ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ

Δομές δεδομένων

Ενότητα 5η: Υλοποίηση Λεξικών με Ισοζυγισμένα Δένδρα

Παναγιώτα Φατούρου

Τμήμα Επιστήμης Υπολογιστών

ΕΝΟΤΗΤΑ 5
ΥΛΟΠΟΙΗΣΗ ΛΕΞΙΚΩΝ ΜΕ ΙΣΟΖΥΓΙΣΜΕΝΑ ΔΕΝΔΡΑ

Ισοζυγισμένα Δένδρα

Χρονική Πολυπλοκότητα αναζήτησης σε δομές που έχουν ήδη διδάχθει:

- Στατική Μη-Ταξινομημένη Λίστα -> O(n), όπου n το πλήθος των κόμβων
- Στατική Ταξινομημένη Λίστα -> O(n)
- Δυναμική μη-ταξινομημένη λίστα -> O(n)
- Δυναμική ταξινομημένη λίστα -> O(n)
- Μη-ταξινομημένα Δένδρα -> O(n), όπου n το πλήθος των κόμβων
- Ταξινομημένα Δένδρα - > O(h), όπου h το ύψος του δένδρου
  Τι τιμές μπορεί να πάρει το h;

Γραμμική πολυπλοκότητα

Αναζητούνται δομές δεδομένων για την υλοποίηση δυναμικών λεξικών που να παρουσιάζουν καλύτερη χρονική πολυπλοκότητα:

- π.χ., όλες οι λειτουργίες να έχουν χρονική πολυπλόκοτητα O(logn)

Δένδρα AVL (Adel'son, Vel'skii & Landis)

Ένα ταξινομημένο δυαδικό δένδρο Τ ονομάζεται AVL (ή ισοζυγισμένο κατ' ύψος) αν και μόνο αν τα ύψη των δύο υποδένδρων κάθε κόμβου v διαφέρουν το πολύ κατά ένα.

Για κάθε κόμβο v ενός δυαδικού δένδρου Τ, ορίζουμε το αριστερό ύψος του v (LeftHeight(v)) ως εξής:

$$LeftHeight(v) = \begin{cases} 0 & \text{αν v->LC == NULL} \\ 1 + Height(v->LC) & \text{διαφορετικά} \end{cases}$$

Ομοίως, ορίζουμε το δεξιό ύψος του v (RightHeight(v)) ως εξής:

$$RightHeight(v) = \begin{cases} 0 & \text{αν v->RC == NULL} \\ 1 + Height(v->RC) & \text{διαφορετικά} \end{cases}$$

Το LeftHeight(T) (RightHeight(T)) του δένδρου ισούται με το LeftHeight(R) (RightHeight(R), αντίστοιχα), όπου R είναι ο κόμβος ρίζα του Τ.

Δένδρα AVL

Το balance (ισοζύγιο) του κόμβου v ορίζεται ως εξής:
balance(v) = RightHeight(v) – LeftHeight(v).

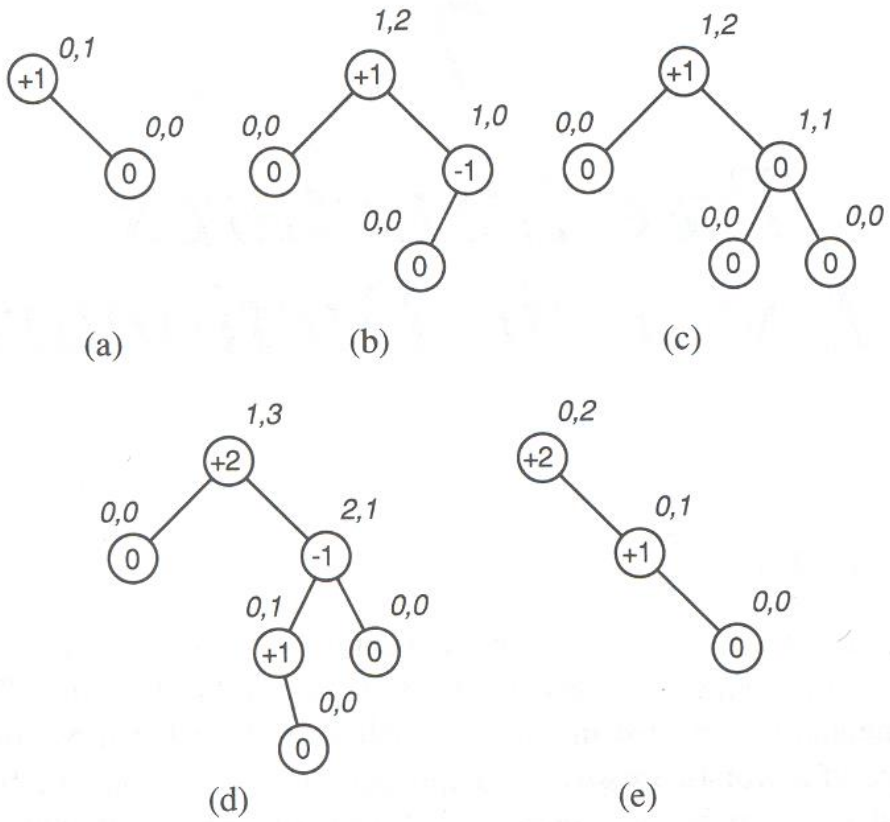
Σε ένα δένδρο AVL, κάθε κόμβος πρέπει να έχει balance 0, +1 ή -1.

Πρόταση 1
Κάθε υποδένδρο ενός δένδρου AVL είναι επίσης δένδρο AVL.

Θεώρημα
Το ύψος h ενός δένδρου AVL με n στοιχεία είναι O(logn).

Τι μπορούμε να συμπεράνουμε για την πολυπλοκότητα της AVLTreeLookUp()?

Σχήμα 7.1: Lewis & Denenberg, Data Structures & Their Algorithms, Addison-Wesley, 1991



Τα δένδρα (a), (b) και (c) είναι δένδρα AVL, ενώ τα (d) και (e) δεν είναι.

Δένδρα AVL – Απόδειξη Θεωρήματος

- Αναζητούμε ένα πάνω φράγμα στο μήκος h του μακρύτερου μονοπατιού οποιουδήποτε δένδρου AVL με n κόμβους.
- Έστω ένας οποιοσδήποτε ακέραιος h. Θα εστιάσουμε στο ερώτημα

  «Ποιος είναι ο μικρότερος ακέραιος n τ.ω. να υπάρχει ένα δένδρο AVL ύψους h με n κόμβους;»
- W_h: σύνολο όλων των δένδρων AVL ύψους h με ελάχιστο πλήθος κόμβων.
- w_h: πλήθος κόμβων σε οποιοδήποτε από αυτά τα δένδρα.
- $w_0 = 1$ και $w_1 = 2$.
- Έστω ότι T είναι οποιοδήποτε δένδρο AVL στο W_h και έστω T_L και T_R το αριστερό και το δεξιό υποδένδρο του T, αντίστοιχα.

Δένδρα AVL – Απόδειξη Θεωρήματος

- Αφού το T έχει ύψος h, είτε το T_L ή το T_R έχει ύψος h-1 (ας υποθέσουμε χωρίς βλάβη της γενικότητας ότι το T_R έχει ύψος h-1).

- Το T_R είναι δένδρο AVL ύψους h-1 και μάλιστα θα πρέπει να έχει το μικρότερο πλήθος κόμβων μεταξύ των δένδρων AVL ύψους h-1 (αφού στην αντίθετη περίπτωση, θα μπορούσε να αντικατασταθεί από κάποιο δένδρο AVL ύψους h-1 με λιγότερους κόμβους και να οδηγηθούμε έτσι σε ένα δένδρο AVL ύψους h με λιγότερους κόμβους από ότι το T, το οποίο αντιτίθεται στον ορισμό του T).

Δένδρα AVL – Απόδειξη Θεωρήματος

- Άρα, $T_R \in W_{h-1}$.
- Αφού το T είναι δένδρο AVL, το T_L έχει ύψος είτε h-1 ή h-2. Εφόσον το T είναι δένδρο με ελάχιστο πλήθος κόμβων μεταξύ εκείνων με ύψος h, θα πρέπει να ισχύει $T_L \in W_{h-2}$.
- Επομένως:

$$w_h = 1 + w_{h-2} + w_{h-1}$$

- Μπορεί να αποδειχθεί (επαγωγικά) ότι $w_h = F_{h+3} - 1$, όπου F_i είναι ο i-οστός όρος της ακολουθίας Fibonacci, για τον οποίο ισχύει ότι $F_i > \varphi^i/\sqrt{5} - 1$,
  όπου $\varphi = (1+\sqrt{5})/2$.
- Επομένως, $n \geq w_h > \varphi^{h+3}/\sqrt{5} - 2$.
- Από την παραπάνω ανίσωση προκύπτει ότι h = O(logn).

Δένδρα AVL – Εισαγωγή

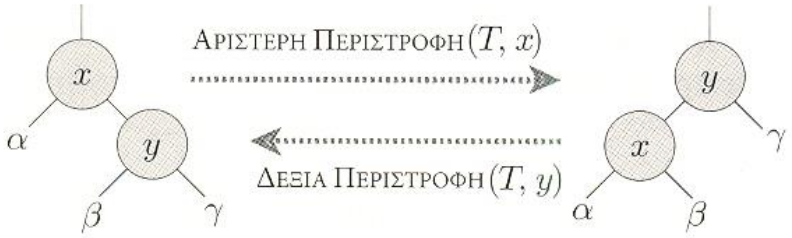
Πως μπορούμε να υλοποιήσουμε την Insert()? Που διαφέρει από την Insert() σε δυαδικό δένδρο αναζήτησης?

1η Προσπάθεια Σχεδίασης Αλγορίθμου Εισαγωγής

- Ακολουθώντας το γνωστό αλγόριθμο εισαγωγής σε δένδρο δυαδικής αναζήτησης, βρίσκουμε το μονοπάτι από τη ρίζα προς τον κατάλληλο κόμβο (έστω v) στον οποίο θα γίνει η εισαγωγή.
- Υπολογίζουμε τα νέα balances για κάθε κόμβο του μονοπατιού.
- Αν το balance όλων των κόμβων του μονοπατιού εξακολουθεί να είναι 0, +1 ή -1, η διαδικασία εισαγωγής τερματίζει.
- Αν το balance κάποιου κόμβου αλλάζει σε +2 ή σε –2, θα πρέπει να ακολουθηθεί διαδικασία προσαρμογής του balance.
  - Η διαδικασία αυτή θα έχει ως αποτέλεσμα το δένδρο να εξακολουθεί να είναι ταξινομημένο δυαδικό δένδρο με τους ίδιους κόμβους και κλειδιά, αλλά με όλους τους κόμβους να έχουν balance 0, 1, ή -1.

Δένδρα AVL – Περιστροφές

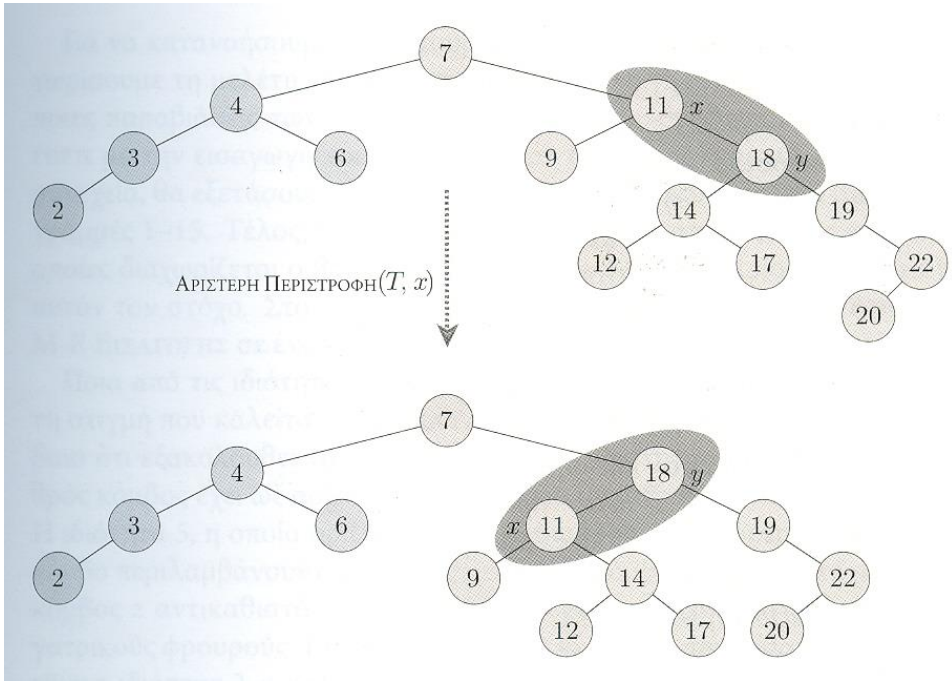
Μια περιστροφή είναι μια τοπική λειτουργία σε ένα ταξινομημένο δένδρο η οποία διατηρεί την ιδιότητα της ταξινόμησης.



Σχήμα 13.2: Cormen, Leiserson, Rivest & Stein, Εισαγωγή στους Αλγορίθμους, Πανεπιστημιακές Εκδόσεις Κρήτης, 2006

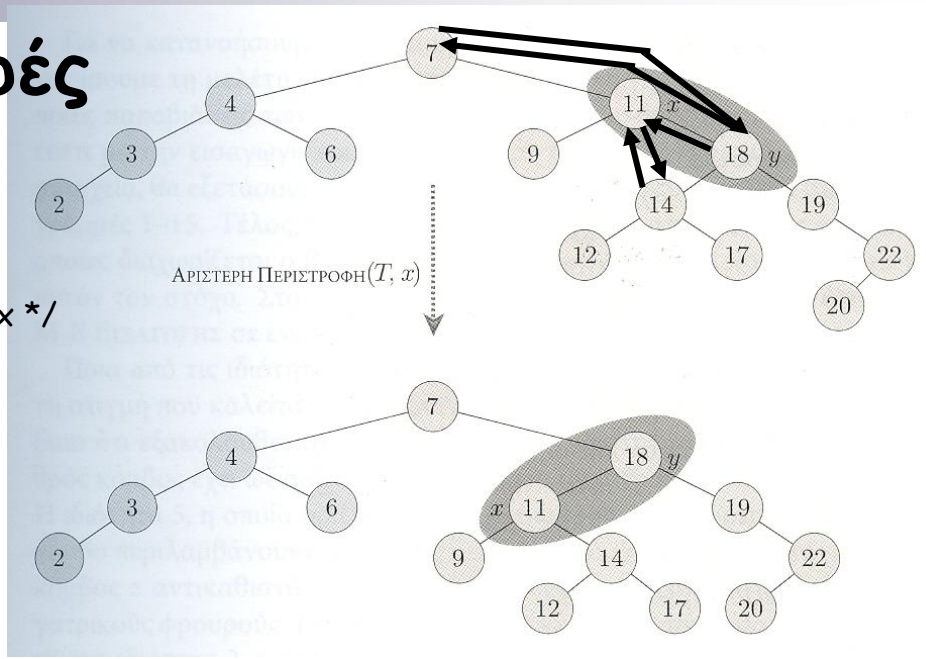
Αριστερή Περιστροφή γύρω από έναν κόμβο x (ο οποίος έχει δεξιό θυγατρικό κόμβο y != nill)

- Στρέφει τους x και y κατά αντίστροφη φορά αυτής των δεικτών του ρολογιού.
- Καθιστά τον y νέα ρίζα του υποδένδρου, μετατρέποντας τον x σε αριστερό παιδί του y και το αριστερό υποδένδρο του y σε δεξιό υποδένδρο του x.



Σχήμα 13.3: Cormen, Leiserson, Rivest & Stein, Εισαγωγή στους Αλγορίθμους, Πανεπιστημιακές Εκδόσεις Κρήτης, 2006

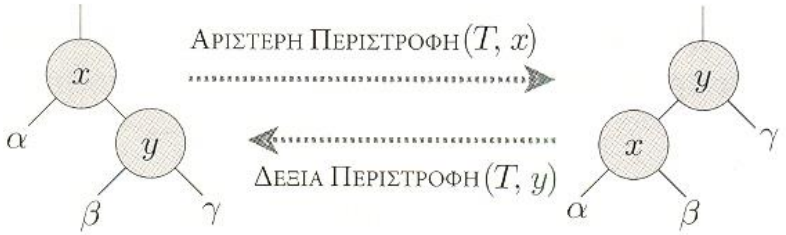
Δένδρα AVL – Περιστροφές




```
Procedure LeftRotate(pointer R, pointer x) {
    y = x->RC;              // ορισμός του y

    /* μετατροπή αριστερού υποδένδρου y σε δεξιό υποδένδρο x */
    x->RC = y->LC;
    if (y->LC != NULL) y->LC->p = x;

/* νέος πατρικός κόμβος του y γίνεται ο πρώην
πατρικός κόμβος του x και αν δεν υπάρχει
τέτοιος κόμβος, νέα ρίζα του δένδρου γίνεται το y */
    y->p = x->p;
    if (x->p == NULL) { y-> LC = x; x->p = y;  return y; }
    if (x == x->p->LC) // αν ο x είναι αριστερό παιδί του πατέρα του
          x->p->LC = y;
    else x->p->RC = y;
    y->LC = x;    // αριστερό παιδί του y γίνεται το x
    x->p = y;  // πατέρας του x γίνεται το y
}
```



Μια δεξιά περιστροφή γίνεται με συμμετρικό τρόπο.

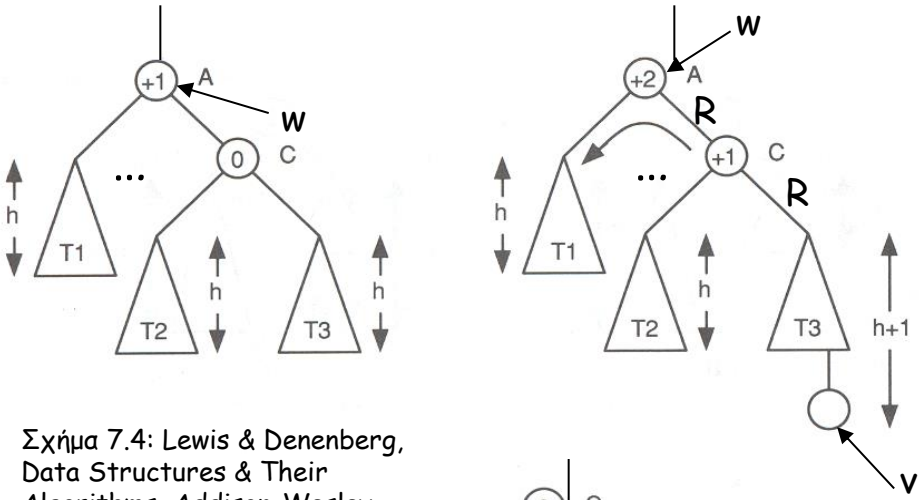
Χρονική Πολυπλοκότητα μιας περιστροφής: $O(1)$

Δένδρα AVL – Εισαγωγή

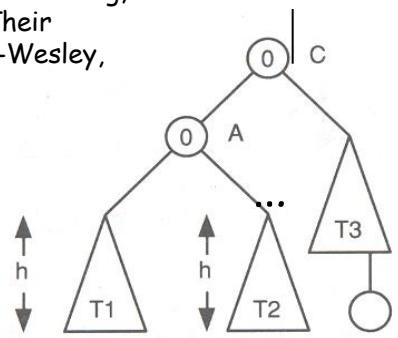
Ο πρώτος κόμβος στο μονοπάτι από τον εισαχθέντα κόμβο v προς τη ρίζα, του οποίου το balance ήταν +1 ή -1 (πριν την εισαγωγή) ονομάζεται κρίσιμος κόμβος.

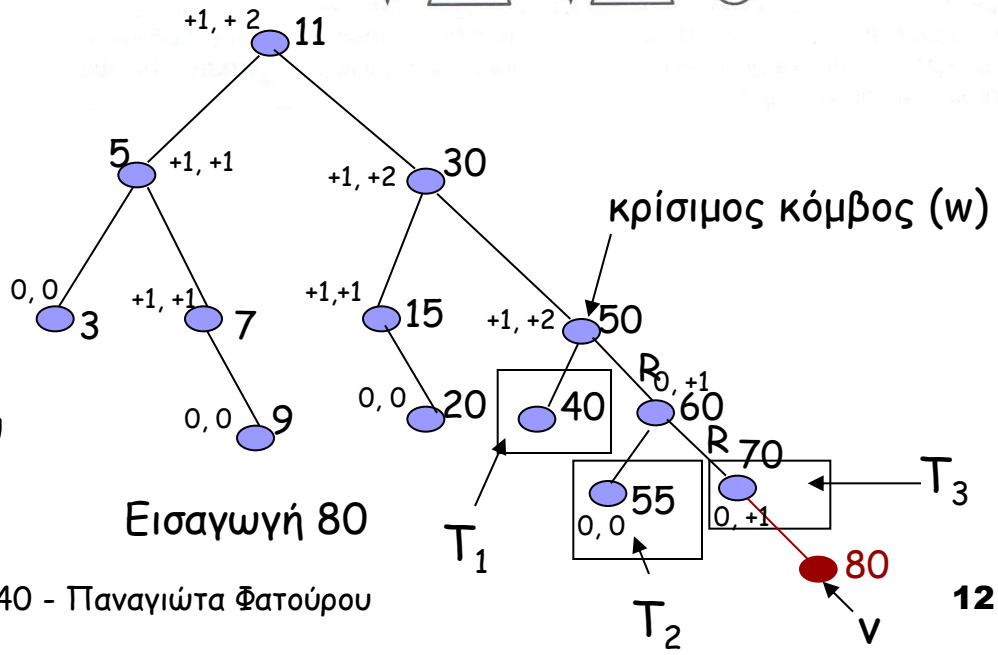
Αν ο κόμβος αυτός αποκτά balance +2 ή -2 μετά την εισαγωγή, τότε πρέπει να γίνουν κατάλληλες ενέργειες ώστε να διορθωθεί το balance του.

Διακρίνουμε περιπτώσεις ανάλογα με το είδος των δύο πρώτων ακμών του μονοπατιού από τον κρίσιμο κόμβο w προς τον εισαχθέντα κόμβο v.



Σχήμα 7.4: Lewis & Denenberg, Data Structures & Their Algorithms, Addison-Wesley, 1991

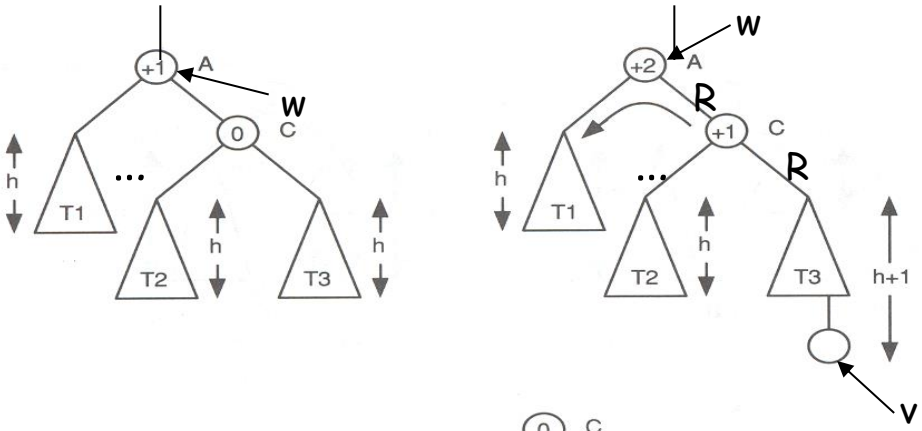


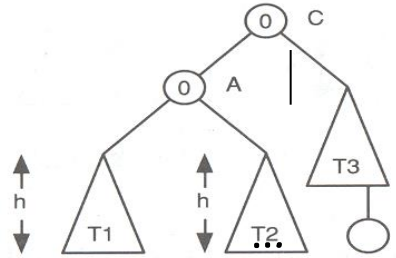


Δένδρα AVL – Εισαγωγή

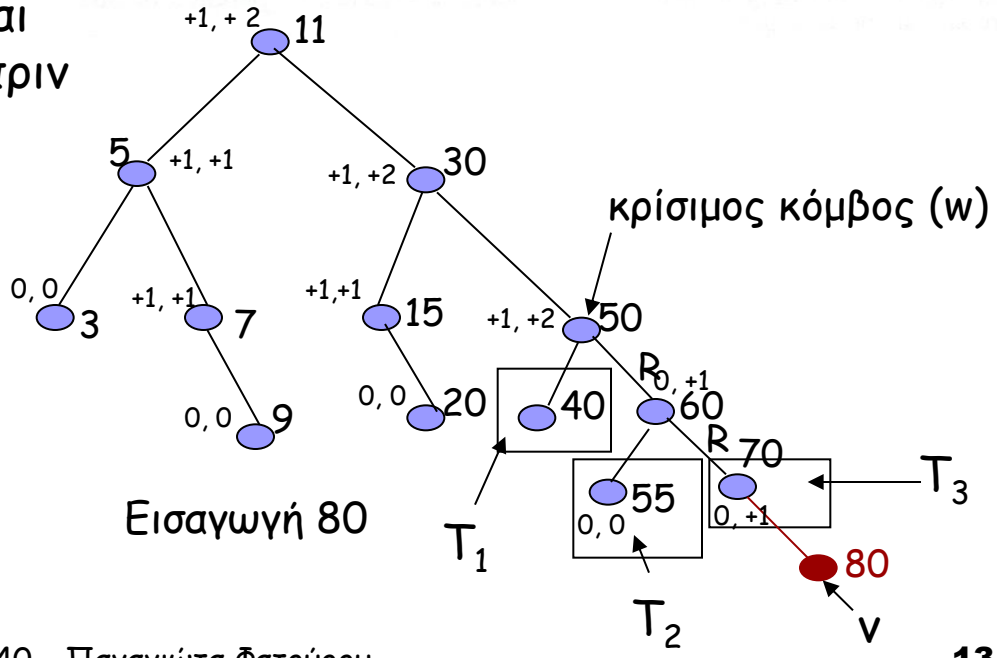
Περίπτωση RR (Right-Right):
Και οι δύο ακμές οδηγούν δεξιά.

- Εκτελούμε μια αριστερή περιστροφή γύρω από τον κρίσιμο κόμβο.



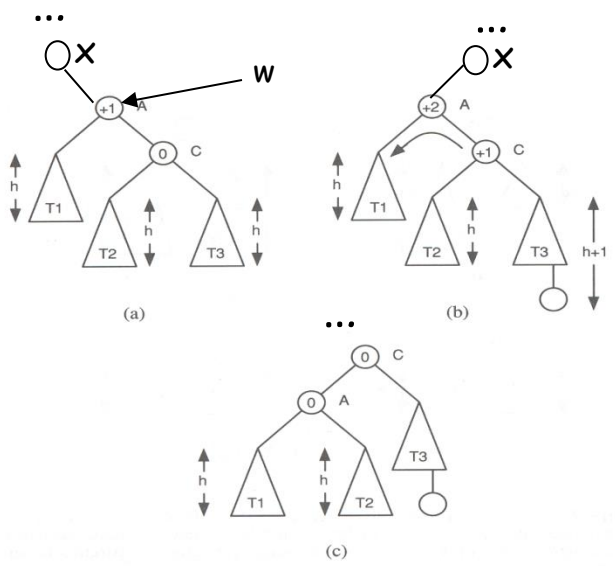


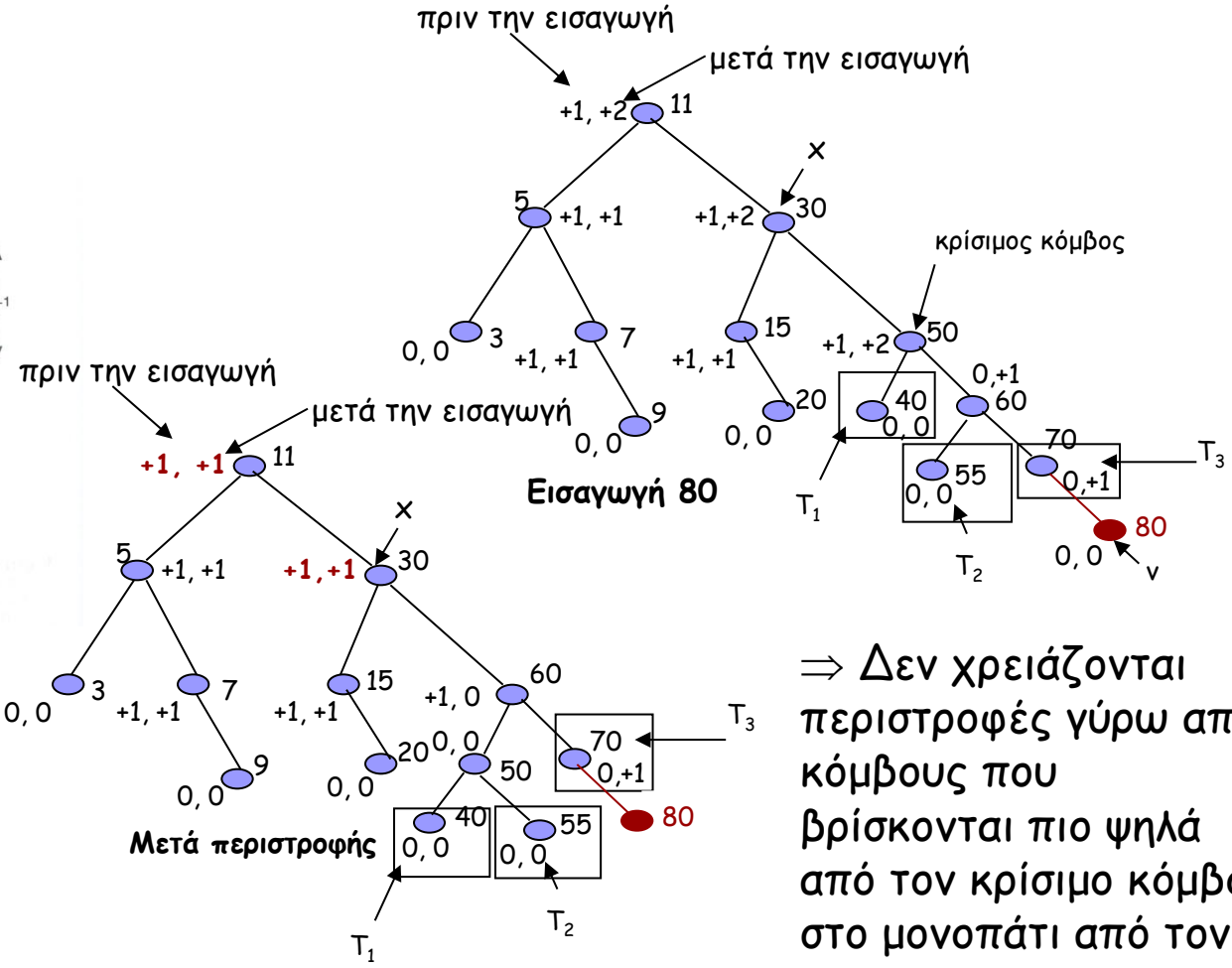
Παρατήρηση: Έστω x ο πατρικός κόμβος του w και έστω ότι ο w είναι δεξιό παιδί του x. Το ύψος του w πριν την εισαγωγή και μετά την περιστροφή είναι το ίδιο (στο παράδειγμα h+2)



Δένδρα AVL – Εισαγωγή – Περίπτωση RR

- Όλοι οι κόμβοι στο μονοπάτι από τον κρίσιμο κόμβο προς τη ρίζα έχουν το ίδιο balance μετά την περιστροφή με αυτό που είχαν πριν την εισαγωγή του v στο δένδρο.





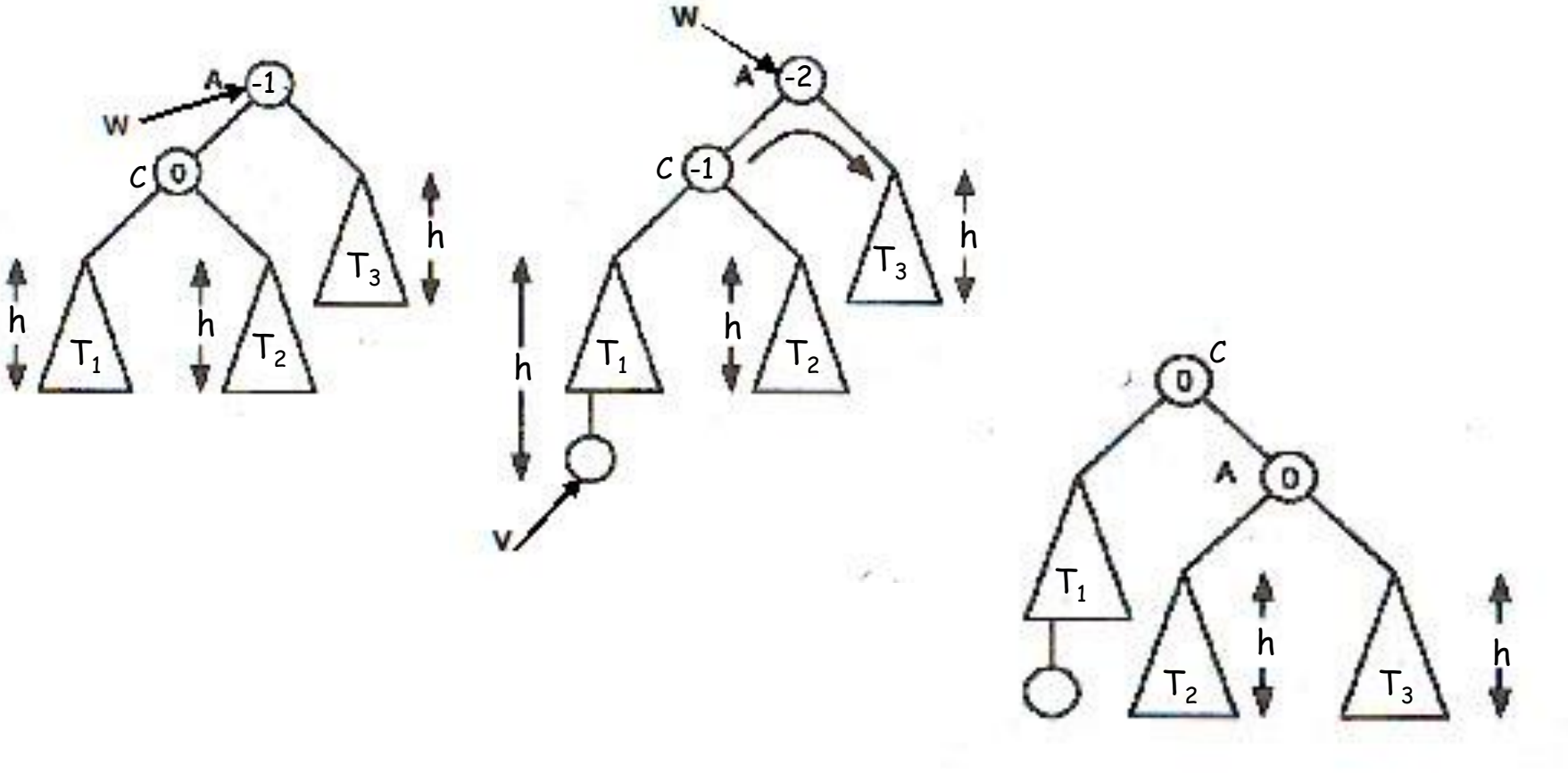
⇒ Μία περιστροφή γύρω από τον κρίσιμο κόμβο αρκεί.

⇒ Δεν χρειάζονται περιστροφές γύρω από κόμβους που βρίσκονται πιο ψηλά από τον κρίσιμο κόμβο στο μονοπάτι από τον v προς τη ρίζα.

Δένδρα AVL – Εισαγωγή – Περίπτωση LL

Περίπτωση LL: Και οι δύο ακμές οδηγούν αριστερά.

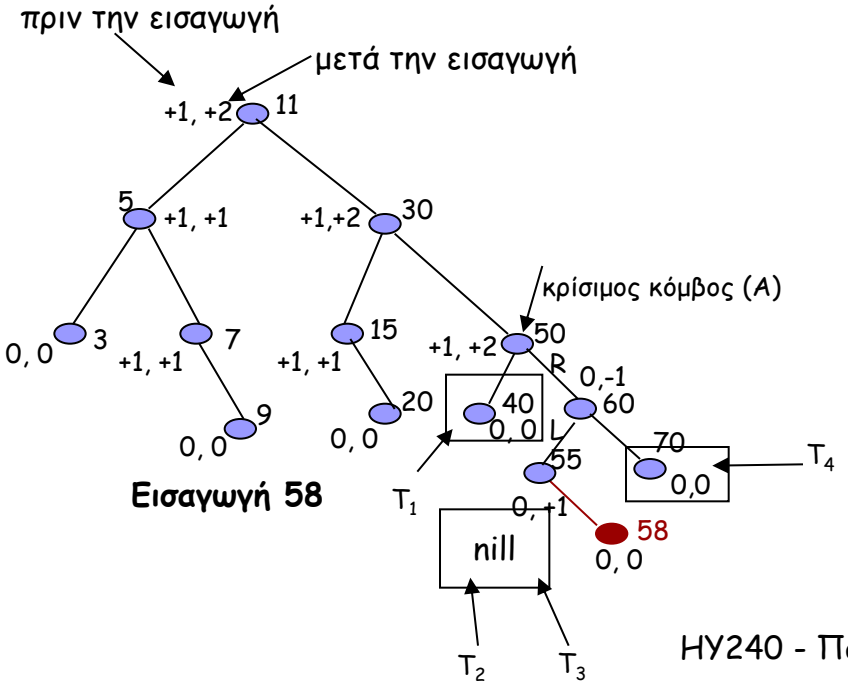
- Είναι συμμετρική της περίπτωσης RR!
- Μία δεξιά περιστροφή γύρω από τον κρίσιμο κόμβο αρκεί για να επιλυθεί το πρόβλημα με το balance του!

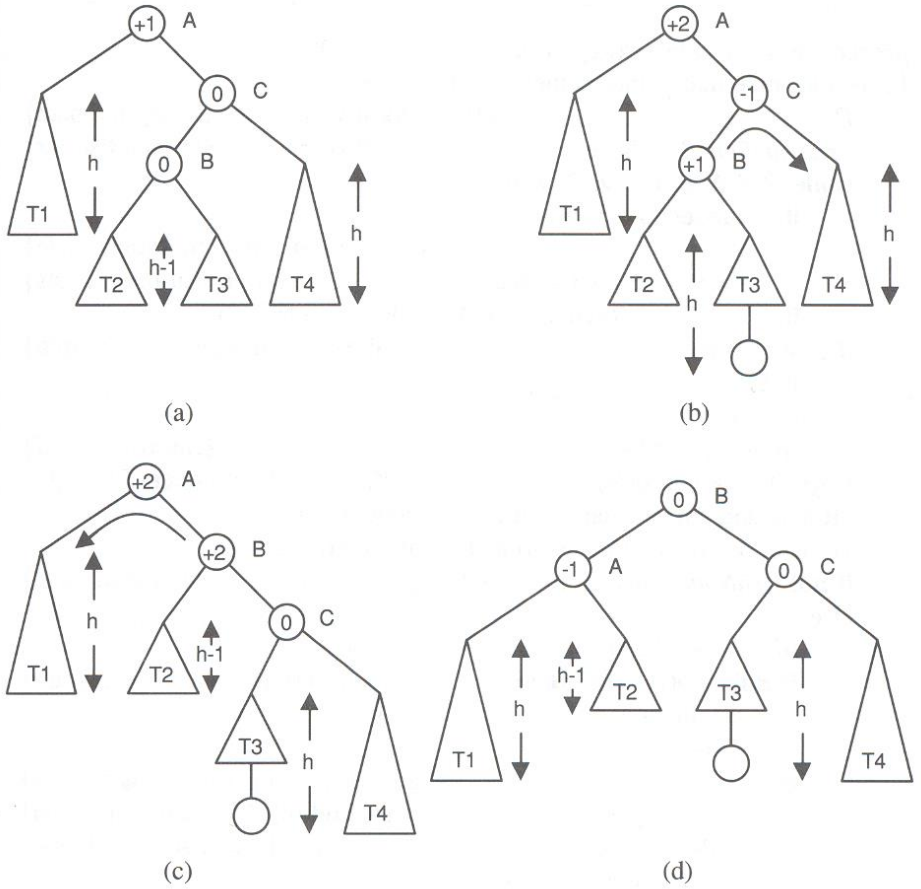


Δένδρα AVL – Εισαγωγή – Περίπτωση RL

Περίπτωση RL: Η πρώτη ακμή οδηγεί δεξιά και η δεύτερη αριστερά.

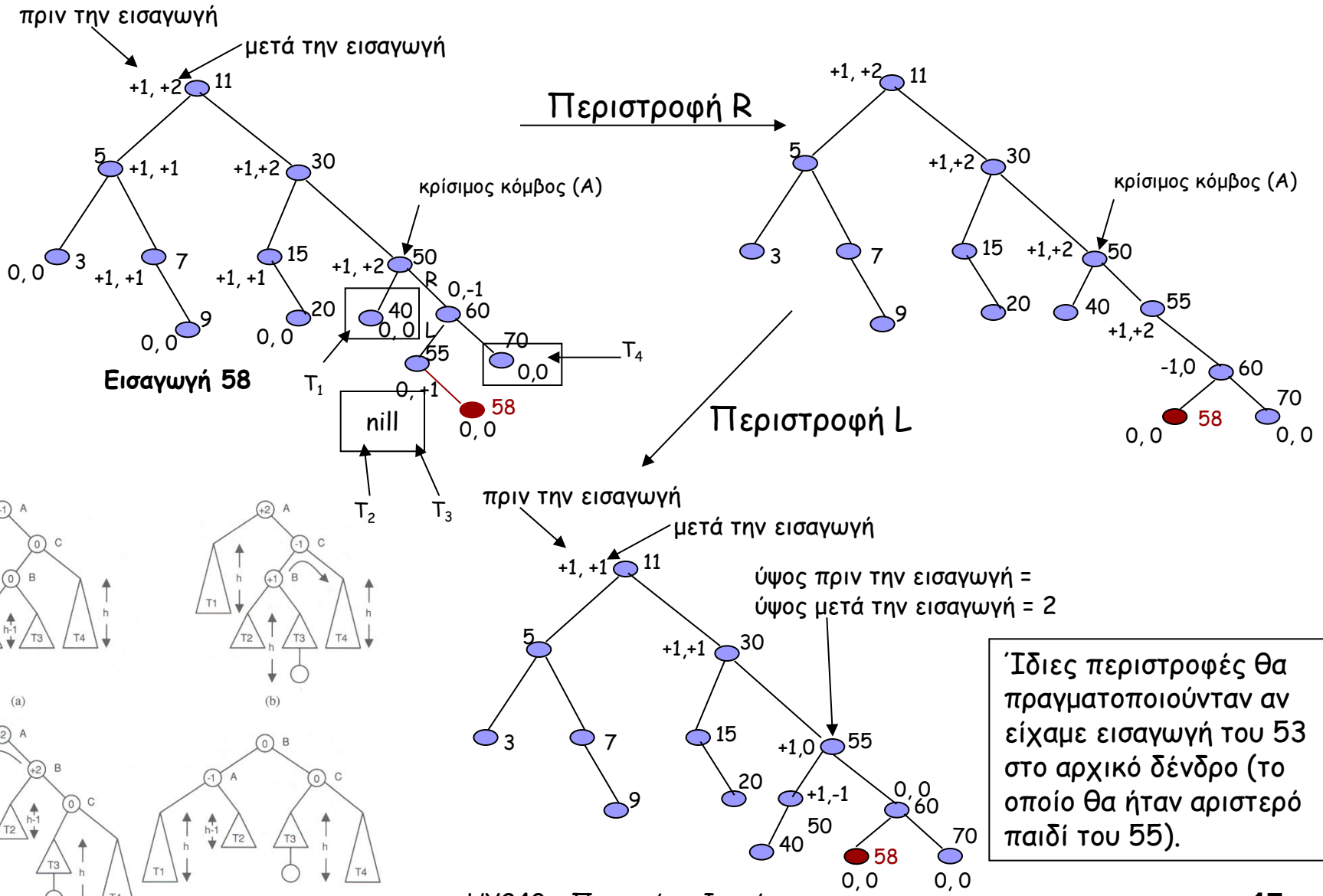
- Απαιτούνται δύο περιστροφές, μια δεξιά περιστροφή γύρω από τον επόμενο του κρίσιμου κόμβου στο μονοπάτι που οδηγεί στον ν και μια αριστερή περιστροφή γύρω από τον κρίσιμο κόμβο.





Σχήμα 7.5: Lewis & Denenberg, Data Structures & Their Algorithms, Addison-Wesley, 1991

Δένδρα AVL – Εισαγωγή – Περίπτωση RL



Ίδιες περιστροφές θα πραγματοποιούνταν αν είχαμε εισαγωγή του 53 στο αρχικό δένδρο (το οποίο θα ήταν αριστερό παιδί του 55).

Δένδρα AVL – Εισαγωγή – Περίπτωση LR

Περίπτωση LR: Η πρώτη ακμή οδηγεί αριστερά και η δεύτερη δεξιά.

- Είναι συμμετρική της περιπτώσεως RL.
- Απαιτούνται δύο περιστροφές, μια αριστερή περιστροφή γύρω από τον επόμενο του κρίσιμου κόμβου στο μονοπάτι που οδηγεί στον v και μια δεξιά περιστροφή γύρω από τον κρίσιμο κόμβο.

Άσκηση για το σπίτι

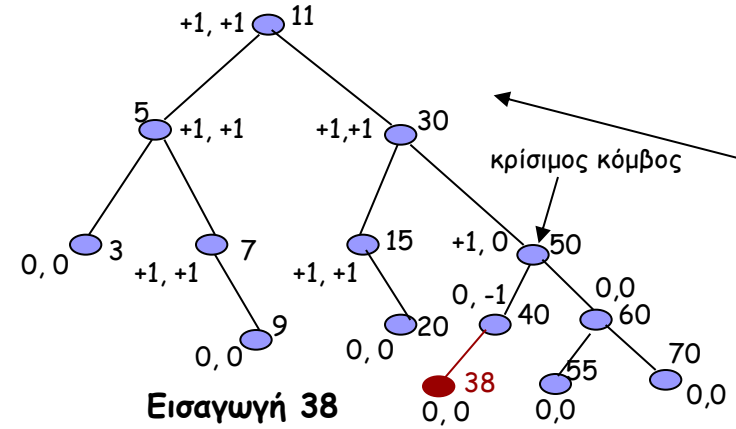
Φτιάξτε τα αντίστοιχα συμμετρικά σχήματα εκείνων που παρουσιάστηκαν στις δύο τελευταίες διαφάνειες και μελετήστε τις ενέργειες που πρέπει να πραγματοποιηθούν στην περίπτωση LR.

Δένδρα AVL – Περιστροφές κατά την Εισαγωγή

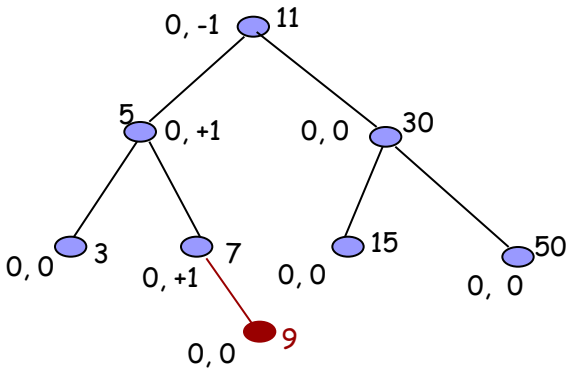
Παρατήρηση

- Οι περιστροφές γίνονται γύρω από το πολύ δύο κόμβους στο δένδρο ανάλογα με την περίπτωση που πρέπει να εφαρμοστεί:
  - μία περιστροφή γύρω από τον κρίσιμο κόμβο για τις περιπτώσεις LL, RR
  - δύο περιστροφές, η πρώτη γύρω από τον επόμενο του κρίσιμου κόμβου στο μονοπάτι που ακολουθήθηκε κατά την τοποθέτηση του εισαχθέντα κόμβου και η δεύτερη γύρω από τον κρίσιμο κόμβο, για τις περιπτώσεις RL, LR

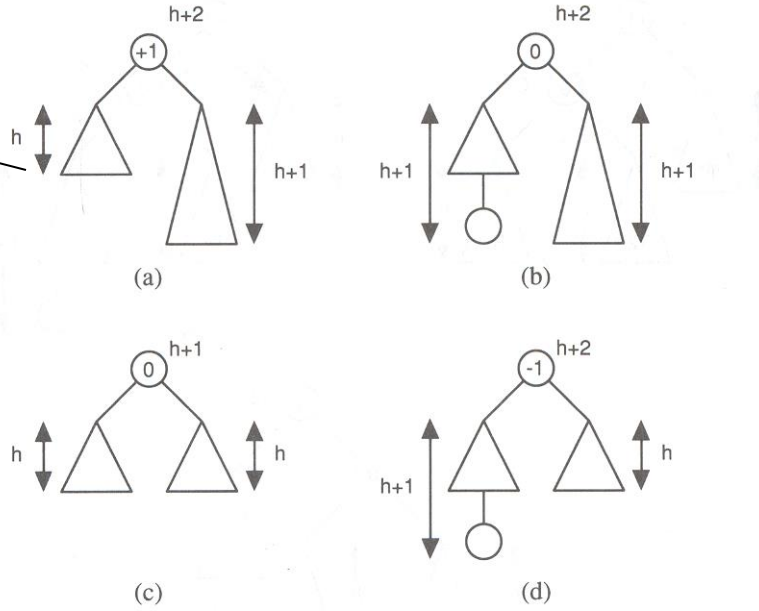
Δένδρα AVL – Εισαγωγή – Περιπτώσεις που δεν απαιτούν περιστροφές



Εισαγωγή 38



Εισαγωγή 9

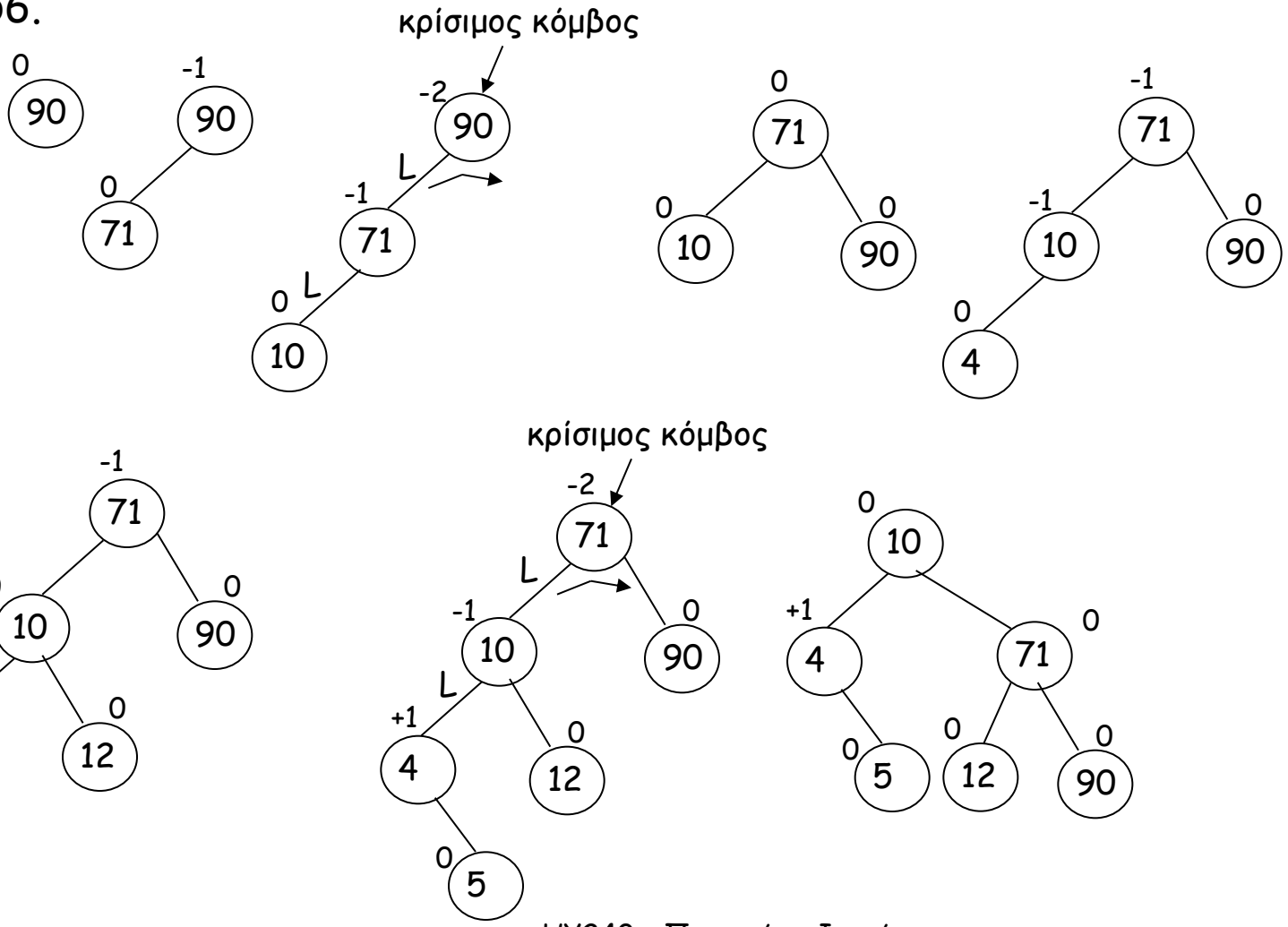


Σχήμα 7.3: Lewis & Denenberg, Data Structures & Their Algorithms, Addison-Wesley, 1991

Εισαγωγές κόμβων σε AVL δένδρο

Παράδειγμα

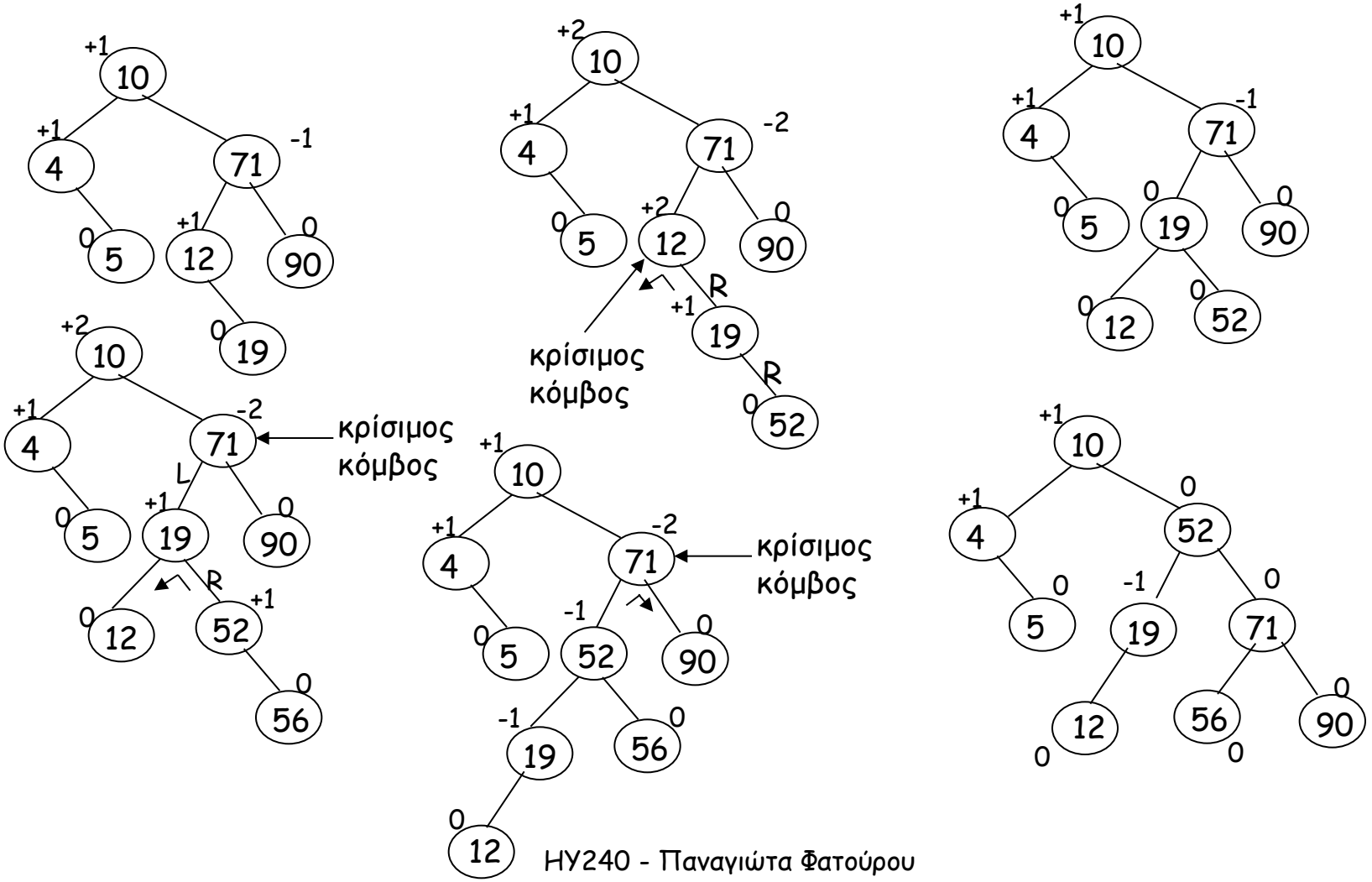
Διαδοχικές εισαγωγές των κλειδιών με τιμές 90, 71, 10, 4, 12, 5, 19, 52 και 56.



Εισαγωγές κόμβων σε AVL δένδρο

Παράδειγμα

Διαδοχικές εισαγωγές των κλειδιών με τιμές 90, 71, 10, 4, 12, 5, 19, 52 και 56.



Δένδρα AVL – Εισαγωγή - Παρατηρήσεις

- Κάθε κόμβος στο μονοπάτι από τον εισαχθέντα κόμβο v μέχρι τον κρίσιμο κόμβο w είχε balance 0 πριν την εισαγωγή. Άρα, θα έχει balance +1 ή –1 μετά την εισαγωγή.

Τι καθορίζει το αν κάποιος τέτοιος κόμβος θα έχει balance +1 ή –1;
Το αν η πρώτη ακμή στο μονοπάτι από τον κόμβο αυτό προς τον εισαχθέντα κόμβο οδηγεί αριστερά ή δεξιά.

- Το balance του κρίσιμου κόμβου w γίνεται είτε 0 ή +2 (ή –2, αντίστοιχα) μετά την τοποθέτηση του νέου κόμβου στην κατάλληλη θέση στο δένδρο.

Γιατί; Πότε το balance γίνεται 0 και πότε +2 (ή –2);
Αν το balance ήταν +1 (-1) και η πρώτη ακμή στο μονοπάτι από τον w στον v οδηγεί δεξιά (αριστερά, αντίστοιχα), ο w θα έχει balance +2 (-2, αντίστοιχα) μετά την τοποθέτηση του v, ενώ αν το μονοπάτι οδηγεί αριστερά (δεξιά, αντίστοιχα), τότε ο w θα έχει balance 0 μετά την τοποθέτηση του v.

Δένδρα AVL – Εισαγωγή - Παρατηρήσεις

- Καθώς η αναζήτηση του πατρικού κόμβου του v εξελίσσεται, αρκεί να αποθηκεύεται ο τελευταίος κόμβος με balance +1 ή -1 στο μονοπάτι που ακολουθείται.
- Μετά την τοποθέτηση του v στο δένδρο (και πριν τις περιστροφές), ξεκινώντας από τον κρίσιμο κόμβο w μπορεί να ακολουθηθεί και πάλι το ίδιο μονοπάτι προς τον εισερχόμενο κόμβο ώστε να διορθωθούν τα balances.

Γιατί μπορούμε να βρούμε και πάλι αυτό το μονοπάτι?

Ακολουθούμε την ίδια διαδικασία αναζήτησης (ξεκινώντας από τον w), δηλαδή αν το κλειδί του προς εισαγωγή κόμβου είναι μικρότερο από εκείνο του τρέχοντος ο επόμενος κόμβος στο μονοπάτι είναι στα αριστερά του τρέχοντος, ενώ διαφορετικά στα δεξιά.

Πως διορθώνουμε τα balances?

Έστω ένας οποιοσδήποτε κόμβος x στο μονοπάτι από τον w στον v. Ο κόμβος αυτός είχε balance 0. Αν η πρώτη ακμή στο μονοπάτι από τον x στον v οδηγεί αριστερά του x, τότε το balance του x γίνεται -1, διαφορετικά η νέα τιμή του balance θα είναι +1.

Δένδρα AVL – Εισαγωγή - Πολυπλοκότητα

Ποια είναι η πολυπλοκότητα της AVLInsert()?

- O(ύψος δένδρου).
- Θεώρημα $\Rightarrow$ ύψος δένδρου AVL = O(logn)

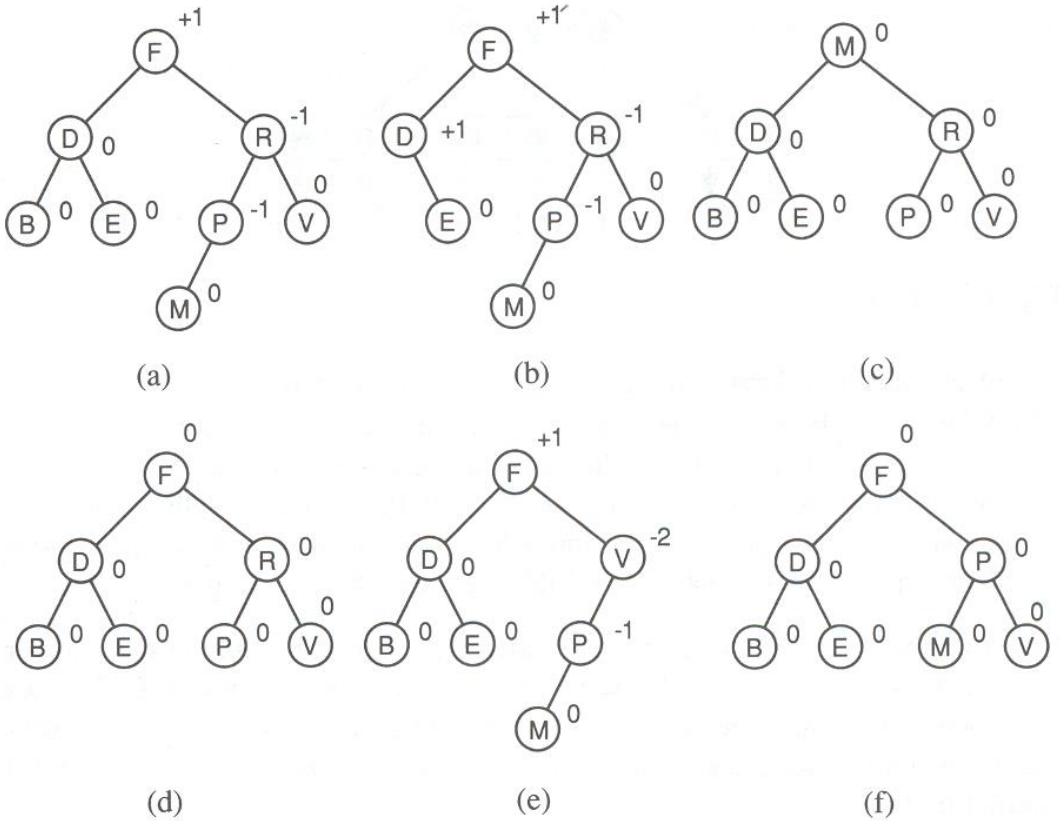
$\Rightarrow$ Χρονική Πολυπλοκότητα εισαγωγής = O(logn)

Δένδρα AVL – Διαγραφή

Σχήμα 7.6: Lewis & Denenberg, Data Structures & Their Algorithms, Addison-Wesley, 1991

Ακολουθούμε το γνωστό αλγόριθμο διαγραφής σε ταξινομημένο δένδρο:

1) Διαγραφή του ίδιου του κόμβου v αν είναι φύλλο.
2) Αντικατάστασή του από το παιδί του αν έχει μόνο ένα παιδί.
3) Αντικατάστασή του από τον επόμενό του στην ενδο-διατεταγμένη διάταξη αν έχει δύο παιδιά.

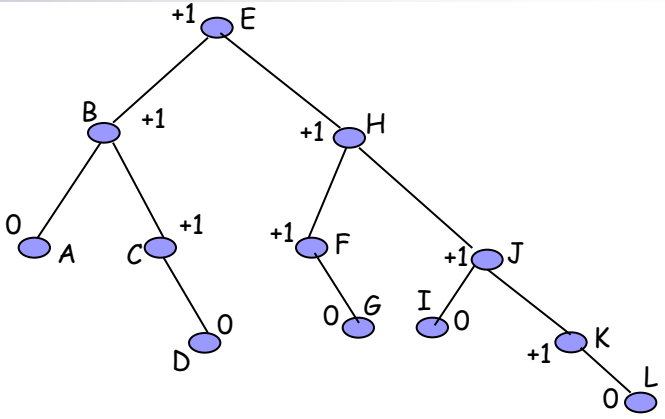


(a) Αρχικό δένδρο, (b) Διαγραφή του B, (c) Διαγραφή του F, (d) Διαγραφή του M, (e) Διαγραφή του R

Balance

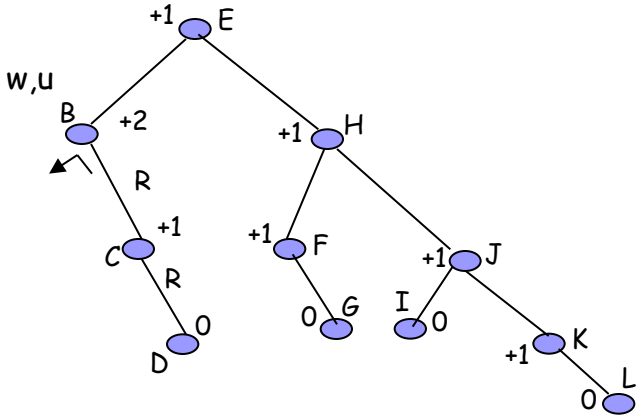
Αν 1) ή 2), το balance του γονικού κόμβου **w** του v αλλάζει.
Αν 3), το balance του γονικού κόμβου **w** του επομένου του v στην ενδο-διατεταγμένη διάσχιση αλλάζει.

Δένδρα AVL – Αλγόριθμος Διαγραφής



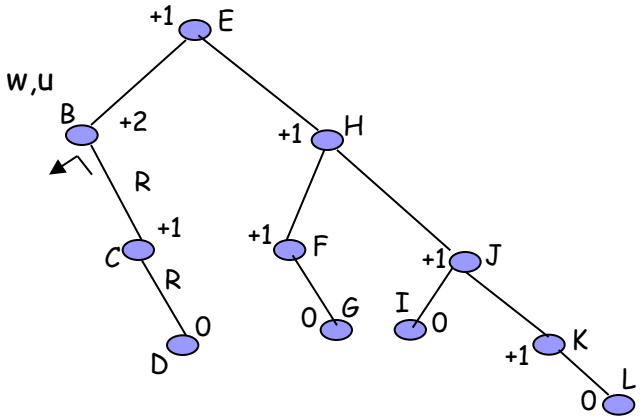
1. Εξετάζουμε όλο το μονοπάτι (ας το συμβολίσουμε με P1) από τον w ως τη ρίζα και για κάθε κόμβο του μονοπατιού αυτού που έχει balance +2 ή -2 μετά τη διαγραφή, ίσως χρειαστεί να γίνουν περιστροφές.

2. Έστω u ο πρώτος τέτοιος κόμβος στο P1.



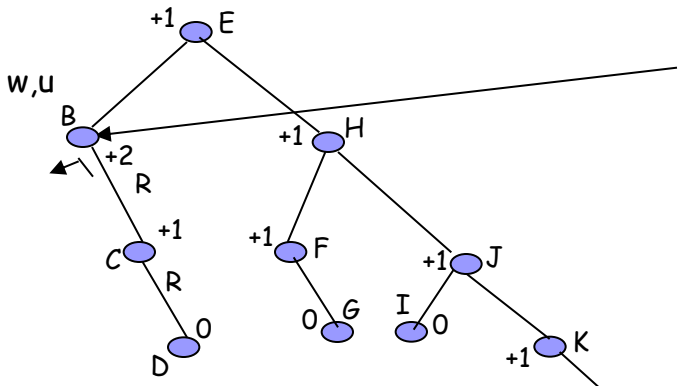
Δένδρα AVL – Αλγόριθμος Διαγραφής

3. Καθορίζουμε το μονοπάτι P2 με το μεγαλύτερο μήκος από τον u προς οποιοδήποτε φύλλο του υποδένδρου του και εξετάζουμε τις δύο πρώτες ακμές σε αυτό το μονοπάτι.
   - Αν LL (ή RR) πραγματοποιούμε μια περιστροφή R (L) γύρω από τον u.
   - Αν RL (ή LR) πραγματοποιούμε δύο περιστροφές, την πρώτη τύπου R (L) γύρω από τον επόμενο του u στο μονοπάτι P2 και τη δεύτερη τύπου L (R) γύρω από τον u.



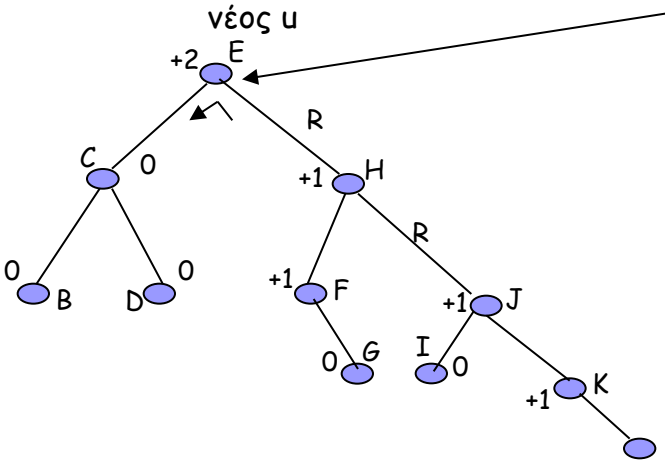
4. Επαναπροσδιορίζουμε balances κόμβων στο P1 & αν εξακολουθούν να υπάρχουν κόμβοι με μη-επιτρεπτό balance σε αυτό:
   - Θέτουμε u = πρώτος κόμβος στο P1 με μη επιτρεπτό balance;
   - Πηγαίνουμε στο Βήμα 3;

Παράδειγμα Διαγραφής που Οδηγεί σε Πολλαπλές Περιστροφές



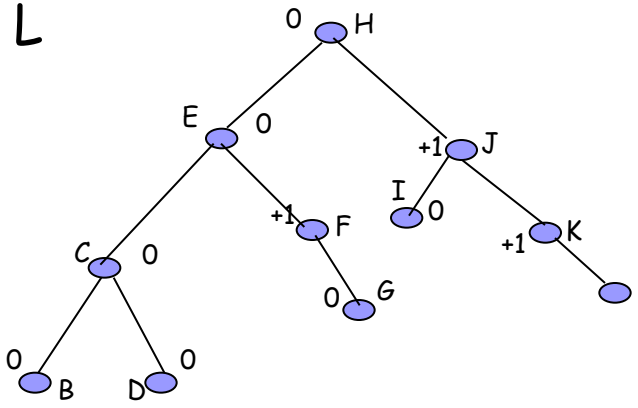
Κόμβος με μη-επιτρεπτό balance

Μακρύτερο μονοπάτι από αυτόν προς οποιοδήποτε φύλλο τύπου RR ⇒ μία περιστροφή τύπου L γύρω από τον κόμβο



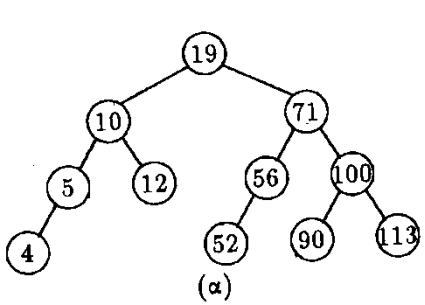
Κόμβος με μη-επιτρεπτό balance

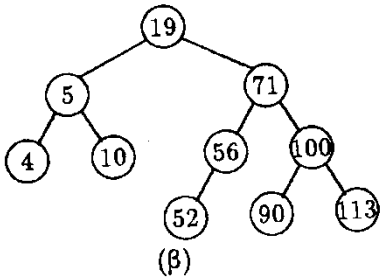
Μακρύτερο μονοπάτι από αυτόν προς οποιοδήποτε φύλλο τύπου RR ⇒ μία περιστροφή τύπου L



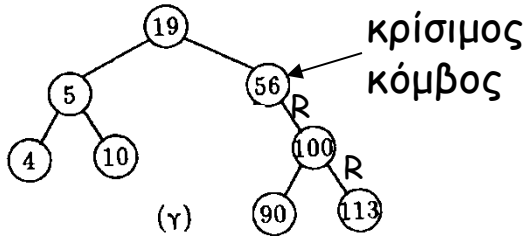
Αν υπάρχουν περισσότερα του ενός μονοπάτια ίδιου μήκους και το μήκος αυτό είναι το μεγαλύτερο, μπορεί να γίνει επιλογή εκείνου του μονοπατιού που οδηγεί στις λιγότερες και πιο απλές περιστροφές!

Δένδρα AVL – Διαγραφή

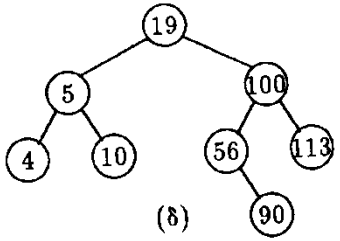




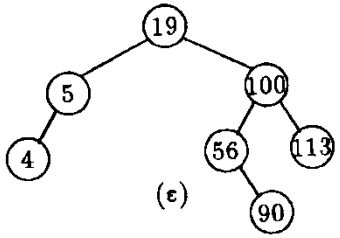
Σχήμα 5.3: Ι. Μανωλόπουλος, Δομές Δεδομένων, Μια προσέγγιση με την Pascal, Εκδόσεις Art of Text.

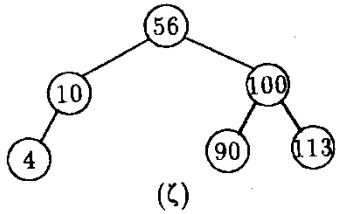


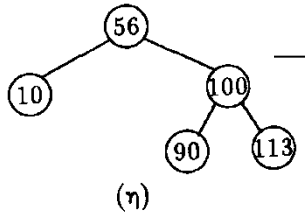
κρίσιμος κόμβος



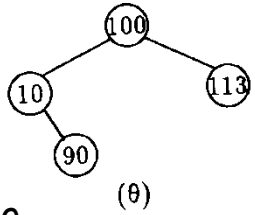
Παράδειγμα
Διαδοχική διαγραφή των κλειδιών 12, 71, 52, 10, 19, 4, και 56.







Αντικατάσταση από 10 και μια αριστερή περιστροφή γύρω από το 10



Δένδρα AVL – Διαγραφή - Πολυπλοκότητα

Ποια είναι η πολυπλοκότητα της AVLDelete()?

O(ύψος δένδρου) για τους ακόλουθους λόγους:

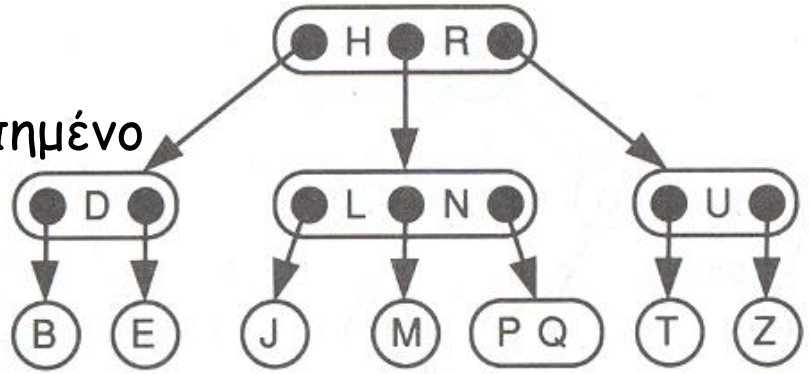
- Για τη διαγραφή του κόμβου το κόστος είναι O(ύψος δένδρου).
- Για κάθε κόμβο u στο P1, θα γίνει εύρεση των δύο πρώτων ακμών του μονοπατιού που καθορίζει το ύψος του u (βάσει του balance του u και του balance του επομένου του u στο μοναπάτι).
- Θα γίνουν O(ύψος δένδρου) περιστροφές και κάθε μια κοστίζει O(1).
- Τα balances αλλάζουν μόνο για τους κόμβους του P1 και τους κόμβους που «εμπλέκονται» στις περιστροφές.

Θεώρημα $\Rightarrow$ ύψος δένδρου AVL = O(logn)

$\Rightarrow$ Χρονική Πολυπλοκότητα = O(logn)

2-3 Δένδρα

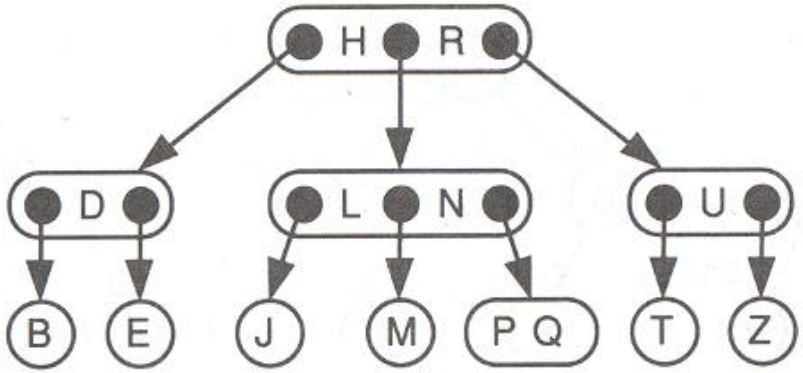
Ιδέα: Το δένδρο είναι τέλεια εξισορροπημένο (όλα τα φύλλα έχουν το ίδιο βάθος) αλλά δεν είναι δυαδικό.



Σχήμα 7.7: Lewis & Denenberg, Data Structures & Their Algorithms, Addison-Wesley, 1991

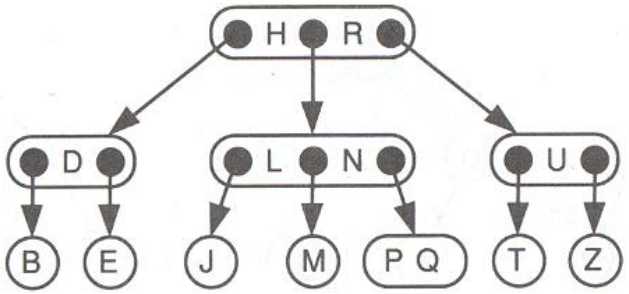
- Σε ένα 2-3 δένδρο ένας κόμβος που δεν είναι φύλλο έχει δύο παιδιά και σε αυτόν αποθηκεύεται ένα στοιχείο του συνόλου (ένας τέτοιος κόμβος λέγεται 2-κόμβος), ή έχει τρία παιδιά και σε αυτόν αποθηκεύονται δύο στοιχεία του συνόλου (τότε ο κόμβος λέγεται 3-κόμβος).
- Κάθε φύλλο που αποθηκεύει ένα στοιχείο του συνόλου είναι 2-κόμβος, ενώ κάθε φύλλο που αποθηκεύει δύο στοιχεία του συνόλου είναι 3-κόμβος.
- Με κατάλληλη διευθέτηση κόμβων και των δύο ειδών, μπορούμε να φτιάξουμε δένδρο στο οποίο όλα τα φύλλα έχουν το ίδιο βάθος και το οποίο αποθηκεύει οποιοδήποτε επιθυμητό πλήθος κλειδιών.

Ιδιότητες 2-3 Δένδρων



1. Όλα τα φύλλα έχουν το ίδιο βάθος και αποθηκεύουν 1 ή 2 στοιχεία.
2. Κάθε εσωτερικός κόμβος:
   - είτε αποθηκεύει ένα στοιχείο και έχει δύο παιδιά,
   - ή αποθηκεύει δύο στοιχεία και έχει τρία παιδιά.
3. Το δένδρο είναι ταξινομημένο.

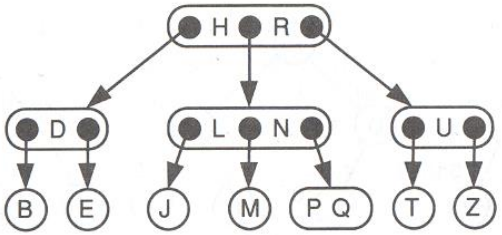
2-3 Δένδρα



Ένα 2-3 δένδρο είναι ταξινομημένο αν για κάθε κόμβο v ισχύουν τα εξής:

- Αν ο v είναι 2-κόμβος, τα κλειδιά όλων των στοιχείων που βρίσκονται στο αριστερό υποδένδρο του v είναι μικρότερα από το κλειδί του v, ενώ εκείνα των στοιχείων που βρίσκονται στο δεξί υποδένδρο του v είναι μεγαλύτερα από εκείνα του v.
- Αν ο v είναι 3-κόμβος, το κλειδί (έστω K_1) του πρώτου στοιχείου του v είναι μικρότερο από το κλειδί (έστω K_2) του δεύτερου στοιχείου του v. Επίσης, τα κλειδιά όλων των στοιχείων στο πρώτο υποδένδρο του v είναι μικρότερα από K_1, τα κλειδιά όλων των στοιχείων στο δεύτερο υποδένδρο του v είναι μεταξύ του K_1 και του K_2, ενώ τα κλειδιά του τρίτου υποδένδρου του v είναι μεγαλύτερα από K_2.

2-3 Δένδρα



Μεταξύ όλων των 2-3 δένδρων ύψους h,
ποιο είναι εκείνο με τους λιγότερους κόμβους;
Εκείνο του οποίου όλοι οι εσωτερικοί κόμβοι είναι 2-κόμβοι.
Ποιο είναι το ύψος αυτού του δένδρου? $\log_2 n$

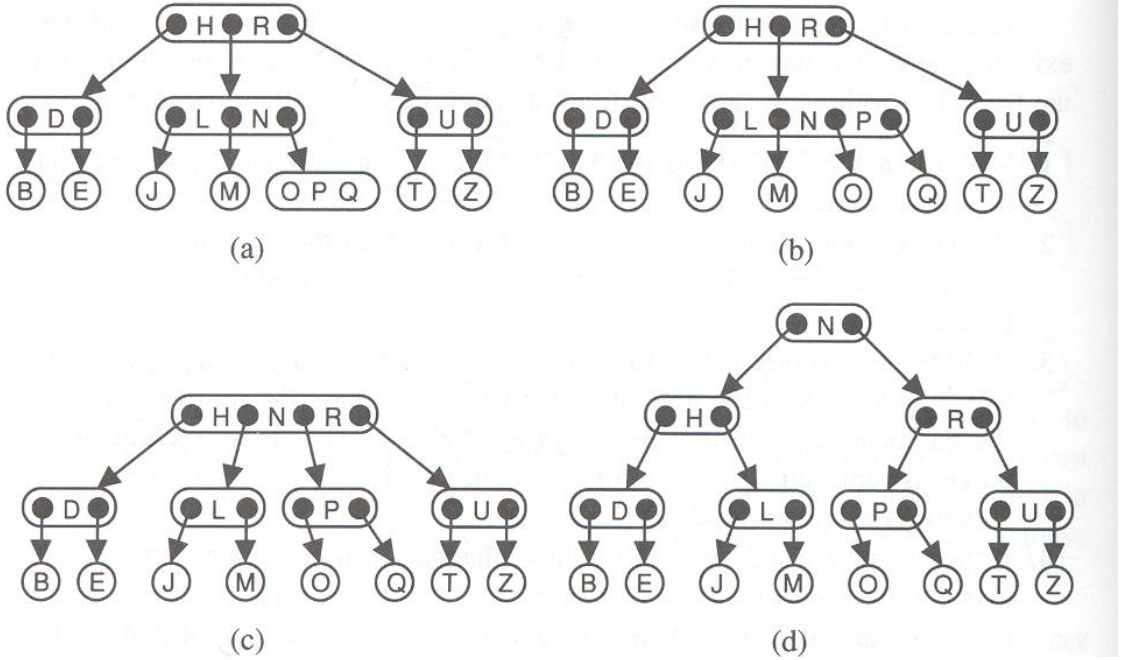
Μεταξύ όλων των 2-3 δένδρων ύψους h, ποιο είναι εκείνο με τους περισσότερους κόμβους;
Εκείνο του οποίου όλοι οι εσωτερικοί κόμβοι είναι 3-κόμβοι.
Ποιο είναι το ύψος αυτού του δένδρου? $\log_3 n$

Πως θα υλοποιήσουμε την LookUp() σε ένα 2-3 δένδρο?
Το δένδρο είναι ταξινομημένο.

Το ύψος ενός 2-3 δένδρου με n κόμβους είναι $\Theta(\log n)$.

Πολυπλοκότητα LookUp; $O(\log n)$

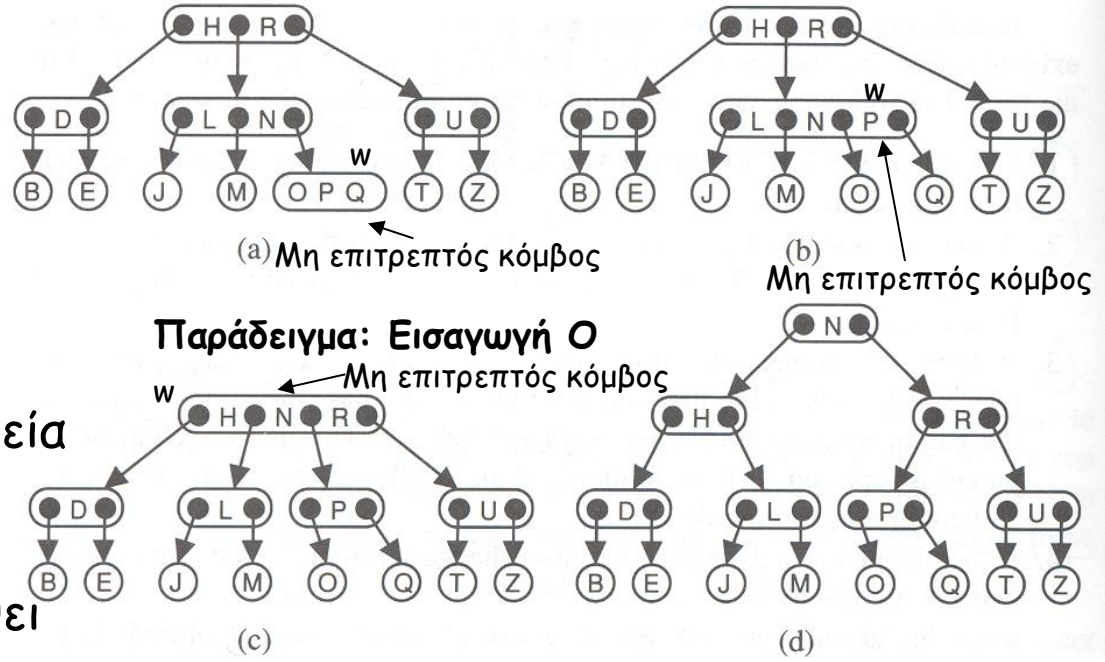
2-3 Δένδρα - Εισαγωγή



Σχήμα 7.8: Lewis & Denenberg, Data Structures & Their Algorithms, Addison-Wesley, 1991

1. Εύρεση του φύλλου w στο οποίο θα έπρεπε σύμφωνα με την ιδιότητα της ταξινόμησης να εισαχθεί το νέο στοιχείο.
2. Διατήρηση των κόμβων του μονοπατιού που ακολουθήθηκε σε στοίβα.
3. Αν το w είναι 2-κόμβος, προσθήκη του νέου στοιχείου στο w και τερματισμός.

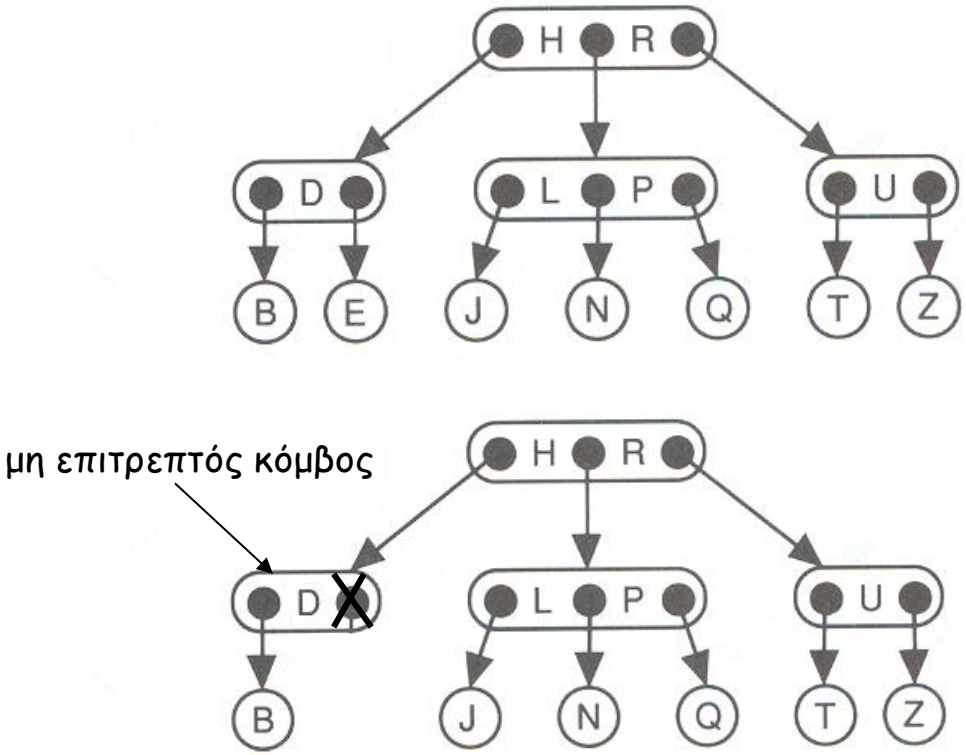
2-3 Δένδρα Εισαγωγή



4. Αν το w είναι 3-κόμβος, μετατρέπεται σε κόμβο που πρέπει να αποθηκεύσει 3 στοιχεία το οποίο είναι μη επιτρεπτό. Αντικαθιστούμε το w με δύο 2-κόμβους, έναν που αποθηκεύει το 1ο και έναν που αποθηκεύει το 3ο στοιχείο του w και το μεσαίο στοιχείο του w μεταφέρεται στον πατρικό του κόμβο. Αν δεν υπάρχει πατρικός κόμβος πηγαίνουμε στο βήμα 6.
5. Αν ο πατρικός κόμβος είναι 2-κόμβος, μετατρέπεται σε 3-κόμβο και ο αλγόριθμος τερματίζει. Διαφορετικά, θέτουμε w = <πατρικός κόμβος του w> και επιστρέφουμε στο βήμα 3.
6. Η διαδικασία αυτή επαναλαμβάνεται μέχρι να βρούμε χώρο για το κλειδί σε κάποιο κόμβο στο μονοπάτι προς τη ρίζα ή να φτάσουμε στη ρίζα. Αν συμβεί το δεύτερο, η ρίζα χωρίζεται σε 2 κόμβους και ένας νέος κόμβος που αποτελεί τον πατρικό κόμβο αυτών των δύο κόμβων αποθηκεύει το μεσαίο στοιχείο που θα έπρεπε να αποθηκευθεί στη ρίζα. Έτσι, το ύψος του δένδρου αυξάνεται κατά 1.

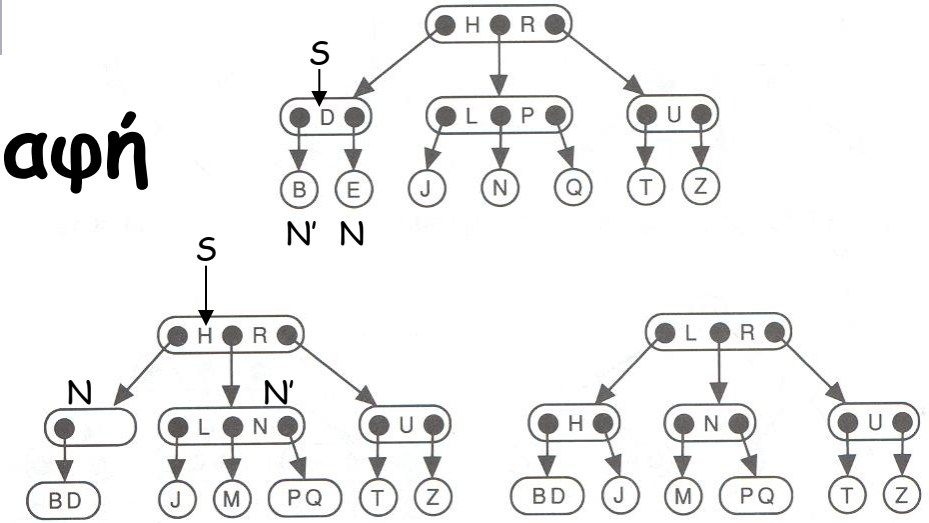
2-3 Δένδρα - Διαγραφή

Μπορεί να προκύψει το αντίστροφο πρόβλημα: μετά τη διαγραφή, κάποιος κόμβος μπορεί να έχει ένα μόνο παιδί.



Παράδειγμα
Διαγραφή E

2-3 Δένδρα - Διαγραφή



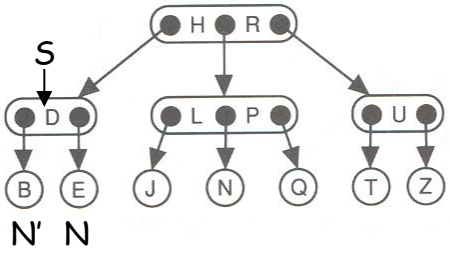
Σχήμα 7.8: Lewis & Denenberg, Data Structures & Their Algorithms, Addison-Wesley, 1991

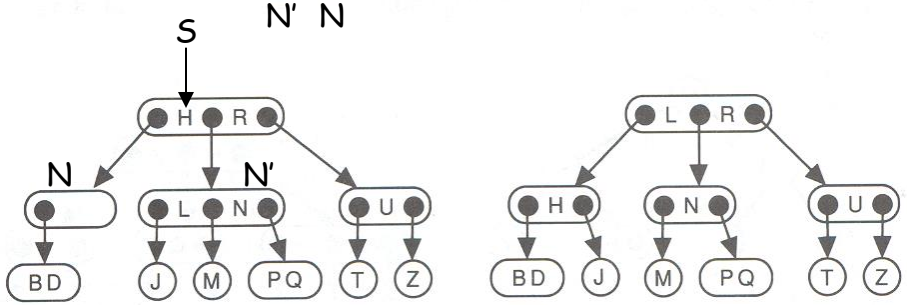
1. Αν το προς διαγραφή κλειδί Κ περιέχεται σε κόμβο φύλλο, το διαγράφουμε. Διαφορετικά, το κλειδί (έστω Κ') που είναι επόμενο του Κ στην ενδο-διατεταγμένη διάσχιση περιέχεται σε φύλλο, οπότε αντικαθιστούμε το Κ με το Κ' και διαγράφουμε το Κ'.

2. Έστω Ν ο κόμβος από τον οποίο διαγράφεται το κλειδί. Αν ο Ν εξακολουθεί να έχει ένα κλειδί, ο αλγόριθμος τερματίζει.

3. Αν ο Ν είναι η ρίζα, τον διαγράφουμε. **Σε αυτή την περίπτωση, ο Ν μπορεί να έχει ένα ή κανένα παιδί.** Αν δεν έχει κανένα παιδί, το δένδρο μετά τη διαγραφή είναι άδειο. Διαφορετικά, το παιδί του Ν είναι η νέα ρίζα του δένδρου.

2-3 Δένδρα - Διαγραφή





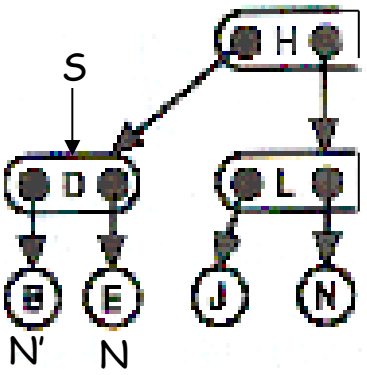
3. Διαφορετικά, ο N έχει τουλάχιστον έναν αδελφικό κόμβο. Έστω N' ένας οποιοσδήποτε αδελφικός κόμβος του N. Έστω P ο πατέρας των N, N' και έστω ότι S είναι το κλειδί του P που χωρίζει τα δύο υπο-δένδρα (που οδηγούν από τον P στους N και N').

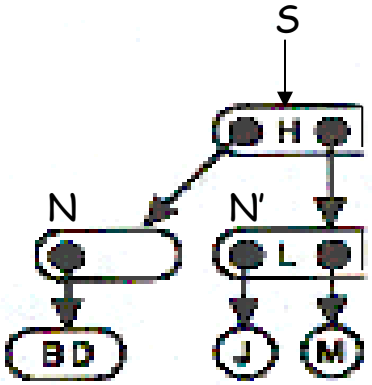
   a. Ο N' είναι 3-κόμβος ακριβώς στα αριστερά (ακριβώς στα δεξιά) του N. Μετακινούμε το S στο N και αντικαθιστούμε το S στον πατέρα του με το δεξιότερο (αριστερότερο) κλειδί του N'. Αν οι N και N' είναι εσωτερικοί κόμβοι, μετατρέπουμε το δεξιότερο (αριστερότερο) υποδένδρο του N' σε αριστερότερο (δεξιότερο) υποδένδρο του N. Οι N, N' έχουν πλέον ένα κλειδί ο καθένας και ο αλγόριθμος τερματίζει.

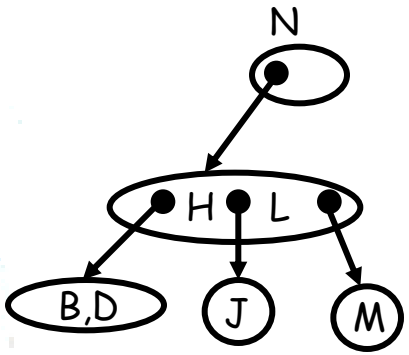
   b. Ο N' είναι 2-κόμβος. Συνενώνουμε το S και το κλειδί του N' σε έναν νέο 3-κόμβο, ο οποίος αντικαθιστά τους N, N' και αποτελεί το μοναδικό παιδί του P (ο οποίος απομένει τώρα χωρίς κλειδί). Θέτουμε N = P και επαναλαμβάνουμε από το βήμα 3.

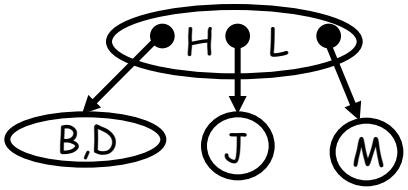
2-3 Δένδρα - Διαγραφή

Ένα ακόμη παράδειγμα







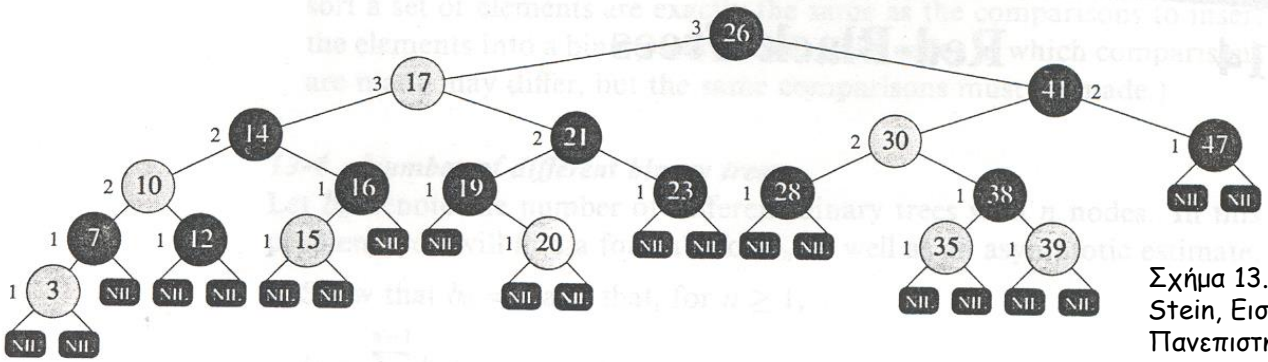


Πολυπλοκότητα Insert(); O(logn)

Πολυπλοκότητα Delete(); O(logn)

Πολυπλοκότητα LookUp(); O(logn)

Κοκκινόμαυρα Δένδρα

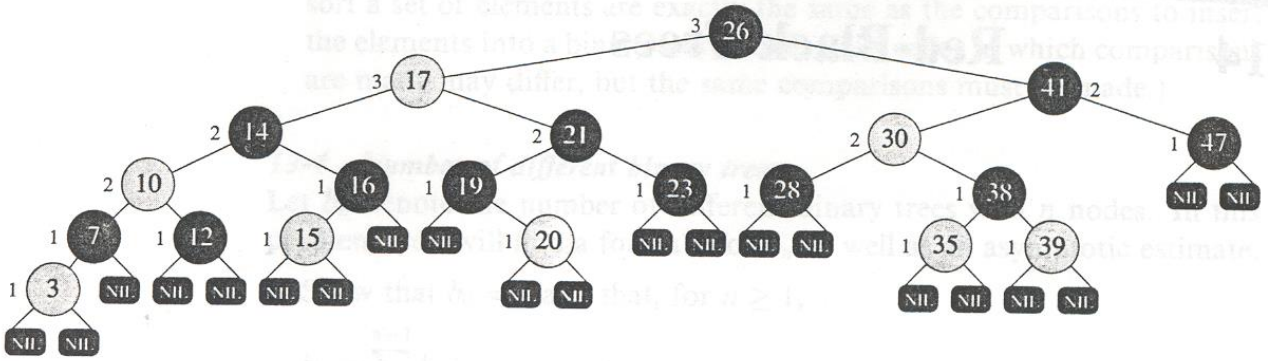


Σχήμα 13.1(α): Cormen, Leiserson, Rivest & Stein, Εισαγωγή στους Αλγορίθμους, Πανεπιστημιακές Εκδόσεις Κρήτης, 2006

- Ένα κοκκινόμαυρο δένδρο είναι ένα ταξινομημένο δυαδικό δένδρο, στο οποίο κάθε κόμβος περιέχει ένα επιπλέον στοιχείο πληροφορίας, το χρώμα του, το οποίο μπορεί να είναι είτε μαύρο ή κόκκινο.

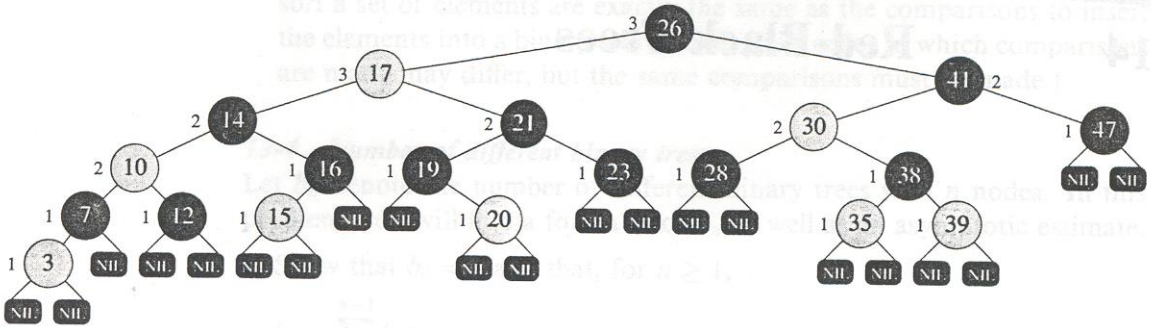
- Αν κάποιο παιδί κάποιου κόμβου δεν υφίσταται, ο αντίστοιχος δείκτης είναι nill. Θεωρούμε ότι δείκτες με τιμή nill είναι δείκτες προς εξωτερικούς κόμβους (φύλλα) του δυαδικού δένδρου, ενώ οι συνήθεις κόμβοι που αποθηκεύουν κλειδιά αποτελούν τους εσωτερικούς κόμβους του δένδρου.

Ιδιότητες Κοκκινόμαυρων Δένδρων



1. Κάθε κόμβος έχει είτε κόκκινο είτε μαύρο χρώμα.
2. Το χρώμα της ρίζας είναι πάντα μαύρο.
3. Κάθε NILL κόμβος έχει μαύρο χρώμα.
4. Αν ένας κόμβος έχει κόκκινο χρώμα, τότε και τα δύο παιδιά του έχουν μαύρο χρώμα.
5. Κάθε μονοπάτι από οποιονδήποτε κόμβο προς οποιοδήποτε εξωτερικό κόμβο φύλλο που είναι απόγονός του περιέχει το ίδιο πλήθος μαύρων κόμβων.

Ύψος Κοκκινόμαυρου Δένδρου

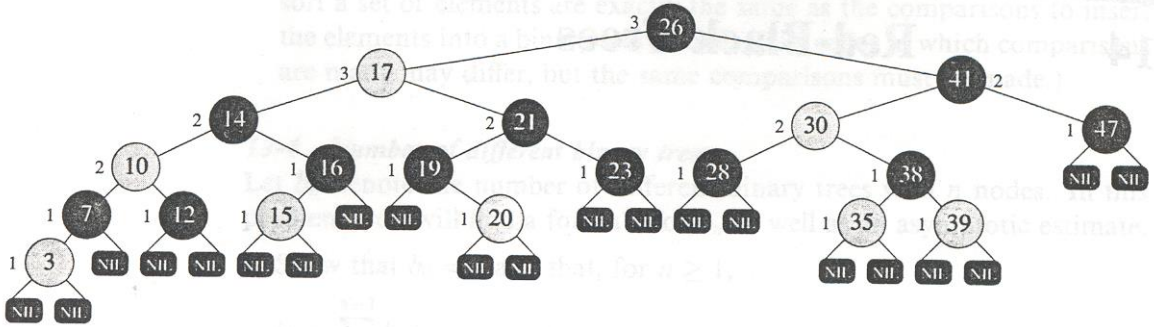


Μαύρο ύψος bh(v) κόμβου v
Πλήθος μαύρων κόμβων σε κάθε μονοπάτι από τον κόμβο v σε οποιοδήποτε εξωτερικό κόμβο nill του υποδένδρου του, χωρίς να συμπεριλαμβάνουμε τον v.

Πρόταση: Ένα κόκκινο μαύρο δένδρο με n εσωτερικούςκόμβους έχει ύψος το πολύ 2log(n+1).

Διαίσθηση: Τουλάχιστον οι μισοί κόμβοι σε κάθε μονοπάτι από τη ρίζα σε φύλλο είναι μαύροι. Γιατί;

Ύψος Κοκκινόμαυρου Δένδρου



Απόδειξη

Αποδεικνύουμε επαγωγικά ότι το υπο-δένδρο με ρίζα κάποιο κόμβο v περιέχει τουλάχιστον $2^{bh(v)}-1$ εσωτερικούς κόμβους (η επαγωγή ως προς το ύψος του v).

Βάση επαγωγής: αν ο v έχει ύψος 0 τότε είναι φύλλο (δηλαδή εξωτερικός κόμβος), οπότε ο αριθμός των εσωτερικών κόμβων στο υποδένδρο που εκφύεται από τον v είναι 0, όπως απαιτείται.

Επαγωγική Υπόθεση: Δοθέντος ενός οποιουδήποτε κόμβου v, υποθέτουμε ότι ο ισχυρισμός ισχύει για τα παιδιά του v.

Ύψος Κοκκινόμαυρου Δένδρου

Απόδειξη (συνέχεια)

Επαγωγικό Βήμα: Αποδεικνύουμε τον ισχυρισμό για τον κόμβο v.

- Έστω w ένα οποιοδήποτε παιδί του v. Τότε, $bh(w) \geq bh(v) - 1$ (αν ο w είναι κόκκινος τότε $bh(w) = bh(v)$, ενώ αν ο w είναι μαύρος τότε $bh(w) \geq bh(v) - 1$).
- Αφού το ύψος του w είναι πιο μικρό από εκείνο του v, από επαγωγική υπόθεση, το υποδένδρο που εκφύεται από τον w έχει $2^{bh(w)}-1 \geq 2^{bh(v)-1} - 1$ εσωτερικούς κόμβους.
- Άρα, το υποδένδρο που εκφύεται από τον v έχει τουλάχιστον $2^* (2^{bh(v)-1} - 1) + 1 = 2^{bh(v)} - 1$ εσωτερικούς κόμβους, όπως απαιτείται.

Άρα, το υπο-δένδρο που εκφύεται από οποιονδήποτε κόμβο v περιέχει τουλάχιστον $2^{bh(v)}-1$ εσωτερικούς κόμβους. (1)

Έστω h το ύψος του δένδρου και r η ρίζα. Είναι $bh(r) \geq h/2$, αφού βάσει της ιδιότητας 4, σε οποιοδήποτε μονοπάτι από τη ρίζα προς οποιονδήποτε εξωτερικό κόμβο, τουλάχιστον οι μισοί κόμβο (εξαιρουμένης της ίδιας της ρίζας) θα πρέπει να είναι μαύροι.

Άρα, από (1), $n \geq 2^{h/2} - 1 \Rightarrow h \leq 2\log(n+1)$.

Πως υλοποιείται η LookUp();
Ποια είναι η πολυπλοκότητα της;

Κοκκινόμαυρα Δένδρα - Εισαγωγή

Αλγόριθμος

- Εισαγωγή του νέου στοιχείου με τον ίδιο τρόπο όπως σε ταξινομημένο δυαδικό δένδρο. Το μονοπάτι που ακολουθείται για την εισαγωγή του νέου κόμβου αποθηκεύεται σε στοίβα.
- Ο νέος κόμβος χαρακτηρίζεται κόκκινος.
- Αν οι ιδιότητες χρωματισμού εξακολουθούν να ισχύουν, ο αλγόριθμος τερματίζεται. Διαφορετικά, εκτελούνται ενέργειες που αποσκοπούν στην αποκατάσταση των ιδιοτήτων του δένδρου (οι οποίες περιγράφονται στη συνέχεια).

Για να απαιτείται η εκτέλεση ενεργειών αποκατάστασης του δένδρου θα πρέπει:

Ο πατρικός κόμβος του νέο-εισαχθέντα κόμβου να είναι κόκκινος. Γιατί;

$\Rightarrow$ Ο πατρικός αυτός κόμβος δεν μπορεί να είναι η ρίζα.

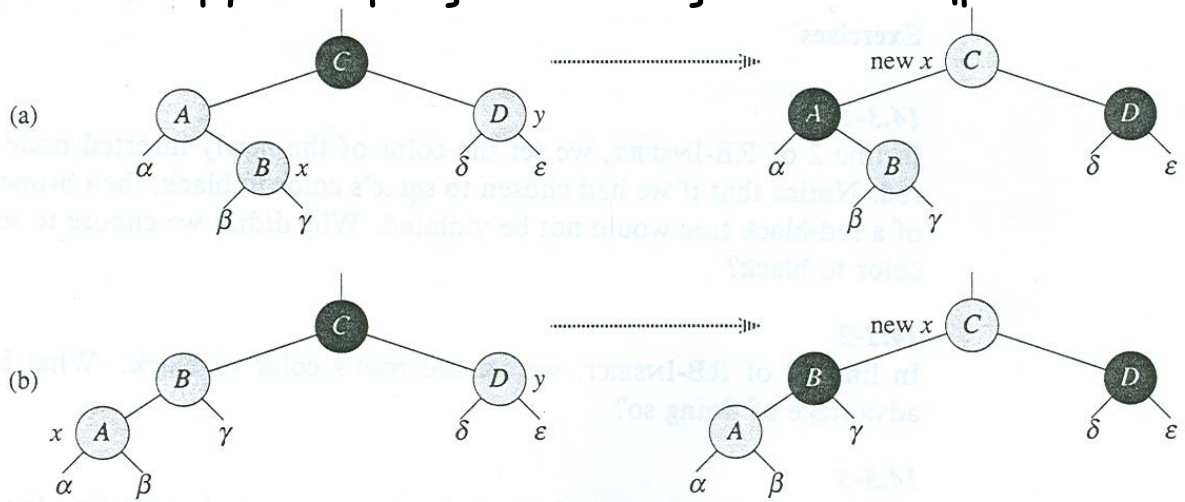
Ο παππούς του νέο-εισαχθέντα κόμβου να είναι μαύρος. Γιατί;

Κοκκινόμαυρα Δένδρα - Εισαγωγή

Περιγραφή Ενεργειών Αποκατάστασης

Υποθέτουμε ότι ο πατέρας του x είναι αριστερό παιδί του παππού του x (η αντίθετη περίπτωση είναι συμμετρική). Αν ο x και ο πατέρας του x είναι κόκκινοι, απαιτούνται ενέργειες αποκατάστασης.

Περίπτωση 1: Ο αδελφικός κόμβος y του πατέρα του x είναι κόκκινος. Τότε, αλλάζουμε το χρώμα του πατέρα του x και του αδελφικού του κόμβου y σε μαύρο και το χρώμα του παππού του x σε κόκκινο. Εκτελούμε την εντολή "x = παππούς του x". Επαναλαμβάνουμε ξεκινώντας από το Βήμα 1.



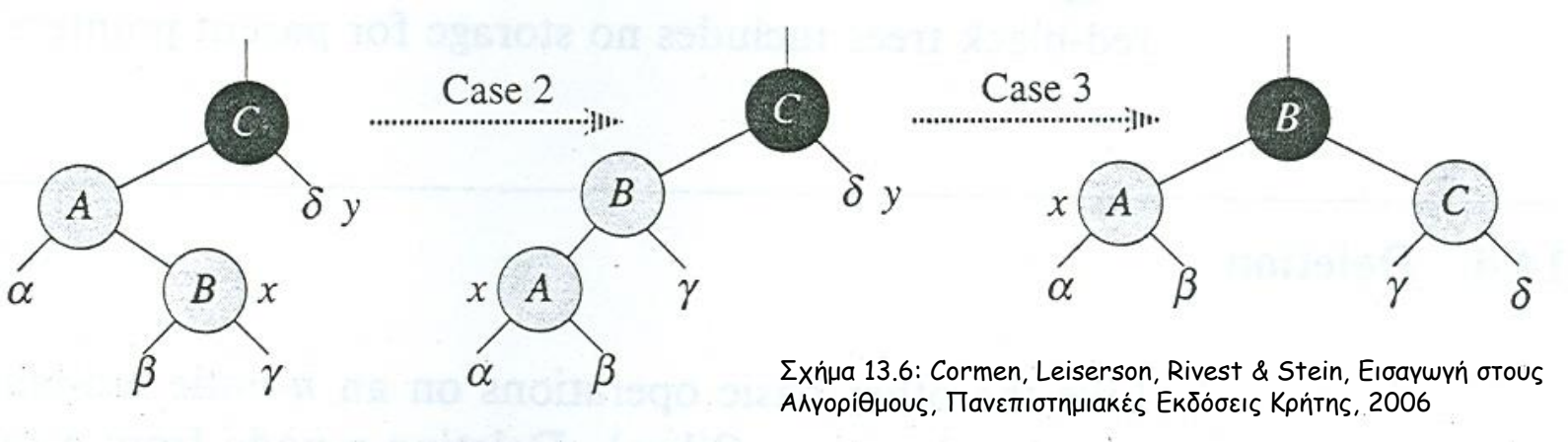
Σχήμα 13.5: Cormen, Leiserson, Rivest & Stein, Εισαγωγή στους Αλγορίθμους, Πανεπιστημιακές Εκδόσεις Κρήτης, 2006

Κοκκινόμαυρα Δένδρα - Εισαγωγή

Περιγραφή Ενεργειών Αποκατάστασης (συνέχεια)

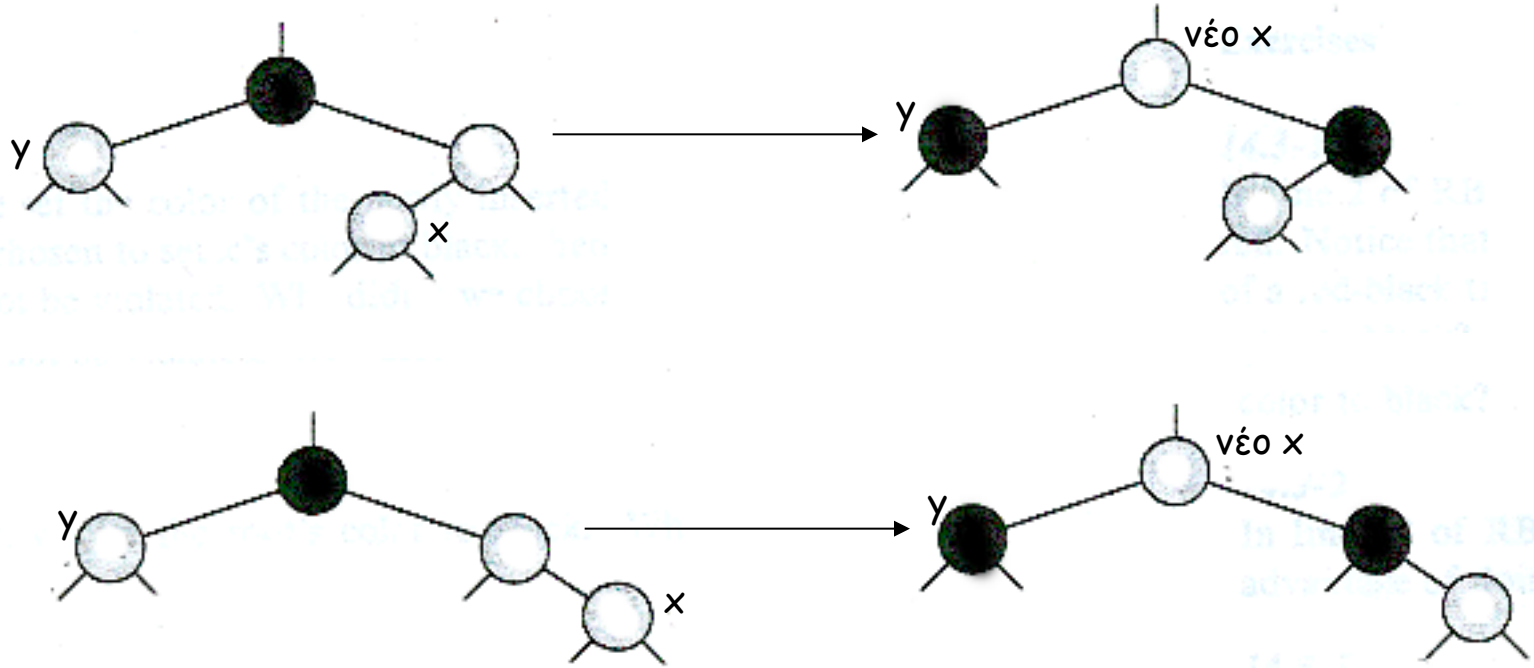
Περίπτωση 2: Ο y (αδελφικός κόμβος του πατέρα του x) είναι μαύρος και ο x είναι δεξί παιδί του πατέρα του. Ανάγουμε την περίπτωση αυτή στην περίπτωση 3 με την εκτέλεση μιας αριστερής περιστροφής γύρω από τον πατέρα του x. (Η περίπτωση που ο x είναι αριστερό παιδί του πατέρα του είναι συμμετρική).

Περίπτωση 3: Ο x είναι αριστερό παιδί του πατέρα του. Το χρώμα του πατέρα του x αλλάζει σε μαύρο και του παππού σε κόκκινο. Εκτελείται μια δεξιά περιστροφή γύρω από τον παππού του x.



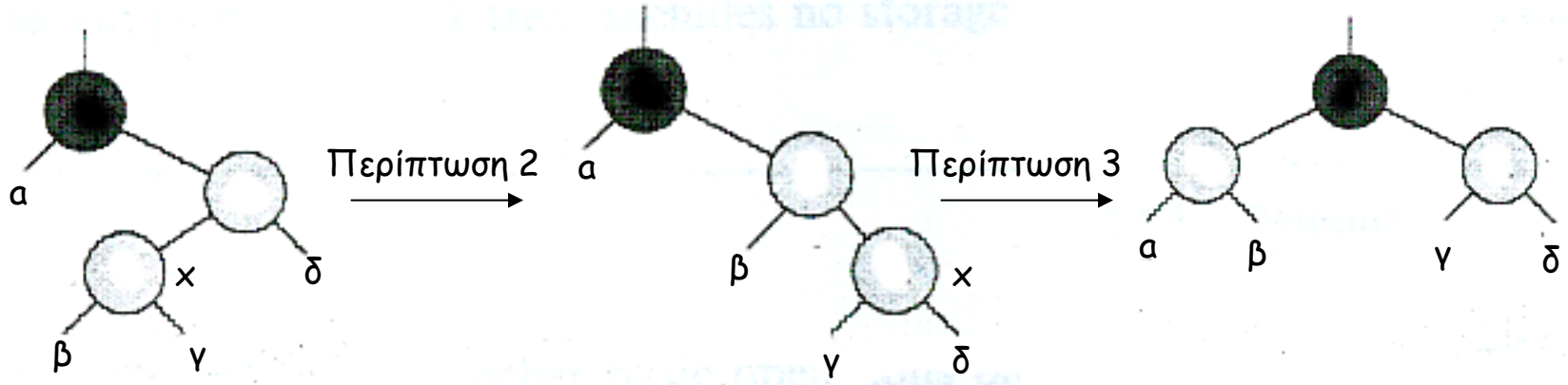
Σχήμα 13.6: Cormen, Leiserson, Rivest & Stein, Εισαγωγή στους Αλγορίθμους, Πανεπιστημιακές Εκδόσεις Κρήτης, 2006

Κοκκινόμαυρα Δένδρα – Εισαγωγή – Κάποιες από τις Συμμετρικές Περιπτώσεις



Περίπτωση 1: Ο αδελφικός κόμβος y του πατρικού κόμβου του x είναι κόκκινος. (Συμμετρική περίπτωση που ο πατρικός κόμβος του x είναι δεξιό παιδί του παππού του x.)

Κοκκινόμαυρα Δένδρα – Εισαγωγή – Κάποιες από τις Συμμετρικές Περιπτώσεις



Περίπτωση 2: Ο γ (αδελφικός κόμβος του πατέρα του x) είναι μαύρος και ο x είναι αριστερό παιδί του πατέρα του. Ανάγουμε την περίπτωση αυτή στην περίπτωση 3 με την εκτέλεση μιας δεξιάς περιστροφής.

Περίπτωση 3: Ο x είναι δεξιό παιδί του πατέρα του. Το χρώμα του πατέρα του x αλλάζει σε μαύρο και του παππού σε κόκκινο. Εκτελείται μια αριστερή περιστροφή.

Κοκκινόμαυρα Δένδρα – Εισαγωγή - Ψευδοκώδικας

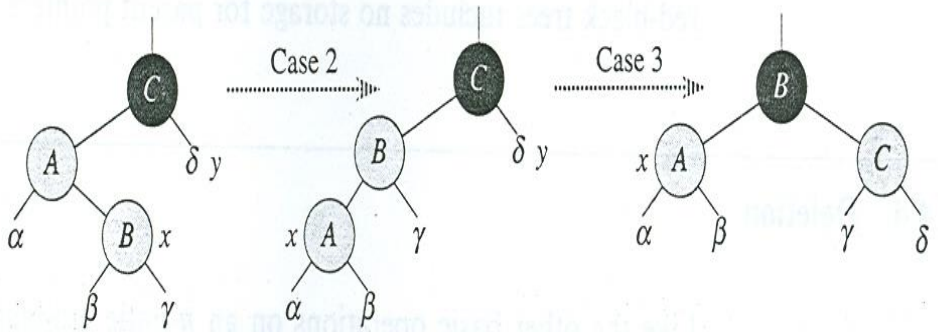
```
RedBlackTree-Insert(pointer R, Key k) {
/* Ο R είναι δείκτης στη ρίζα του δένδρου και το K είναι το
   προς-εισαγωγή κλειδί. Θεωρώ ότι το δένδρο είναι διπλά συνδεδεμένο */

   y = nill; z = R;
   while (z != null) {   // έχω αγνοήσει τους ελέγχους για το αν το κλειδί K βρίσκεται ήδη στο δένδρο
        y = z;           // οι οποίοι χρειάζονται
        if (K < z->Key) z = z->LC;
        else z = z->RC;
   }
   x = newcell();  x->Key = K;  x->LC = x->RC = null;  x->color = RED;
   x->p = y;
   if (y == null) (κάνε το x ρίζα του δένδρου);
   else if (x->key < y->key) y->lc = x;      // τοποθέτηση του x ως
   else y->rc = x;                           // κατάλληλου παιδιού του y
   ColorPropertiesInsert(R,x)
}
```

Κοκκινόμαυρα Δένδρα – Εισαγωγή

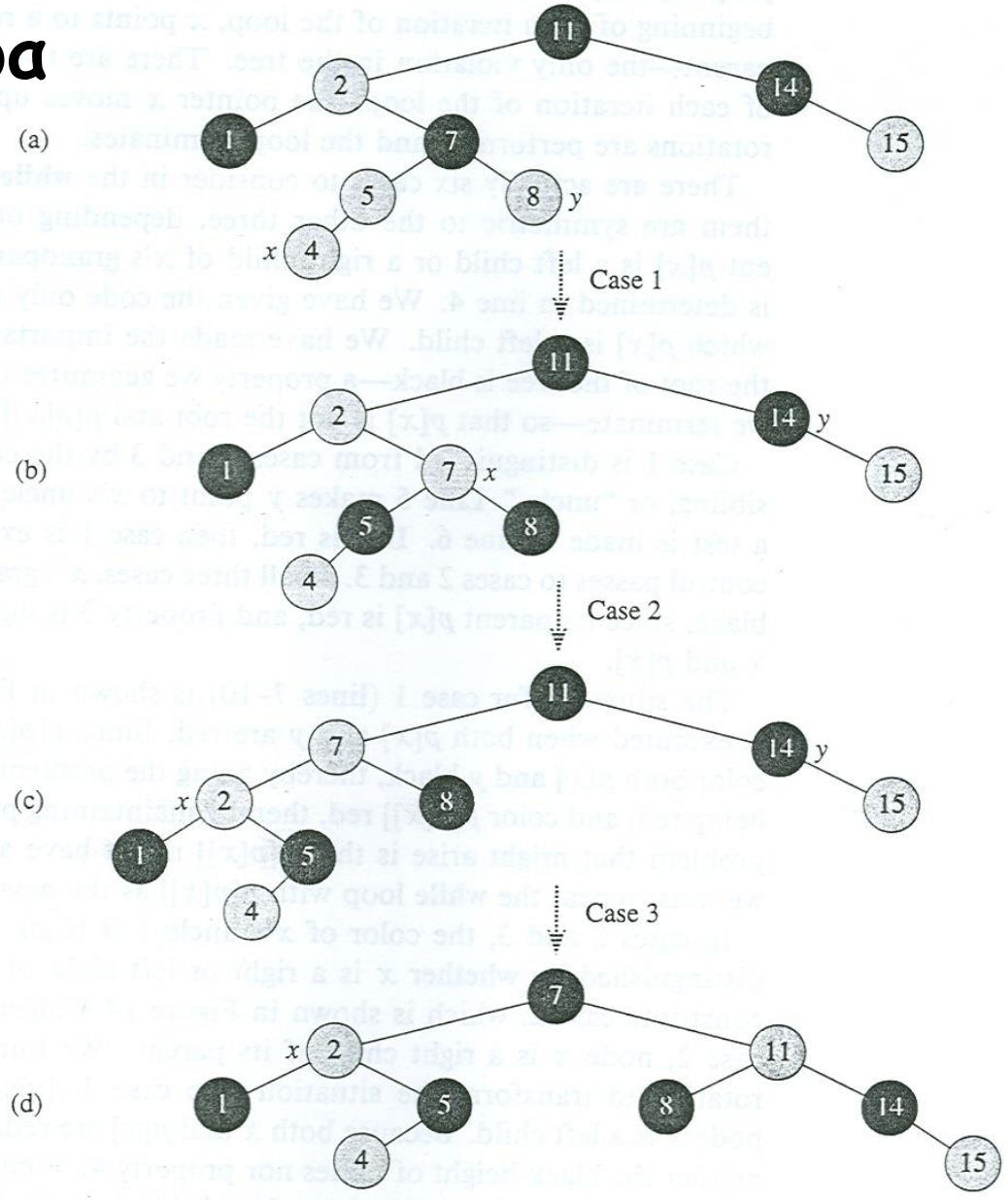
```
ColorPropertiesInsert(pointer R, pointer x) {
    while (x->p->color == RED) {     //αν το χρώμα του πατέρα του x είναι επίσης κόκκινο
        if (x->p == x->p->p->LC) {     // αν ο πατέρας του x είναι αριστερό παιδί του παππού του x
            y = x->p->p->RC;          // ο y είναι ο αδελφικός κόμβος του πατέρα του x
            if (y->color == RED) {
                x->p->color = BLACK;    // πατέρας και αδελφικός κόμβος πατέρα
                y->color = BLACK;        // γίνονται και οι δυο μαύροι
                x->p->p->color = RED;   // ο παππούς γίνεται κόκκινος
                x = x->p->p;           // (νέο x = παππούς x); και επαναλαμβάνουμε
             }
            else {
                if (x == x->p->RC) {      // ο x είναι δεξιό παιδί του πατέρα του ⇒ Περίπτωση 2
                    x = x->p;              // ο πατέρας του x θα παίξει το ρόλο του x (δείτε σχήματα)
                    LeftRotate(R, x);    // αριστερή περιστροφή γύρω από τον πατέρα του x
                  }
                x->p->color = BLACK;
                x->p->p->color = RED;
                RightRotate(R, x->p->p);
             }
        }
        else ( (1) με εναλλαγή του LC με RC
                και το αντίστροφο, καθώς και του LeftRotate με RightRotate
          και το αντίστροφο);
    } }
```

(1)



Κοκκινόμαυρα Δένδρα Εισαγωγή Παράδειγμα

Ποια είναι η χρονική πολυπλοκότητα της RBInsert()?



Σχήμα 13.4: Cormen, Leiserson, Rivest & Stein, Εισαγωγή στους Αλγορίθμους, Πανεπιστημιακές Εκδόσεις Κρήτης, 2006

Κοκκινόμαυρα Δένδρα – Διαγραφή

- Η διαγραφή γίνεται με παρόμοιο τρόπο όπως σε ταξινομημένο δυαδικό δένδρο και στη συνέχεια ελέγχεται αν οι ιδιότητες των κοκκινόμαυρων δένδρων ισχύουν ή όχι και γίνονται κατάλληλες ενέργειες αν απαιτούνται.
- Έστω y ο κόμβος που θα διαγραφεί από το δένδρο.
- Αν το χρώμα του y είναι κόκκινο, ο y διαγράφεται και ο αλγόριθμος τερματίζει.

- Γιατί δεν προκύπτει πρόβλημα με τις ιδιότητες χρωματισμού του δένδρου σε αυτή την περίπτωση?
  - Δεν υπάρχουν κόμβοι των οποίων το μαύρο ύψος να έχει μεταβληθεί στο δένδρο
  - Δεν έχει πραγματοποιηθεί καμία απευθείας σύνδεση μεταξύ δύο κόκκινων κόμβων
  - Δεδομένου ότι, αν ο y ήταν κόκκινος, δεν μπορεί να είναι ο ριζικός κόμβος, ο ριζικός κόμβος παραμένει μαύρος.

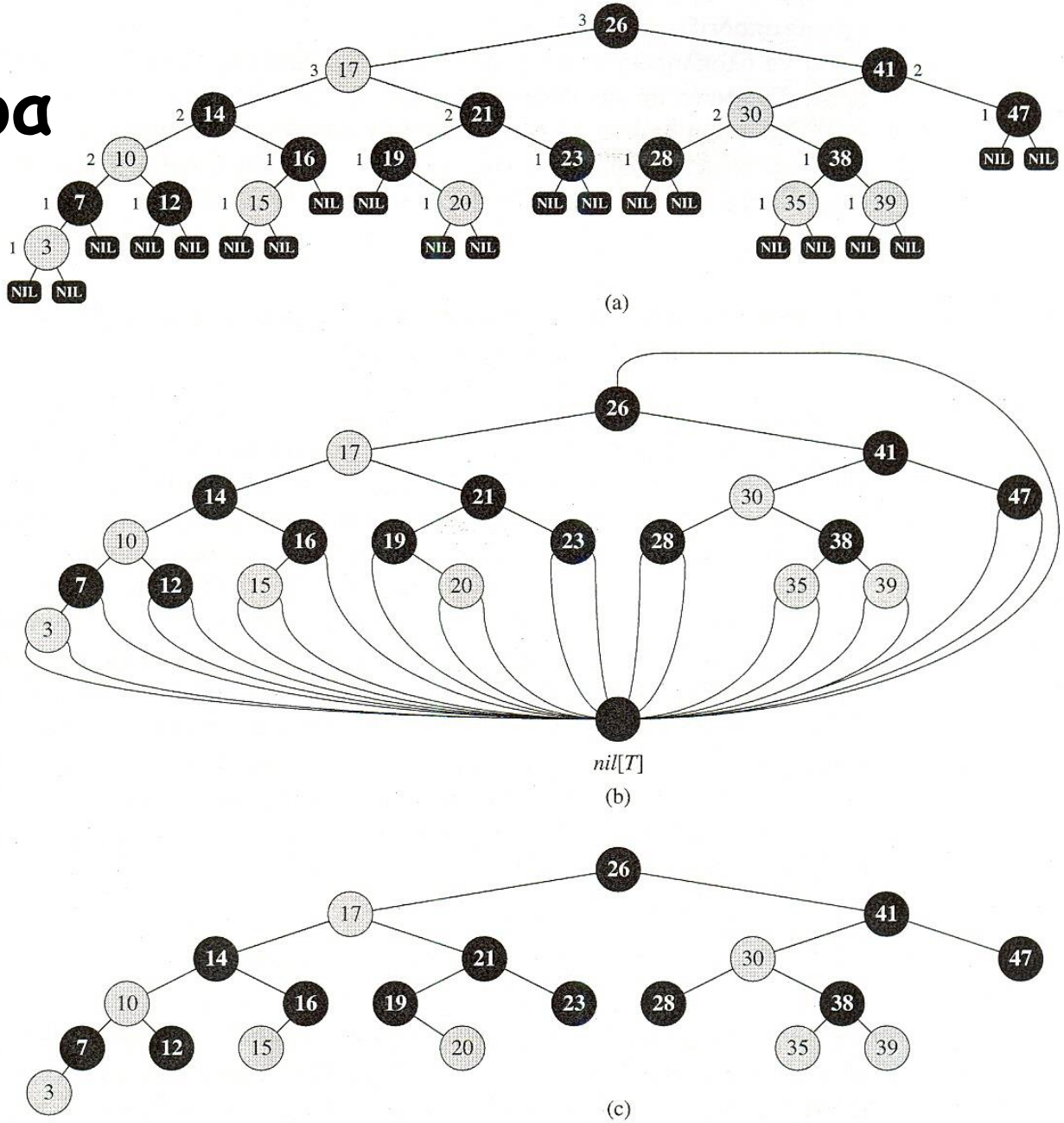
Κοκκινόμαυρα Δένδρα – Διαγραφή

- Αν το χρώμα του y είναι μαύρο, η διαγραφή του μπορεί να δημιουργήσει τα εξής προβλήματα:
  - Αν ο y ήταν ο ριζικός κόμβος και τεθεί ως νέος ριζικός κάποιος κόκκινος κόμβος, έχει παραβιαστεί η ιδιότητα 2.
  - Αν ο μοναδικός θυγατρικός κόμβος x του y και ο πατρικός κόμβος p του y ήταν και οι δυο κόκκινοι, έχει παραβιαστεί η ιδιότητα 4.
  - Λόγω της διαγραφής του y, όλα τα μονοπάτια που περιείχαν τον y (πριν τη διαγραφή) έχουν ένα μαύρο κόμβο λιγότερο (μετά τη διαγραφή) $\Rightarrow$ μετά τη διαγραφή, όλοι οι πρόγονοι του y στο δένδρο παραβιάζουν την ιδιότητα 5.

Κοκκινόμαυρα Δένδρα – Διαγραφή

- Υποθέτουμε ότι η μαύρη ιδιότητα του y μεταφέρεται στο παιδί του (έστω x), το οποίο αν είναι μαύρο γίνεται διπλά μαύρο (που είναι μη επιτρεπτό). Φανταζόμαστε πως κάθε κόμβος έχει ένα και μοναδικό κουπόνι που καθορίζει το χρώμα του (μαύρο ή κόκκινο). Τότε, ένας διπλά-μαύρος κόμβος είναι σαν να έχει δύο κουπόνια με μαύρο χρώμα. Αυτό είναι μη επιτρεπτό και ο κόμβος πρέπει να απαλλαγεί από το επιπρόσθετο μαύρο κουπόνι.
- Προκειμένου να επιτευχθεί αυτό, το επιπρόσθετο μαύρο κουπόνι μεταφέρεται προς τα πάνω (ακολουθώντας το μονοπάτι από τον κόμβο προς τη ρίζα) μέχρι είτε:
  - να φθάσουμε στη ρίζα, ή
  - να βρούμε έναν κόκκινο κόμβο που τον επαναχρωματίζουμε μαύρο και ο αλγόριθμος τερματίζει, ή
  - να μπορούν να εκτελεστούν κατάλληλες περιστροφές και επαναχρωματισμοί κάποιων κόμβων ώστε να επιλυθεί το πρόβλημα.

Κοκκινόμαυρα Δένδρα Διαγραφή



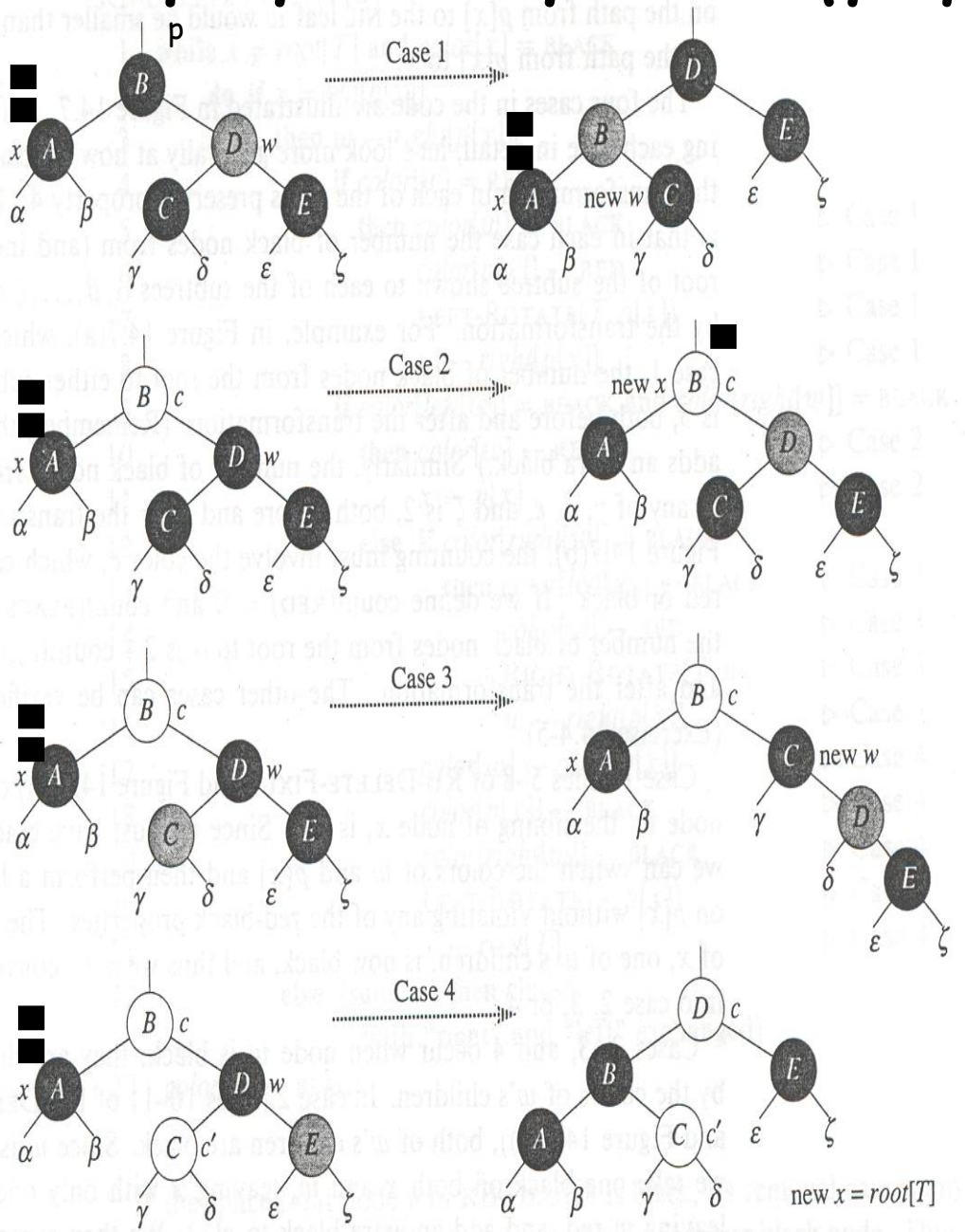
Θεωρώ ότι το δένδρο υλοποιείται με κόμβο φρουρό. Έτσι, όλοι οι nill δείκτες δείχνουν στον κόμβο φρουρό.

(Αυτό εξυπηρετεί στη μείωση των ελέγχων που απαιτούνται στον κώδικα που θα παρουσιαστεί σε επόμενες διαφάνειες).

Σχήμα 13.1: Cormen, Leiserson, Rivest & Stein, Εισαγωγή στους Αλγορίθμους, Πανεπιστημιακές Εκδόσεις Κρήτης, 2006

Κοκκινόμαυρα Δένδρα- Διαγραφή

- Έστω x το παιδί του κόμβου που διαγράφεται, w ο αδελφικός κόμβος και p ο πατέρας του x.
- Αν ο x είναι διπλά μαύρος, ο w δεν μπορεί να είναι ο κόμβος φρουρός. Γιατί;
- Υποθέτουμε ότι ο x είναι αριστερό παιδί του πατρικού του κόμβου (η αντίθετη περίπτωση είναι συμμετρική)
- Διακρίνουμε περιπτώσεις ως προς το χρώμα του w.

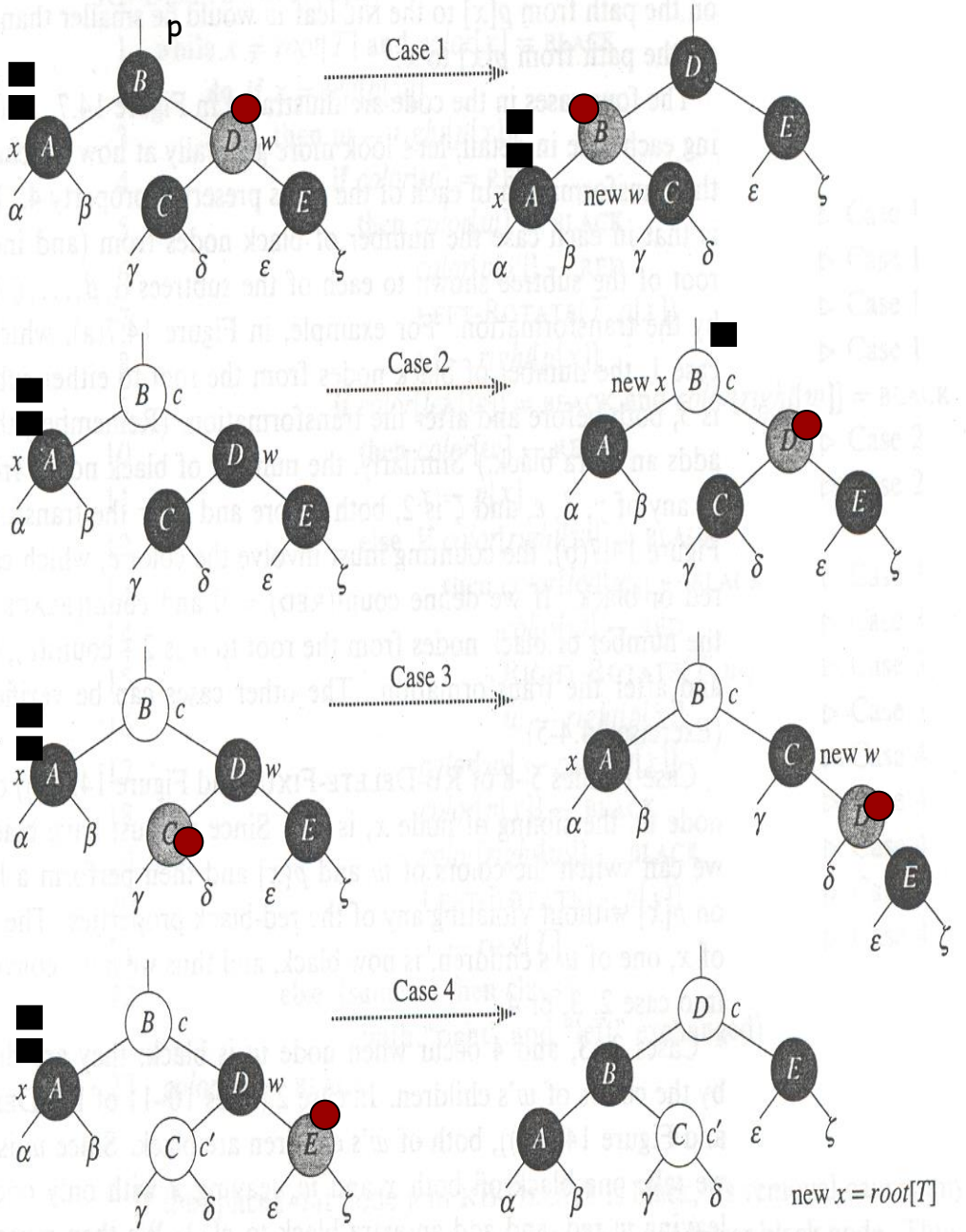


Σχήμα 13.7: Cormen, Leiserson, Rivest & Stein, Εισαγωγή στους Αλγορίθμους, Πανεπιστημιακές Εκδόσεις Κρήτης, 2006

Κοκκινόμαυρα Δένδρα - Διαγραφ

Βασική Ιδέα

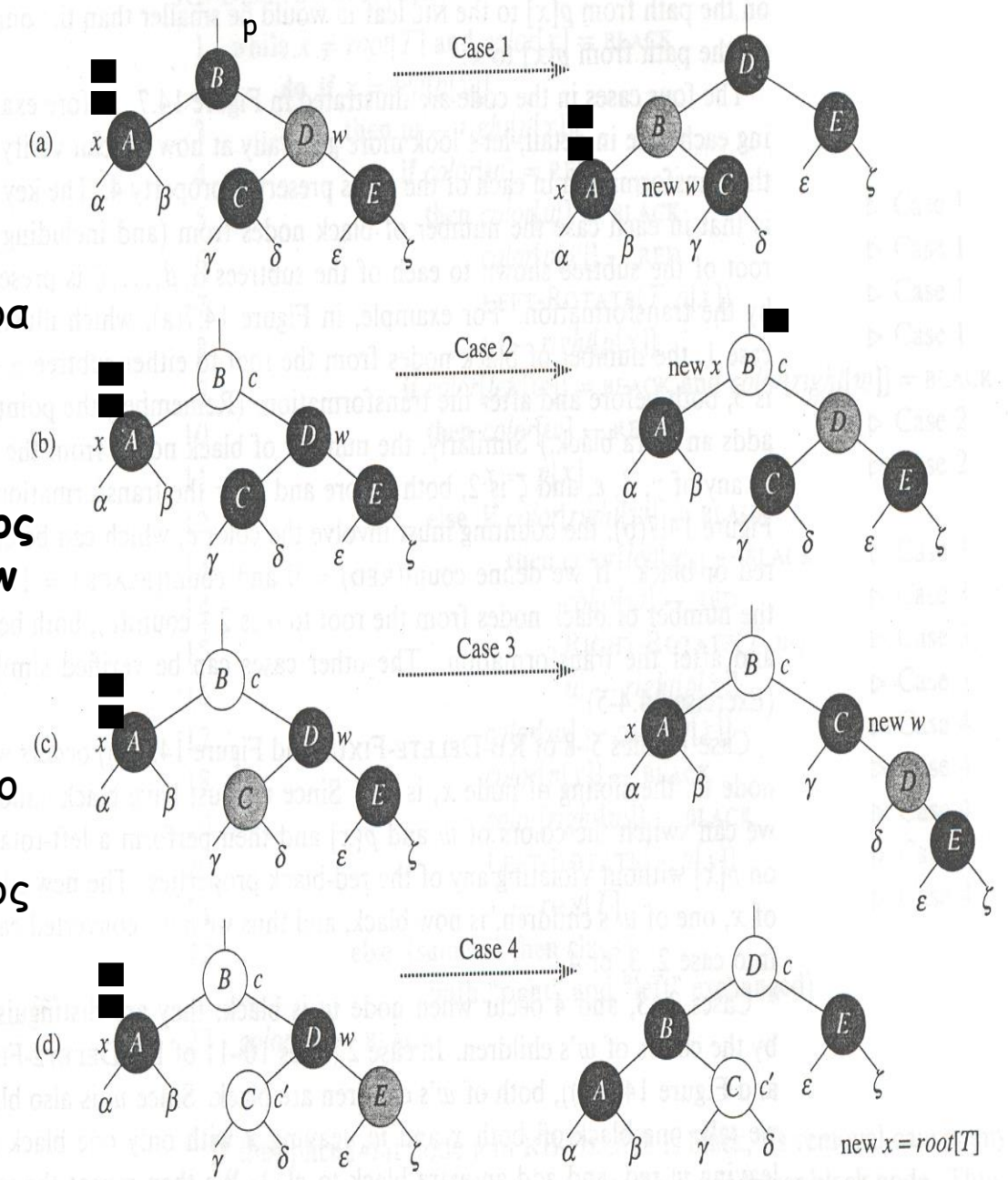
Σε όλες τις περιπτώσεις, ο μετασχηματισμός διατηρεί το πλήθος των μαύρων κόμβων (συμπεριλαμβανομένου και του extra μαύρου κουπονιού του x) από τον ριζικό κόμβο του εικονιζόμενου υποδένδρου (συμπεριλαμβανομένου του ριζικού) μέχρι καθένα από τα υποδένδρα α,β,γ,δ,ε και ζ.



Κοκκινόμαυρα Δένδρα - Διαγραφή

1. **Περίπτωση 1: Ο w είναι κόκκινος.**
   Αλλάζουμε το χρώμα του w σε μαύρο και του p σε κόκκινο και εκτελούμε μια αριστερή περιστροφή γύρω από τον πατέρα του x. Έτσι, μεταπίπτουμε στην περίπτωση 2, 3, ή 4.

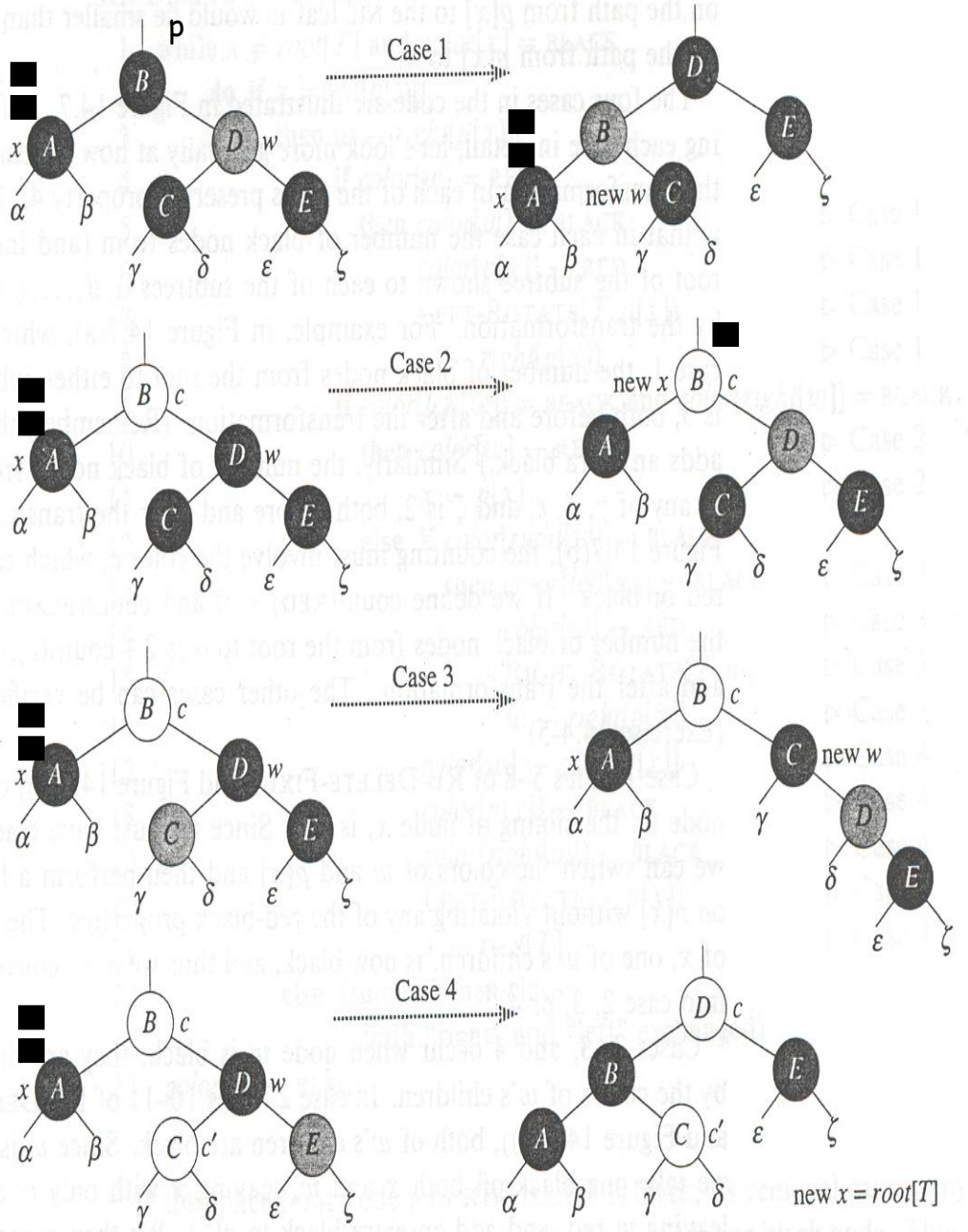
- **Περίπτωση 2: Ο w είναι μαύρος κόμβος και τα δύο παιδιά του w είναι μαύρα.**
  Αλλάζουμε το χρώμα του w σε κόκκινο, του x σε μαύρο (από διπλά μαύρο) και μεταφέρουμε το μαύρο που αφαιρέσαμε από τους w, x στον p. Αν ο p ήταν κόκκινος γίνεται μαύρος και ο αλγόριθμος τερματίζει. Διαφορετικά, ο p γίνεται διπλά μαύρος και ο αλγόριθμος επαναλαμβάνεται με x = p.



Κοκκινόμαυρα Δένδρα - Διαγραφή

3. **Περίπτωση 3: Το w->lc είναι κόκκινο και το w->rc μαύρο.** Αλλάζω το χρώμα του w σε κόκκινο και του w->lc σε μαύρο και εκτελώ μια περιστροφή γύρω από το w ⇒ Μεταπίπτουμε στην περίπτωση 4

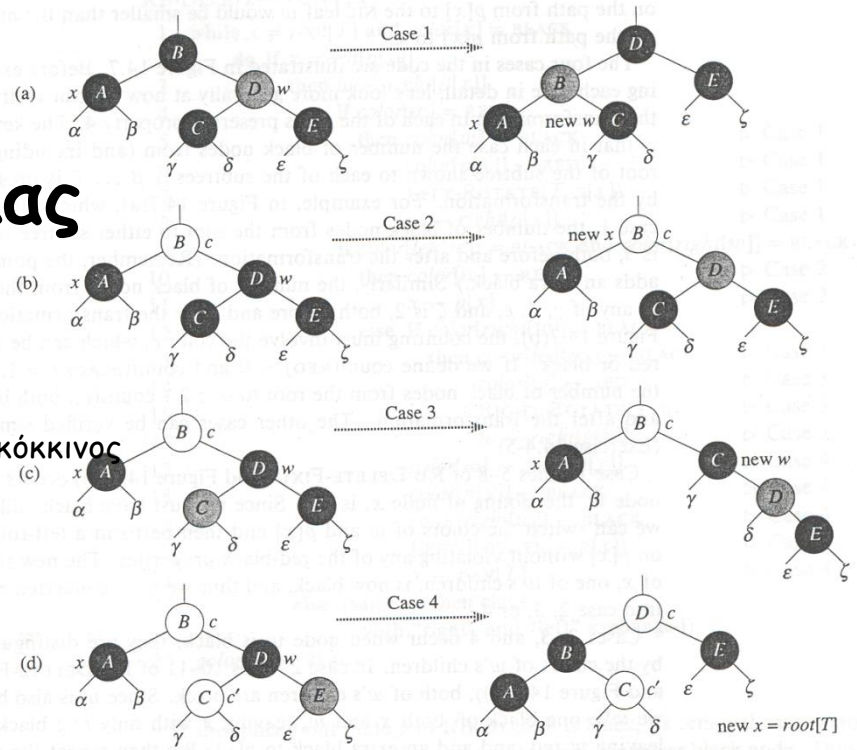
4. **Περίπτωση 4: Το w->rc είναι κόκκινο.** Αλλάζω το χρώμα του w->rc σε μαύρο, του w σε ότι ήταν το χρώμα του p και του p σε μαύρο και εκτελώ μια περιστροφή γύρω από τον p. Ο αλγόριθμος τερματίζει.



Κοκκινόμαυρα Δένδρα – Διαγραφή - Ψευδοκώδικας

```
RedBlack-Delete(pointer R, pointer z) {
   if (z->RC == NULL OR z->LC == NULL)
      y = z;                                  // αν ο z έχει ένα ή κανένα παιδί, διαγράφεται ο ίδιος ο z
   else y = TreeSuccessor(R,z);               // διαφορετικά, διαγράφεται ο επόμενος κόμβος στην ενδ/γμένη διάσχιση
   if (y->LC != NULL) x = y->LC;              // ο x δείχνει στο μοναδικό παιδί του y
   else x = y->RC;
   x->p = y->p;
   if (y->p == NULL) (κάνε το x ρίζα του δένδρου);
   else if (y == y->p->LC) y->p->LC = x;      // αν ο y είναι αριστερό παιδί του πατέρα του
   else y->p->RC = x;
   if (y != z) {                              // αν ο προς διαγραφή κόμβος είναι ο επόμενος στην ενδ/γμένη διάσχιση
      z->Key = y->key;                        // αντιγραφή όλων των πεδίων δεδομένων
      (αντιγραφή των υπολοίπων πεδίων δεδομένων του y στο z);
   }
    if (y->color == BLACK) ColorPropertiesDelete(R,x);
                  // αν η διαγραφή αφορούσε κόμβο με χρώμα μαύρο, εκτέλεση κατάλληλων
                  //   ενεργειών για την επαναφορά των ιδιοτήτων χρωματισμού του δένδρου
    return y;
}
```

Κοκκινόμαυρα Δένδρα – Διαγραφή – Ψευδοκώδικας




```
ColorPropertiesDelete(pointer R, pointer x) {
  while (x != R AND x->color == BLACK) {
    if (x == x->p->LC) {  // αν ο x είναι αριστερό παιδί
      w = x->p->RC;
      if (w->color == RED) {      // αν αδελφικός του x κόκκινος
        w->color = BLACK;         // Περίπτωση 1
        x->p->color = RED;        // Περίπτωση 1
        LeftRotation(R, x->p);    // Περίπτωση 1
        w = x->p->RC;             // Περίπτωση 1
      }
      if (w->LC->color == BLACK AND
          w->RC->color == BLACK) {
                  // αν και τα δύο παιδιά του w είναι μαύρα
(2)     w->color = RED;           // Περίπτωση 2
        x = x->p;                 // Περίπτωση 2
      }
      else {
        if (w->RC->color == BLACK) {
                                  // αν δεξιό παιδί του x μαύρο
          w->LC->color = BLACK;   // Περίπτωση 3
          w->color = RED;         // Περίπτωση 3
          RightRotation(R,w);     // Περίπτωση 3
          w = x->p->RC;           // Περίπτωση 3
        }
        w->color = x->p->color;   // Περίπτωση 4
        x->p->color = BLACK;      // Περίπτωση 4
        w->RC->color = BLACK;//Περίπτωση 4
        LeftRotation(R, x->p);//Περίπτωση 4
        x = R;                    // Περίπτωση 4
      } /* else */
    } /* if */
    else (ίδια με (2) με εναλλαγή του LC με RC
          και το αντίστροφο, καθώς και του
          LeftRotate με RightRotate και το
          αντίστροφο);
  }
  x->color = BLACK;
}
```

Ποια είναι η χρονική πολυπλοκότητα της RBDelete()?

Αναφορές

Το υλικό της ενότητας αυτής περιέχεται στα ακόλουθα βιβλία:

- Harry Lewis and Larry Denenberg, *Data Structures and Their Algorithms*, Harper Collins Publishers, Inc., New York, 1991
  - Chapter 7: Tree Structures for Dynamic Dictionaries
- Cormen, Leiserson, Rivest & Stein, *Εισαγωγή στους Αλγορίθμους,* Πανεπιστημιακές Εκδόσεις Κρήτης, 2006.
  - Κεφάλαιο 13: Μελανέρυθρα Δένδρα
- Ι. Μανωλόπουλος, Δομές Δεδομένων, Μια προσέγγιση με την Pascal, Εκδόσεις Art of Text.
  - Κεφάλαιο 5: Ισοζυγισμένα Δένδρα

Τέλος Ενότητας

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Χρηματοδότηση

•Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.

•Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Κρήτης**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.

•Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Σημειώματα

Σημείωμα αδειοδότησης

.

Σημείωμα Αναφοράς