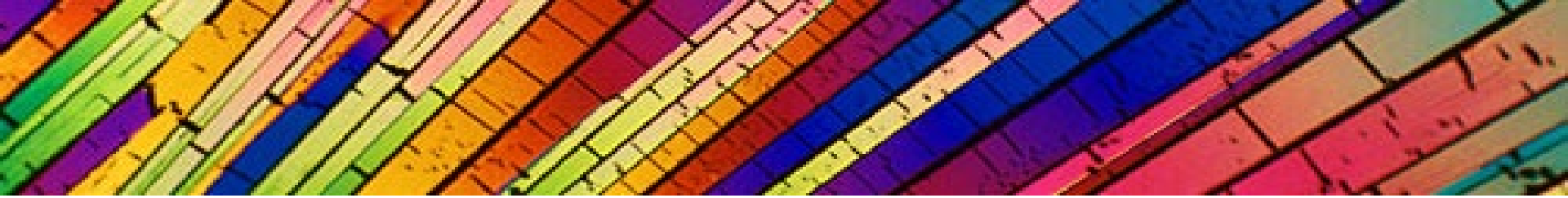
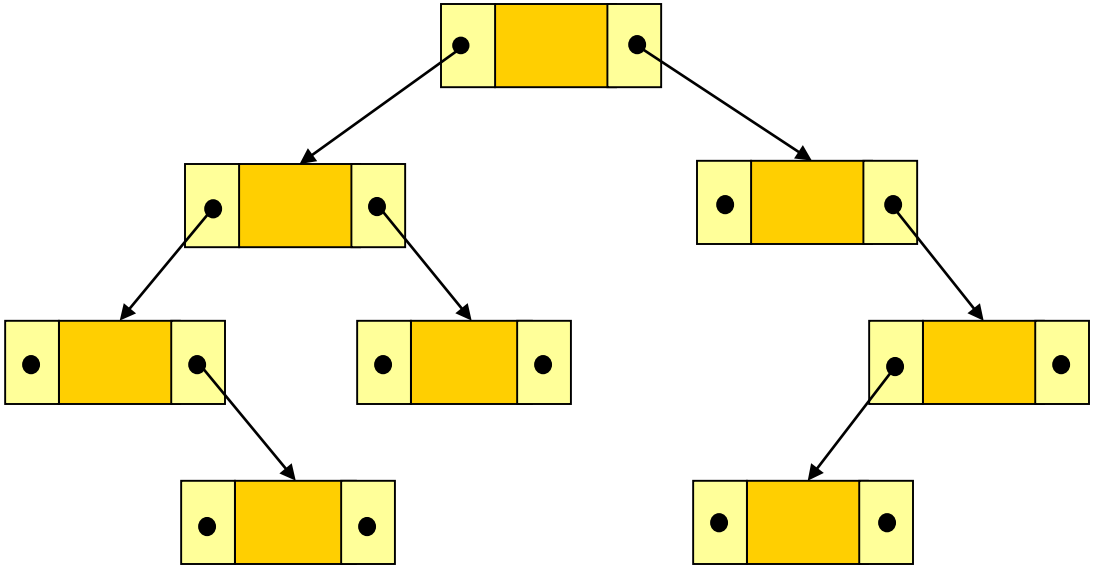


Alberi binari
(introduzione)

Ver. 3

Albero binario

- Struttura di dati bidimensionale formata da nodi costituiti ciascuno dai dati da memorizzare e da due link

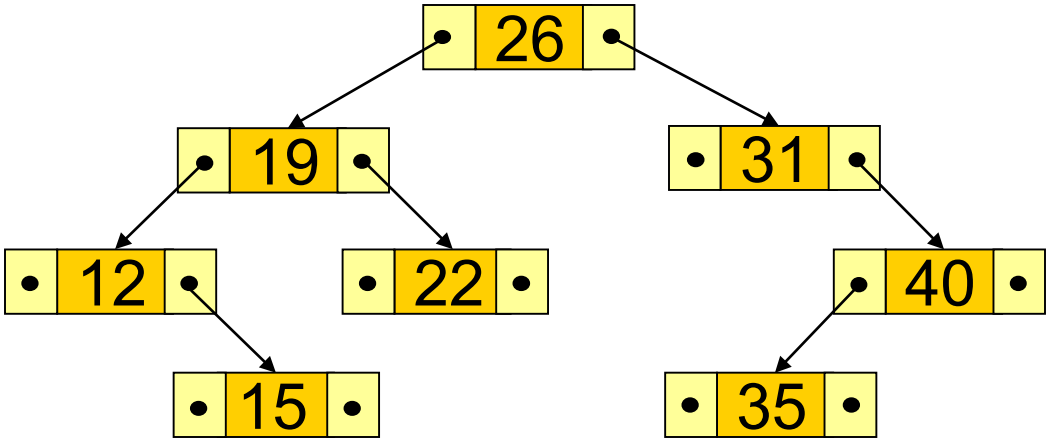


Terminologia

- **Nodo radice:** nodo da cui discende un albero
- **Figli destro e sinistro di un nodo:** nodi puntati dai link di quel nodo (**padre**)
- **Sottoalbero destro e sinistro di un nodo:** alberi che hanno come radici i figli destro e sinistro di quel nodo
- **Nodi fratelli:** nodi figli dello stesso nodo padre
- **Foglia:** nodo terminale (senza figli)

Alberi di ricerca binaria (BST)

- I valori contenuti in ogni sottoalbero sinistro sono minori del valore contenuto nei padri
- I valori contenuti in ogni sottoalbero destro sono maggiori del valore contenuto nei padri
- Non ci sono nodi duplicati



Inserimento di un valore

```
insNode(Tipo valore, Nodo **radice)
  if (*radice == NULL) {
     crea un nodo ;
     inserisci il valore nel nodo;
     imposta a NULL i puntatori nel nodo;
     aggancia il nodo a radice; }
  else {
     if (valore > (*radice)->valore)
        insNode(valore,&(*radice)->destra);
     else if (valore < (*radice)->valore)
        insNode(valore,&(*radice)->sinist);
     else gestisce la duplicazione; }
```

- Si deve poter cambiare il contenuto di `radice` (che è un puntatore): se ne passa il puntatore

Attraversamento dell'albero

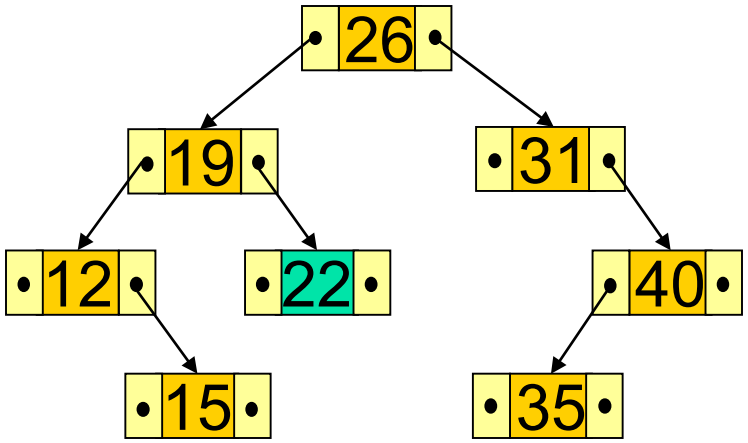
- In order (simmetrica – *fornisce i valori ordinati*)
  - Visita il sottoalbero del figlio di sinistra
  - Elabora il valore del nodo attuale
  - Visita il sottoalbero del figlio di destra
- Pre-order (anticipata)
  - Elabora il valore del nodo attuale
  - Visita il sottoalbero del figlio di sinistra
  - Visita il sottoalbero del figlio di destra
- Post-order (posticipata)
  - Visita il sottoalbero del figlio di sinistra
  - Visita il sottoalbero del figlio di destra
  - Elabora il valore del nodo attuale

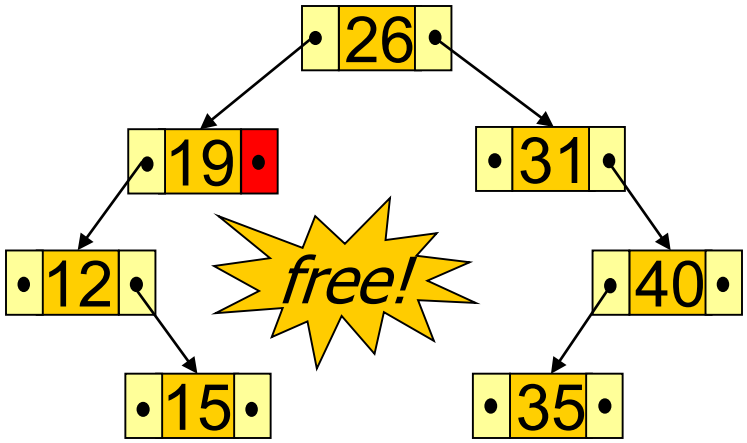
Attraversamento dell'albero

- A livelli (dalla radice, da sinistra a destra)
  - Inserire il nodo radice in una coda
  - Finché la coda non è vuota:
    - Prendere un elemento dalla coda
    - Utilizzare il valore dell'elemento
    - Se il figlio di sinistra non è nullo
      $\Rightarrow$ inserirlo nella coda
    - Se il figlio di destra non è nullo
      $\Rightarrow$ inserirlo nella coda

Rimozione di un valore
Il nodo da eliminare è una foglia

- Nel nodo padre si imposta a NULL il puntatore all'elemento da rimuovere (22)

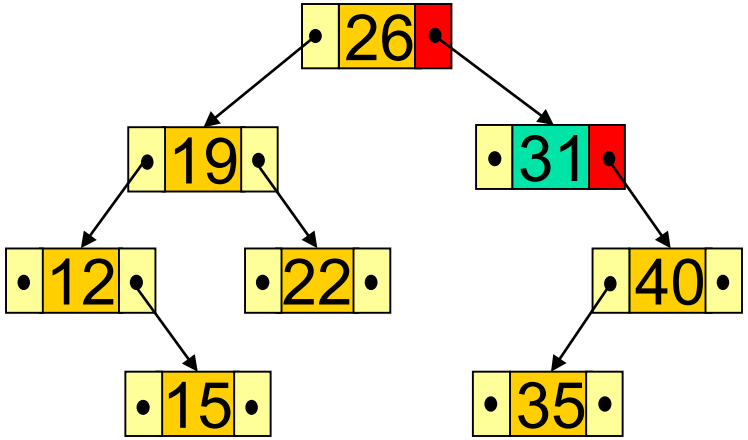


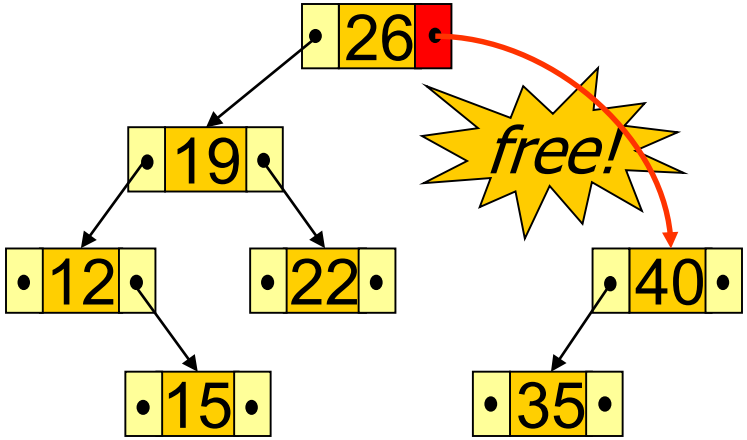


Rimozione di un valore

Il nodo da eliminare ha un solo nodo figlio

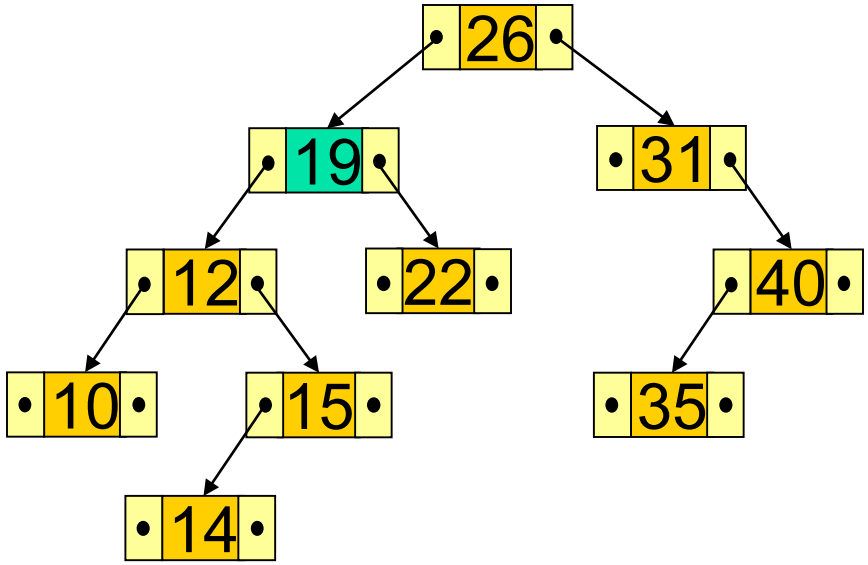
- Nel nodo padre (26) al puntatore al nodo figlio da rimuovere (31) viene assegnato il puntatore all'unico figlio (40) del nodo da rimuovere (31)





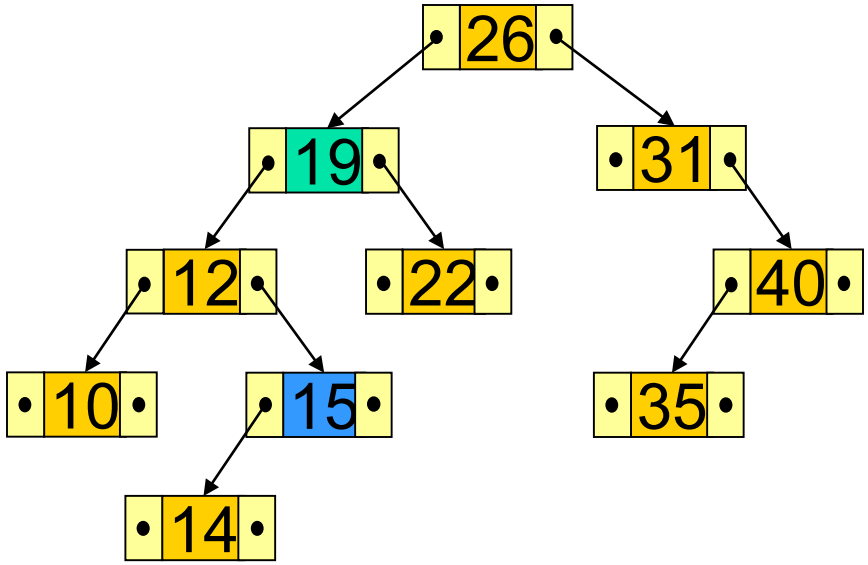
Rimozione di un valore
Il nodo da eliminare ha due nodi figli - 1/7

- Si identifica l'elemento da rimuovere (19)



Rimozione di un valore
Il nodo da eliminare ha due nodi figli - 2/7

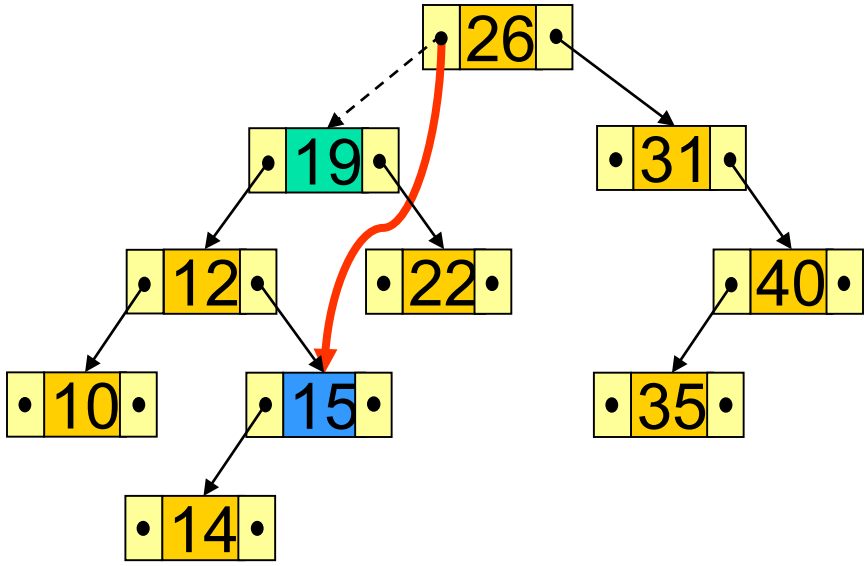
- Si cerca il *nodo sostitutivo* (15): è il nodo contenente il più grande valore minore di quello nel nodo da rimuovere (19), lo si trova percorrendo tutto il ramo destro del suo figlio di sinistra



Rimozione di un valore

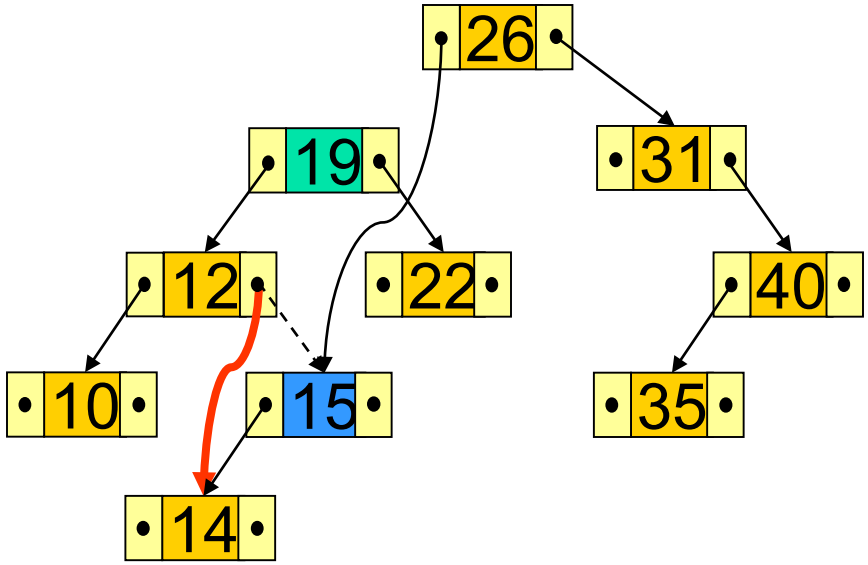
Il nodo da eliminare ha due nodi figli - 3/7

- Il link del padre (26) al nodo da rimuovere (19) viene fatto puntare al nodo sostitutivo (15)



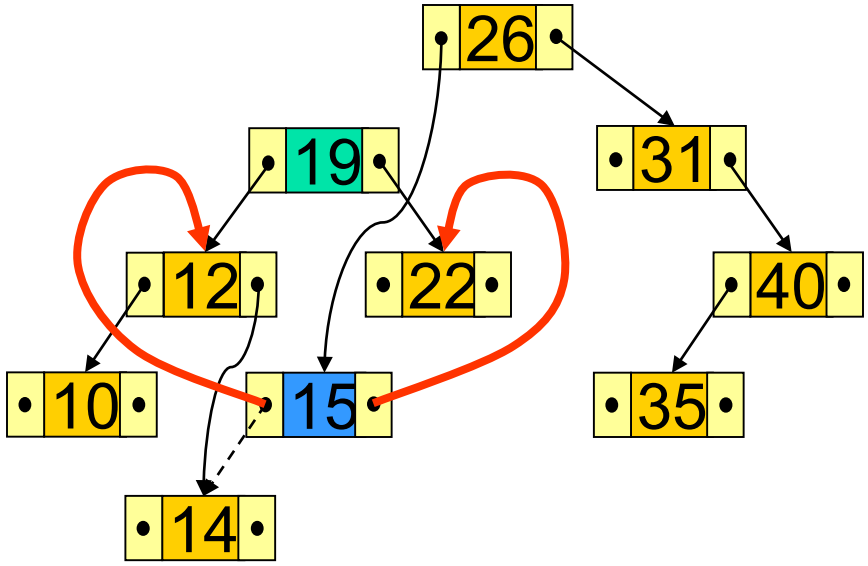
Rimozione di un valore
Il nodo da eliminare ha due nodi figli - 4/7

- Il link destro del padre (12) del nodo sostitutivo viene fatto puntare al sottoalbero sinistro del nodo sostitutivo (14), se non c'è si mette NULL. N.B. non può esserci il sottoalbero destro perché sarebbe maggiore



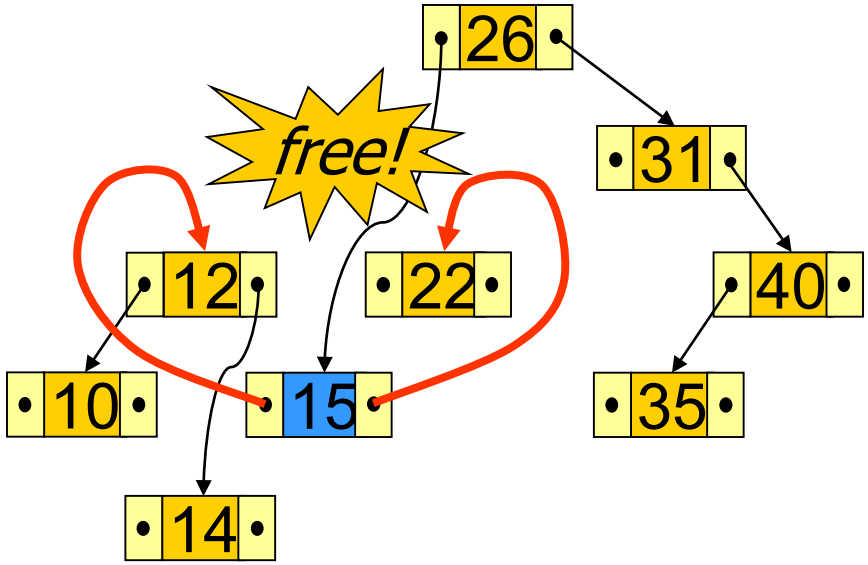
Rimozione di un valore
Il nodo da eliminare ha due nodi figli - 5/7

- Il link destro del nodo sostitutivo (15) viene fatto puntare al sottoalbero destro (22) del nodo da sostituire, il link sinistro al sinistro (12)



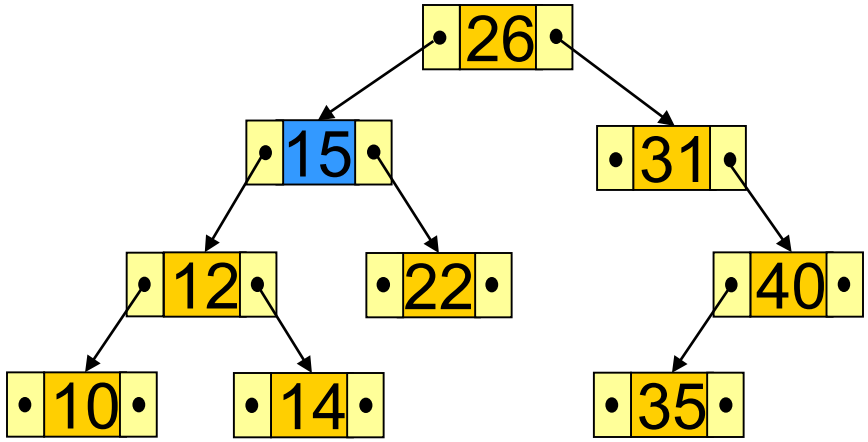
Rimozione di un valore
Il nodo da eliminare ha due nodi figli - 6/7

- Si dealloca il nodo da rimuovere



Rimozione di un valore
Il nodo da eliminare ha due nodi figli - 7/7

- Risultato finale



Bilanciamento

- La forma dell'albero cambia a seconda dell'ordine con cui vengono memorizzati i valori
- Un albero *bilanciato* è circa simmetrico e il tempo di ricerca è logaritmico rispetto al numero dei valori memorizzati
- Se i valori vengono introdotti già ordinati la struttura dell'albero degenera in una lista e il tempo di ricerca è lineare

Esercizi

1. Si scriva un programma costruisca un albero binario introducendo dei valori. I valori duplicati vengano conteggiati. Si scrivano funzioni per:
   1. la visita dell'albero nei modi: inOrder, preOrder e postOrder
   2. la ricerca di un elemento

Esercizi

2. Scrivere un programma che analizzi un testo memorizzando e contando ogni occorrenza delle parole che lo costituiscono. Si usi una struttura ad albero che tenga conto del numero delle ripetizioni delle parole. L'output finale delle parole ordinate alfabeticamente con il corrispondente numero di ripetizioni avvenga in un file di testo sequenziale.