In questa lezione

Alberi binari di ricerca.

- **operazioni già studiate:**

ricerca, inserimento, determinazione del minimo e del massimo, del successivo e del precedente.

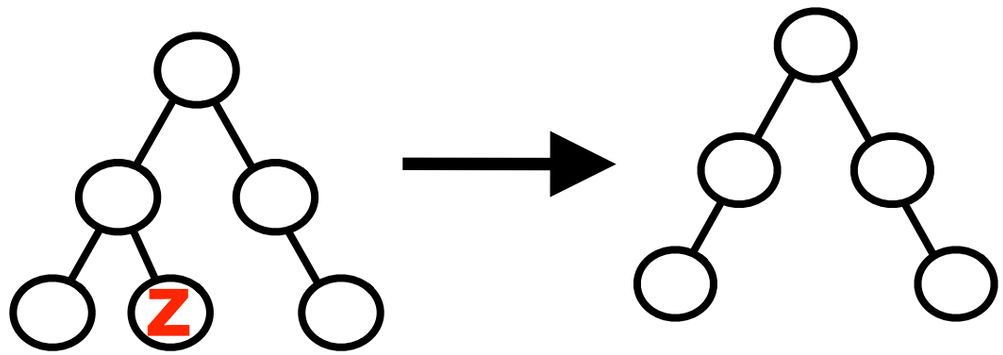
Sappiamo inoltre che la visita inorder produce le chiavi in ordine crescente e come verificare se un albero binario è un albero binario di ricerca in O(n).

- **terza operazione fondamentale: la cancellazione**

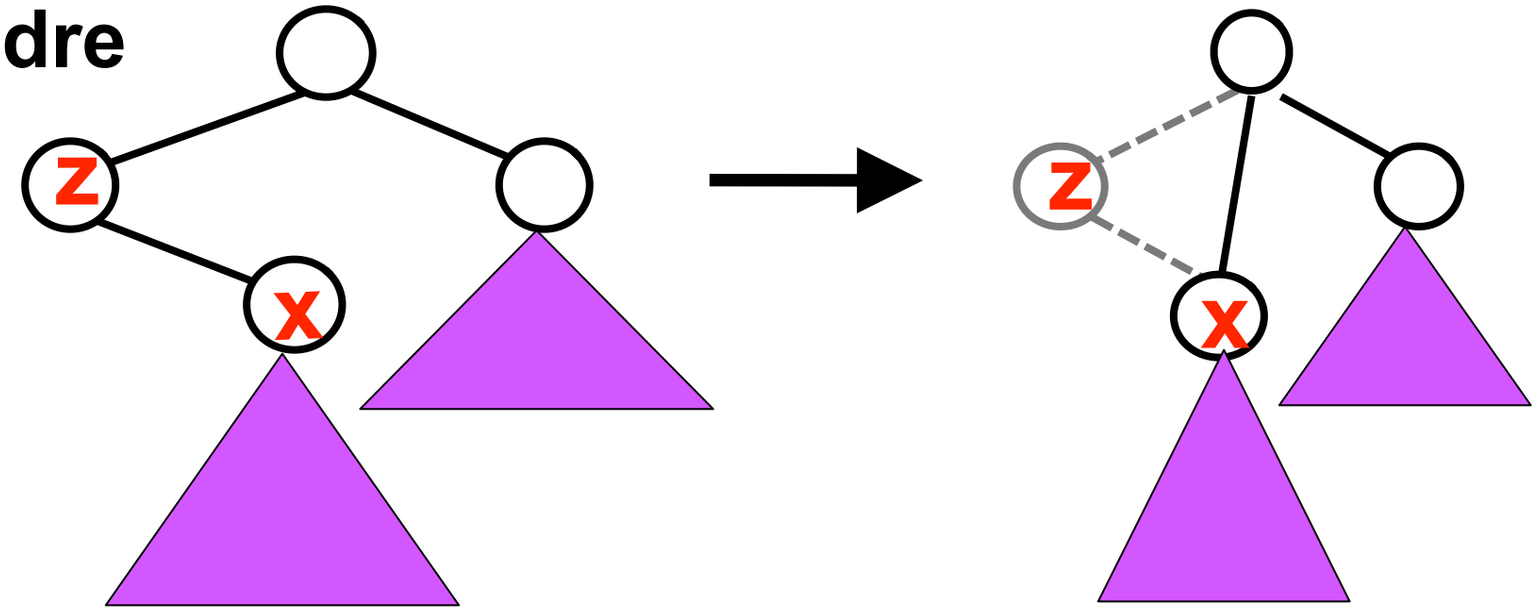
L'algoritmo di Hibbard per cancellare (1962)

Sia z il nodo da cancellare:

1) z è una foglia: si rimuove



2) z ha un solo figlio x:
si rende x figlio del padre di z

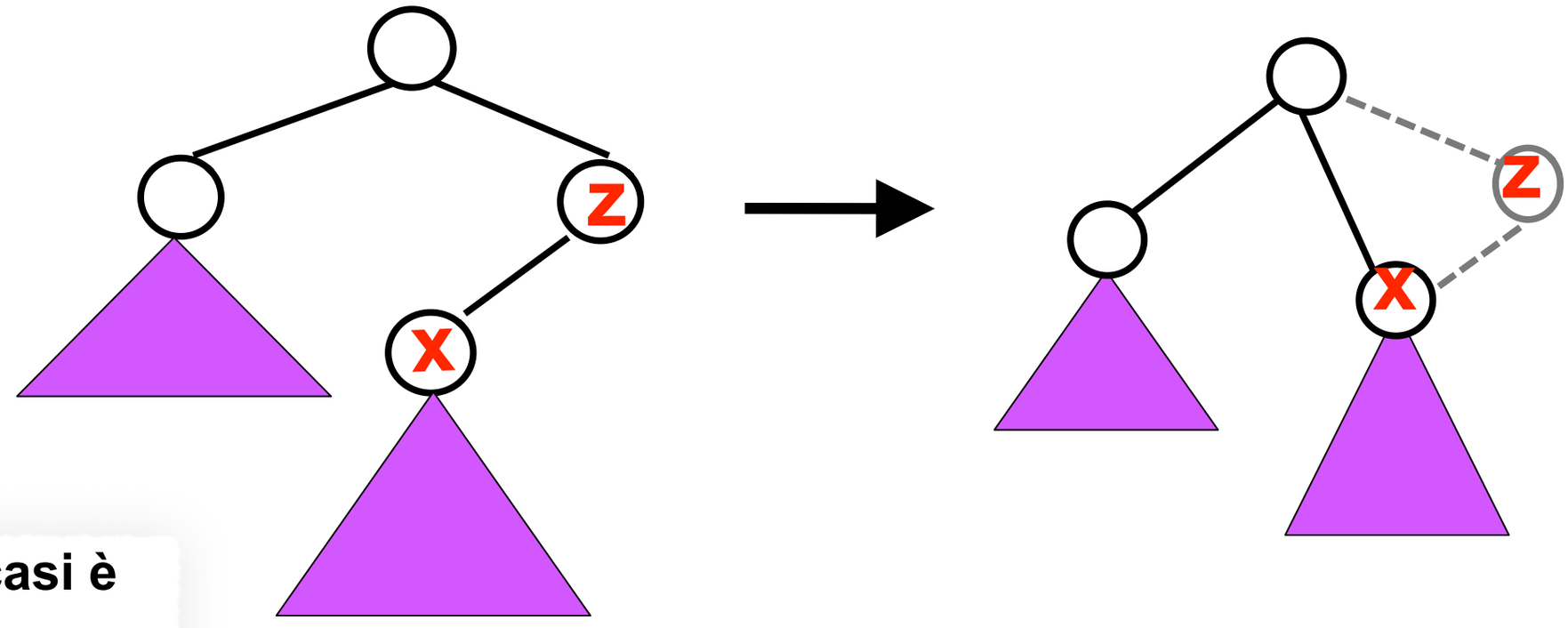


La cancellazione 2

il caso simmetrico

z ha un solo figlio x:

x diventa figlio del padre di z

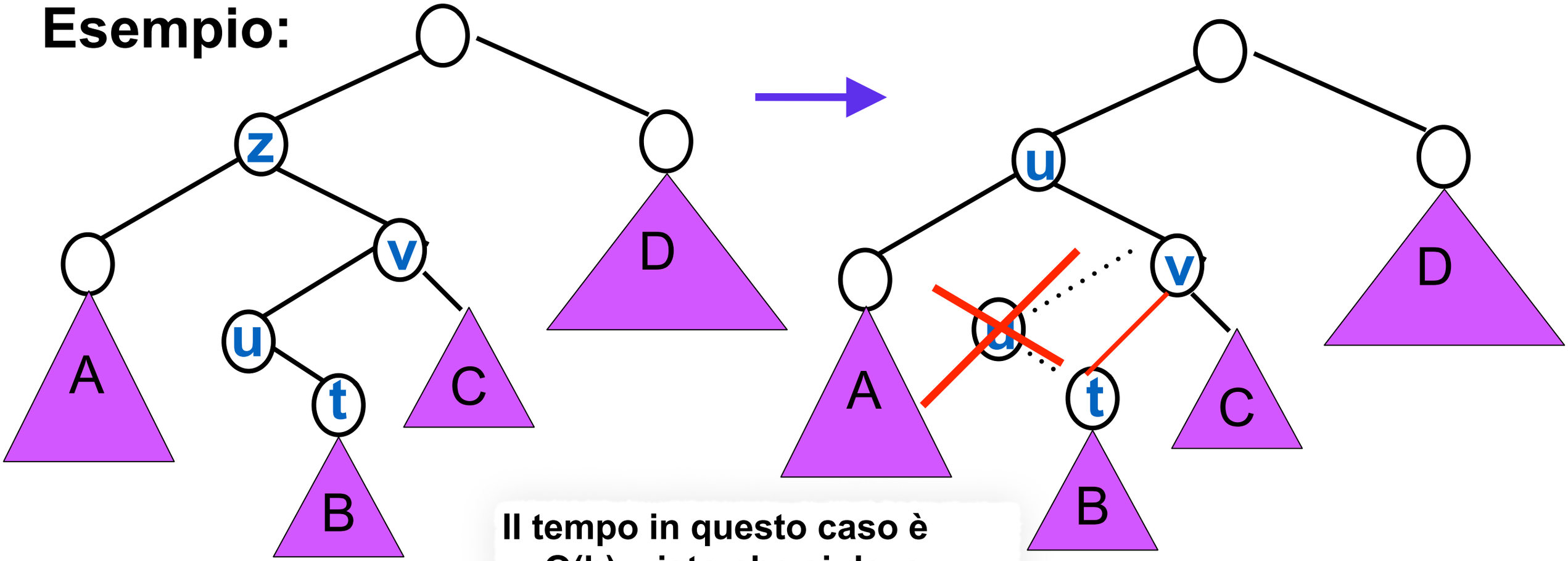


Il tempo in questi due casi è costante, visto che supponiamo di avere il riferimento a z

la cancellazione 3

3) z ha due figli: la chiave di z viene sostituita con quella minima del suo sottoalbero destro, che è il suo successivo nell'albero. Il nodo di chiave minima del sottoalbero destro di z, che ha al più un figlio, viene rimosso fisicamente.

Esempio:



Il tempo in questo caso è O(h), visto che si deve raggiungere il minimo

La cancellazione ricorsiva: caso 1

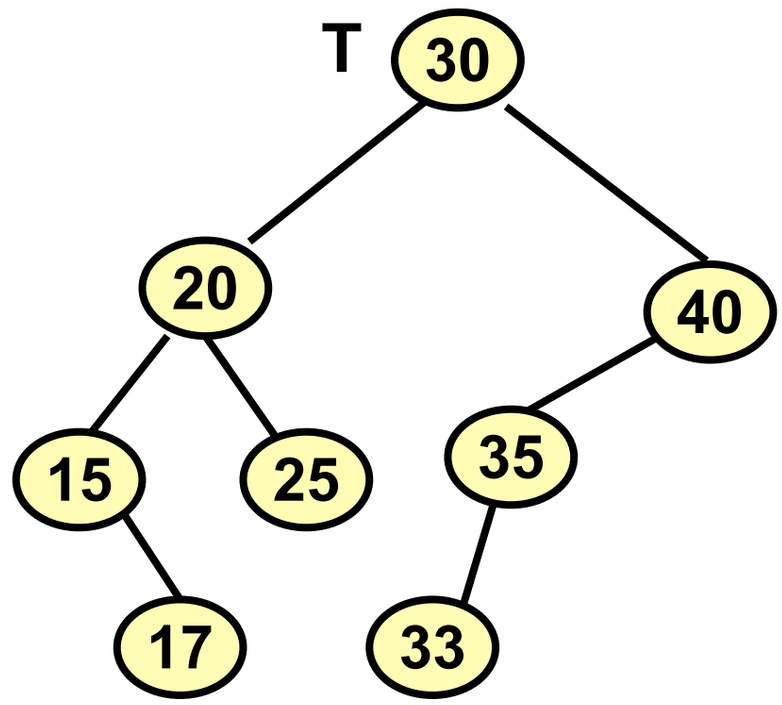
Delete(T,x) — **Solo puntatori ai figli**

input: T è il puntatore alla radice di un albero binario, x un intero

prec: T è un ABR

output: T privato del nodo di chiave x

```
if T == NIL return T
if  T.key == x  ▷    trovato!
    then if   T.left == NIL return T.right
         if   T.right == NIL return T.left
    ▷T ha un solo figlio o è  una foglia
  .
  .
  .
else if  x < T.key
        then T.left = Delete(T.left,x)
         else  if  x > T.key
                 then  T.right = Delete(T.right,x)
  return T
```



T.key = 30, x= 35, T.right = Delete(T.right,35)

T.key = 40, T.left = Delete(T.left,35)

T.key = 35 == x restituisce T.left, il (puntatore al) nodo con key=33

La cancellazione ricorsiva: caso 1

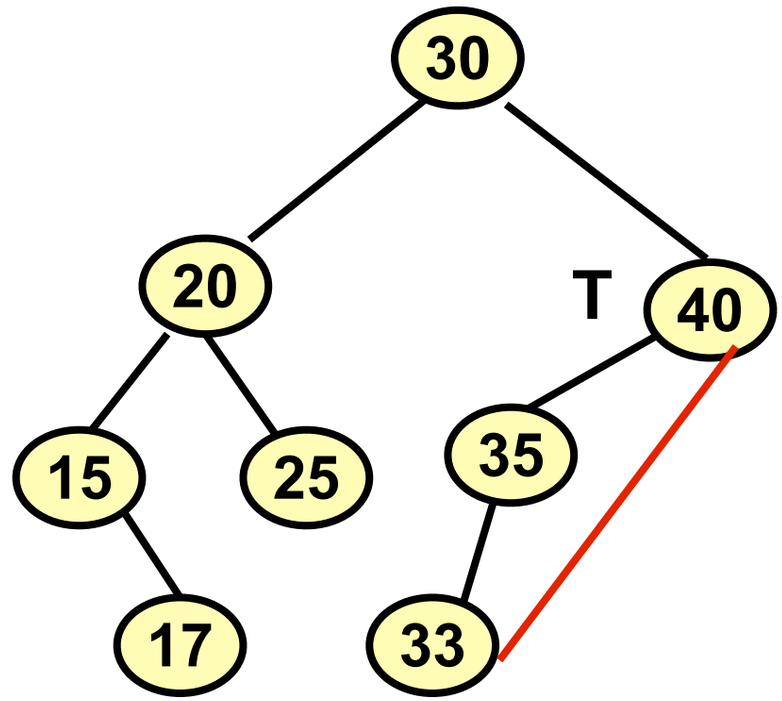
Delete(T,x)

Solo puntatori ai figli

prec: T è un ABR

postc: restituisce T privato del nodo di chiave x

```
if T == NIL return T
if  T.key == x  ▷    trovato!
    then if   T.left == NIL return T.right
              if   T.right == NIL return T.left
    ▷T ha un solo figlio o è  una foglia
  .
  .
  .
else if  x < T.key
        then T.left = Delete(T.left,x)
         else  if  x > T.key
                   then  T.right = Delete(T.right,x)
  return T
```



T.key = 30, x= 35, T.right = Delete(T.right,35)

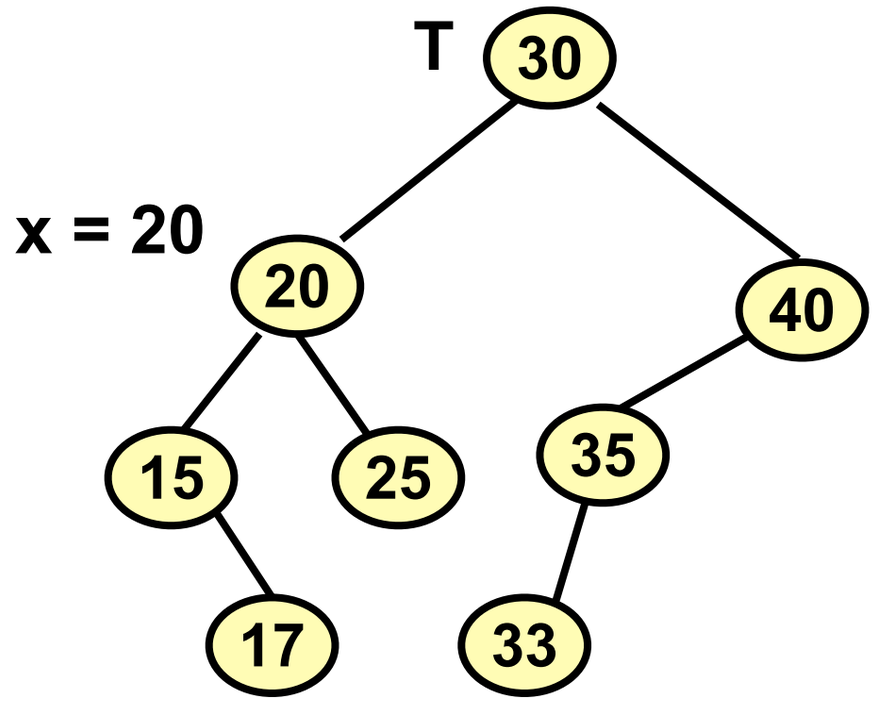
T.key = 40, T.left = Delete(T.left,35)

T.left è il nodo con key=33 e al rientro nella chiamata su 40, diventa il figlio sinistro di 40.

La cancellazione ricorsiva: caso 2

Solo puntatori ai figli

```
Delete(T,x)
 if T == NIL return T
 if  T.key == x  //    trovato!
    then if   T.left == NIL return T.right
              if   T.right == NIL return T.left
       // T ha un solo figlio o è  una foglia
      // T    ha due figli, si eliminerà il
       successivo di T
          z = Minimum(T.right)
          Delete(T.right,z.key)
       // z è un nodo con al più un figlio
          T.key = z.key  // il valore del
       successivo viene copiato in T
else  if  x < T.key
        then T.left = Delete(T.left,x)
         else   T.right = Delete(T.right,x)
return T
```



T.key = 30, T.left = Delete(T.left,20)

T.key = 20, z.key = 25

Delete(T.right,25)

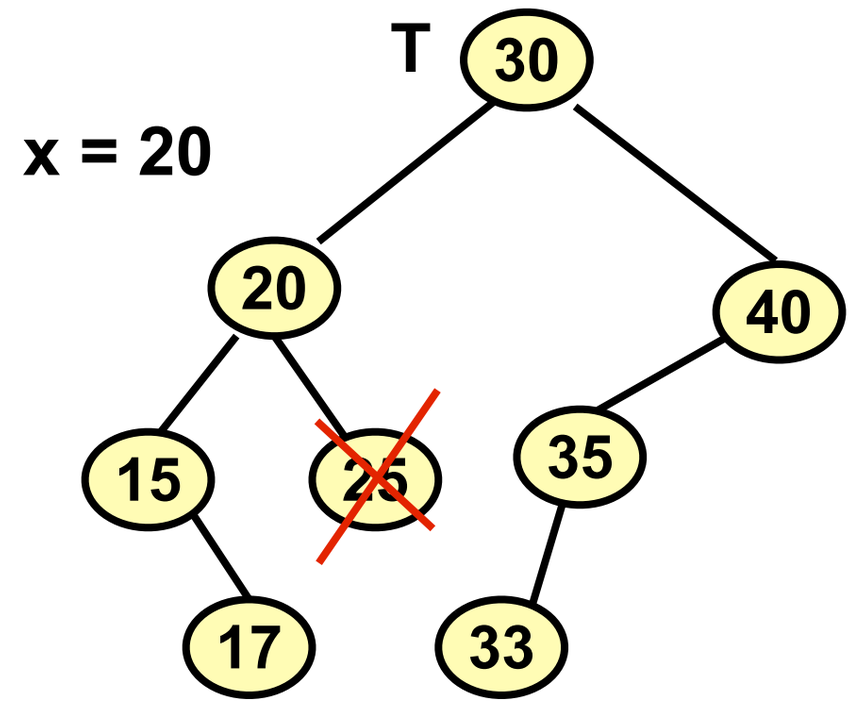
T.key = 25

La cancellazione ricorsiva: caso 2

Solo puntatori ai figli

```
Delete(T,x)
 if T == NIL return T
 if  T.key == x  //    trovato!
    then if   T.left == NIL return T.right
              if   T.right == NIL return T.left
       // T ha un solo figlio o è  una foglia

     // T    ha due figli, si eliminerà il
      successivo di T
         z = Minimum(T.right)
         Delete(T.right,z.key)  // z è un nodo
       con al più un figlio
         T.key = z.key  // il valore del
       successivo viene copiato in T
else  if  x < T.key
       then T.left = Delete(T.left,x)
        else   T.right = Delete(T.right,x)
return T
```



T.key = 30, T.left =Delete(T.left,20)

T.key = 20, z.key = 25

Delete(T.right,25)

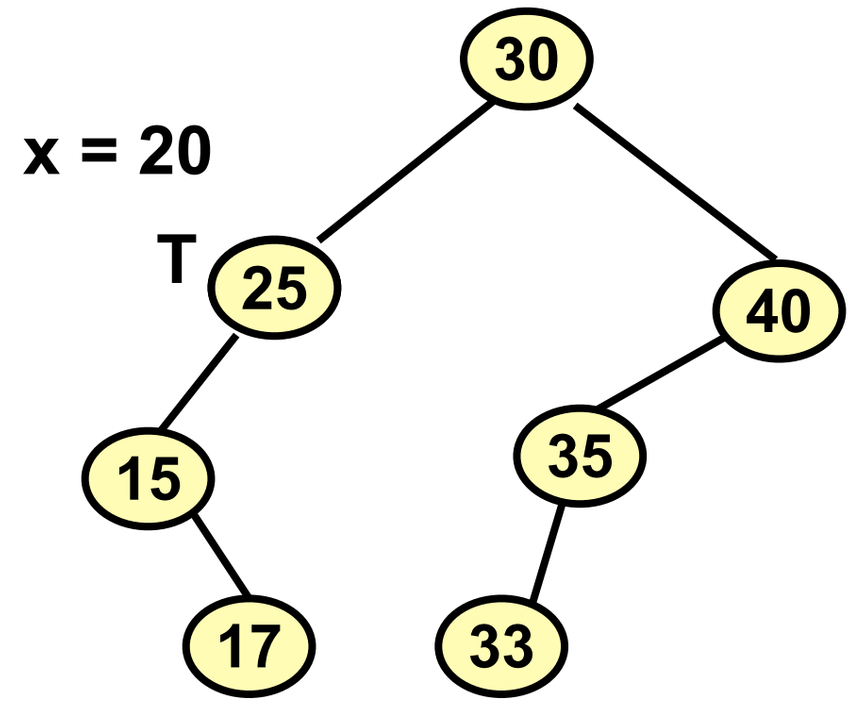
T.key = 25

La cancellazione ricorsiva: caso 2

Solo puntatori ai figli

```
Delete(T,x)
 if T == NIL return T
 if  T.key == x  //    trovato!
     then if   T.left == NIL return T.right
          if   T.right == NIL return T.left
       // T ha un solo figlio o è  una foglia

       // T    ha due figli, si eliminerà il
successivo di T
        z = Minimum(T.right)
        Delete(T.right,z.key)  // z è un nodo con
        al più un figlio
        T.key = z.key  // il valore del successivo
        viene copiato in T
  else  if  x < T.key
            then T.left = Delete(T.left,x)
             else   T.right = Delete(T.right,x)
 return T
```



T.key = 30, T.left = Delete(T.left,20)

T.key = 20, z.key = 25

Delete(T.right,25)

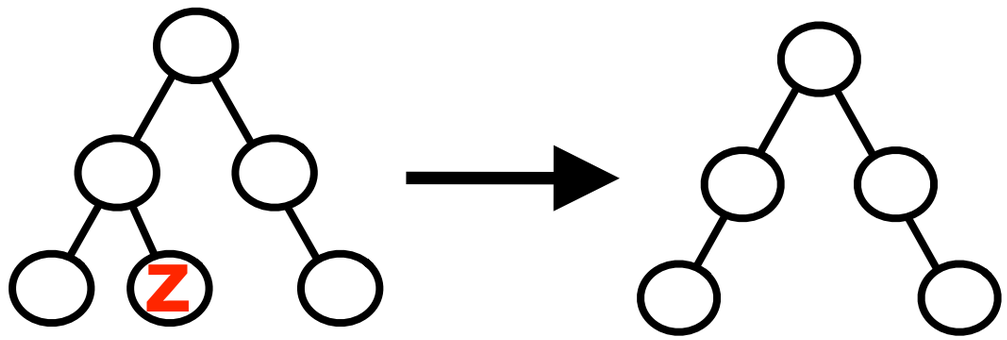
T.key = 25

La cancellazione rivista

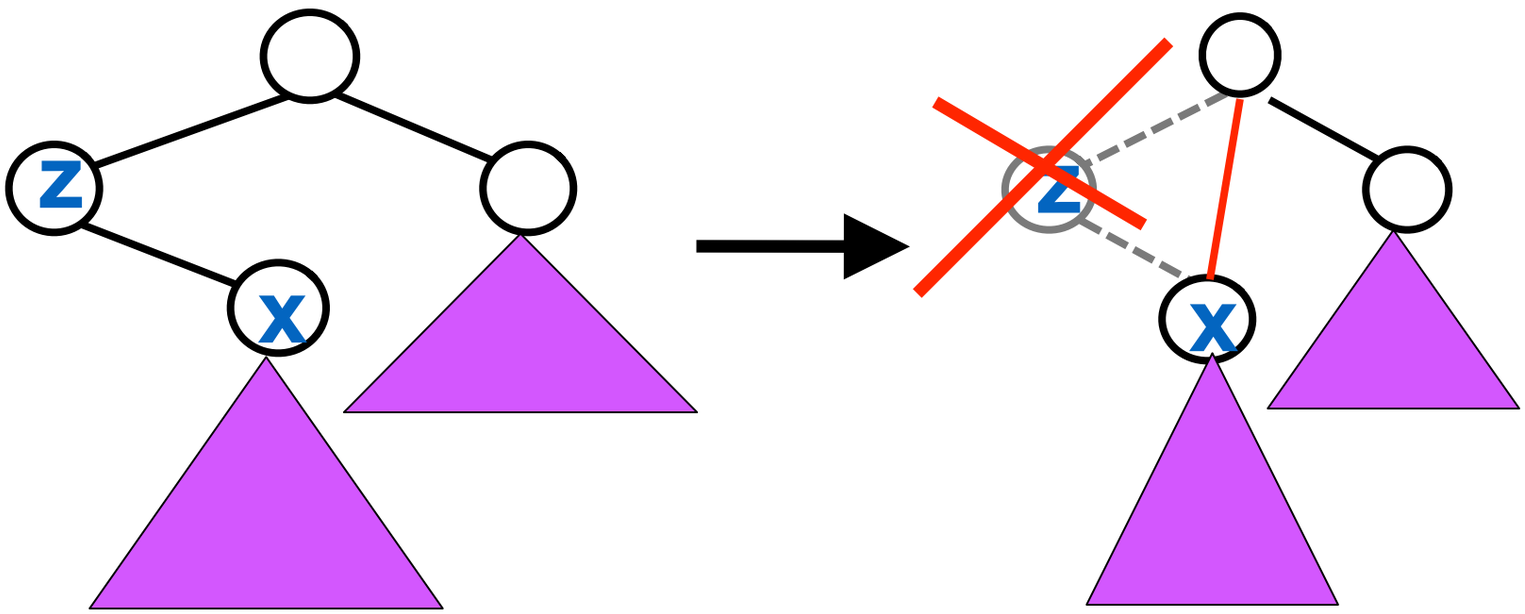
E se qualche struttura dati fa riferimento ai nodi dell'albero?

Nei casi in cui z, il nodo da cancellare, abbia un solo figlio il nodo che si vuole cancellare è quello che viene fisicamente cancellato, quindi non ci sono difficoltà.

1) z è una foglia: la si rimuove

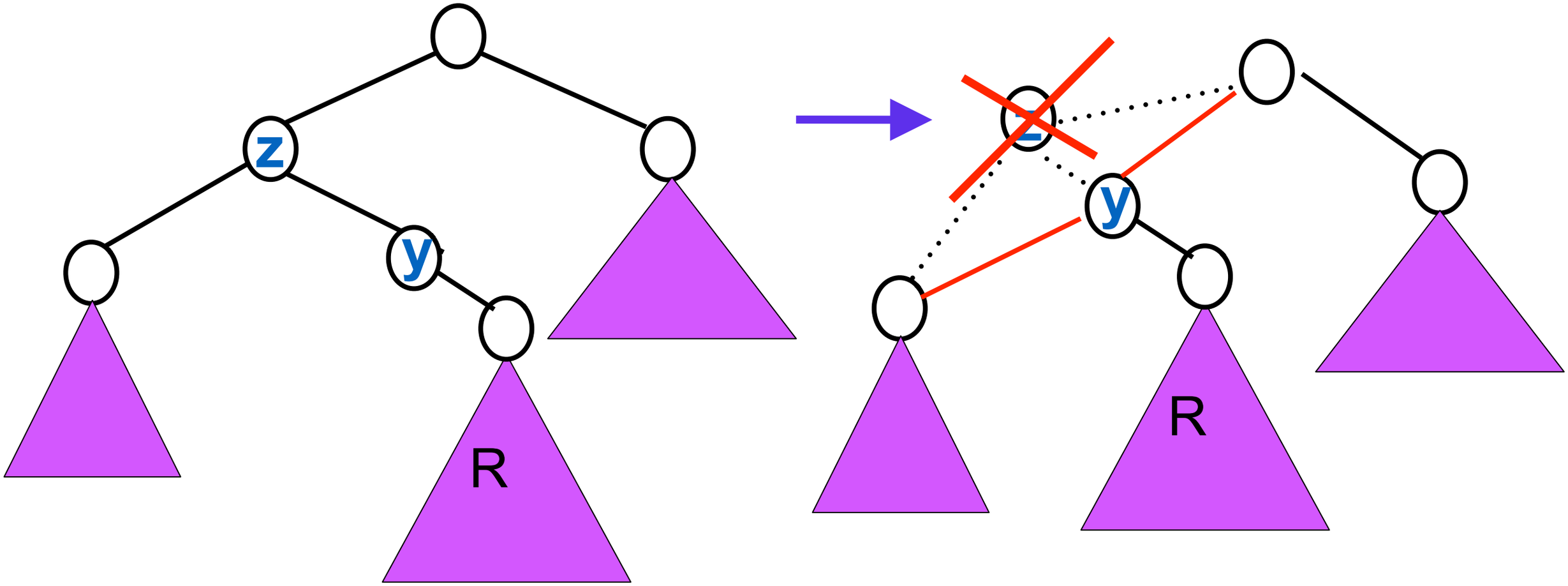


2) z ha un solo figlio x: x diventa figlio del padre di z



E se qualche struttura dati fa riferimento ai nodi dell'albero?

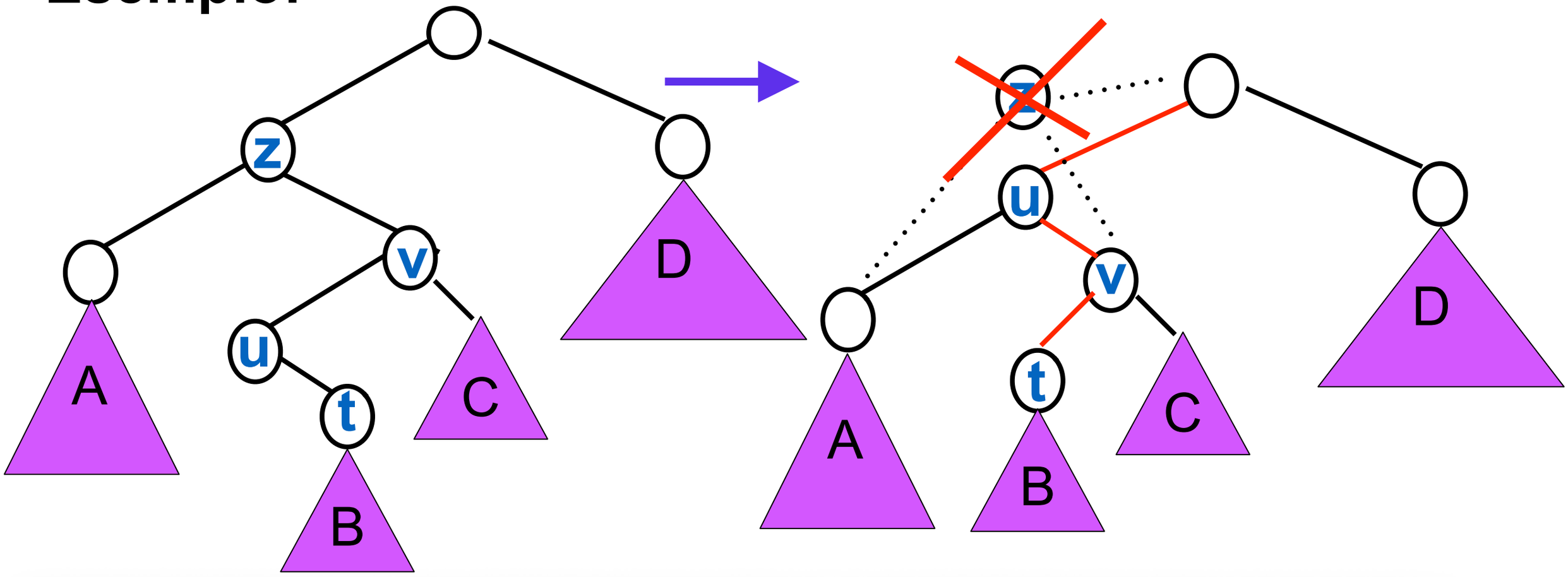
Se z, il nodo da cancellare, ha due figli e il successivo è il suo figlio destro, cioè un nodo senza figlio sinistro, ancora si può cancellare il nodo z mettendo y al suo posto.



E se qualche struttura dati fa riferimento ai nodi dell'albero?

Se z, il nodo da cancellare, ha due figli, si rimpiazza tutto il nodo e non solo la chiave con quella minima nel sottoalbero destro

Esempio:



Il tempo rimane O(h), visto che è sempre determinato dalla necessità di raggiungere il nodo da cancellare e il minimo e le operazioni vere e proprie di cancellazione sono modifiche di puntatori e quindi eseguire in tempo costante

Capire un ABR - 1

Se si inserisce un elemento k in un ABR (BST) T e poi lo si cancella si ottiene lo stesso albero T. Si dica perchè è vero.

Capire un ABR - 1

Se si inserisce un elemento k in un ABR (BST) T e poi lo si cancella si ottiene lo stesso albero T. Si motivi la risposta.

Perchè il nodo inserito è una foglia che viene eliminata senza altri cambiamenti sull'albero.

Capire un ABR - 2

Se si cancella un elemento k da ABR (BST) T e poi lo si reinserisce si ottiene sempre lo stesso albero T?
Si motivi la risposta.

Capire un ABR - 2 - sol

NO, perchè se il nodo cancellato ha uno o due figli poi viene invece inserito come foglia.
Esempio: