Δομές Δεδομένων

Μέρος 7ο: Δυαδικά Δένδρα Αναζήτησης

Εξάμηνο Σπουδών: 6ο

Κωδικός Μαθήματος: 681

Τμήμα Μαθηματικών
Πανεπιστήμιο Ιωαννίνων

Μιχάλης Α. Μπέκος
bekos@uoi.gr

Μέρος 1ο

Δυαδικά Δένδρα Αναζήτησης

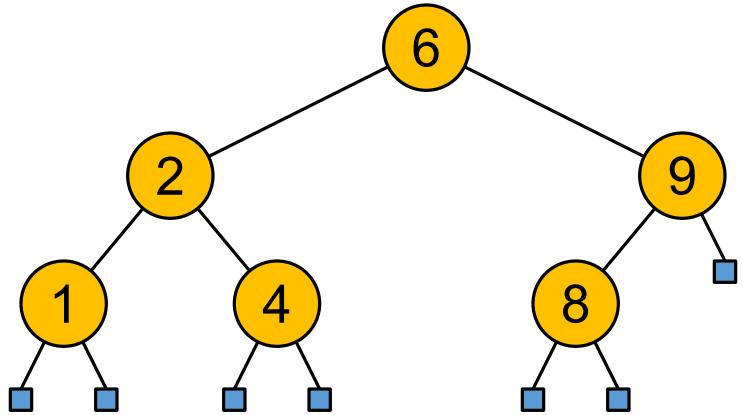
Δυαδικά Δένδρα Αναζήτησης

Binary Search Trees

- Ένα δυαδικό δένδρο αναζήτησης είναι ένα δυαδικό δένδρο το οποίο αποθηκεύει κλειδιά (ή καταχωρήσεις κλειδιών-τιμών) στους εσωτερικούς του κόμβους και ικανοποιεί την παρακάτω ιδιότητα:

  Έστω `u`, `v` και `w` τρεις κόμβοι, τέτοιοι ώστε ο `u` είναι στο αριστερό υποδένδρο του `v` και ο `w` είναι στο δεξί υποδένδρο του `v`. Τότε: `key(u) < key(v) < key(w)`

- Παράδειγμα:

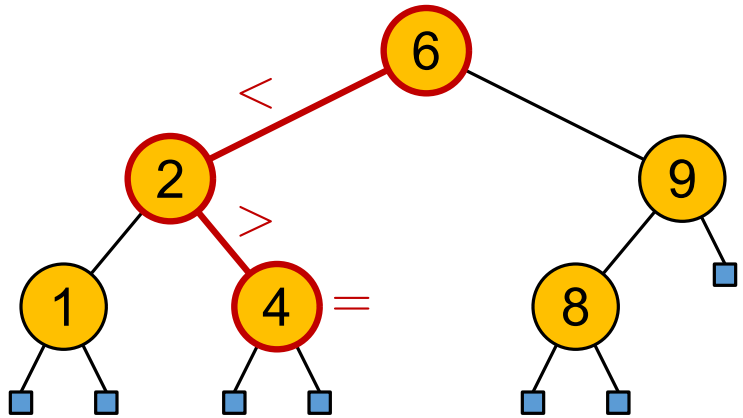


- Παρατηρήσεις:
  - Οι εξωτερικοί κόμβοι δεν αποθηκεύουν καταχωρήσεις
  - Μία inorder διαπέραση του δυαδικού δένδρου αναζήτησης επισκέπτεται τα κλειδιά σε αύξουσα διάταξη

Δυαδικά Δένδρα Αναζήτησης: Αναζήτηση στοιχείου

- Για την αναζήτηση του κλειδιού `k`, ιχνιλατούμε ένα μονοπάτι από τη ρίζα προς τα φύλλα
- Ο επόμενος κόμβος που επισκεπτόμαστε εξαρτάται από τη σύγκριση του `k` με το κλειδί του τρέχοντος κόμβου
- Εάν φτάσουμε σε ένα φύλλο, το κλειδί δεν βρέθηκε

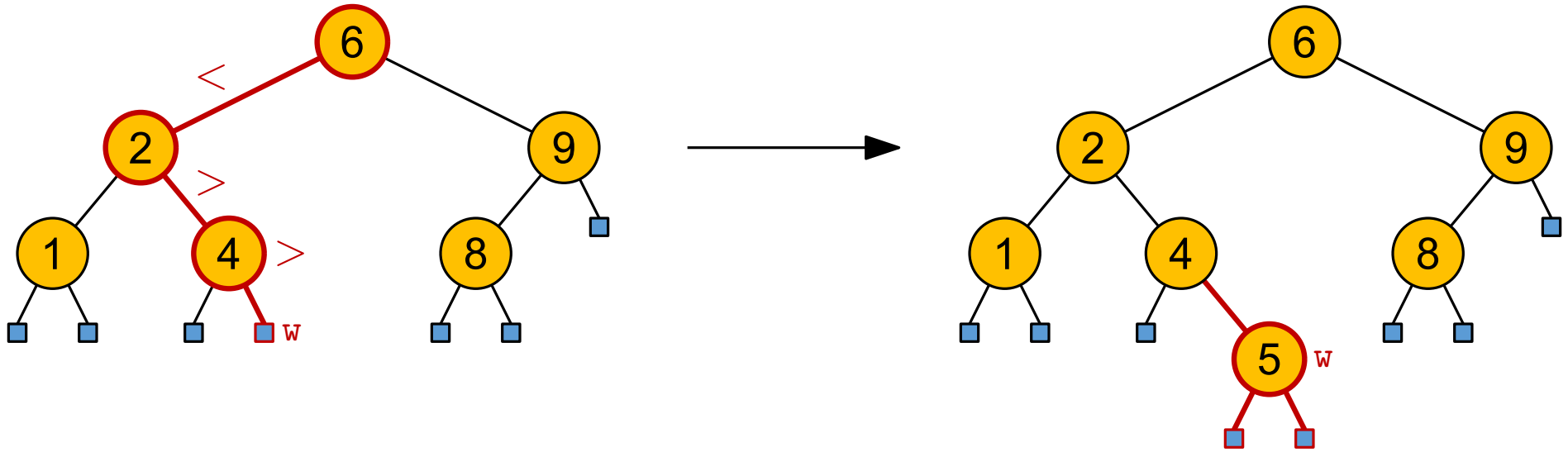
- Παράδειγμα: Αναζήτηση του κλειδιού 4




```
1 Algorithm TreeSearch(k, v)
2 |   if v.isExternal() then
3 |   |   return null;
4 |   if k < v.key() then
5 |   |   return TreeSearch(k, v.left());
6 |   else if k > v.key() then
7 |   |   return TreeSearch(k, v.right());
8 |   return v;
```

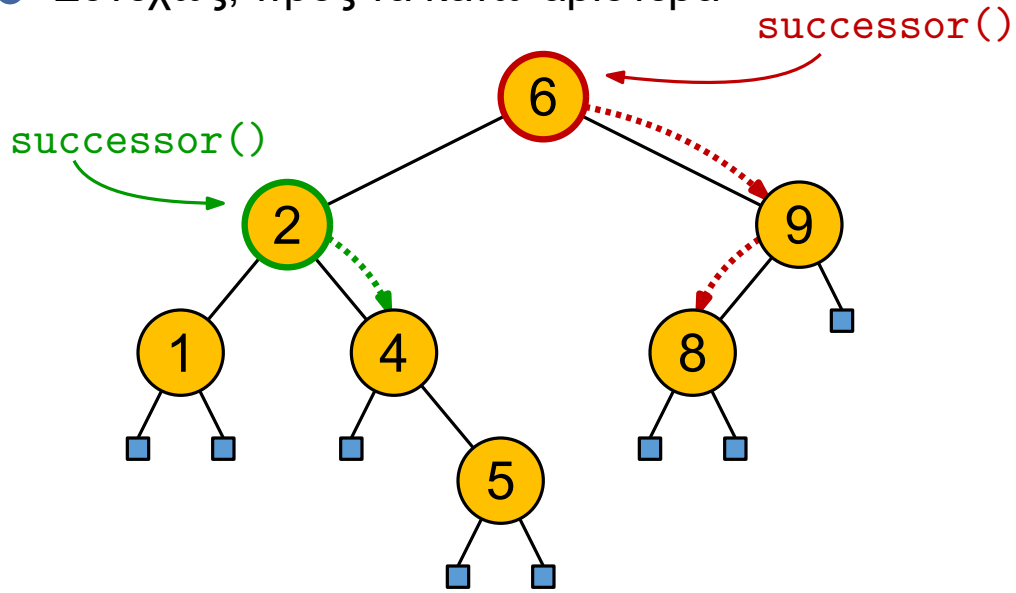
Δυαδικά Δένδρα Αναζήτησης: Εισαγωγή Στοιχείου

- Για την εισαγωγή μιας εγγραφής `(k, d)` κάνουμε μια αναζήτηση για το κλειδί `k` (χρησιμοποιώντας τον αλγόριθμο `TreeSearch`)
- Έστω ότι το στοιχείο `k` δεν υπάρχει ήδη στο δένδρο και ότι το `w` είναι το φύλλο που καταλήγει η (αποτυχημένη) αναζήτηση του `k`
- Εισάγουμε το `k` στον κόμβο `w` και επεκτείνουμε τον `w` σε εσωτερικό κόμβο
- Παράδειγμα: Εισαγωγή του στοιχείου 5

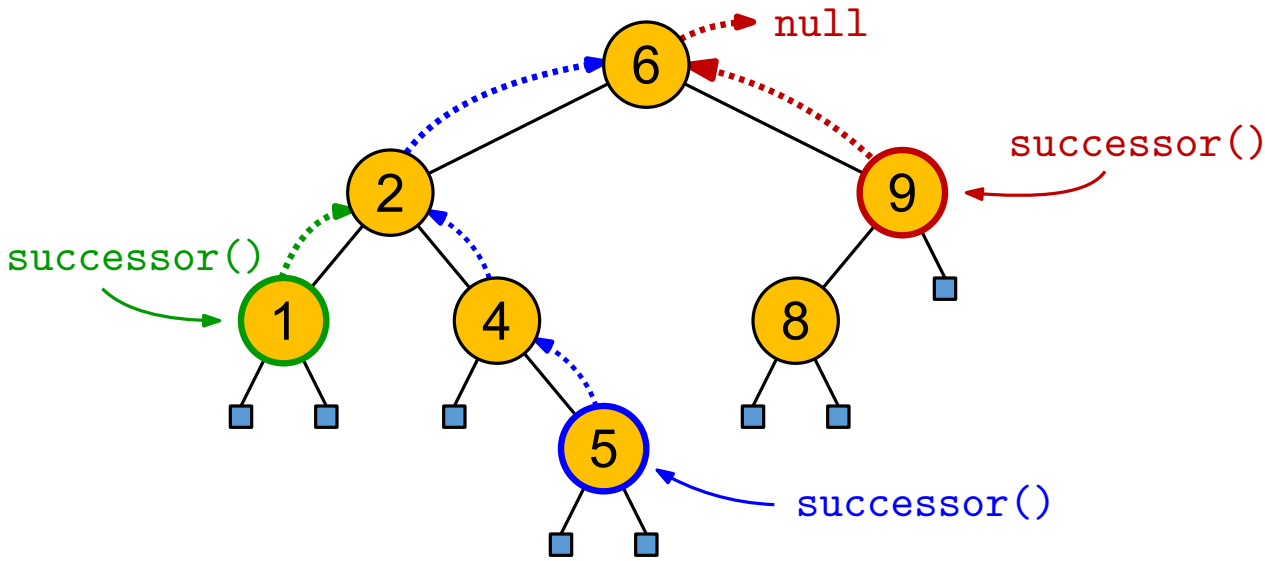


Δυαδικά Δένδρα Αναζήτησης: Εύρεση επόμενου στοιχείου

- Έστω ο κόμβος `v` ο οποίος περιέχει το κλειδί `k`. Θέλουμε να βρούμε τον κόμβο που περιέχει το αμέσως επόμενο σε μέγεθος κλειδί.
- Ο ζητούμενος κόμβος `w` είναι ο κόμβος που ακολουθεί τον `v` στην inorder διαπέραση
- **Περίπτωση-1:** Το δεξί παιδί του κόμβου `v` δεν είναι φύλλο.
  - Ένα βήμα προς τα κάτω-δεξιά
  - Συνεχώς, προς τα κάτω-αριστερά

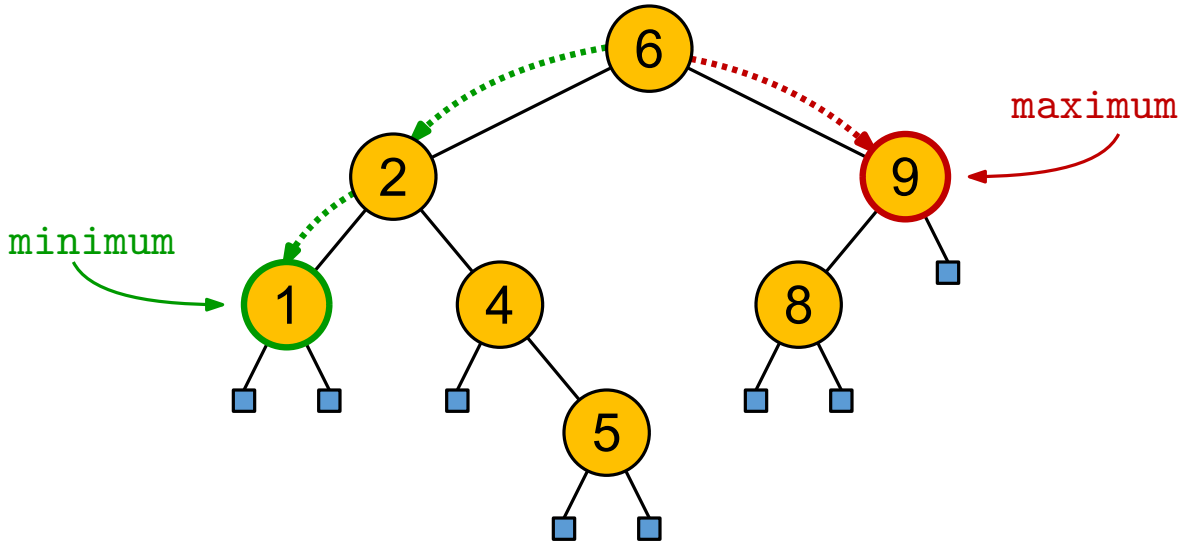


- **Περίπτωση-2:** Το δεξί παιδί του κόμβου `v` είναι φύλλο.
  - Συνεχώς, προς τα πάνω-αριστερά
  - Ένα βήμα προς τα πάνω-δεξιά (εάν αδύνατον, return null)



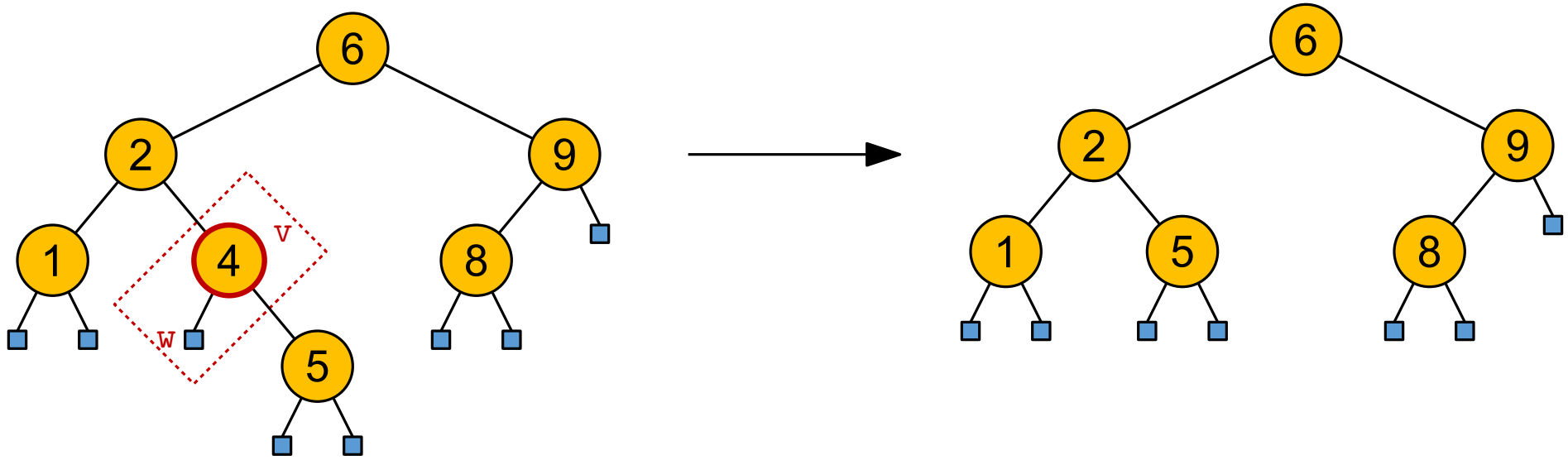
Δυαδικά Δένδρα Αναζήτησης: Εύρεση μέγιστου / ελάχιστου

- Εύρεση προηγούμενου στοιχείου: Συμμετρικά με την εύρεση του επόμενου στοιχείου
- Εύρεση μικρότερου στοιχείου:
  Πήγαινε στη ρίζα $\rightarrow$ συνεχώς, κάτω-αριστερά
- Εύρεση μέγιστου στοιχείου:
  Πήγαινε στη ρίζα $\rightarrow$ συνεχώς, κάτω-δεξιά
- Παράδειγμα:



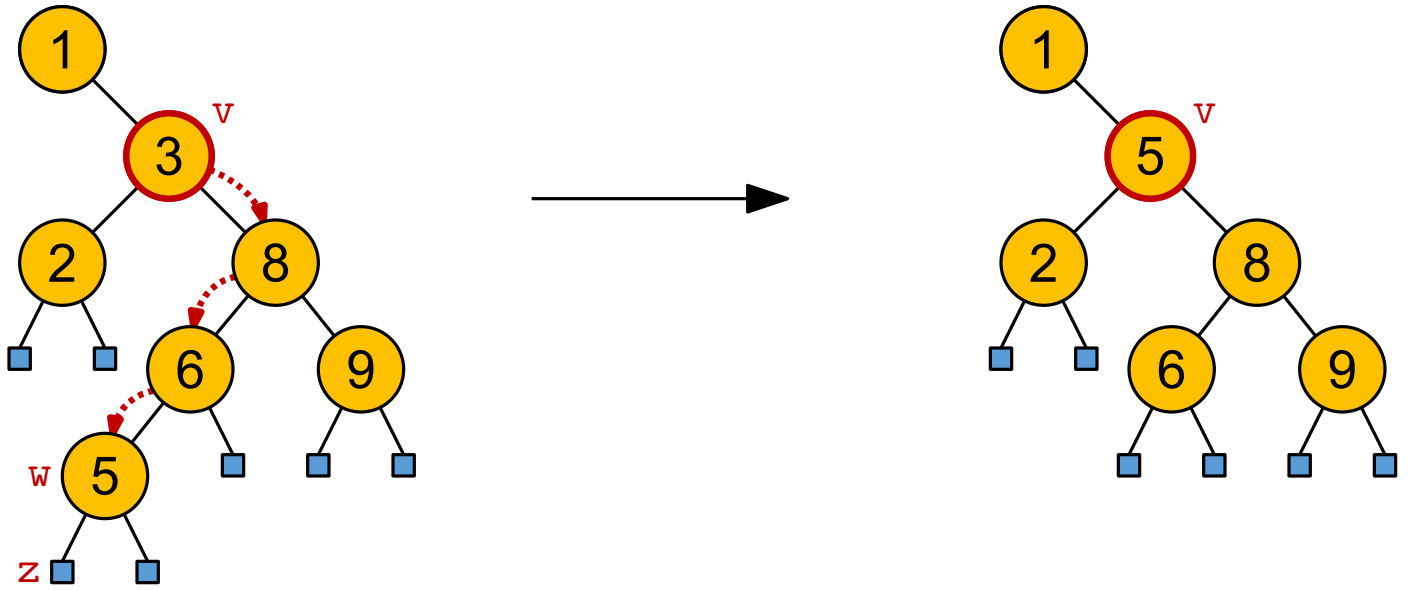
Δυαδικά Δένδρα Αναζήτησης: Διαγραφή στοιχείου

- Για τη διαγραφή μιας εγγραφής με κλειδί `k`, κάνουμε μια αναζήτηση για το κλειδί `k`.
- Έστω ότι η εγγραφή με κλειδί `k` είναι αποθηκευμένη στον κόμβο `v`
- **Περίπτωση-1:** : Ο κόμβος `v` έχει ως παιδί ένα φύλλο `w`
  - Διαγράφουμε τους `v` και `w` από το δένδρο
  - Συνδέουμε τον πατέρα του `v` με το άλλο παιδί του `v`
- Παράδειγμα: Διαγραφή του στοιχείου 4



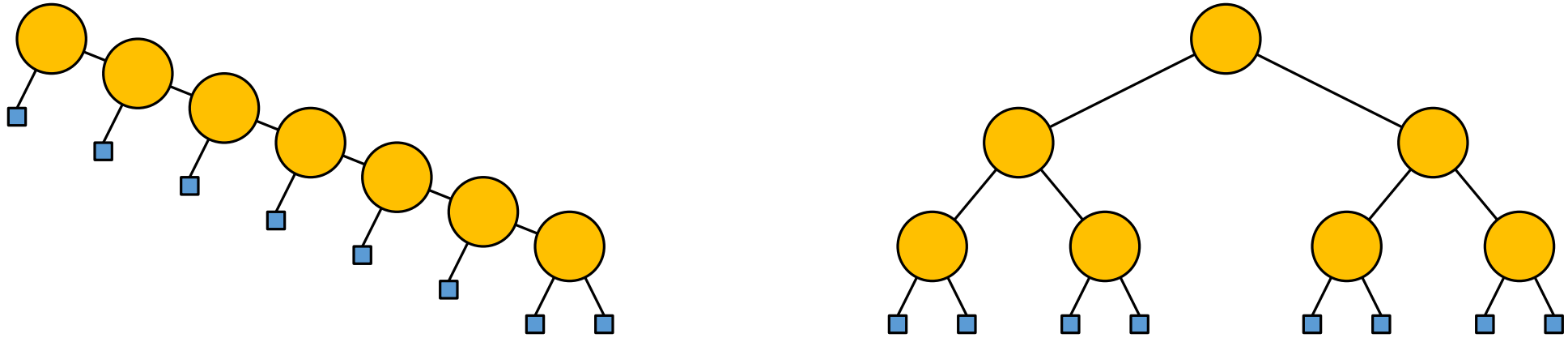
Δυαδικά Δένδρα Αναζήτησης: Διαγραφή στοιχείου

- Για τη διαγραφή μιας εγγραφής με κλειδί `k`, κάνουμε μια αναζήτηση για το κλειδί `k`.
- Έστω ότι η εγγραφή με κλειδί `k` είναι αποθηκευμένη στον κόμβο `v`
- **Περίπτωση-2:** : Τα παιδιά του κόμβου `v` είναι (και τα δύο) εσωτερικοί κόμβοι
  - Βρίσκουμε τον εσωτερικό κόμβο `w` οποίος ακολουθεί τον `v` σε μια inorder διαπέραση του δένδρου
  - Αντιγράφουμε το κλειδί του `w` στον κόμβο `v` και αφαιρούμε τον κόμβο `w` και το αριστερό παιδί (φύλλο) `z` αυτού
- Παράδειγμα: Διαγραφή του στοιχείου 3



Δυαδικά Δένδρα Αναζήτησης: Απόδοση

- Υποθέτουμε ένα δυαδικό δένδρο αναζήτησης με n εγγραφές και ύψος h
- Ο χώρος που χρησιμοποιείται είναι $O(n)$
- Η εισαγωγή, διαγραφή, εύρεση επόμενου/προηγούμενου, ελαχίστου και μεγίστου ολοκληρώνονται σε $O(h)$ χρόνο
- Το ύψος h είναι $O(n)$ στη χειρότερη περίπτωση και $O(\log n)$ στη καλύτερη περίπτωση



- Σημείωση: Όταν τα στοιχεία εισάγονται σε τυχαία σειρά, το αναμενόμενο ύψος h είναι $O(\log n)$

ΑΤΔ: Ουρά Προτεραιότητας

	Unsorted list	Sorted list	Heap	Binary Search Tree	
				worst case	expected
`empty()`	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
`size()`	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
`insert(e)`	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(\log n)$	$\mathcal{O}(n)$	$\mathcal{O}(\log n)$
`removeMin()`	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(\log n)$	$\mathcal{O}(n)$	$\mathcal{O}(\log n)$
`min()`	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$

Μέρος 2ο

AVL Δένδρα

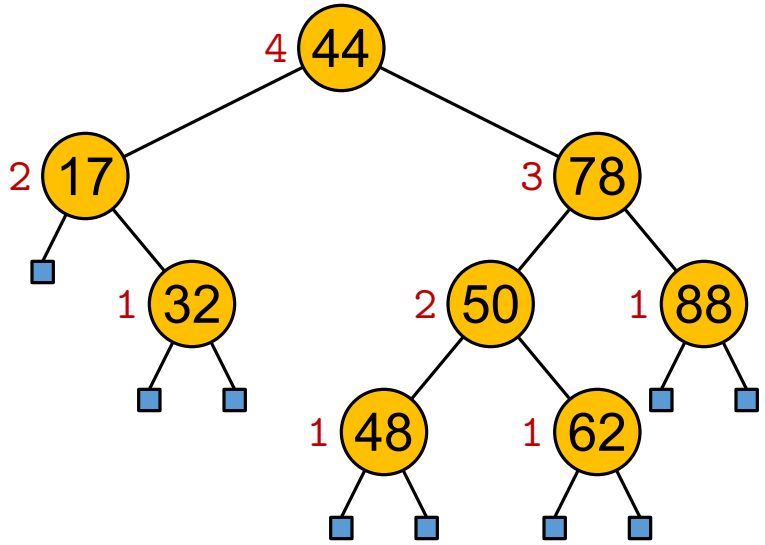
AVL Δένδρα

AVL Trees

- Ένα AVL δένδρο είναι ένα δυαδικό δένδρο αναζήτησης, τέτοιο ώστε για κάθε εσωτερικό κόμβο του v ισχύει ότι τα ύψη των υπο-δένδρων του v διαφέρουν το πολύ κατά 1.

- Τα AVL δένδρα είναι “ζυγισμένα”

- Παράδειγμα AVL δέντρου (τα ύψη εμφανίζονται δίπλα στους κόμβους)



Το ύψος AVL δένδρου

- Θεώρημα: Το ύψος ενός AVL δένδρου το οποίο έχει αποθηκευμένα n κλειδιά είναι $O(\log n)$.

  Απόδειξη:

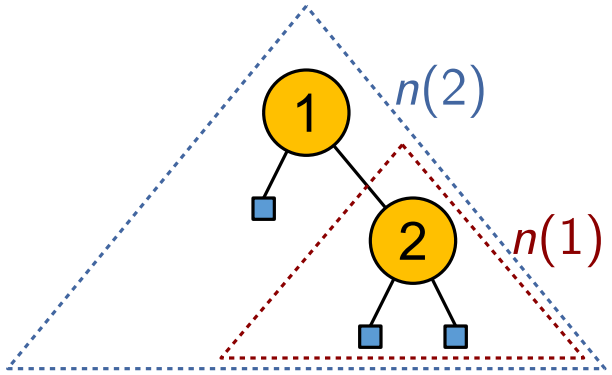
  $n(h) \leftarrow$ ο ελάχιστος αριθμός εσωτερικών κόμβων ενός AVL δένδρο ύψους h.

  Ισχυρισμός: $n(h) > 2^{h/2-1}$

  - Εύκολα βλέπουμε ότι $n(1) = 1$ και $n(2) = 2$
  - Για $h > 2$, ισχύει*: $n(h) = 1 + n(h-1) + n(h-2)$

    Προφανώς $n(h-1) > n(h-2)$, οπότε: $n(h) > 2 \cdot n(h-2)$.

$$\left.\begin{array}{l} n(h) > 2^1 \cdot n(h-2) \\ n(h) > 2^2 \cdot n(h-4) \\ \ldots \\ n(h) > 2^i \cdot n(h-2i) \end{array}\right\} \quad n(h) > 2^{h/2-1} \quad \Rightarrow \quad \boxed{h < 2 \cdot \log n(h) + 2}$$



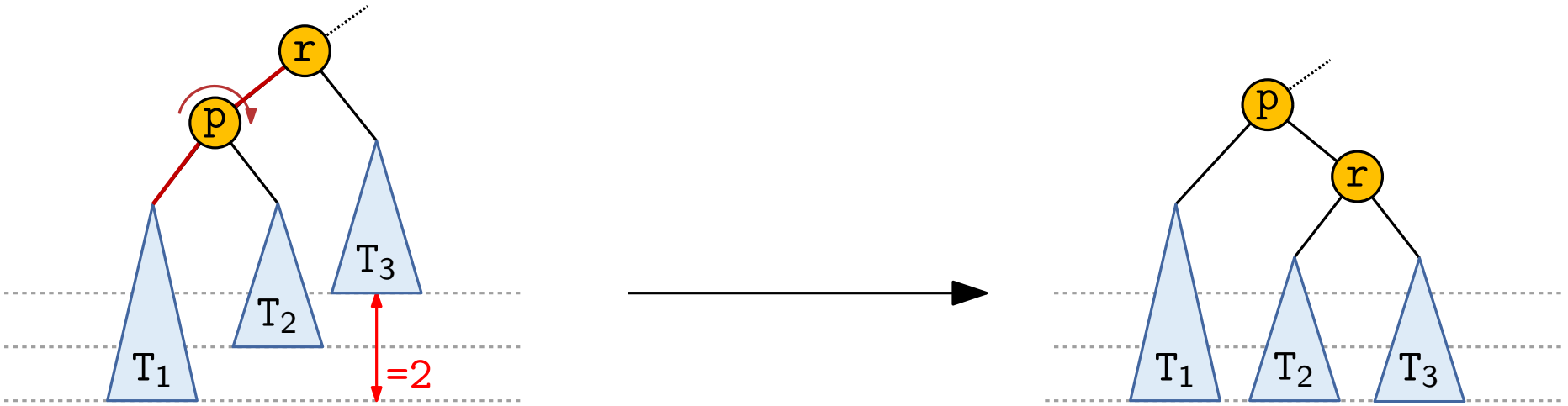
* Ένα AVL δέντρο ύψους h περιέχει τον κόμβο-ρίζα, ένα AVL υποδέντρο ύψους $h-1$ και ένα άλλο ύψους $h-2$

AVL Δένδρα: Αναδιάταξη μέσω περιστρεφών

AVL Trees

- Περιστροφή: Τοπικός μετασχηματισμός γύρω από έναν κόμβο χωρίς να αλλάζει η inorder διαπέραση των στοιχείων.
  Rotation

- Απλές περιστροφές: Δεξιά περιστροφή γύρω από τον κόμβο `p`

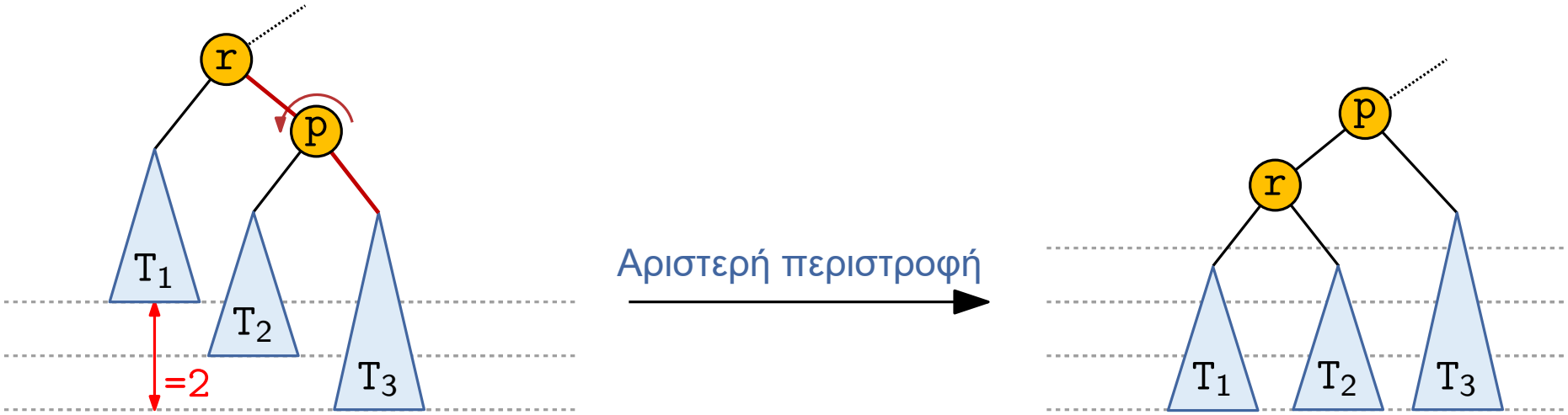


AVL Δένδρα: Αναδιάταξη μέσω περιστρεφών

AVL Trees

- Περιστροφή: Τοπικός μετασχηματισμός γύρω από έναν κόμβό χωρίς να αλλάζει η inorder διαπέραση των στοιχείων.
  Rotation

- Απλές περιστροφές: Αριστερή περιστροφή γύρω από τον κόμβο `p`, όπου `r` είναι ο κόμβος στον οποίο χάνεται η συνθήκη ισορροπίας AVL



AVL Δένδρα: Αναδιάταξη μέσω περιστρεφών

AVL Trees

- Περιστροφή: Τοπικός μετασχηματισμός γύρω από έναν κόμβο χωρίς να αλλάζει η inorder διαπέραση των στοιχείων.
  Rotation

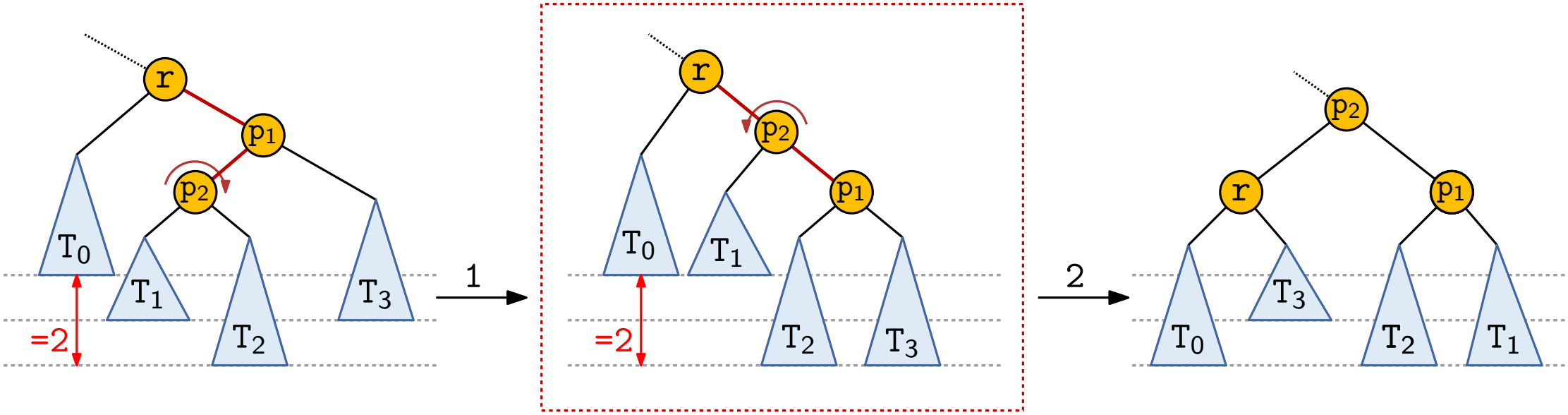
- Απλές περιστροφές: Δεξιά περιστροφή γύρω από τον κόμβο `p`, όπου `r` είναι ο κόμβος στον οποίο χάνεται η συνθήκη ισορροπίας AVL

AVL Δένδρα: Αναδιάταξη μέσω περιστρεφών

AVL Trees

- **Περιστροφή:** Rotation
  Τοπικός μετασχηματισμός γύρω από έναν κόμβό χωρίς να αλλάζει η inorder διαπέραση των στοιχείων.

- **Διπλές περιστροφές:** Διπλή περιστροφή γύρω από τους κόμβους p_1 και p_2, όπου r είναι ο κόμβος στον οποίο χάνεται η συνθήκη ισορροπίας AVL



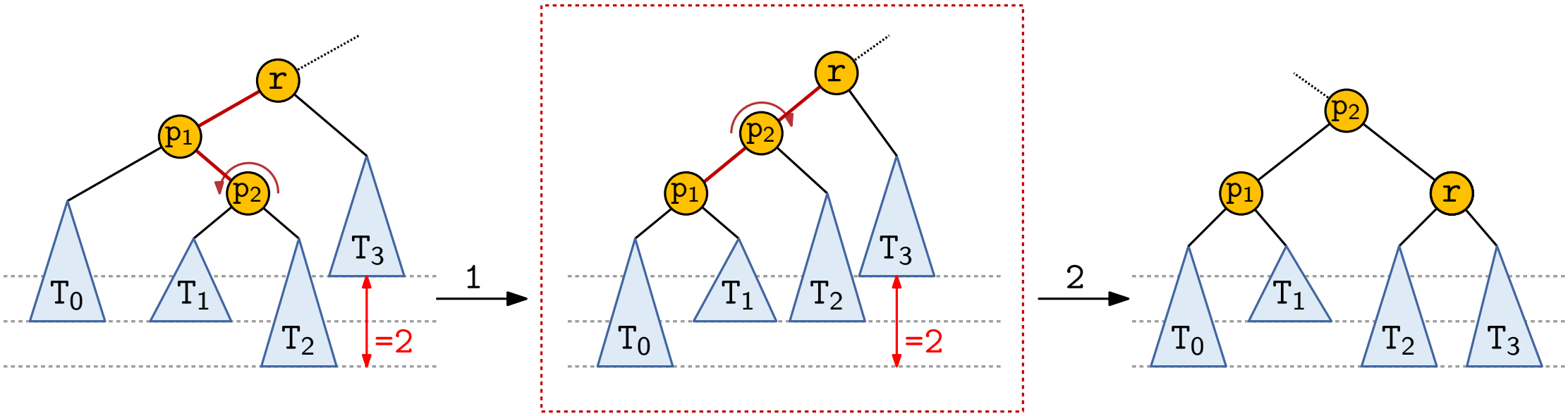
[πρώτα “δεξιά”, μετά “αριστερά”]

AVL Δένδρα: Αναδιάταξη μέσω περιστρεφών

AVL Trees

- **Περιστροφή:** Rotation
  Τοπικός μετασχηματισμός γύρω από έναν κόμβό χωρίς να αλλάζει η inorder διαπέραση των στοιχείων.

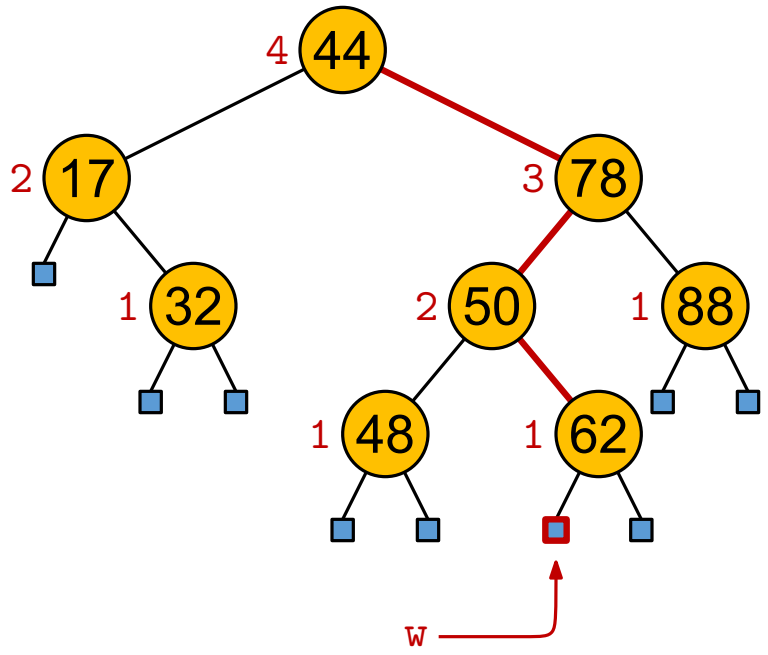
- **Διπλές περιστροφές:** Διπλή περιστροφή γύρω από τους κόμβους p_1 και p_2, όπου r είναι ο κόμβος στον οποίο χάνεται η συνθήκη ισορροπίας AVL



[πρώτα “αριστερά”, μετά “δεξιά”]

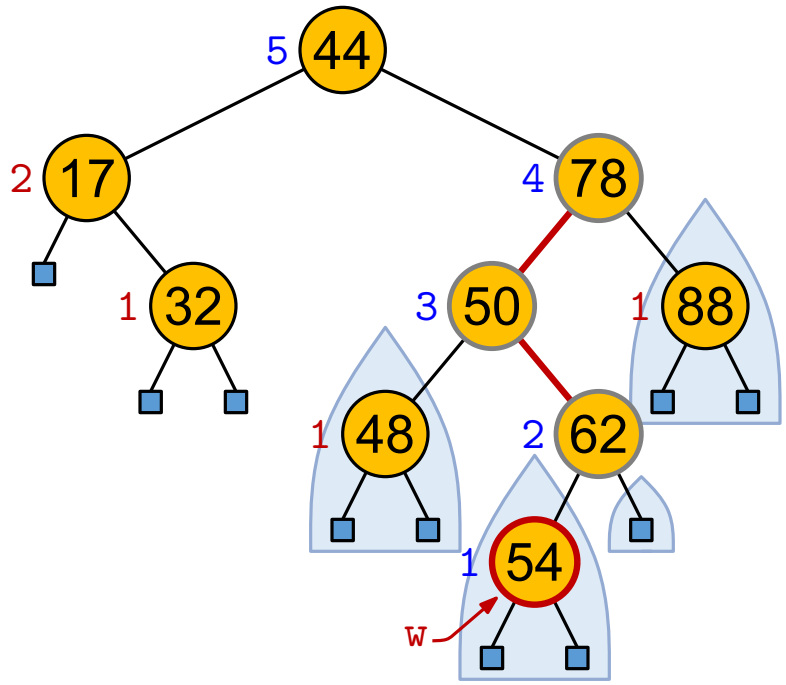
AVL Δένδρα: Εισαγωγή Στοιχείου

- Η εισαγωγή είναι ίδια όπως και στα δυαδικά δένδρα αναζήτησης.
- Πάντα ολοκληρώνεται με μία “επέκταση” ενός εξωτερικού κόμβου.
- Παράδειγμα: Εισαγωγή του στοιχείου 54



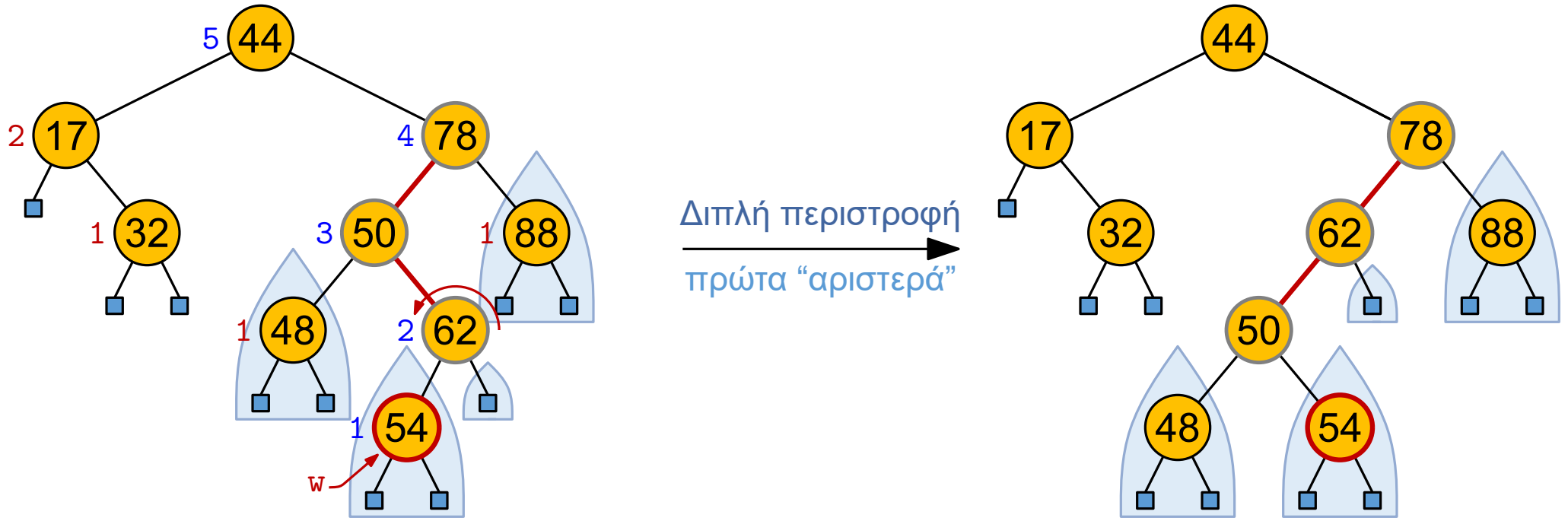
AVL Δένδρα: Εισαγωγή Στοιχείου

- Η εισαγωγή είναι ίδια όπως και στα δυαδικά δένδρα αναζήτησης.
- Πάντα ολοκληρώνεται με μία “επέκταση” ενός εξωτερικού κόμβου.
- Παράδειγμα: Εισαγωγή του στοιχείου 54



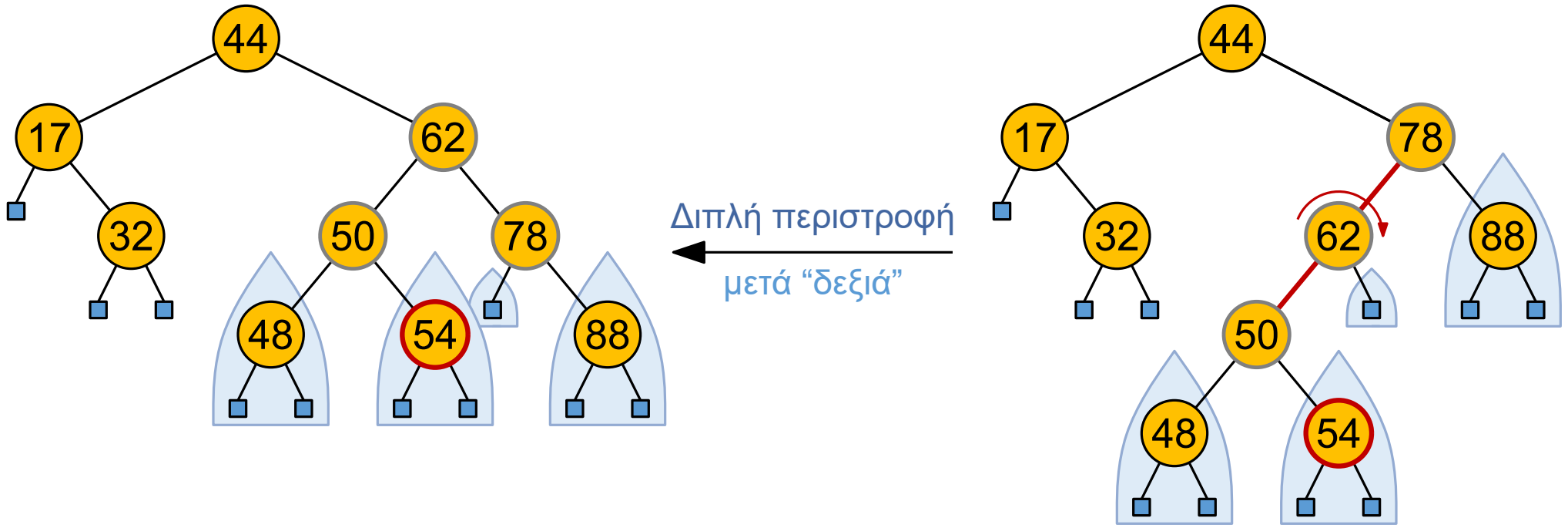
AVL Δένδρα: Εισαγωγή Στοιχείου

- Η εισαγωγή είναι ίδια όπως και στα δυαδικά δένδρα αναζήτησης.
- Πάντα ολοκληρώνεται με μία “επέκταση” ενός εξωτερικού κόμβου.
- Παράδειγμα: Εισαγωγή του στοιχείου 54



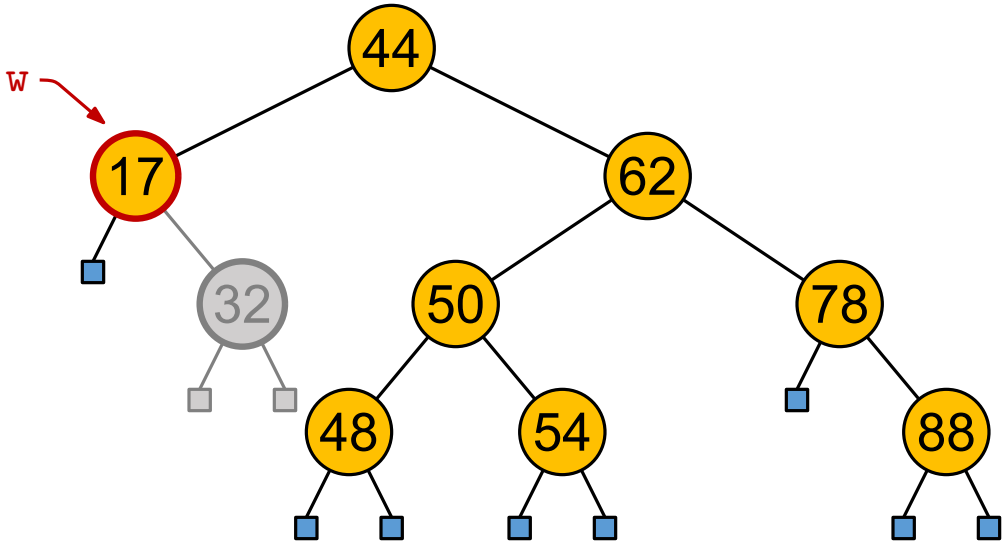
AVL Δένδρα: Εισαγωγή Στοιχείου

- Η εισαγωγή είναι ίδια όπως και στα δυαδικά δένδρα αναζήτησης.
- Πάντα ολοκληρώνεται με μία “επέκταση” ενός εξωτερικού κόμβου.
- Παράδειγμα: Εισαγωγή του στοιχείου 54



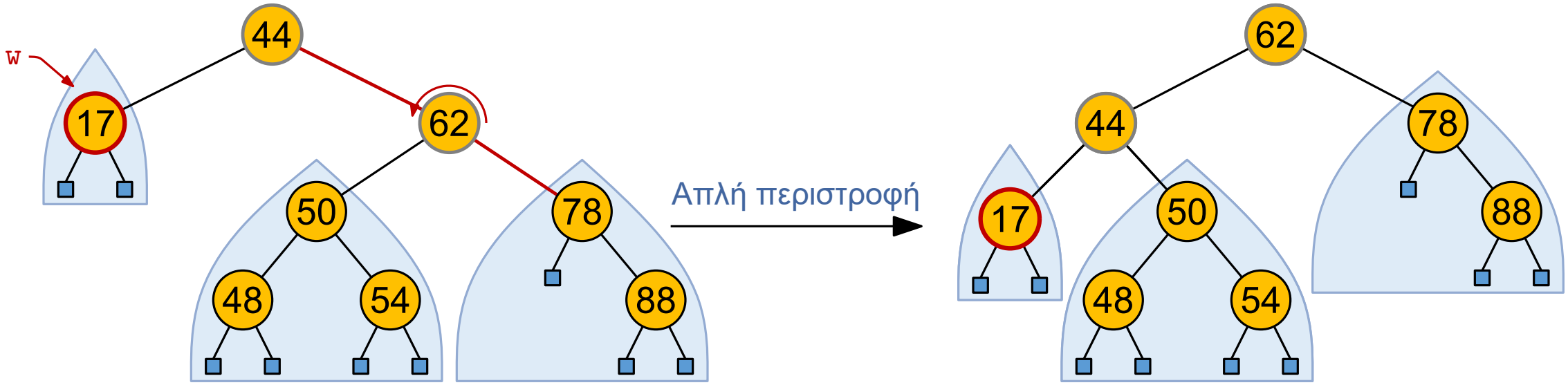
AVL Δένδρα: Διαγραφή Στοιχείου

- Η διαγραφή γίνεται όπως και στα δυαδικά δένδρα αναζήτησης.
- Στον πατέρα, `w`, του κόμβου που διαγράφεται μπορεί να προκληθεί ανισορροπία.
- Παράδειγμα: Διαγραφή του στοιχείου 32



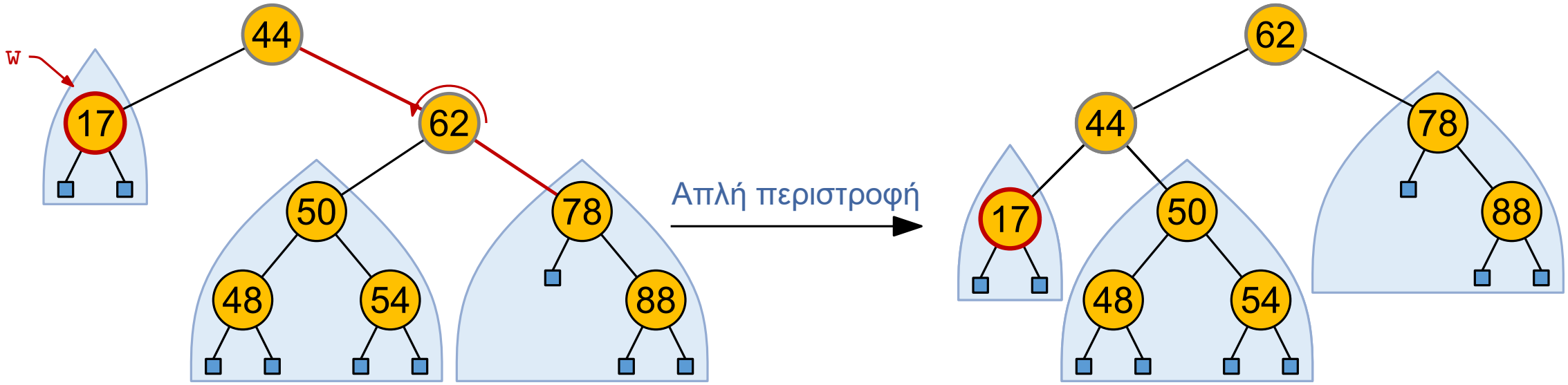
AVL Δένδρα: Διαγραφή Στοιχείου

- Η διαγραφή γίνεται όπως και στα δυαδικά δένδρα αναζήτησης.
- Στον πατέρα, `w`, του κόμβου που διαγράφεται μπορεί να προκληθεί ανισορροπία.
- Παράδειγμα: Διαγραφή του στοιχείου 32



AVL Δένδρα: Διαγραφή Στοιχείου

- Η διαγραφή γίνεται όπως και στα δυαδικά δένδρα αναζήτησης.
- Στον πατέρα, `w`, του κόμβου που διαγράφεται μπορεί να προκληθεί ανισορροπία.
- Παράδειγμα: Διαγραφή του στοιχείου 32



- Σημείωση: Η περιστροφή μπορεί να προκαλέσει ανισορροπία σε άλλο κόμβο ψηλότερα στο δέντρο $\Rightarrow$ Συνεχίζουμε τον έλεγχο της ισορροπίας μέχρι να φτάσουμε στη ρίζα του δένδρου

AVL Δένδρα: Απόδοση

- Ένα AVL δένδρο στο οποίο είναι αποθηκευμένα n στοιχεία χρησιμοποιεί $O(n)$ χώρο
- Μία περιστροφή γίνεται σε $O(1)$ χρόνο
- Η αναζήτηση στοιχείου γίνεται σε $O(\log n)$ χρόνο
  - Δεν είναι απαραίτητη καμία αναδιάταξη
  - Το ύψος του δένδρου είναι $O(\log n)$
- Η εισαγωγή στοιχείου γίνεται σε $O(\log n)$ χρόνο
  - Η αρχική αναζήτηση γίνεται σε $O(\log n)$ χρόνο
  - Απαιτείται μόνο μία αναδιάταξη
- Η διαγραφή στοιχείου γίνεται σε $O(\log n)$ χρόνο
  - Η αρχική αναζήτηση γίνεται σε $O(\log n)$ χρόνο
  - Το πολύ $O(\log n)$ αναδιατάξεις

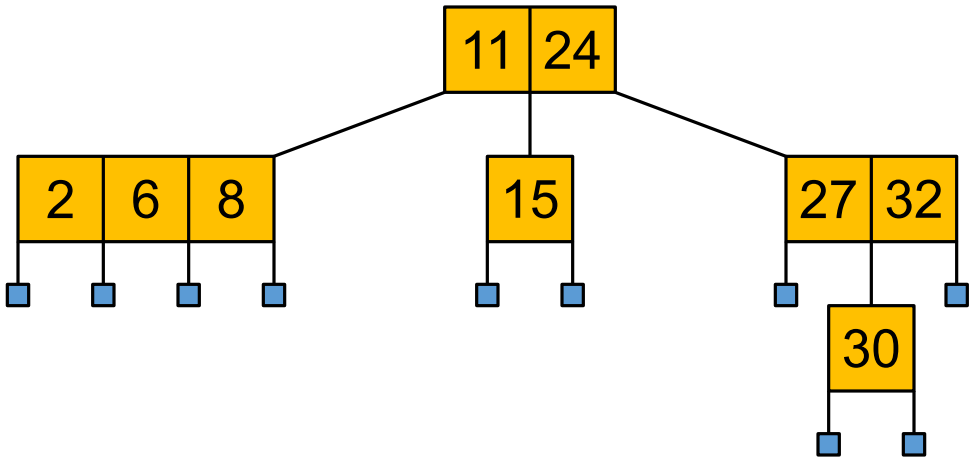
Μέρος 3ο

(2,4) Δένδρα

Πολυκατευθυνόμενο δέντρο αναζήτησης

Multi-Way search tree

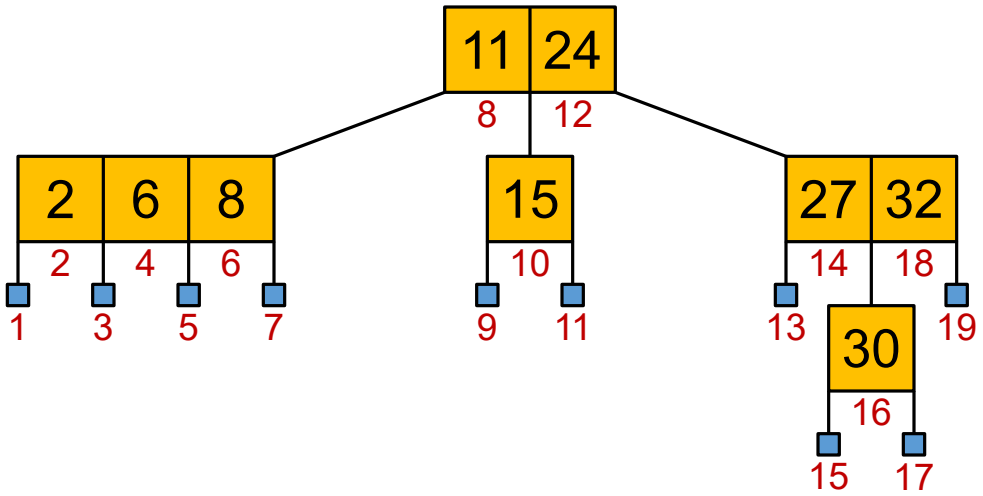
- Ένα πολυκατευθυνόμενο δέντρο αναζήτησης είναι ένα διατεταγμένο δέντρο τέτοιο ώστε:
  - Κάθε εσωτερικός κόμβος έχει τουλάχιστον δύο παιδιά και αποθηκεύει $d - 1$ στοιχεία (ζεύγη κλειδιών-τιμών (k_i, o_i)), όπου d είναι ο αριθμός των παιδιών του.
  - Για έναν κόμβο με παιδιά $v_1, \ldots, v_d$ που περιέχουν τα κλειδιά $k_1, \ldots, k_{d-1}$:
    - τα κλειδιά στο υποδέντρο του v_1 είναι μικρότερα του k_1
    - τα κλειδιά στο υποδέντρο του v_i είναι μεταξύ k_{i-1} και k_i $(i = 2, \ldots, d - 1)$
    - τα κλειδιά στο υποδέντρο του v_d είναι μικρότερα του k_{d-1}
  - Τα φύλλα δεν περιέχουν στοιχεία
- Παράδειγμα:



Εσωδιάταξη ενός πολυκατευθυνόμενου δένδρου αναζήτησης

Multi-Way inorder traversal

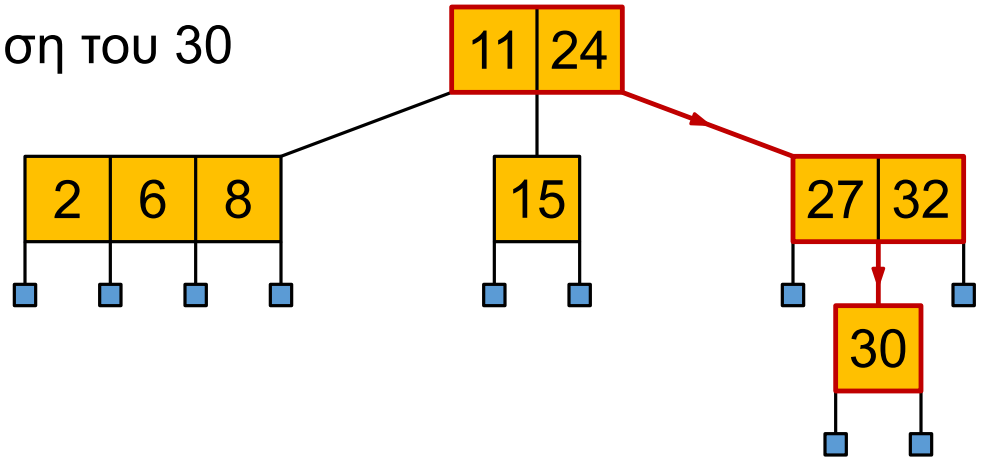
- Η εσωδιάταξη ενός δυαδικού δένδρου μπορεί εύκολα να επεκταθεί σε πολυκατευθυνόμενα δέντρα αναζήτησης.
- Επισκεπτόμαστε το στοιχείο (k_i, o_i) του κόμβου v μεταξύ των αναδρομικών διατάξεων των υποδέντρων του v με ρίζα τα παιδιά v_i και v_{i+1}
- Μια εσωδιάταξη ενός πολυκατευθυνόμενου δέντρου αναζήτησης επισκέπτεται τα κλειδιά του δένδρου σε αύξουσα σειρά
- Παράδειγμα:



Αναζήτηση σε πολυκατευθυνόμενο δένδρο αναζήτησης

Multi-Way searching

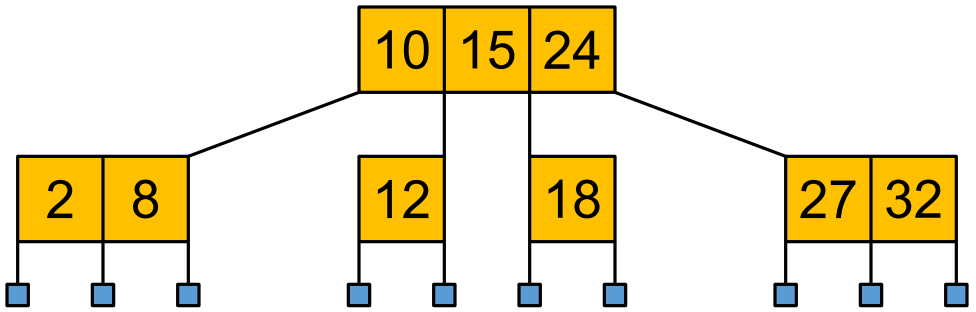
- Παρόμοια με την αναζήτηση ενός κλειδιού k σε ένα δυαδικό δέντρο αναζήτησης
- Για κάθε εσωτερικό κόμβο με παιδιά $v_1, \ldots, v_d$ και κλειδιά $k_1, \ldots, k_{d-1}$, αν:
  - $k = k_i$ $(i = 1, \ldots, d-1)$: η αναζήτηση τερματίζει επιτυχώς
  - $k < k_1$: η αναζήτηση συνεχίζει στο παιδί v_1
  - $k_{i-1} < k < k_i$ $(i = 1, \ldots, d-1)$: η αναζήτηση συνεχίζει στο παιδί v_i
  - $k > k_{d-1}$: η αναζήτηση συνεχίζει στο παιδί v_d
- Η αναζήτηση τερματίζει ανεπιτυχώς αν καταλήξει σε εξωτερικό κόμβο
- Παράδειγμα: Αναζήτηση του 30



(2,4) δένδρα

(2,4) trees

- Ένα (2,4) δένδρο είναι ένα πολυκατευθυνόμενο δέντρο αναζήτησης με τις παρακάτω ιδιότητες:
  - Μέγεθος κόμβου: κάθε εσωτερικός κόμβος έχει το πολύ 4 παιδιά
  - Βάθος φύλλων: όλοι οι εξωτερικοί κόμβοι έχουν το ίδιο βάθος
- Ανάλογα με το πλήθος των παιδιών του, ένας εσωτερικός κόμβος ενός (2,4) δέντρου λέγεται 2-κόμβος, 3-κόμβος ή 4-κόμβος
- Παράδειγμα:



Το ύψος ενός (2,4) δένδρου

- Θεώρημα: Το ύψος ενός (2,4) δένδρου το οποίο έχει αποθηκευμένα n κλειδιά είναι $O(\log n)$.

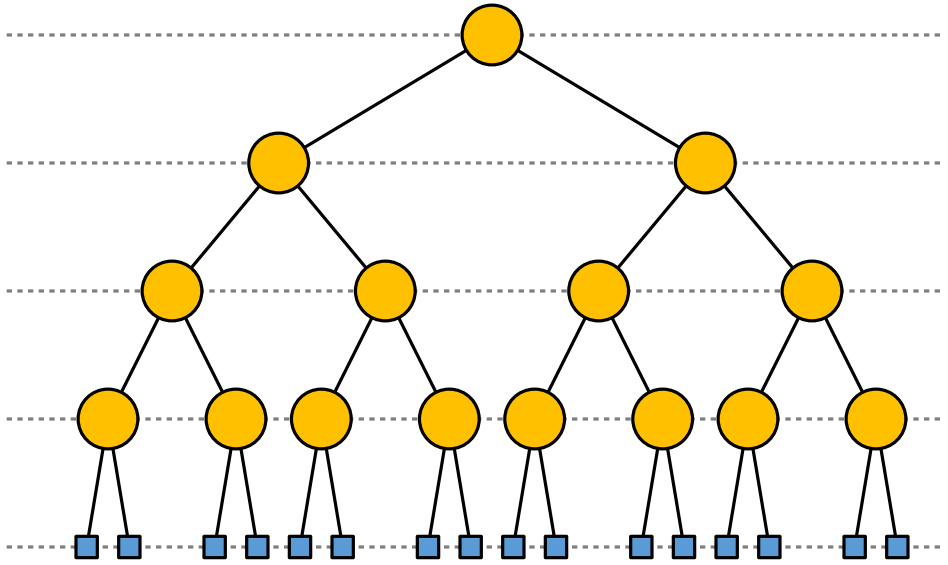
  Απόδειξη:

  - $h \leftarrow$ το ύψος του (2,4) δένδρου
  - Για κάθε $i = 0, 1, \ldots, h-1$, υπάρχουν τουλάχιστον 2^i στοιχεία βάθους i, και κανένα βάθους h. Οπότε:

    $n \geq 2^0 + 2^1 + 2^2 + \ldots + 2^{h-1}$

    $\Rightarrow n \geq \frac{2^h - 1}{2-1} \Rightarrow n \geq 2^h - 1$

    $\Rightarrow h \leq \log(n+1)$

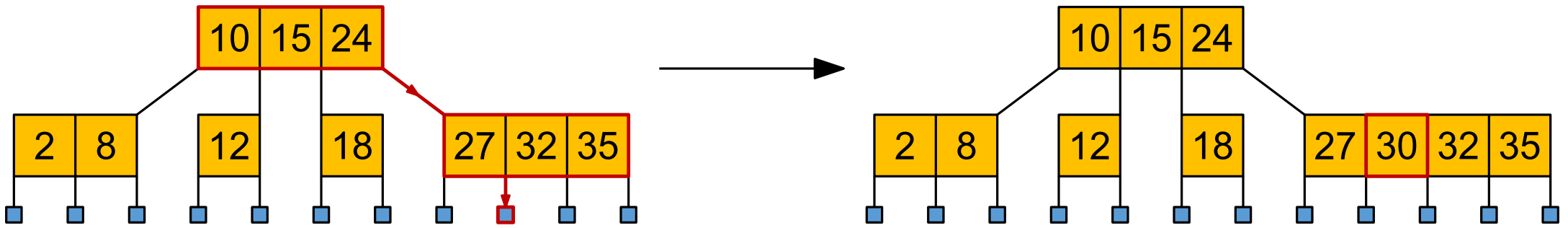


βάθος	στοιχεία
0	1
1	2
⋮	⋮
$h-1$	2^{h-1}
h	0

Εισαγωγή σε (2,4)-δένδρο

- Η εισαγωγή ενός νέου στοιχείου `(k, o)` γίνεται στον γονέα `v` του φύλλου στο οποίο κατέληξε η αναζήτηση του `k`
  - διατηρείται η ιδιότητα του βάθους αλλά
  - μπορεί να προκληθεί υπερχείλιση (π.χ., ο κόμβος `v` μπορεί να γίνει ένας 5-κόμβος)

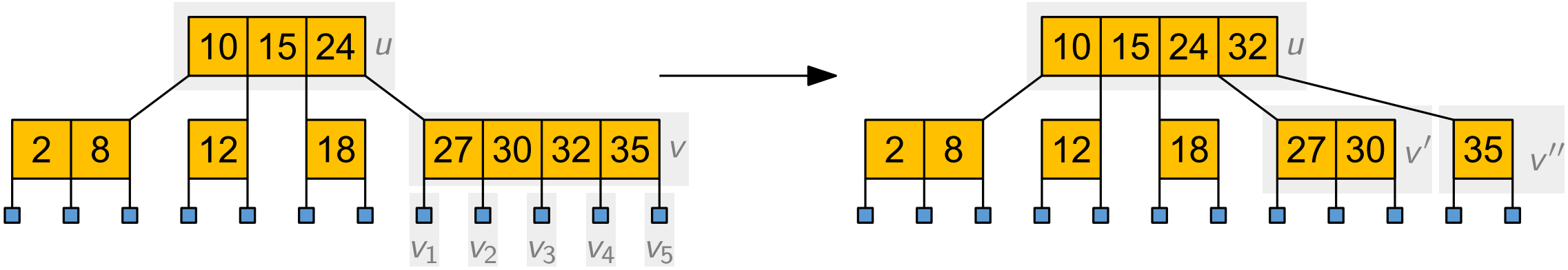
- Παράδειγμα: η εισαγωγή του κλειδιού 30 προκαλεί υπερχείλιση



Υπερχείλιση και διάσπαση

Overflow και split

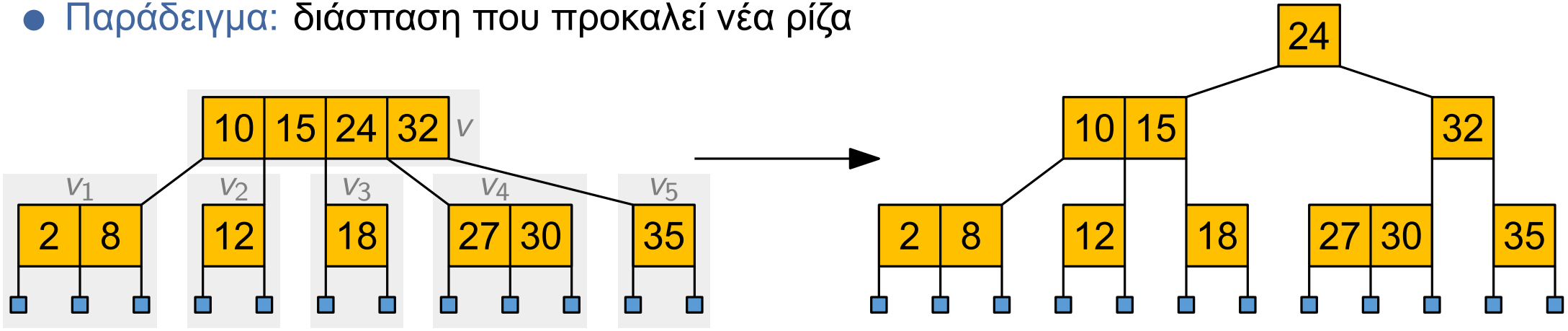
- Η υπερχείλιση σε ένα 5-κόμβο v αντιμετωπίζεται με μία διεργασία διάσπασης:
  - έστω $v_1, \ldots, v_5$ τα παιδιά του v και $k_1, \ldots, k_4$ τα κλειδιά του v
  - ο κόμβος v αντικαθίσταται από τους κόμβους v' και v
    - ο v' είναι ένας 3-κόμβος με κλειδιά k_1, $k2$ και παιδιά v_1, v_2, v_3
    - ο v είναι ένας 2-κόμβος με κλειδί k_4 και παιδιά v_4, v_5
  - το κλειδί k_3 εισάγεται στο γονέα u του v (μπορεί να δημιουργηθεί νέα ρίζα ή υπερχείλιση)
- Παράδειγμα: διάσπαση που προκαλεί υπερχείλιση



Υπερχείλιση και διάσπαση

Overflow και split

- Η υπερχείλιση σε ένα 5-κόμβο v αντιμετωπίζεται με μία διεργασία διάσπασης:
  - έστω $v_1, \ldots, v_5$ τα παιδιά του v και $k_1, \ldots, k_4$ τα κλειδιά του v
  - ο κόμβος v αντικαθίσταται από τους κόμβους v' και v
    - ο v' είναι ένας 3-κόμβος με κλειδιά k_1, $k2$ και παιδιά v_1, v_2, v_3
    - ο v είναι ένας 2-κόμβος με κλειδί k_4 και παιδιά v_4, v_5
  - το κλειδί k_3 εισάγεται στο γονέα u του v (μπορεί να δημιουργηθεί νέα ρίζα ή υπερχείλιση)
- Παράδειγμα: διάσπαση που προκαλεί νέα ρίζα



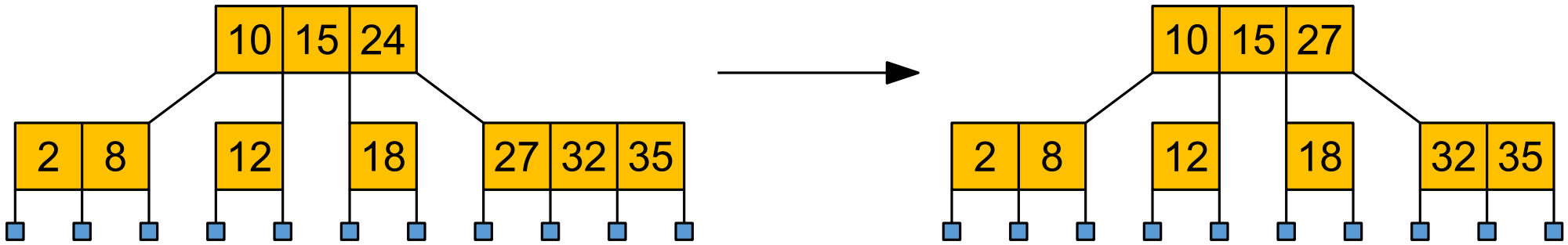
Αλγόριθμος εισαγωγής σε (2,4)-δένδρο

```
1 Algorithm insert(T, (k, o))
2     Search for k in T to locate the insertion node v;
3     Add a new entry (k, o) at node v;
4     while overflow(v) do
5         if v.isRoot() then
6             Create a new empty root above v;
7         v ← split(v);
```

- Πολυπλοκότητα: Γραμμική ως προς το ύψος του (2,4)-δένδρου $\Rightarrow O(\log n)$
  - Η αναζήτηση του κλειδιού k γίνεται σε $O(\log n)$ χρόνο γιατί εμπλέκει $O(\log n)$ κόμβους
  - Η προσθήκη του ζεύγους (k, o) γίνεται σε $O(1)$ χρόνο
  - Οι διαχείριση των υπερχειλίσεων γίνεται σε $O(\log n)$ χρόνο γιατί κάθε διάσπαση γίνεται σε $O(1)$ χρόνο και πραγματοποιούνται $O(\log n)$ διασπάσεις

Διαγραφή σε (2,4)-δένδρο

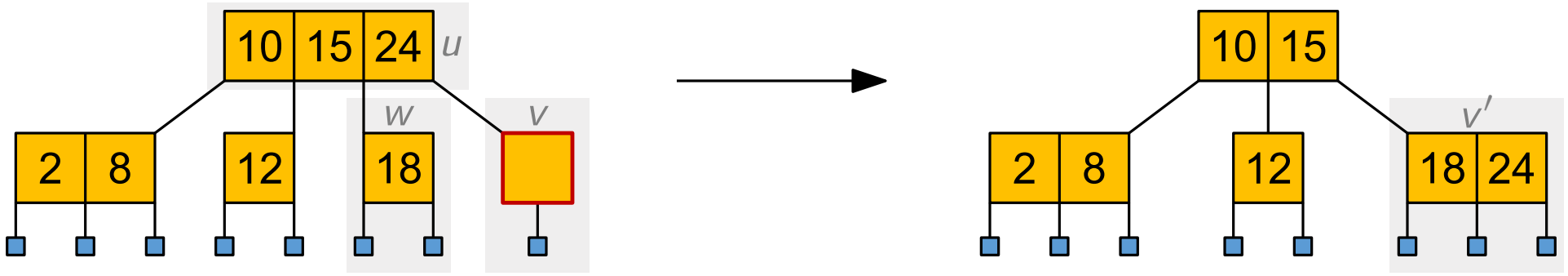
- Επικεντρώνουμε στη διαδικασία διαγραφής όταν το στοιχείο προς διαγραφή είναι σε κόμβο με παιδιά φύλλα
- Αλλιώς, αντικαθιστούμε το στοιχείο προς διαγραφή με το εν σειρά διάδοχο του (ή αντίστοιχα με το εν σειρά προκάτοχό του) και διαγράφουμε το δεύτερο στοιχείο
- Παράδειγμα: Η διαγραφή του κλειδιού 24 γίνεται με αντικατάσταση του με το εν σειρά διάδοχο του κλειδί 27



Υποχείλιση και σύντηξη

Underflow και fusion

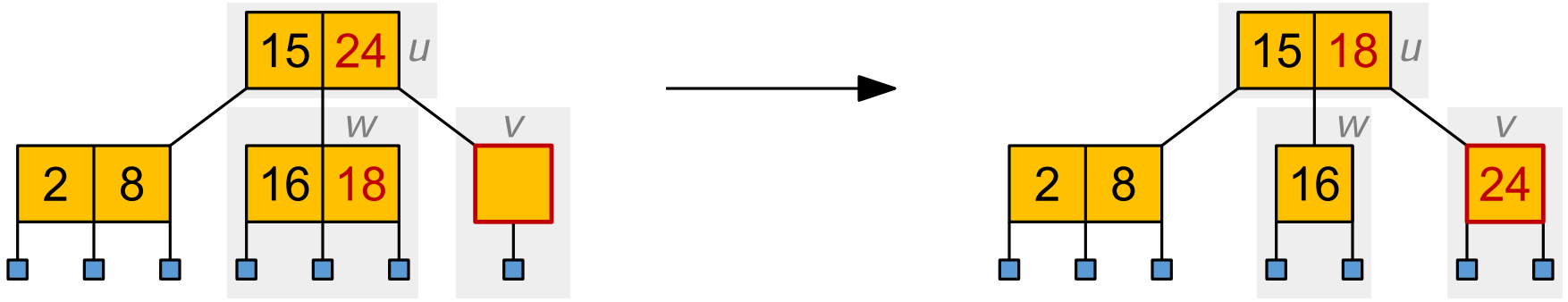
- Η διαγραφή ενός στοιχείου από ένα κόμβο `v` μπορεί να οδηγήσει σε υποχείλιση (ο κόμβος `v` μπορεί να γίνει 1-κόμβος). Αυτή αντιμετωπίζεται θεωρώντας δύο περιπτώσεις.
- Περίπτωση 1: Τα γειτονικά αδέρφια του `v` είναι 2-κόμβοι
  - Σύντηξη: 1. συγχωνεύουμε σε νέο κόμβο `v'` τον κόμβο `v` με έναν γειτονικό του αδελφό `w`
    2. μετακινούμε ένα στοιχείο από τον γονέα `u` του κόμβου `v` στον κόμβο `v'`
  - Μετά την σύντηξη, η υποχείλιση μπορεί να μεταφερθεί στον γονέα `u` του κόμβου `v`
- Παράδειγμα:



Υποχείλιση και μεταφορά

Underflow και transfer

- Η διαγραφή ενός στοιχείου από ένα κόμβο v μπορεί να οδηγήσει σε υποχείλιση (ο κόμβος v μπορεί να γίνει 1-κόμβος). Αυτή αντιμετωπίζεται θεωρώντας δύο περιπτώσεις.
- Περίπτωση 2: Ένας γειτονικός αδερφός w του κόμβου v είναι 3- ή 4-κόμβος
  - Μεταφορά: 1. μετακινούμε ένα παιδί του w στον v
    2. μετακινούμε ένα στοιχείο του u στον v και ένα στοιχείο του w στον u
  - Μετά την μεταφορά, δεν παρουσιάζεται υποχείλιση
- Παράδειγμα:



Διαγραφή σε (2,4)-δένδρο

- Έστω T ένα (2,4)-δέντρο με n στοιχεία
  - Το δέντρο T έχει $O(\log n)$ ύψος
- Κατά την διαγραφή ενός στοιχείου από το δένδρο T:
  - Για τον εντοπισμό του κόμβου απ'όπου θα διαγραφεί το στοιχείο, επισκεπτόμαστε $O(\log n)$ κόμβους
  - Αντιμετωπίσουμε μια υποχείλιση με μια σειρά από $O(\log n)$ συντήξεις, ακολουθούμενες από το πολύ μια μεταφορά
  - Κάθε σύντηξη και μεταφορά γίνεται σε $O(1)$ χρόνο
- Άρα, η διαγραφή ενός στοιχείου από ένα (2,4)-δέντρο γίνεται σε $O(\log n)$ χρόνο

ΑΤΔ: Ουρά Προτεραιότητας

	Unsorted list	Sorted list	Heap	Binary Search Tree		AVL / (2,4) Tree
				worst case	on average	
`empty()`	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
`size()`	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
`insert(e)`	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(\log n)$	$\mathcal{O}(n)$	$\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$
`removeMin()`	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(\log n)$	$\mathcal{O}(n)$	$\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$
`min()`	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$

Επιπλέον υλικό

- Ενότητες 3.1, 4.1, 4.2, 20.2:
  Michael T. Goodrich, Roberto Tamassia, Αλγόριθμοι σχεδίαση και εφαρμογές
  ISBN: 9789605126971, εκδόσεις Γκιούρδα, 2016.