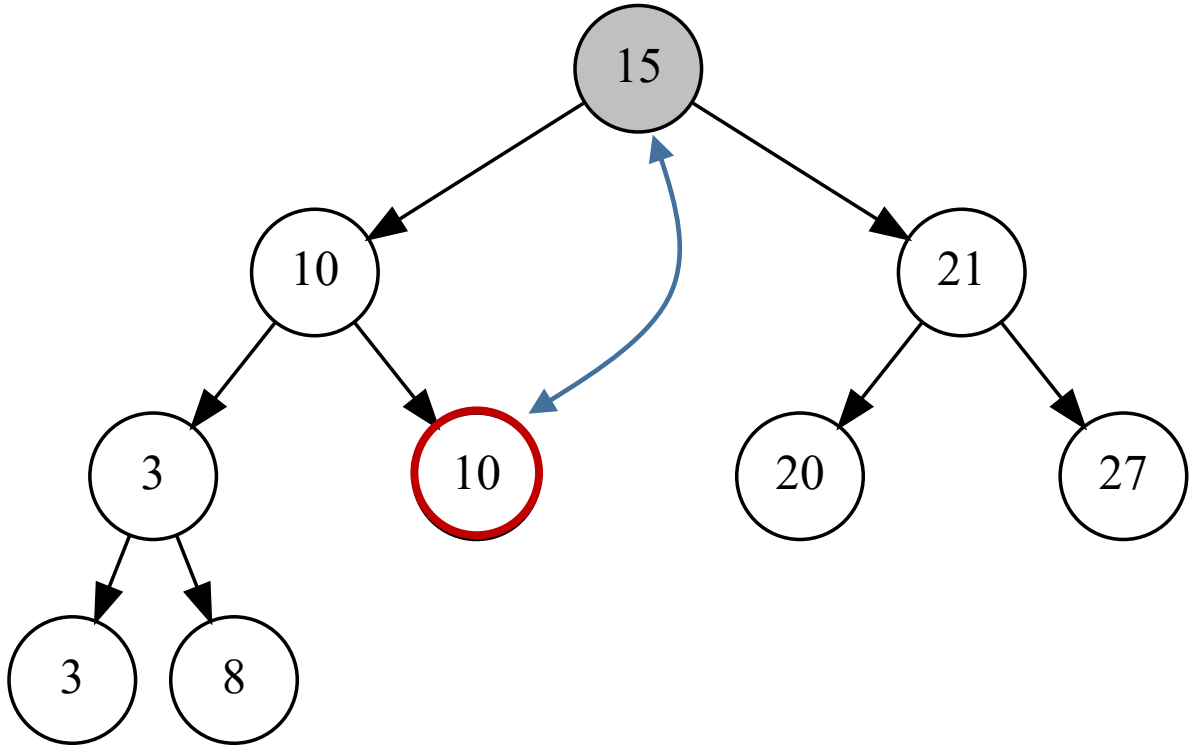
Delete Node – Caso 3

- Qual è il predecessore di un nodo?
- Il nodo con chiave massima nel sottoalbero sinistro del nodo stesso.



Delete Node – Caso 3

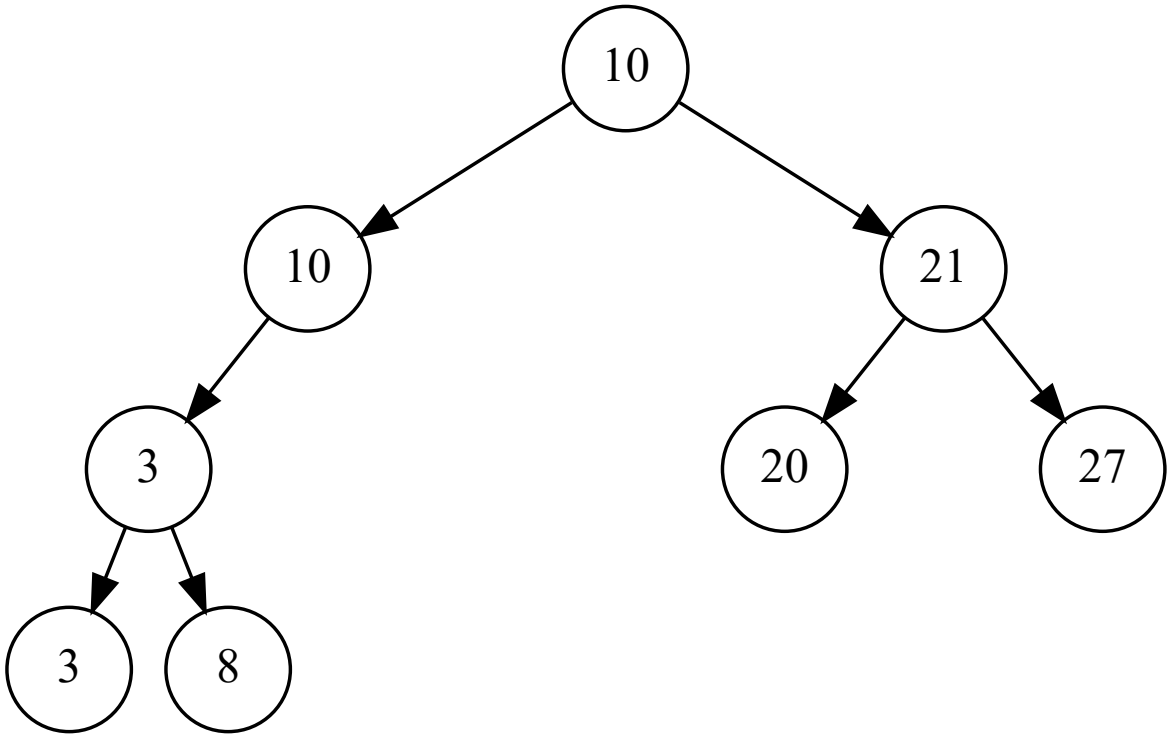
- Come procediamo? Il meccanismo è del tutto analogo a quello visto per il successore.



Delete Node – Caso 3

- Il risultato è sempre corretto?
- Dipende dalla definizione che abbiamo scelto per il BST!

1. $left \leq root < right$
2. $left < root \leq right$
3. $left \leq root \leq right$
4. $left < root < right$



Binary Search Tree - Definizione

- Se non diversamente specificato, negli esercizi utilizzeremo la prima definizione delle proprietà:
  1. $\boldsymbol{left \leq root < right}$**, le chiavi duplicate, se presenti, si trovano sempre nel sottoalbero sinistro di un nodo;**
  2. $left < root \leq right$, le chiavi duplicate, se presenti, si trovano sempre nel sottoalbero destro di un nodo;
  3. $left \leq root \leq right$, le chiavi duplicate, se presenti, possono trovarsi sia nel sottoalbero destro che in quello sinistro;
  4. $left < root < right$, non sono ammesse chiavi duplicate.