Λειτουργίες συνόλων βάσει Δυαδικών Δέντρων Αναζήτησης.

Λειτουργία join

- Δίνονται δύο δυαδικά δέντρα αναζήτησης T_1 και T_2 , RB ή AVL και τα δύο, και ένας αριθμός k.
- Ισχύει:
  - όλα τα στοιχεία του T_1 είναι μικρότερα του k και όλα τα στοιχεία του T_2 μεγαλύτερα του T_2
  - Η λειτουργία join(T_1,k,T_2) επιστρέφει ένα δέντρο T που περιέχει το k και όλα τα στοιχεία των T_1 και T_2 αντίστοιχα. Το T θα πρέπει να είναι RB ή AVL αντίστοιχα.

Λειτουργία join – RB δέντρα

- **T_1 και T_2 είναι RB**
- Έστω h_1 και h_2 το «μαύρο» ύψος των δέντρων T_1 και T_2 δηλ. το πλήθος των μαύρων κόμβων που συναντάμε από τη ρίζα στους εξωτερικούς κόμβους.

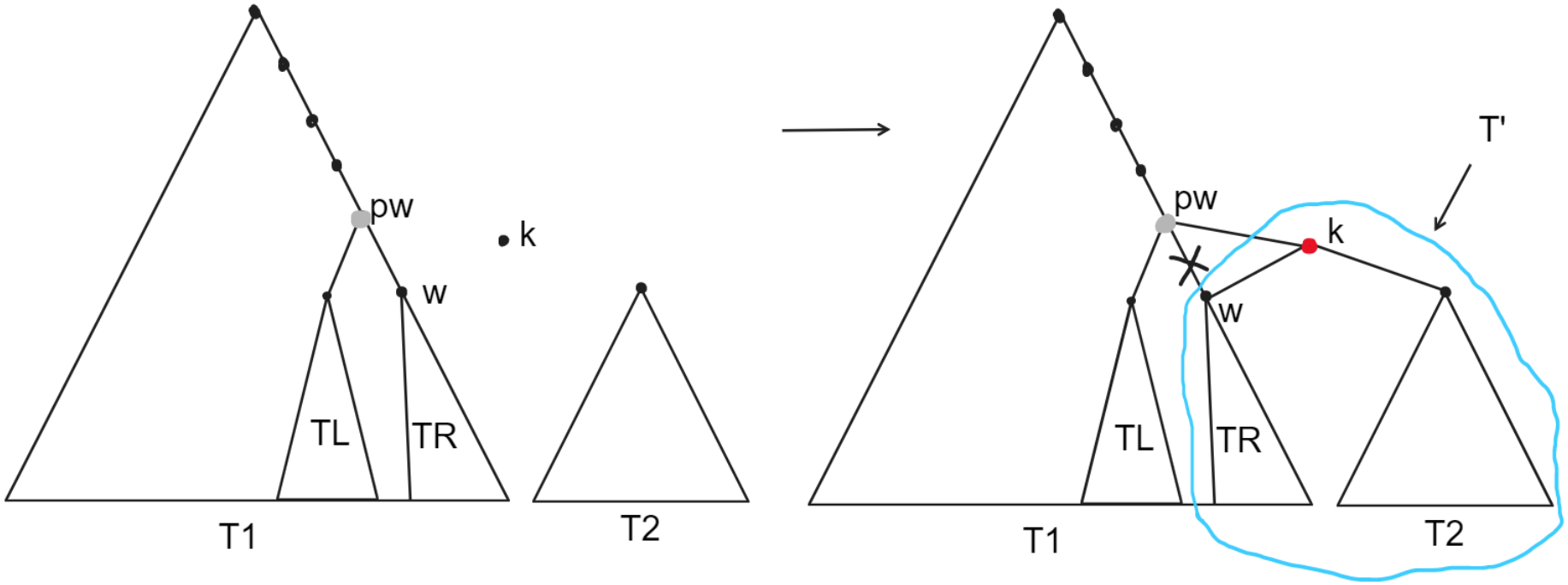
$1^η$ περίπτωση:

- Αν $h_1=h_2$ τότε η λειτουργία join γίνεται εύκολα, θέτοντας ως ρίζα του T το k και ως αριστερό και δεξιό υποδέντρο του T τα δέντρα T_1 και T_2 αντίστοιχα.
- Έστω το μαύρο ύψος h_1 είναι μεγαλύτερο του h_2. Συμμετρικά η άλλη περίπτωση.

Λειτουργία join – RB δέντρα

- Έστω το μαύρο ύψος h_1 είναι μεγαλύτερο του h_2. Συμμετρικά η άλλη περίπτωση.
- Κατεβαίνουμε συνεχώς το πιο δεξιό μονοπάτι του T_1 μέχρι να εντοπίσουμε τον κόμβο w που έχει μαύρο ύψος ίσο με h_2.
- Έστω pw ο γονιός του w στο δέντρο T_1.
- Δημιουργούμε ένα νέο δέντρο T’ που έχει ως ρίζα το στοιχείο k, αριστερό υποδέντρο, το υποδέντρο με ρίζα τον κόμβο w, και δεξί υποδέντρο το δέντρο T_2.
- Θέτουμε το χρώμα του κόμβου k κόκκινο.
- Τοποθετούμε το δέντρο T’ ως δεξί υποδέντρο του κόμβου pw στο δέντρο T.
- Αν ο κόμβος pw είναι κόκκινος: δύο γειτονικοί κόμβοι μετά τη προσάρτηση του υποδέντρου T’ στο δέντρο T_1.
- Το πρόβλημα επιλύεται με τις επαναχρωματισμό κόμβων ή περιστροφές ακριβώς όπως και στην απλή εισαγωγή σε ένα RB δέντρο.
- Συνολική πολυπλοκότητα: Θ(ύψος T_1- ύψος T_2)

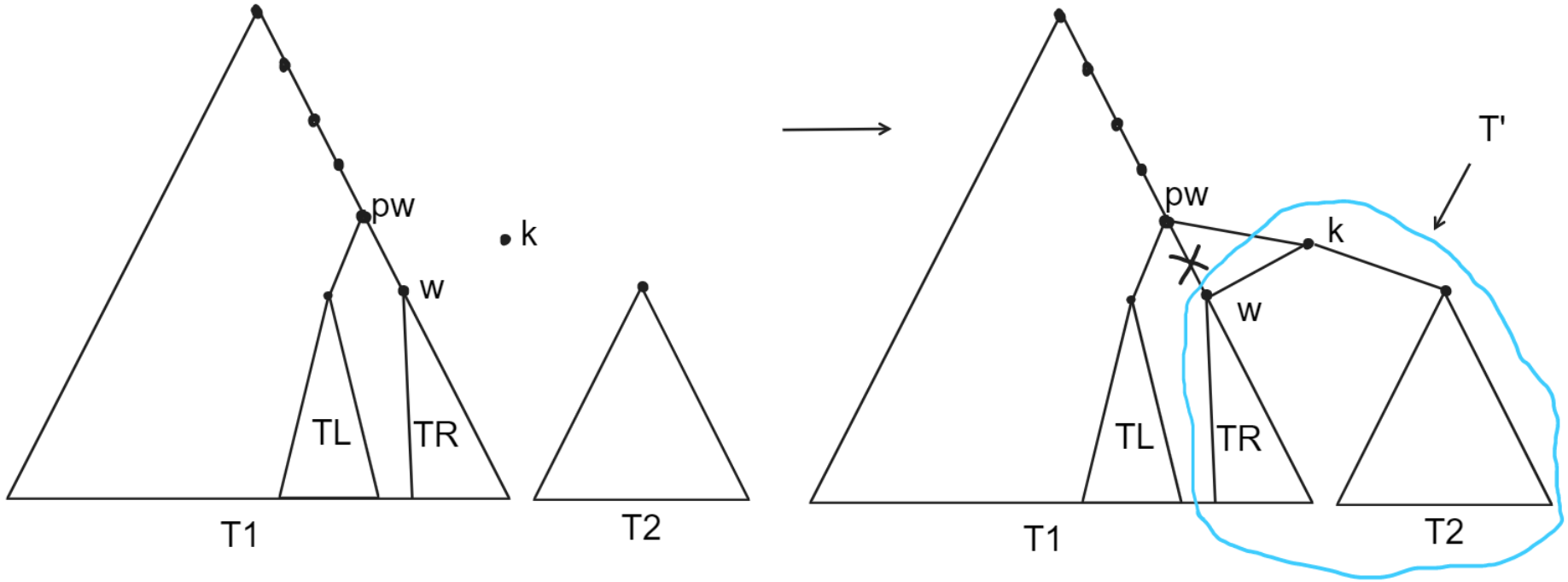
Λειτουργία join – RB δέντρα



Λειτουργία join – AVL δέντρα

- **T_1 και T_2 είναι AVL**
- Έστω τα ύψη των T_1 και T_2 είναι αντίστοιχα h_1 και h_2 αντίστοιχα.
- Αν $|h_1-h_2|\leq 1$ τότε εύκολη η συνένωση των δύο δέντρων:
  - Το στοιχείο k είναι ρίζα του δέντρου και τα T_1 και T_2 το αριστερό και δεξιό υποδέντρο του k αντίστοιχα. Το νέο δέντρο είναι AVL.
- Αν $|h_1-h_2|> 1$ τότε κατεβαίνουμε το πιο δεξιό μονοπάτι μέχρι να βρούμε τον πρώτο κόμβο w για τον $h(w) \leq h(T_2)+1$.
- Επομένως $h(pw)= h_2+2$ ή $h(pw)= h_2+3$
- Έστω T_L και T_R το αριστερό και δεξιό υποδέντρο αντίστοιχα του κόμβου pw ενώ h_L και h_R τα ύψη τους.

Λειτουργία join – AVL δέντρα



Λειτουργία join – AVL δέντρα

- Δημιουργείται ένα νέο δέντρο T’, θέτοντας ως ρίζα τον κόμβο k αριστερό δέντρο το δέντρο T_R και δεξιό υποδέντρο το δέντρο T_2.
- Το δέντρο T’ είναι AVL δέντρο.
- Το δέντρο T’ τίθεται ως δεξιό υποδέντρο του κόμβου pw.
- Προκύπτουν οι εξής περιπτώσεις όσον αφορά το παλαιό και νέο ύψος του w και των υποδέντρων του.

	h_{pw}	h_L	h_R	h_{pw}^{new}	ΔI_{pw}	AVL;	Ενέργεια
1	h_2+2	h_2+1	h_2	h_2+2	0	Ναι	
2	h_2+2	h_2	h_2+1	h_2+3	-2	Όχι	Διπλή περιστροφή
3	h_2+2	h_2+1	h_2+1	h_2+3	-1	Πιθανώς όχι	Μονή περιστροφή
4	h_2+3	h_2+2	h_2+1	h_2+3	0	Ναι	

Λειτουργία join – AVL δέντρα

- Ενδέχεται να υπάρχει παραβίαση της βασικής ιδιότητας των AVL δέντρων στο δέντρο που έχει δημιουργηθεί από την προσάρτηση του δέντρου Τ’ στο δέντρο T_1.
- Συγκεκριμένα, στη δεύτερη περίπτωση, πρόβλημα στον κόμβο pw: επίλυση με περιστροφή RL.
- Στην τρίτη περίπτωση, μπορεί να προκύψει πρόβλημα στους προγόνους: επίλυση με περιστροφή RR.

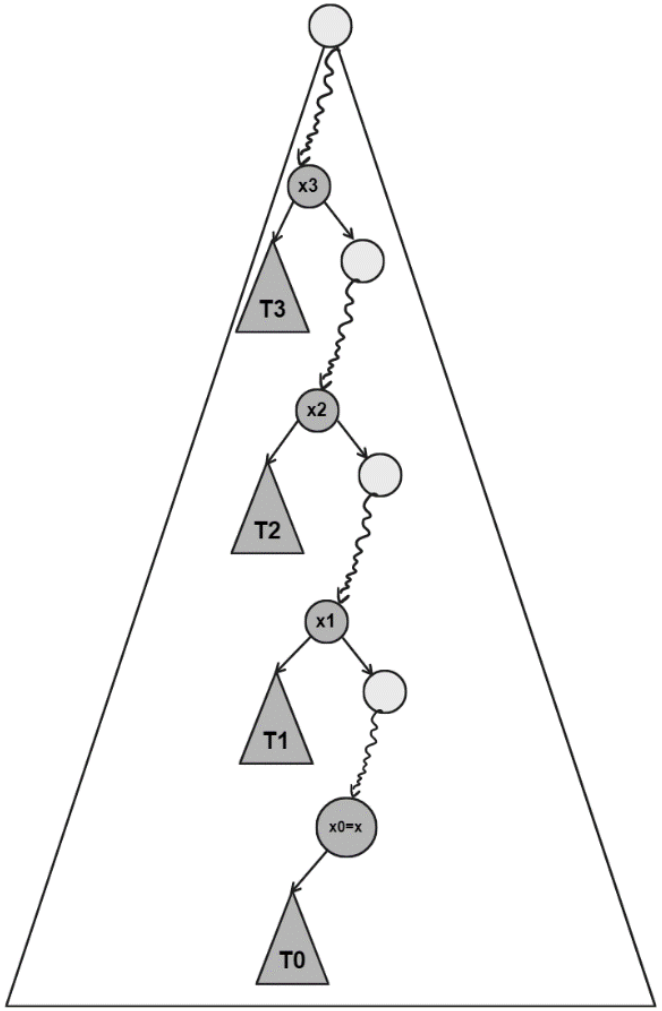
Λειτουργία join

- Η λειτουργία Join(T_1,k,T_2) απαιτεί την ύπαρξη του στοιχείου k.
- Πως εκτελείται η λειτουργία Join(T_1,T_2);
- Βρίσκουμε το μεγαλύτερο στοιχείο του T_1 και το αφαιρούμε από το T_1: O(log(ύψος του T_1)) χρόνος.
- Έστω k’ το στοιχείο αυτό και T'_1 το δέντρο T_1 μετά τη διαγραφή.
- Στη συνέχεια, Join(T'_1,k’,T_2).

Διαμελισμός δέντρου (Split)

- Δίνεται ένα δέντρο T AVL ή RB και ένα στοιχείο του δέντρου x.
- Η λειτουργία Split(x,T) διαιρεί το δέντρο σε δύο υποδέντρα $T^{<}$ και $T^{>}$ τέτοια ώστε όλα τα στοιχεία του $T^{<}$ είναι μικρότερα του x και όλα τα στοιχεία του $T^{>}$ είναι μεγαλύτερα του x.
- Ακολουθούμε το μονοπάτι από τη ρίζα στο στοιχείο x:
  - Αν σε ένα κόμβο συνεχίζουμε στο δεξιό υποδέντρο τότε το στοιχείο αυτού του κόμβου καθώς και το αριστερό υποδέντρο του κόμβου ανήκει σίγουρα στο δέντρο $T^{<}$.
  - Αν σε ένα κόμβο συνεχίζουμε στο αριστερό υποδέντρο τότε το στοιχείο αυτού του κόμβου καθώς και το δεξιό υποδέντρο του κόμβου ανήκει σίγουρα στο δέντρο $T^{>}$.
- Στη συνέχεια, όλα τα στοιχεία του μονοπατιού προς το x και των δέντρων που ανήκουν στο $T^{<}$ ενώνονται με τη λειτουργία join.
- Το ίδιο εφαρμόζεται για το δέντρο $T^{>}$.

Διαμελισμός δέντρου (Split)



- $T^{<} = Join(T_3,x_3,Join(T_2,x_2,Join(T_1,x_1,T_0)))$
- Αναδρομικός ορισμός της λειτουργίας Split:

```
Split(x,T) {
If x > T.root then
        (T<_R, T>_R)=Split(x, T_R)
        Join(T_L, T.root, T<_R)
Else if x < T.root then
        (T<_L, T>_L)=Split(x, T_L)
        Join(T>_L, T.root, T_R)
Else
        Return (T_L,T_R)
}
```

- Πολυπλοκότητα λειτουργίας Split:
  - Το κόστος της λειτουργίας Join μεταξύ του δέντρου T_i, του στοιχείου x_i, και του αποτελέσματος των μέχρι τώρα ενώσεων δέντρων είναι O(ύψος του x_i - ύψος του x_{i-1})
  - Έτσι το συνολικό κόστος όλων των ενώσεων είναι O(ύψος x_3 - ύψος του x_0)=O(log N)

Διαμελισμός δέντρου (Split)

- Τι γίνεται όταν το στοιχείο x δεν είναι στο δέντρο T όταν εφαρμόζουμε την Split(x,T);
- Θεωρούμε προσωρινά ότι το x έχει εισαχθεί στο δέντρο (ως φύλλο) και μετά ακολουθούμε την ίδια λογική.

Ένωση δύο δέντρων

- Η λειτουργία Union(T_1,T_2) δέχεται δύο δέντρα T_1, T_2 και επιστρέφει την ένωσή τους.
- Αν κάποιο στοιχείο είναι κοινό και στα δύο δέντρα αποθηκεύεται μόνο μία φορά στο τελικό δέντρο.

Union(T_1,T_2) {

If T_1 is empty return T_2

If T_2 is empty return T_1

($T^{<}_2$, $T^{>}_2$)=Split(T_1.root, T_2)

return Join(Union(T_1^L, $T^{<}_2$), T_1.root, Union(T_1^R, $T^{>}_2$))

}

Τομή δύο δέντρων

- Η λειτουργία Intersect(T_1,T_2) δέχεται δύο δέντρα T_1, T_2 και επιστρέφει την τομή τους.

Intersect(T_1,T_2) {

If T_1 or T_2 empty then return empty

($T^<_2$, $T^>_2$)=Split(T_1.root, T_2)

If T_1.root was found in T_2 then

return Join(Interesect(T_1^L, $T^<_2$), T_1.root, Intersect(T_1^R, $T^>_2$)

Else

return Join(Interesect(T_1^L, $T^<_2$), Intersect(T_1^R, $T^>_2$))

}

όπου T_1^L και T_1^R είναι το αριστερό και δεξιό υποδέντρο της ρίζας του T_1.