Κοκκινόμαυρα Δέντρα [1]

Έχουμε δει ότι ένα δυαδικό δέντρο αναζήτησης ύψους h επιτρέπει την εκτέλεση των λειτουργιών της Εισαγωγής, της Διαγραφής και της Αναζήτησης σε $O(h)$ χρόνο. Συνεπώς, όταν το ύψος του δέντρου είναι μικρό, οι διαδικασίες αυτές εκτελούνται γρήγορα. Όταν όμως το ύψος του δέντρου είναι μεγάλο, οι διαδικασίες αυτές μπορεί να είναι το ίδιο χρονοβόρες όσο και οι αντίστοιχες διαδικασίες για γραμμικές λίστες, δηλαδή, να απαιτούν χρόνο γραμμικό στο πλήθος των κόμβων της λίστας ή του δέντρου.

Τα δέντρα που θα μελετήσουμε σ' αυτό το κεφάλαιο ανήκουν στην κατηγορία των "ισοζυγισμένων" δυαδικών δέντρων αναζήτησης. Τα δέντρα αυτά επιτρέπουν την εκτέλεση των παραπάνω λειτουργιών σε χρόνο λογαριθμικό στο πλήθος των κόμβων του δέντρου στη χειρότερη περίπτωση.

1 Ιδιότητες των κοκκινόμαυρων δέντρων

Ένα **κοκκινόμαυρο δέντρο** *(red-black tree)* είναι ένα δυαδικό δέντρο αναζήτησης με ένα επιπλέον στοιχείο πληροφορίας ανά κόμβο: το χρώμα του κόμβου, που μπορεί να είναι είτε **κόκκινο** είτε **μαύρο**. Με την επιβολή περιορισμών στον τρόπο που μπορούν να χρωματιστούν οι κόμβοι, τα κοκκινόμαυρα δέντρα εξασφαλίζουν ότι καμία διαδρομή από τη ρίζα σε κάποιο φύλλο δεν έχει μήκος που υπερβαίνει το διπλάσιο του μήκους οποιασδήποτε άλλης τέτοιας διαδρομής, και έτσι το δέντρο είναι σχεδόν **ισοζυγισμένο** *(balanced)*.

Κάθε κόμβος ενός κοκκινόμαυρου δέντρου περιλαμβάνει τα πεδία *color*, *key*, *p*, *left* και *right* που καταχωρούν το χρώμα, το κλειδί, και δείκτες στον πατέρα, στο αριστερό και στο δεξί παιδί του κόμβου, αντίστοιχα. Εάν κάποιο παιδί ενός κόμβου δεν υπάρχει, ο αντίστοιχος δείκτης είναι NIL. Θα θεωρήσουμε αυτούς τους δείκτες NIL ως δείκτες σε εξωτερικούς NIL-κόμβους (φύλλα) του δυαδικού δέντρου αναζήτησης και τους κανονικούς κόμβους που αποθηκεύουν κλειδιά ως εσωτερικούς κόμβους του δέντρου.

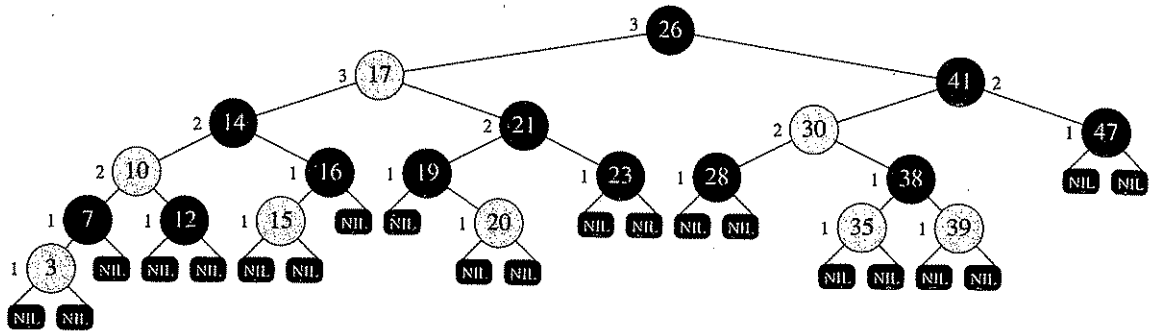
Ένα δυαδικό δέντρο αναζήτησης είναι κοκκινόμαυρο δέντρο εάν πληροί τις ακόλουθες ιδιότητες οι οποίες στη συνέχεια θα αναφέρονται ως **ΚΜ-Ιδιότητες**:

1. Κάθε κόμβος έχει είτε κόκκινο είτε μαύρο χρώμα.
2. Κάθε NIL-κόμβος έχει μαύρο χρώμα.
3. Εάν ένας κόμβος έχει κόκκινο χρώμα, τότε και τα δυο παιδιά του έχουν μαύρο χρώμα.
4. Κάθε απλή διαδρομή από έναν κόμβο σε έναν NIL-κόμβο που είναι απόγονός του περιέχει το ίδιο πλήθος μαύρων κόμβων.

Ένα παράδειγμα ενός κοκκινόμαυρου δέντρου παρουσιάζεται στο Σχήμα 1.

Το πλήθος των μαύρων κόμβων σε οποιαδήποτε διαδρομή από έναν κόμβο x (και χωρίς να τον συμπεριλαμβάνει) σε έναν NIL-κόμβο που είναι απόγονος του x ονομάζεται **μαύρο ύψος** *(black height)* του κόμβου και συμβολίζεται με $bh(x)$. Σύμφωνα με την Ιδιότητα 4, η έννοια

[1] Μετάφραση από το κεφάλαιο 14 του βιβλίου *Introduction to Algorithms* των T.H. Cormen, C.E. Leiserson και R.L. Rivest.



Σχήμα 1. Ένα κοκκινόμαυρο δέντρο στο οποίο οι μαύροι κόμβοι δείχνονται σκιασμένοι και οι κόκκινοι κόμβοι λευκοί. Κάθε κόμβος σε ένα κοκκινόμαυρο δέντρο έχει είτε κόκκινο είτε μαύρο χρώμα, κάθε NIL-κόμβος έχει μαύρο χρώμα, τα παιδιά ενός κόκκινου κόμβου έχουν και τα δύο μαύρο χρώμα, και κάθε απλή διαδρομή από έναν κόμβο σε έναν NIL-κόμβο που είναι απόγονός του περιέχει το ίδιο πλήθος μαύρων κόμβων. Δίπλα σε κάθε εσωτερικό κόμβο σημειώνεται το μαύρο ύψος του· οι NIL-κόμβοι έχουν μαύρο ύψος 0.

του μαύρου ύψους είναι καλά ορισμένη, αφού όλες οι διαδρομές από τον κόμβο x στους NIL-απογόνους του έχουν το ίδιο πλήθος μαύρων κόμβων. Το μαύρο ύψος ενός κοκκινόμαυρου δέντρου ορίζεται ως το μαύρο ύψος της ρίζας του.

Το ακόλουθο λήμμα δείχνει γιατί τα κοκκινόμαυρα δέντρα είναι δέντρα αναζήτησης στα οποία οι διαδικασίες Εισαγωγής, Διαγραφής και Αναζήτησης γίνονται πολύ αποτελεσματικά.

Λήμμα 1
Ένα κοκκινόμαυρο δέντρο με n εσωτερικούς κόμβους έχει ύψος το πολύ $2\log(n+1)$.

Απόδειξη. Θα δείξουμε πρώτα ότι το υποδέντρο με ρίζα έναν κόμβο, έστω x, περιέχει τουλάχιστον $2^{bh(x)}-1$ εσωτερικούς κόμβους. Θα αποδείξουμε τον ισχυρισμό αυτό με επαγωγή στο ύψος[2] του κόμβου x. Εάν το ύψος του είναι 0, τότε ο x πρέπει να είναι NIL-κόμβος. Άρα $bh(x)=0$ και το δέντρο με ρίζα τον x πραγματικά έχει $2^{bh(x)}-1=2^0-1=0$ εσωτερικούς κόμβους. Για το επαγωγικό βήμα, θεωρούμε ότι ο κόμβος x έχει θετικό ύψος. Συνεπώς, ο x είναι ένας εσωτερικός κόμβος με δύο παιδιά. Κάθε παιδί έχει μαύρο ύψος ίσο με $bh(x)$ ή $bh(x)-1$, ανάλογα με το εάν το χρώμα του είναι κόκκινο ή μαύρο, αντίστοιχα. Αφού το ύψος των παιδιών του x είναι μικρότερο από το ύψος του ίδιου του x, μπορούμε να εφαρμόσουμε την επαγωγική υπόθεση και να συμπεράνουμε ότι κάθε παιδί έχει τουλάχιστον $2^{bh(x)-1}-1$ εσωτερικούς κόμβους. Άρα, το υποδέντρο με ρίζα τον x έχει τουλάχιστον $(2^{bh(x)-1}-1)+(2^{bh(x)-1}-1)+1=2^{bh(x)}-1$ εσωτερικούς κόμβους, πράγμα που αποδεικνύει τον ισχυρισμό.

Για να ολοκληρώσουμε την απόδειξη, θεωρούμε ότι το ύψος του δέντρου είναι h. Σύμφωνα με την Ιδιότητα 3, τουλάχιστον οι μισοί κόμβοι σε οποιαδήποτε απλή διαδρομή από τη ρίζα σε έναν NIL-κόμβο, μη συμπεριλαμβανομένης της ρίζας, έχουν μαύρο χρώμα. Συνεπώς, το μαύρο ύψος της ρίζας πρέπει να είναι τουλάχιστον $h/2$, οπότε

$$n \geq 2^{h/2}-1.$$

Μεταφέροντας το 1 στο αριστερό μέλος της ανισότητας και υπολογίζοντας τους λογαρίθμους (με βάση το 2) των δύο μελών βρίσκουμε $\log(n+1) \geq h/2$ ή $h \leq 2\log(n+1)$. ∎

[2]Εδώ πρόκειται για το ύψος του κόμβου στο δέντρο και όχι το μαύρο ύψος του.

Άμεση συνέπεια του παραπάνω λήμματος είναι ότι οι διαδικασίες της Εισαγωγής (TREE-INSERT), της Διαγραφής (TREE-DELETE) και της Αναζήτησης (TREE-SEARCH) για δυαδικά δέντρα αναζήτησης απαιτούν $O(\log n)$ χρόνο όταν εκτελεστούν σε ένα κοκκινόμαυρο δέντρο, δεδομένου ότι απαιτούν $O(h)$ χρόνο σε ένα δυαδικό δέντρο αναζήτησης ύψους h. Ωστόσο, σε κοκκινόμαυρα δέντρα, ενώ η διαδικασία TREE-SEARCH αρκεί για Αναζήτηση, οι διαδικασίες TREE-INSERT και TREE-DELETE δεν είναι επαρκείς από μόνες τους για εισαγωγή και διαγραφή κόμβων, γιατί δεν εξασφαλίζουν ότι το δυαδικό δέντρο που θα προκύψει θα είναι ένα κοκκινόμαυρο δέντρο. [Σημειώνεται ότι οι διαδικασίες Εισαγωγής και Διαγραφής κόμβων σε κοκκινόμαυρα δέντρα εκτελούνται πράγματι σε $O(\log n)$ χρόνο, όπως περιγράφεται στις Παραγράφους 3 και 4 αντίστοιχα.]

Ασκήσεις

1-1
Σχεδιάστε το πλήρες δυαδικό δέντρο αναζήτησης ύψους 3 με κλειδιά $\{1, 2, \ldots, 15\}$ και προσθέστε τους NIL-κόμβους. Χρωματίστε τους κόμβους με τρεις διαφορετικούς τρόπους έτσι ώστε τα μαύρα ύψη των κοκκινόμαυρων δέντρων που θα προκύψουν να είναι 2, 3 και 4.

1-2
Έστω ότι η ρίζα ενός κοκκινόμαυρου δέντρου έχει κόκκινο χρώμα. Αλλάζουμε το χρώμα της από κόκκινο σε μαύρο. Το δέντρο που προκύπτει από την αλλαγή αυτή παραμένει κοκκινόμαυρο;

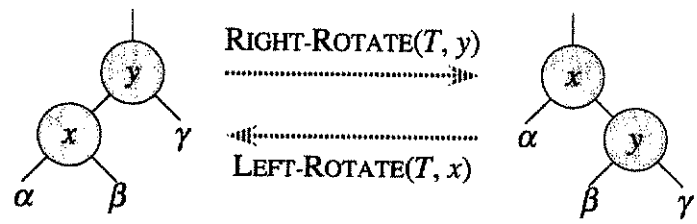
1-3
Δείξτε ότι η μέγιστη απλή διαδρομή από έναν κόμβο x σε έναν NIL-απόγονο έχει μήκος το πολύ διπλάσιο του μήκους της ελάχιστης απλής διαδρομής από τον x σε έναν NIL-απόγονο.

1-4
Ποιο είναι το μέγιστο δυνατό πλήθος εσωτερικών κόμβων σε ένα κοκκινόμαυρο δέντρο με μαύρο ύψος k; Ποιο είναι το ελάχιστο δυνατό πλήθος;

1-5
Περιγράψτε ένα κοκκινόμαυρο δέντρο με n κλειδιά που επιτυγχάνει τον μέγιστο δυνατό λόγο κόκκινων προς μαύρους εσωτερικούς κόμβους. Ποιος είναι αυτός ο λόγος; Ποιο δέντρο έχει τον ελάχιστο δυνατό λόγο, και ποιος είναι αυτός ο λόγος;

2 Περιστροφές

Οι διαδικασίες της Εισαγωγής και Διαγραφής κόμβων σε δυαδικά δέντρα αναζήτησης απαιτούν $O(\log n)$ χρόνο, όταν αυτές εκτελεστούν σε ένα κοκκινόμαυρο δέντρο με n κλειδιά. Επειδή οι διαδικασίες αυτές τροποποιούν το δέντρο, το δέντρο που προκύπτει μπορεί να μην πληροί πλέον τις ΚΜ-Ιδιότητες που παρουσιάστηκαν στην Παράγραφο 1. Για να επαναφέρουμε σε ισχύ αυτές τις ιδιότητες, θα πρέπει να αλλάξουμε τόσο τη δομή του δέντρου (αλλαγή δεικτών προς συγκεκριμένους κόμβους) όσο και το χρώμα κάποιων από τους κόμβους του.



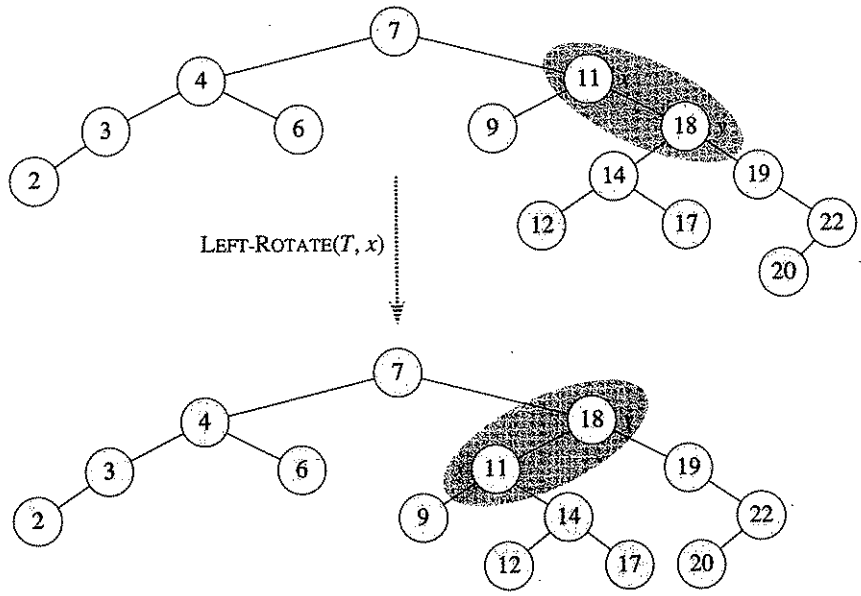
Σχήμα 2. Τα δύο είδη περιστροφών σε ένα δυαδικό δέντρο αναζήτησης. Η διαδικασία RIGHT-ROTATE(T, x) μετασχηματίζει τον συνδυασμό των δύο κόμβων στα αριστερά στον συνδυασμό κόμβων στα δεξιά αλλάζοντας ένα σταθερό πλήθος δεικτών. Ο συνδυασμός κόμβων στα δεξιά μπορεί να μετασχηματιστεί στον συνδυασμό κόμβων στα αριστερά μέσω της αντίστροφης διαδικασίας LEFT-ROTATE(T, y). Οι δύο κόμβοι θα μπορούσαν να βρίσκονται οπουδήποτε στο δέντρο αναζήτησης. Τα γράμματα α, β και γ αντιπροσωπεύουν τυχόντα υποδέντρα. Οποιαδήποτε από τις δύο περιστροφές διατηρεί την ενδοδιατεταγμένη σειρά των κλειδιών: τα κλειδιά στο α προηγούνται του $key[x]$, το οποίο προηγείται των κλειδιών στο β, τα οποία προηγούνται του $key[y]$, το οποίο τέλος προηγείται των κλειδιών στο γ.

Η αλλαγή της δομής του δέντρου επιτυγχάνεται χρησιμοποιώντας **περιστροφές** *(rotations)*· αυτές είναι τοπικές δομικές μεταβολές σε ένα δέντρο αναζήτησης που διατηρούν την ενδοδιατεταγμένη σειρά των κλειδιών. Το Σχήμα 2 παρουσιάζει τα δύο είδη περιστροφών: τις αριστερές περιστροφές *(left rotations)* και τις δεξιές περιστροφές *(right rotations)*. Όταν εκτελούμε μια αριστερή περιστροφή σε έναν κόμβο x, υποθέτουμε ότι το δεξί παιδί του, έστω y, δεν είναι NIL-κόμβος. Η αριστερή περιστροφή εκτελείται γύρω από την ακμή που συνδέει τους κόμβους x και y. Η περιστροφή αυτή θέτει τον κόμβο y ως τη νέα ρίζα του υποδέντρου, τον κόμβο x ως αριστερό παιδί του y και το αριστερό παιδί του y ως δεξί παιδί του x.

Στον ψευδοκώδικα για την αριστερή περιστροφή –LEFT-ROTATE– που ακολουθεί υποθέτουμε ότι το δεξί παιδί $right[x]$ του x δεν είναι NIL-κόμβος.

```
LEFT-ROTATE(T, x)
 1  y ← right[x]              ▷ ανάθεση στον y
 2  right[x] ← left[y]        ▷ το αριστερό υποδέντρο του y γίνεται δεξί υποδέντρο του x
 3  if left[y] ≠ NIL
 4    then p[left[y]] ← x
 5  p[y] ← p[x]               ▷ ο πατέρας του x συνδέεται με τον y
 6  if p[x] = NIL
 7    then root[T] ← y
 8    else if x = left[p[x]]
 9           then left[p[x]] ← y
10           else right[p[x]] ← y
11  left[y] ← x               ▷ ο x συνδέεται ως αριστερό παιδί του y
12  p[x] ← y
```

Το Σχήμα 3 δείχνει τα βήματα της διαδικασίας LEFT-ROTATE. Η διαδικασία δεξιάς περιστροφής (RIGHT-ROTATE) είναι παρόμοια. Τόσο η διαδικασία LEFT-ROTATE όσο και η



Σχήμα 3. Ένα παράδειγμα της δομικής μεταβολής που η διαδικασία LEFT-ROTATE(T, x) επιφέρει σε ένα δυαδικό δέντρο αναζήτησης. Οι NIL-κόμβοι έχουν παραλειφθεί. Οι ενδοδιατεταγμένες διασχίσεις του αρχικού και του μετασχηματισμένου δέντρου παράγουν την ίδια σειρά κλειδιών.

RIGHT-ROTATE εκτελείται σε $O(1)$ χρόνο. Κατά την περιστροφή, αλλάζουν οι τιμές δεικτών μόνον, ενώ τα υπόλοιπα πεδία των κόμβων παραμένουν ως έχουν.

Ασκήσεις

2-1
Σχεδιάστε το κοκκινόμαυρο δέντρο που προκύπτει μετά από την εκτέλεση της διαδικασίας Εισαγωγής κόμβου σε δυαδικό δέντρο αναζήτησης στο δέντρο του Σχήματος 1 για το κλειδί 36. Είναι το δέντρο αυτό κοκκινόμαυρο, εάν ο κόμβος που εισάγεται χρωματιστεί με κόκκινο χρώμα; Τι συμβαίνει εάν ο κόμβος χρωματιστεί με μαύρο χρώμα;

2-2
Γράψτε ψευδοκώδικα για τη διαδικασία δεξιάς περιστροφής RIGHT-ROTATE.

2-3
Δείξτε ότι η αριστερή και η δεξιά περιστροφή σε ένα δυαδικό δέντρο διατηρούν την ενδοδιατεταγμένη σειρά των κλειδιών.

2-4
Θεωρείστε το αριστερό δέντρο του Σχήματος 2 και έστω a, b και c τυχαίοι κόμβοι των υποδέντρων α, β και γ αντίστοιχα. Πώς αλλάζει το βάθος των a, b και c όταν εκτελεστεί μία αριστερή περιστροφή στον κόμβο x του σχήματος;

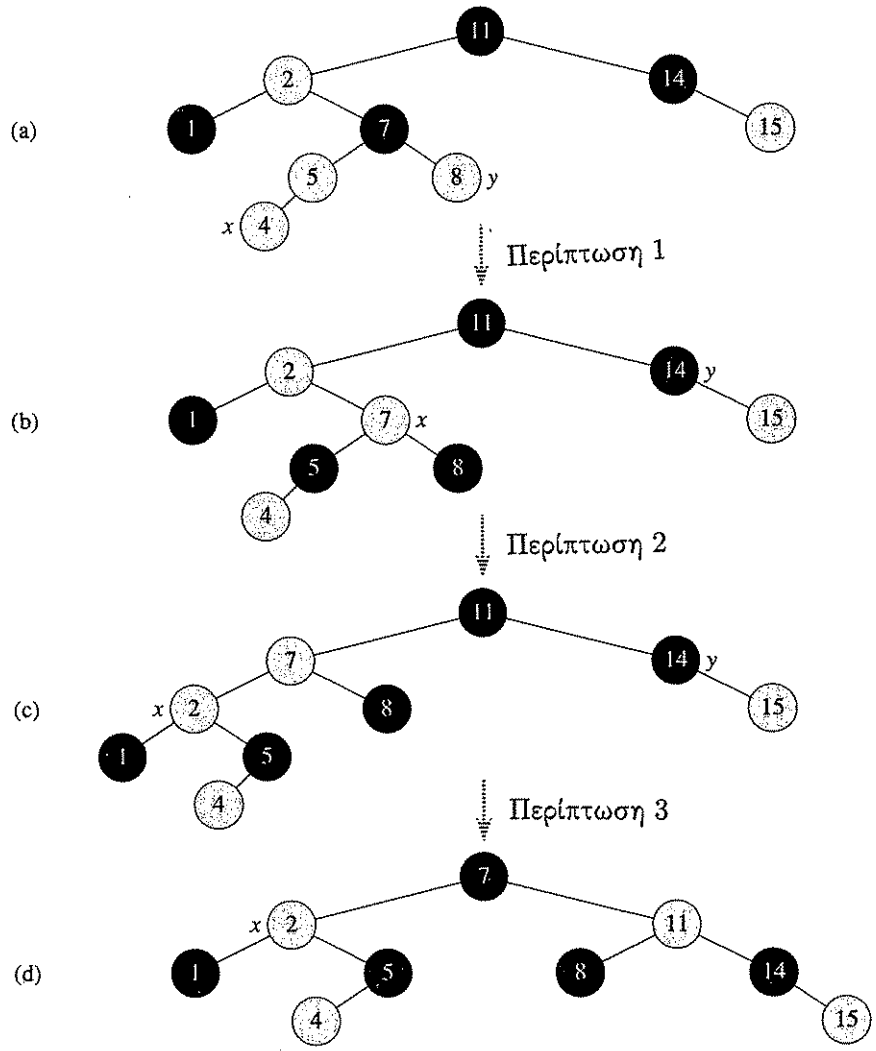
2-5
Δείξτε ότι ένα τυχαίο δέντρο με n κόμβους μπορεί να μετατραπεί σε ένα οποιοδήποτε άλλο δέντρο με n κόμβους χρησιμοποιώντας $O(n)$ περιστροφές. (*Υπόδειξη:* Δείξτε πρώτα ότι ένα οποιοδήποτε δέντρο μπορεί να μετασχηματιστεί σε μια "δεξιά αλυσίδα" από κόμβους εκτελώντας το πολύ $n-1$ περιστροφές.)

3 Εισαγωγή

Η εισαγωγή κόμβου σε ένα κοκκινόμαυρο δέντρο T με n κόμβους μπορεί να επιτευχθεί σε $O(\log n)$ χρόνο. Χρησιμοποιούμε τη διαδικασία εισαγωγής κόμβου σε δυαδικά δέντρα αναζήτησης (TREE-INSERT) για να εισάγουμε τον κόμβο x στο δέντρο T όπως εάν αυτό ήταν ένα απλό δυαδικό δέντρο αναζήτησης και μετά χρωματίζουμε τον x με κόκκινο χρώμα. Για να εξασφαλίσουμε ότι οι ΚΜ-Ιδιότητες εξακολουθούν να ισχύουν, τροποποιούμε το δέντρο που έχει προκύψει αλλάζοντας το χρώμα συγκεκριμένων κόμβων και εκτελώντας περιστροφές. Το μεγαλύτερο μέρος του κώδικα για τη διαδικασία εισαγωγής κόμβου RB-INSERT σε κοκκινόμαυρο δέντρο χειρίζεται τις διάφορες περιπτώσεις που μπορεί να παρουσιαστούν κατά την τροποποίηση του δέντρου.

```
RB-INSERT(T, x)
 1  TREE-INSERT(T, x)                          ▷ εισαγωγή σε δυαδικό δέντρο αναζήτησης
 2  color[x] ← κόκκινο
 3  while x ≠ root[T] and color[p[x]] = κόκκινο do
 4        if p[x] = left[p[p[x]]]
 5          then y ← right[p[p[x]]]
 6               if color[y] = κόκκινο
 7                 then color[p[x]] ← μαύρο                    ▷ Περίπτωση 1
 8                      color[y] ← μαύρο                       ▷ Περίπτωση 1
 9                      color[p[p[x]]] ← κόκκινο               ▷ Περίπτωση 1
10                      x ← p[p[x]]                            ▷ Περίπτωση 1
11                 else if x = right[p[x]]
12                        then x ← p[x]                       ▷ Περίπτωση 2
13                             LEFT-ROTATE(T, x)              ▷ Περίπτωση 2
14                      color[p[x]] ← μαύρο                   ▷ Περίπτωση 3
15                      color[p[p[x]]] ← κόκκινο              ▷ Περίπτωση 3
16                      RIGHT-ROTATE (T, p[p[x]])             ▷ Περίπτωση 3
17          else (όμοια με τον ψευδοκώδικα των γραμμών 5-16, όπου
                  τα πεδία "left" αντικαθίστανται από πεδία "right" και το αντίστροφο)
18  color[root[T]] ← μαύρο
```

Η διαδικασία RB-INSERT είναι λιγότερο περίπλοκη από όσο φαίνεται. Χωρίζουμε την εξέταση του ψευδοκώδικα σε τρία κύρια βήματα. Πρώτον, θα προσδιορίσουμε ποιες από τις ΚΜ-Ιδιότητες παραβιάζονται ως συνέπεια των γραμμών 1-2 του ψευδοκώδικα, στις οποίες εισάγουμε τον κόμβο x και τον χρωματίζουμε με κόκκινο χρώμα. Δεύτερον, θα περιγράψουμε τι προσπαθούμε να επιτύχουμε με τον βρόχο `while` (γραμμές 3-17). Τέλος, θα δούμε πως επιτυγχάνουμε αυτόν τον στόχο, αναλύοντας καθεμία από τις τρεις περιπτώσεις που συγκροτούν τον βρόχο `while`. Το Σχήμα 4 δίνει ένα παράδειγμα εκτέλεσης της διαδικασίας RB-INSERT σε ένα ενδεικτικό κοκκινόμαυρο δέντρο.



Σχήμα 4. Εκτέλεση της διαδικασίας RB-INSERT. **(α)** Ο κόμβος x μετά από την εισαγωγή του. Αφού τόσο ο x όσο και ο πατέρας $p[x]$ του x έχουν κόκκινο χρώμα, παρατηρείται παραβίαση της Ιδιότητας 3. Καθώς ο θείος y του x έχει επίσης κόκκινο χρώμα, εκτελείται ο κώδικας της Περίπτωσης 1. Κόμβοι επαναχρωματίζονται και ο κόμβος x μεταφέρεται σε υψηλότερο επίπεδο στο δέντρο, οπότε προκύπτει το δέντρο που εμφανίζεται στο **(b)**. Και πάλι, τόσο ο x όσο και ο πατέρας του έχουν κόκκινο χρώμα, αλλά αυτή τη φορά ο θείος y του x έχει μαύρο χρώμα. Αφού ο x είναι το δεξί παιδί του πατέρα του, εκτελείται ο κώδικας της Περίπτωσης 2. Εκτελείται δηλαδή μία αριστερή περιστροφή και το δέντρο που προκύπτει εμφανίζεται στο **(c)**. Αυτή τη φορά, ο x είναι το αριστερό παιδί του πατέρα του και εκτελείται ο κώδικας της Περίπτωσης 3. Μια δεξιά περιστροφή οδηγεί στο δέντρο του **(d)**, το οποίο είναι πλέον ένα αποδεκτό κοκκινόμαυρο δέντρο.

Ποιες από τις ΚΜ-Ιδιότητες μπορεί να παραβιάζονται μετά από την εκτέλεση των εντολών στις γραμμές 1-2; Η Ιδιότητα 1, όπως και η Ιδιότητα 2, εξακολουθούν προφανώς να ισχύουν, καθώς ο νεο-εισαχθείς κόκκινος κόμβος έχει για παιδιά NIL-κόμβους. Η Ιδιότητα 4, σύμφωνα με την οποία το πλήθος των μαύρων κόμβων είναι ο ίδιος σε κάθε διαδρομή από δοθέντα κόμβο επίσης ισχύει, καθώς ο κόμβος x αντικαθιστά έναν (μαύρο) NIL-κόμβο και είναι κόκκινος με παιδιά NIL-κόμβους. Τελικά, η μόνη ιδιότητα που πιθανόν να παραβιάζεται είναι η Ιδιότητα 3, σύμφωνα με την οποία ένας κόκκινος κόμβος δεν μπορεί να έχει κόκκινα παιδιά. Συγκεκριμένα, δεδομένου ότι ο x χρωματίζεται με κόκκινο χρώμα στη γραμμή 2, η Ιδιότητα 3 παραβιάζεται εάν ο πατέρας του κόμβου x έχει κόκκινο χρώμα. Το Σχήμα 4(α) δείχνει μια τέτοια παραβίαση μετά από την εισαγωγή του κόμβου x.

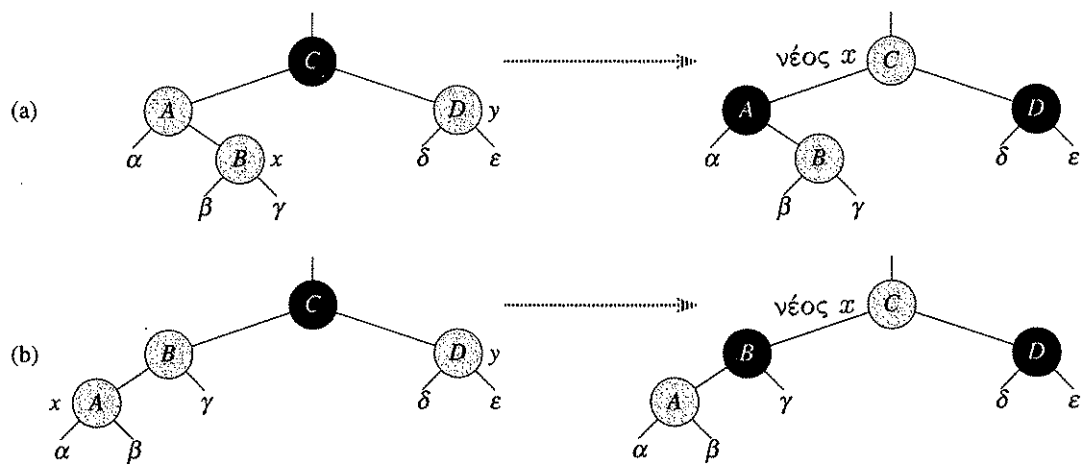
Ο σκοπός που εξυπηρετεί ο βρόχος `while` (γραμμές 3-17) είναι να μεταφέρει τη μοναδική παραβίαση της Ιδιότητας 3 σε υψηλότερο επίπεδο στο δέντρο διατηρώντας παράλληλα την Ιδιότητα 4 σε ισχύ. Στην αρχή κάθε επανάληψης του βρόχου, τόσο ο κόμβος x όσο και ο πατέρας του έχουν κόκκινο χρώμα –η μόνη παραβίαση στο δέντρο. Υπάρχουν δύο πιθανά ενδεχόμενα σε κάθε επανάληψη του βρόχου: η παραβίαση μεταφέρεται σε υψηλότερο επίπεδο στο δέντρο, ή εκτελούνται κάποιες περιστροφές και η εκτέλεση του βρόχου ολοκληρώνεται.

Υπάρχουν για την ακρίβεια έξι περιπτώσεις που πρέπει να εξετάσουμε στον βρόχο `while`· οι τρεις από αυτές είναι συμμετρικές με τις υπόλοιπες τρεις με μόνη διαφορά ότι ο πατέρας $p[x]$ του κόμβου x είναι δεξί αντί για αριστερό παιδί του παππού $p[p[x]]$ του x (γραμμή 4). Στον ψευδοκώδικα που παραθέσαμε εστιάσαμε την προσοχή μας στην περίπτωση που ο $p[x]$ είναι αριστερό παιδί. Επίσης, κάναμε τη σημαντική παραδοχή ότι η ρίζα του δέντρου έχει μαύρο χρώμα, γεγονός που εξασφαλίζεται στη γραμμή 18 στο τέλος της διαδικασίας RB-INSERT. Έτσι ο $p[x]$ δεν είναι η ρίζα του δέντρου και επομένως ο $p[p[x]]$ υπάρχει.

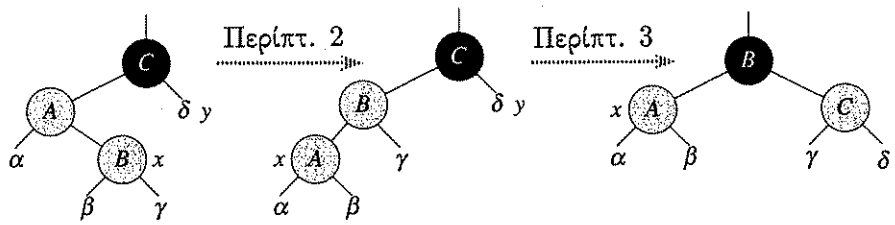
Η Περίπτωση 1 διαφέρει από τις Περιπτώσεις 2 και 3 ως προς το χρώμα του "θείου" του x, δηλαδή, του αδερφού του πατέρα του x ($right[p[p[x]]]$). Έστω y ο κόμβος αυτός (γραμμή 5). Εάν το χρώμα του y είναι κόκκινο, τότε εκτελείται ο κώδικας της Περίπτωσης 1. Αλλιώς, ο έλεγχος μεταβιβάζεται στις Περιπτώσεις 2 και 3. Και στις τρεις περιπτώσεις, ο παππούς $p[p[x]]$ του x έχει μαύρο χρώμα, καθώς ο πατέρας $p[x]$ του x έχει κόκκινο χρώμα και η Ιδιότητα 3 παραβιάζεται μόνο ανάμεσα στους κόμβους x και $p[x]$.

Τα βήματα του ψευδοκώδικα για την Περίπτωση 1 (γραμμές 7-10) απεικονίζονται στο Σχήμα 5. Στην Περίπτωση 1, τόσο ο πατέρας $p[x]$ όσο και ο θείος y του x έχουν κόκκινο χρώμα. Καθώς ο παππούς $p[p[x]]$ έχει μαύρο χρώμα, μπορούμε να χρωματίσουμε τους $p[x]$ και y με μαύρο χρώμα, διορθώνοντας έτσι το πρόβλημα που οφειλόταν στο γεγονός ότι οι x και $p[x]$ είχαν και οι δύο κόκκινο χρώμα. Επίσης, χρωματίζουμε τον $p[p[x]]$ με κόκκινο χρώμα, ώστε να διατηρήσουμε την Ιδιότητα 4. Το μόνο πρόβλημα που μπορεί να προκύψει τώρα είναι ότι ο πατέρας του, $p[p[x]]$, μπορεί να έχει κόκκινο χρώμα και άρα θα πρέπει να επαναλάβουμε τον βρόχο `while` για τον $p[p[x]]$ ως τον καινούργιο κόμβο x.

Στις Περιπτώσεις 2 και 3 (Σχήμα 6), το χρώμα του θείου y του κόμβου x είναι μαύρο. Οι δύο περιπτώσεις διαφέρουν ως προς το ότι ο x είναι δεξί ή αριστερό παιδί του πατέρα του. Στην Περίπτωση 2 (γραμμές 12-13), ο κόμβος x είναι δεξί παιδί: τότε εκτελούμε μία αριστερή περιστροφή ώστε να αναχθούμε στην Περίπτωση 3 (γραμμές 14-16), όπου ο x είναι αριστερό παιδί. Επειδή τόσο ο x όσο και ο πατέρας του έχουν κόκκινο χρώμα, η περιστροφή δεν επηρεάζει ούτε το μαύρο ύψος των κόμβων, ούτε την Ιδιότητα 4. Ανεξάρτητα από το εάν μεταβήκαμε στην Περίπτωση 3 κατευθείαν ή δια μέσου της Περίπτωσης 2, ο θείος y του x έχει μαύρο χρώμα· σε αντίθετη περίπτωση, θα μεταβαίναμε στην Περίπτωση 1. Εκτελούμε αλλαγές χρωμάτων σε συγκεκριμένους κόμβους (γραμμές 14-15) και μία δεξιά περιστροφή, η οποία διατηρεί την Ιδιότητα 4. Ως αποτέλεσμα, δεν υπάρχουν πλέον δύο κόκκινοι κόμβοι



Σχήμα 5. Η Περίπτωση 1 της διαδικασίας RB-INSERT. Η Ιδιότητα 3 παραβιάζεται, αφού τόσο ο x όσο και ο πατέρας του έχουν κόκκινο χρώμα. Τα ίδια βήματα εκτελούνται είτε (**a**) ο x είναι δεξί παιδί είτε (**b**) ο x είναι αριστερό παιδί. Καθένα από τα υποδέντρα α, β, γ, δ και ε έχει μαύρη ρίζα και το ίδιο μαύρο ύψος. Ο κώδικας για την Περίπτωση 1 αλλάζει το χρώμα κάποιων κόμβων διατηρώντας την Ιδιότητα 4: όλες οι διαδρομές από έναν κόμβο σε έναν NIL-απόγονο έχουν το ίδιο πλήθος μαύρων κόμβων. Ο βρόχος `while` ξαναεκτελείται, όπου x είναι τώρα ο παππούς $p[p[x]]$ του x. Τυχόν παραβίαση της Ιδιότητας 3 μπορεί πλέον να παρουσιασθεί μόνο ανάμεσα στον νέο κόμβο x, που έχει κόκκινο χρώμα, και στον πατέρα του, εάν έχει επίσης κόκκινο χρώμα.



Σχήμα 6. Οι Περιπτώσεις 2 και 3 της διαδικασίας RB-INSERT. Και στις δύο περιπτώσεις (όπως και στην Περίπτωση 1), η Ιδιότητα 3 παραβιάζεται, αφού ο x και ο πατέρας του έχουν και οι δύο κόκκινο χρώμα. Καθένα από τα υποδέντρα α, β, γ και δ έχει μαύρη ρίζα και το ίδιο μαύρο ύψος. Η Περίπτωση 2 ανάγεται στην Περίπτωση 3 με την εκτέλεση μίας αριστερής περιστροφής που διατηρεί την Ιδιότητα 4: όλες οι διαδρομές από έναν κόμβο σε έναν NIL-απόγονο περιέχουν το ίδιο πλήθος μαύρων κόμβων. Η Περίπτωση 3 περιλαμβάνει την αλλαγή χρώματος συγκεκριμένων κόμβων και μία δεξιά περιστροφή που επίσης διατηρεί την Ιδιότητα 4. Ο βρόχος `while` δεν ξαναεκτελείται, καθώς ικανοποιείται η Ιδιότητα 3: δεν υπάρχουν πλέον δύο κόκκινοι κόμβοι στη σειρά.

στη σειρά και η διαδικασία ολοκληρώνεται. Ο βρόχος `while` δεν ξαναεκτελείται καθώς το χρώμα του πατέρα του x είναι πλέον μαύρο.

Ποια είναι η πολυπλοκότητα χρόνου της διαδικασίας RB-INSERT; Καθώς το ύψος ενός κοκκινόμαυρου δέντρου με n κόμβους είναι $O(\log n)$, η κλήση της διαδικασίας TREE-INSERT απαιτεί $O(\log n)$ χρόνο. Ο βρόχος `while` επαναλαμβάνεται μόνο στην Περίπτωση 1 και σε κάθε επανάληψη ο κόμβος x μετακινείται σε υψηλότερο επίπεδο στο δέντρο. Το συνολικό πλήθος επαναλήψεων του βρόχου `while` είναι συνεπώς $O(\log n)$. Επομένως, η διαδικασία RB-INSERT απαιτεί συνολικά $O(\log n)$ χρόνο. Είναι ενδιαφέρον να παρατηρήσει κανείς ότι κατά την εισαγωγή ενός κόμβου εκτελούνται το πολύ δύο περιστροφές, καθώς ο βρόχος `while` δεν ξαναεκτελείται εάν προκύψουν οι Περιπτώσεις 2 και 3.

Ασκήσεις

3-1
Στη γραμμή 2 της διαδικασίας RB-INSERT, χρωματίζουμε με κόκκινο χρώμα τον κόμβο x που μόλις εισαγάγαμε. Παρατηρήστε ότι εάν τον είχαμε χρωματίσει με μαύρο χρώμα, η Ιδιότητα 3 των κοκκινόμαυρων δέντρων δεν θα παραβιαζόταν. Γιατί λοιπόν δεν επιλέξαμε να χρωματίσουμε τον x με μαύρο χρώμα;

3-2
Στη γραμμή 18 της διαδικασίας RB-INSERT, χρωματίζουμε τη ρίζα του δέντρου με μαύρο χρώμα. Ποιο είναι το πλεονέκτημα αυτής της ενέργειας;

3-3
Σχεδιάστε τα κοκκινόμαυρα δέντρα που προκύπτουν κατά τη διαδοχική εισαγωγή των κλειδιών 41, 38, 31, 12, 19 και 8 σε ένα αρχικά κενό κοκκινόμαυρο δέντρο.

3-4
Έστω ότι το μαύρο ύψος καθενός από τα υποδέντρα α, β, γ, δ και ε στα Σχήματα 5 και 6 είναι k. Τοποθετήστε σε κάθε κόμβο των δύο αυτών σχημάτων το μαύρο ύψος του ώστε να επιβεβαιώσετε ότι η Ιδιότητα 4 εξακολουθεί να ισχύει κατά τον εκτελούμενο μετασχηματισμό.

3-5
Θεωρείστε ένα κοκκινόμαυρο δέντρο που προκύπτει από την εισαγωγή n κόμβων με τη χρήση της διαδικασίας RB-INSERT. Αποδείξτε ότι εάν $n > 1$, τότε το δέντρο έχει τουλάχιστον έναν κόκκινο κόμβο.

3-6
Προτείνετε έναν αποτελεσματικό τρόπο υλοποίησης της διαδικασίας RB-INSERT για την περίπτωση που η παράσταση των κοκκινόμαυρων δέντρων δεν περιλαμβάνει πεδίο δείκτη προς τον πατέρα του κόμβου.

4. Διαγραφή

Όπως και οι άλλες βασικές διαδικασίες σε ένα κοκκινόμαυρο δέντρο με n κόμβους, η διαγραφή ενός κόμβου απαιτεί $O(\log n)$ χρόνο. Είναι όμως λίγο πιο περίπλοκη από τη διαδικασία της εισαγωγής κόμβου.

Για να απλοποιήσουμε τις συνθήκες ελέγχου στον ψευδοκώδικα, χρησιμοποιούμε έναν **κόμβο-φρουρό** *(sentinel)* $nil[T]$ για να αναπαραστήσουμε τους NIL-κόμβους ενός κοκκινόμαυρου δέντρου T. Ο κόμβος-φρουρός έχει τα ίδια ακριβώς πεδία όπως ένας συνηθισμένος κόμβος του T: το χρώμα του είναι μαύρο, ενώ οι τιμές των άλλων πεδίων p, $left$, $right$, και key λαμβάνουν αυθαίρετες τιμές. Η χρήση του κόμβου-φρουρού συνεπάγεται ότι όλοι οι δείκτες NIL του κοκκινόμαυρου δέντρου T αντικαθιστώνται από δείκτες στον κόμβο-φρουρό $nil[T]$.

Ο κόμβος-φρουρός μάς επιτρέπει να χειριζόμαστε τα NIL παιδιά κάποιου κόμβου x ως κανονικά παιδιά του x. Θα μπορούσαμε να προσθέσουμε έναν ξεχωριστό κόμβο-φρουρό στη θέση κάθε NIL-κόμβου στο δέντρο, ώστε ο πατέρας κάθε τέτοιου κόμβου να είναι καλά ορισμένος, αλλά αυτό απλά θα οδηγούσε σε σπατάλη μνήμης. Αντίθετα, χρησιμοποιούμε μόνον έναν κόμβο-φρουρό, τον $nil[T]$, για να αναπαραστήσουμε όλους τους NIL-κόμβους. Ωστόσο, όταν επιθυμούμε να χειριστούμε το NIL παιδί κάποιου κόμβου x, πρέπει να φροντίσουμε πρώτα να αναθέσουμε στο $p[nil[T]]$ την τιμή του x.

Η διαδικασία διαγραφής RB-DELETE από ένα κοκκινόμαυρο δέντρο είναι μια ελαφριά παραλλαγή της διαδικασίας TREE-DELETE που επιτυγχάνει τη διαγραφή κλειδιού από δυαδικό δέντρο αναζήτησης. Αφού αφαιρέσουμε τον αντίστοιχο κόμβο από το δέντρο, καλούμε μια βοηθητική διαδικασία RB-DELETE-FIXUP η οποία τροποποιεί το χρώμα κάποιων κόμβων και εκτελεί περιστροφές για να αποκαταστήσει τις ΚΜ-Ιδιότητες.

RB-DELETE(T, z)
1 if $left[z] = nil[T]$ or $right[z] = nil[T]$
2 then $y \leftarrow z$
3 else $y \leftarrow$ δείκτης στον κόμβο με το αμέσως μικρότερο κλειδί από αυτό του z
4 if $left[y] \neq nil[T]$
5 then $x \leftarrow left[y]$
6 else $x \leftarrow right[y]$
7 $p[x] \leftarrow p[y]$
8 if $p[y] = nil[T]$
9 then $root[T] \leftarrow x$
10 else if $y = left[p[y]]$
11 then $left[p[y]] \leftarrow x$
12 else $right[p[y]] \leftarrow x$
13 if $y \neq z$
14 then $key[z] \leftarrow key[y]$
15 ▷ Εάν ο y έχει και άλλα πεδία, αντίγραψέ τα επίσης.
16 if $color[y] =$ μαύρο
17 then RB-DELETE-FIXUP (T, x)
18 return y

Η RB-DELETE-FIXUP καλείται μόνον εάν ο y έχει μαύρο χρώμα. Εάν ο y έχει κόκκινο χρώμα, οι ΚΜ-Ιδιότητες παραμένουν σε ισχύ μετά από την αφαίρεση του y από το δέντρο, καθώς τα μαύρα ύψη των κόμβων δεν μεταβάλλονται και δεν υπάρχουν γειτονικοί κόκκινοι κόμβοι. Ο κόμβος x που παίρνει ως όρισμα η RB-DELETE-FIXUP είναι είτε το μοναδικό παιδί του y πριν την αφαίρεση του y στην περίπτωση που ο y είχε ένα παιδί που δεν ήταν NIL, είτε ο κόμβος-φρουρός $nil[T]$ στην περίπτωση που και τα δύο παιδιά του y ήταν NIL. Στη δεύτερη περίπτωση, η ανάθεση στη γραμμή 7 εξασφαλίζει ότι ο πατέρας του x είναι τώρα ο κόμβος που ήταν προηγουμένως ο πατέρας του y, ανεξάρτητα από το εάν ο x ήταν κανονικός εσωτερικός κόμβος του δέντρου ή ο κόμβος-φρουρός $nil[T]$.

Ας εξετάσουμε τώρα πως η διαδικασία RB-DELETE-FIXUP αποκαθιστά τις ΚΜ-Ιδιότητες στο δέντρο που προκύπτει μετά από την αφαίρεση του y.

```
RB-DELETE-FIXUP(T, x)
 1  while x ≠ root[T] and color[x] = μαύρο do
 2        if x = left[p[x]]
 3          then w ← right[p[x]]
 4               if color[w] = κόκκινο
 5                 then color[w] ← μαύρο                          ▷ Περίπτωση 1
 6                      color[p[x]] ← κόκκινο                     ▷ Περίπτωση 1
 7                      LEFT-ROTATE(T, p[x])                      ▷ Περίπτωση 1
 8                      w ← right[p[x]]                           ▷ Περίπτωση 1
 9               if color[left[w]] = μαύρο and color[right[w]] = μαύρο
10                 then color[w] ← κόκκινο                        ▷ Περίπτωση 2
11                      x ← p[x]                                  ▷ Περίπτωση 2
12                 else if color[right[w]] = μαύρο
13                        then color[left[w]] ← μαύρο             ▷ Περίπτωση 3
14                             color[w] ← κόκκινο                 ▷ Περίπτωση 3
15                             RIGHT-ROTATE(T, w)                 ▷ Περίπτωση 3
16                             w ← right[p[x]]                    ▷ Περίπτωση 3
17                      color[w] ← color[p[x]]                    ▷ Περίπτωση 4
18                      color[p[x]] ← μαύρο                       ▷ Περίπτωση 4
19                      color[right[w]] ← μαύρο                   ▷ Περίπτωση 4
20                      LEFT-ROTATE(T, p[x])                      ▷ Περίπτωση 4
21                      x ← root[T]                               ▷ Περίπτωση 4
22          else (όμοια με τον ψευδοκώδικα των γραμμών 3-21, όπου
                  τα πεδία "left" αντικαθίστανται από πεδία "right" και το αντίστροφο)
23  color[x] ← μαύρο
```

Εάν ο κόμβος y (στην RB-DELETE) ο οποίος αφαιρείται έχει μαύρο χρώμα, η αφαίρεσή του από το δέντρο κάνει κάθε διαδρομή που προηγούμενα περιλάμβανε τον κόμβο y να περιέχει τώρα έναν μαύρο κόμβο λιγότερο. Συνεπώς, η Ιδιότητα 4 πλέον παραβιάζεται από όλους τους προγόνους του y στο δέντρο. Μπορούμε να διορθώσουμε αυτό το πρόβλημα θεωρώντας ότι ο κόμβος x έχει μία επιπλέον "μονάδα μαύρου χρώματος." Δηλαδή, η Ιδιότητα 4 ισχύει, αν θεωρήσουμε ότι οποιαδήποτε απλή διαδρομή που περιλαμβάνει τον x περιέχει έναν επιπλέον μαύρο κόμβο. Όταν αφαιρούμε τον μαύρο κόμβο y, αναθέτουμε τη μονάδα μαύρου χρώματος που έχει στο παιδί του. Το μόνο πρόβλημα είναι ότι τώρα ο κόμβος x μπορεί να είναι "διπλά μαύρος" παραβιάζοντας έτσι την Ιδιότητα 1.

Η διαδικασία RB-DELETE-FIXUP προσπαθεί να αποκαταστήσει την ισχύ της Ιδιότητας 1. Με τον βρόχο `while` στις γραμμές 1-22 φροντίζουμε να μετακινήσουμε την επιπλέον "μονάδα μαύρου χρώματος" ψηλότερα στο δέντρο μέχρις ότου

α) είτε ο x έχει κόκκινο χρώμα, οπότε τον χρωματίζουμε με μαύρο χρώμα (γραμμή 23),

β) είτε ο x είναι η ρίζα του δέντρου, οπότε η επιπλέον μονάδα μαύρου χρώματος μπορεί απλά να αγνοηθεί,

γ) ή τέλος μπορούμε να εκτελέσουμε κατάλληλους επαναχρωματισμούς και περιστροφές.

Σημειώστε ότι σε κάθε εκτέλεση του βρόχου `while`, ο x είναι ένας μαύρος κόμβος που δεν είναι η ρίζα του δέντρου και φέρει μία επιπλέον μονάδα μαύρου χρώματος. Στη γραμμή 2 ελέγχουμε εάν ο x είναι αριστερό ή δεξί παιδί του πατέρα του ($p[x]$). (Έχουμε παραθέσει

τον κώδικα για την περίπτωση στην οποία ο x είναι αριστερό παιδί· η περίπτωση όπου ο x είναι δεξί παιδί –γραμμή 22– είναι συμμετρική.) Έστω w ο αδελφός του κόμβου x. Καθώς ο κόμβος x είναι "διπλά μαύρος", ο κόμβος w δεν μπορεί να είναι $nil[T]$, αλλιώς το πλήθος των μαύρων κόμβων στη διαδρομή από τον $p[x]$ στον NIL-κόμβο w θα ήταν μικρότερο από το αντίστοιχο πλήθος στη διαδρομή από τον $p[x]$ στον x.

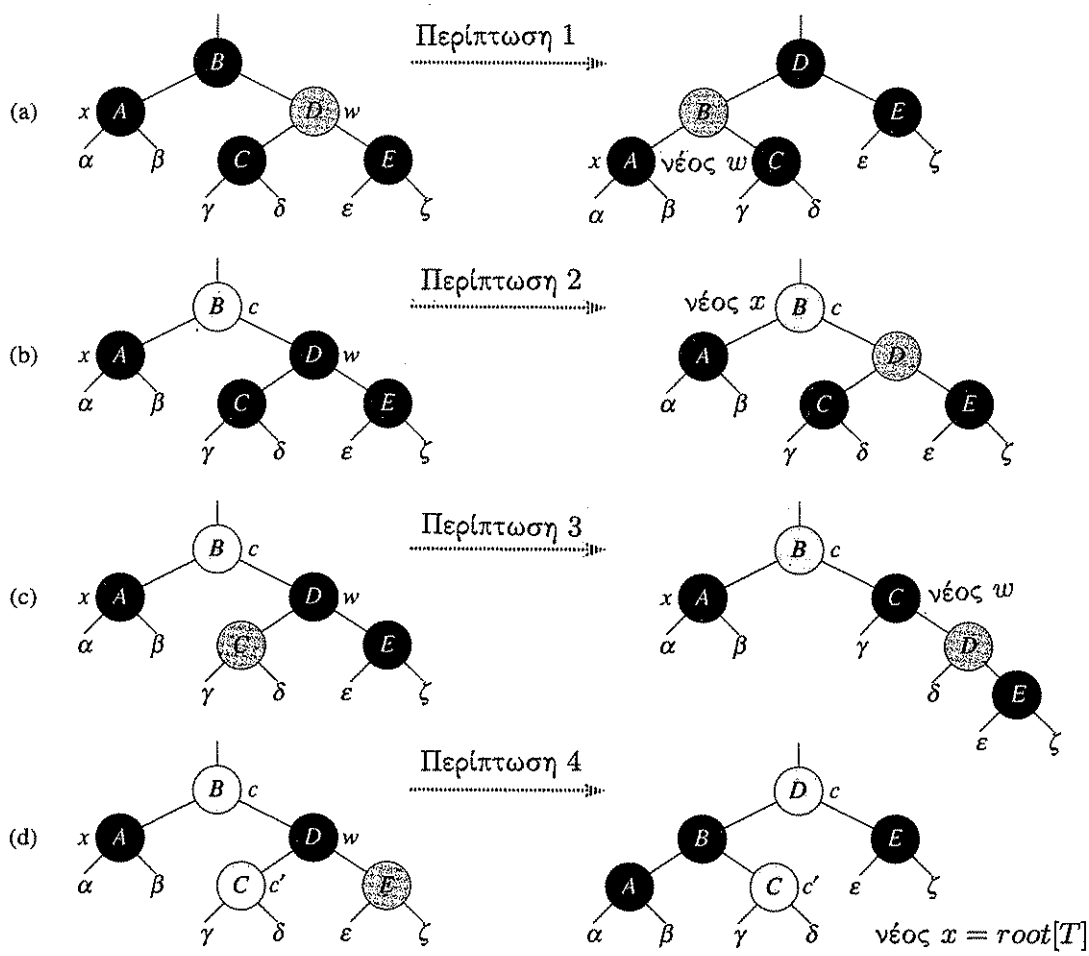
Οι τέσσερις περιπτώσεις στον κώδικα απεικονίζονται στο Σχήμα 7. Πριν να εξετάσουμε την κάθε περίπτωση με λεπτομέρεια, ας δούμε πιο γενικά πως μπορούμε να επιβεβαιώσουμε ότι ο μετασχηματισμός διατηρεί την Ιδιότητα 4 σε κάθε περίπτωση. Η βασική ιδέα είναι ότι σε κάθε περίπτωση το πλήθος των μαύρων κόμβων από (και συμπεριλαμβανομένης) της ρίζας του υποδέντρου που απεικονίζεται μέχρι καθένα από τα υποδέντρα α, β, ..., ζ διατηρείται από τον μετασχηματισμό. Για παράδειγμα, στο Σχήμα 7(a), που απεικονίζει την Περίπτωση 1, το πλήθος των μαύρων κόμβων από τη ρίζα στο υποδέντρο α όπως και στο β είναι 3 τόσο πριν όσο και μετά από τον μετασχηματισμό. (Υπενθυμίζεται ότι ο κόμβος x φέρει μία επιπλέον μονάδα μαύρου χρώματος.) Όμοια, το πλήθος των μαύρων κόμβων από τη ρίζα σε καθένα από τα γ, δ, ε και ζ είναι 2 τόσο πριν όσο και μετά από τον μετασχηματισμό. Στο Σχήμα 7(b), ο αντίστοιχος υπολογισμός πρέπει να λάβει υπόψη του το χρώμα c, το οποίο μπορεί να είναι είτε κόκκινο είτε μαύρο. Εάν ορίσουμε τις τιμές $count(\text{κόκκινο}) = 0$ και $count(\text{μαύρο}) = 1$, τότε το πλήθος των μαύρων κόμβων από τη ρίζα στο α είναι $2 + count(c)$ τόσο πριν όσο και μετά από τον μετασχηματισμό. Οι υπόλοιπες περιπτώσεις μπορούν να επιβεβαιωθούν με παρόμοιο τρόπο (Άσκηση 4-5).

Η Περίπτωση 1 (γραμμές 5-8 της RB-DELETE-FIXUP και Σχήμα 7(a)) προκύπτει όταν ο κόμβος w, ο αδελφός του κόμβου x, έχει κόκκινο χρώμα. Καθώς ο w πρέπει τότε να έχει μαύρα παιδιά, μπορούμε να εναλλάξουμε τα χρώματα των w και $p[x]$ και να εκτελέσουμε μία αριστερή περιστροφή στον $p[x]$ χωρίς να παραβιάσουμε καμία από τις ΚΜ-Ιδιότητες. Ο νέος αδελφός του x, ένα από τα παιδιά του w, έχει τώρα μαύρο χρώμα, και έχουμε συνεπώς μετατρέψει την Περίπτωση 1 σε μία από τις Περιπτώσεις 2, 3, ή 4.

Οι Περιπτώσεις 2, 3, και 4 προκύπτουν όταν ο κόμβος w έχει μαύρο χρώμα· οι περιπτώσεις διαφέρουν ως προς τα χρώματα των παιδιών του w. Στην Περίπτωση 2 (γραμμές 10-11 της RB-DELETE-FIXUP και Σχήμα 7(b)), και τα δύο παιδιά του w έχουν μαύρο χρώμα. Καθώς ο w έχει μαύρο χρώμα επίσης, αφαιρούμε μία μονάδα μαύρου χρώματος τόσο από τον x όσο και από τον w, αφήνοντας τον x με μόνο μία μονάδα μαύρου χρώματος και χρωματίζοντας τον w κόκκινο, και προσθέτουμε μία επιπλέον μονάδα μαύρου χρώματος στον $p[x]$. Κατόπιν επαναλαμβάνουμε τον βρόχο `while` με τον $p[x]$ ως τον καινούργιο κόμβο x. Παρατηρήστε ότι εαν καταλήξουμε στην Περίπτωση 2 μέσω της Περίπτωσης 1, το χρώμα c του νέου κόμβου x είναι κόκκινο, καθώς ο αρχικός $p[x]$ είχε κόκκινο χρώμα· συνεπώς ο βρόχος δεν ξαναεκτελείται.

Η Περίπτωση 3 (γραμμές 13-16 και Σχήμα 7(c)) προκύπτει όταν ο w και το δεξί παιδί του έχουν μαύρο χρώμα, ενώ το αριστερό παιδί του w έχει κόκκινο χρώμα. Μπορούμε να εναλλάξουμε τα χρώματα των w και του αριστερού του παιδιού, $left[w]$, και κατόπιν να εκτελέσουμε μία δεξιά περιστροφή στον w χωρίς να παραβιάσουμε καμία από τις ΚΜ-Ιδιότητες. Ο νέος αδελφός w του x είναι τώρα ένας μαύρος κόμβος με ένα κόκκινο δεξί παιδί και συνεπώς έχουμε μετασχηματίσει την Περίπτωση 3 στην Περίπτωση 4.

Η Περίπτωση 4 (γραμμές 17-21 και Σχήμα 7(d)) προκύπτει όταν ο αδελφός w του κόμβου x έχει μαύρο χρώμα και το δεξί παιδί του w έχει κόκκινο χρώμα. Κάνοντας κάποιες αλλαγές χρωμάτων και εκτελώντας μία αριστερή περιστροφή στον $p[x]$, μπορούμε να αφαιρέσουμε την επιπλέον μονάδα μαύρου χρώματος από τον x χωρίς να παραβιάσουμε καμία



Σχήμα 7. Οι περιπτώσεις του βρόχου `while` της διαδικασίας RB-DELETE-FIXUP. Οι κόμβοι που απεικονίζονται με μαύρο χρώμα είναι μαύροι κόμβοι του κοκκινόμαυρου δέντρου, οι ελαφρά σκιασμένοι κόμβοι είναι κόκκινοι κόμβοι του δέντρου, ενώ οι λευκοί κόμβοι μπορεί να είναι είτε μαύροι είτε κόκκινοι και σημειώνονται με c ή c'. Τα γράμματα α, β, ..., ζ αναπαριστούν τυχόντα υποδέντρα. Σε κάθε περίπτωση, ο συνδυασμός κόμβων στα αριστερά μετασχηματίζεται στον συνδυασμό κόμβων στα δεξιά με αλλαγή του χρώματος κάποιων κόμβων και/ή μία περιστροφή. Κάθε κόμβος x φέρει μία επιπλέον "μονάδα μαύρου χρώματος." Η μόνη περίπτωση που οδηγεί σε επανάληψη του βρόχου είναι η Περίπτωση 2. (**a**) Η Περίπτωση 1 μετασχηματίζεται στις Περιπτώσεις 2, 3 ή 4 με εναλλαγή των χρωμάτων των κόμβων B και D και με εκτέλεση μίας αριστερής περιστροφής. (**b**) Στην Περίπτωση 2, μεταφέρουμε την επιπλέον "μονάδα μαύρου χρώματος" (την οποία φέρει ο κόμβος x) ψηλότερα στο δέντρο χρωματίζοντας τον κόμβο D κόκκινο και κάνοντας τον κόμβο x να δείχνει στον κόμβο B. Εάν αναχθούμε στην Περίπτωση 2 μέσω της Περίπτωσης 1, ο βρόχος `while` δεν ξαναεκτελείται αφού το χρώμα c είναι κόκκινο. (**c**) Η Περίπτωση 3 μετασχηματίζεται στην Περίπτωση 4 με εναλλαγή των χρωμάτων των κόμβων C και D και την εκτέλεση μίας δεξιάς περιστροφής. (**d**) Στην Περίπτωση 4, η επιπλέον "μονάδα μαύρου χρώματος", την οποία φέρει ο κόμβος x, μπορεί να αφαιρεθεί με αλλαγές κάποιων χρωμάτων και την εκτέλεση μίας αριστερής περιστροφής (χωρίς να παραβιασθεί καμία από τις ΚΜ-Ιδιότητες), και ο βρόχος δεν ξαναεκτελείται.

από τις ΚΜ-Ιδιότητες. Η ανάθεση $x \leftarrow root[T]$ οδηγεί στον τερματισμό της εκτέλεσης του βρόχου `while` στον επόμενο έλεγχο της συνθήκης του βρόχου.

Ποια είναι η πολυπλοκότητα χρόνου της RB-DELETE; Καθώς το ύψος ενός κοκκινόμαυρου δέντρου με n κόμβους είναι $O(\log n)$, ο συνολικός χρόνος που απαιτεί η διαδικασία χωρίς την κλήση της RB-DELETE-FIXUP είναι $O(\log n)$. Στην RB-DELETE-FIXUP, σε καθεμία από τις Περιπτώσεις 1, 3, και 4 εκτελούμε ένα σταθερό πλήθος αλλαγών χρώματος και το πολύ τρεις περιστροφές και ο βρόχος δεν ξαναεκτελείται. Η Περίπτωση 2 είναι η μόνη περίπτωση που οδηγεί σε επανάληψη του βρόχου `while`, και κάθε φορά ο x μετακινείται σε κόμβο ψηλότερα στο δέντρο (δηλαδή, το πολύ $O(\log n)$ φορές), ενώ δεν εκτελείται καμία περιστροφή. Συνεπώς, η διαδικασία RB-DELETE-FIXUP απαιτεί $O(\log n)$ χρόνο και εκτελεί το πολύ τρεις περιστροφές. Επομένως, ο συνολικός χρόνος για την RB-DELETE είναι επίσης $O(\log n)$.

Ασκήσεις

4-1.
Δείξτε ότι η ρίζα ενός κοκκινόμαυρου δέντρου έχει πάντοτε μαύρο χρώμα μετά από την εκτέλεση της RB-DELETE.

4-2.
Στην άσκηση 3-3 βρήκατε το κοκκινόμαυρο δέντρο που προκύπτει από τη διαδοχική εισαγωγή των κλειδιών 41, 38, 31, 12, 19, 8 σε ένα αρχικά κενό δέντρο. Τώρα σχεδιάστε τα κοκκινόμαυρα δέντρα που προκύπτουν μετά από τις διαδοχικές διαγραφές των κλειδιών 8, 12, 19, 31, 38, 41 κατά σειρά.

4-3.
Σε ποιες γραμμές του ψευδοκώδικα της RB-DELETE-FIXUP είναι πιθανόν να εξετάσουμε ή να αλλάξουμε τιμές στα πεδία του κόμβου-φρουρού $nil[T]$;

4-4.
Απλοποιήστε τον ψευδοκώδικα για την LEFT-ROTATE χρησιμοποιώντας έναν κόμβο-φρουρό για τους NIL-κόμβους και έναν άλλο κόμβο-φρουρό που αποθηκεύει έναν δείκτη στη ρίζα του δέντρου.

4-5.
Σε καθεμία από τις περιπτώσεις του Σχήματος 7, βρήτε το πλήθος των μαύρων κόμβων στις διαδρομές από τη ρίζα του εικονιζόμενου υποδέντρου σε καθένα από τα υποδέντρα α, β, ..., ζ, και επιβεβαιώστε ότι το πλήθος αυτό δεν αλλάζει μετά από κάθε μετασχηματισμό. Σε περίπτωση που ένας κόμβος σημειώνεται με c ή c', χρησιμοποιήστε συμβολικά τις ποσότητες $count(c)$ και $count(c')$ στους υπολογισμούς σας.

4-6.
Έστω ότι ο κόμβος x εισάγεται σε ένα κοκκινόμαυρο δέντρο με την RB-INSERT και αμέσως μετά διαγράφεται με την RB-DELETE. Είναι το κοκκινόμαυρο δέντρο που προκύπτει τελικά ίδιο με το αρχικό κοκκινόμαυρο δέντρο; Δικαιολογήστε την απάντησή σας.