

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ - ΤΜΗΥΠ

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ Ι

Β. Μεγαλοοικονόμου
Δ. Χριστοδουλάκης

Δομές Ευρετηρίων και Κατακερματισμός Αρχείων Ι

(παρουσίαση βασισμένη εν μέρη σε σημειώσεις των Silberchatz, Korth και Sudarshan και του C. Faloutsos)



Σύνοψη Ύλης

- **Σχεσιακό μοντέλο** (Relational model) - **SQL**
 - Επίσημες (Formal) & Εμπορικές γλώσσες ερωτήσεων (commercial query languages)
- **Συναρτησιακές Εξαρτήσεις** (Functional Dependencies)
- **Κανονικοποίηση** (Normalization)
- **Φυσικός Σχεδιασμός** (Physical Design)
- **Ευρετηριοποίηση** (Indexing)



Ευρετηριοποίηση (Indexing)- Περίληπτικά

- **Πρωτεύοντα / δευτερεύοντα ευρετήρια**
- Σειριακή μέθοδος προσπέλασης (ISAM)
- B - trees, B+ - trees
- Κατακερματισμός
 - Στατικός κατακερματισμός
 - Δυναμικός κατακερματισμός

Βασική ιδέα

- Οι ευρετηριοποιήσεις επιταχύνουν την διαδικασία ανάκτησης των επιθυμητών δεδομένων
 - Π.χ., κατάλογος συγγραφέων των βιβλίων μιας βιβλιοθήκης
- **Κλειδί αναζήτησης (Search Key)** – γνώρισμα ή σύνολο γνωρισμάτων που χρησιμοποιείται για την εύρεση εγγραφών στο αρχείο
- Ένα **αρχείο ευρετηρίου (index file)** αποτελείται από εγγραφές (**καταχωρήσεις ευρετηρίου - index entries**) της **μορφής**

Κλειδί αναζήτησης	Δείκτης
-------------------	---------
- Τα αρχεία ευρετηρίου είναι συνήθως μικρότερα σε μέγεθος από τα αρχεία δεδομένων
- Δύο βασικά είδη ευρετηρίων :
 - **Ταξινομημένα ευρετήρια (Ordered indices):** οι εγγραφές του κλειδιού αναζήτησης αποθηκεύονται ταξινομημένα
 - **Ευρετήρια Κατακερματισμού (Hash indices):** οι εγγραφές των κλειδιών αναζήτησης διανέμονται ομοιόμορφα στους «κάδους» (buckets) με χρήση κάποιας συνάρτησης κατακερματισμού.

Ευρετηριοποίηση

- Πως “ψάχνουμε” αποδοτικά όταν οι εγγραφές είναι αποθηκευμένες σε ένα αρχείο; (Π.χ., AM=123?)

Φοιτητής		
<u>AM</u>	Όνομα	Διεύθυνση
123	Σταύρου	Αιόλου
234	Αντωνίου	Θράκης
125	tomson	main str



Ευρετηριοποίηση

- Πως “ψάχνουμε” αποδοτικά εφόσον οι εγγραφές είναι αποθηκευμένες σε ένα αρχείο;
- **brute force**: ανέκτησε όλες τις εγγραφές, επέστρεψε μόνο αυτές που πληρούν το κριτήριο αναζήτησης
- **..καλύτερα**: χρησιμοποίησε δείκτες για τον απευθείας εντοπισμό των εγγραφών

Ευρετηριοποίηση – Βασική ιδέα:

123
125
234

Φοιτητής		
<u>Ssn</u>	Name	Address
123	smith	main str
234	jones	forbes ave
125	tomson	main str



Μέτρηση της καταλληλότητας μιας μεθόδου ευρετηριοποίησης

- Χρόνος ανάκτησης;
- Εισαγωγής / Διαγραφής;
- Επιβάρυνση χώρου;
- Αναδιοργάνωση;
- Ερωτήματα διαστήματος (range queries);



Βασική Ιδέα

- Τα κλειδιά αναζήτησης αποθηκεύονται στο αρχείο ευρετηρίου και δείχνουν στα πραγματικά δεδομένα
- Πρωτεύοντα ή Δευτερεύοντα ευρετήρια (indices)
- Συσταδοποίηση (Clustering) (αραιά sparse) ή ευρετήρια χωρίς συσταδοποίηση non-clustering (πυκνά - dense) ευρετήρια

Ευρετηριοποίηση

Ευρετήριο Πρωτεύοντος κλειδιού :

- Κλειδί αναζήτησης το πρωτεύον κλειδί
- Δεν περιέχει διπλότυπες εγγραφές

123
234
345
456
567

STUDENT		
<u>Ssn</u>	Name	Address
123	smith	main str
234	jones	forbes ave
678	tomson	main str
456	stevens	forbes ave
345	smith	forbes ave

Ευρετηριοποίηση

Ευρετήριο Δευτερεύοντος κλειδιού :

- Μπορεί να περιέχει διπλότυπες εγγραφές

forbes ave
main str

Διευθύνσεις-
ευρετήριο

STUDENT		
<u>Ssn</u>	Name	Address
123	smith	main str
234	jones	forbes ave
345	tomson	main str
456	stevens	forbes ave
567	smith	forbes ave

Indexing

Ευρετήριο Δευτερεύοντος κλειδιού : ΤΥΠΙΚά
με **'postings lists'**

Postings lists

forbes ave
main str



STUDENT

<u>Ssn</u>	Name	Address
123	smith	main str
234	jones	forbes ave
345	tomson	main str
456	stevens	forbes ave
567	smith	forbes ave



Βασική Ιδέα (συνέχεια)

- Αραιό ευρετήριο:
 - οι εγγραφές είναι διατεταγμένες στο αρχείο ως προς το κλειδί (δεν συμπεριλαμβάνονται στο ευρετήριο όλες οι τιμές του πεδίου ευρετηριοποίησης)
- Πυκνό ευρετήριο:
 - το αντίθετο
- Για παράδειγμα:...

Ευρετηριοποίηση – Αραιά ευρετήρια

αραιό ευρετήριο στο γνώρισμα AM (ssn)

123
456
...

≥ 123

≥ 456

STUDENT			
<u>Ssn</u>	Name	Address	
123	smith	main str	
234	jones	forbes ave	
345	tomson	main str	
456	stevens	forbes ave	
567	smith	forbes ave	



Sparse Index Files

- **Αραιό ευρετήριο:** περιέχει εγγραφές ευρετηρίου μόνο για μερικές τιμές κλειδιών
 - Κατάλληλο για εγγραφές σειριακά ταξινομημένες με βάση το κλειδί αναζήτησης
- **Για τον εντοπισμό μιας εγγραφής :** με το κλειδί αναζήτησης να έχει τιμή K :
 - Βρες την εγγραφή του ευρετηρίου με την μεγαλύτερη τιμή κλειδιού αναζήτησης η οποία είναι μικρότερη από την τιμή K
 - Αναζήτησε το αρχείο ξεκινώντας από την εγγραφή στην οποία δείχνει η εγγραφή του ευρετηρίου που μας επιστράφηκε στο προηγούμενο βήμα
- Λιγότερος χώρος και κόστος συντήρησης για εισαγωγές και διαγραφές
- Στην γενική περίπτωση σχετικά πιο αργή διαδικασία, για τον εντοπισμό εγγραφών, από τα πυκνά ευρετήρια
- **Καλό tradeoff:** αραιά ευρετήρια με μια εγγραφή ευρετηρίου για κάθε block σε ένα αρχείο αντιστοιχεί σε λιγότερες τιμές κλειδιών σε ένα block

Ευρετηριοποίηση – Πυκνό ευρετήριο

Πυκνό ευρετήριο

123
234
345
456
567

<u>Ssn</u>	Name	Address
345	tomson	main str
234	jones	forbes ave
567	smith	forbes ave
456	stevens	forbes ave
123	smith	main str



Σύνοψη

- Όλοι συνδυασμοί είναι δυνατοί...

	Πυκνό	Αραιό
Πρωτεύον		Σύνηθες
Δευτερέων	Σύνηθες	Σπάνιο

- **Αραιό ευρετήριο** : το πολύ ένα
- **Πυκνά ευρετήρια** : όσα επιθυμούμε

Συνήθως:

- Ένα πρωτεύον ευρετήριο στο πρωτεύον κλειδί (ίσως αραιό)
- Μερικά δευτερεύοντα ευρετήρια, σε κάποια απο τα υπόλοιπα γνωρίσματα (πυκνά ευρετήρια)



Ευρετηριοποίηση – Περιληπτικά

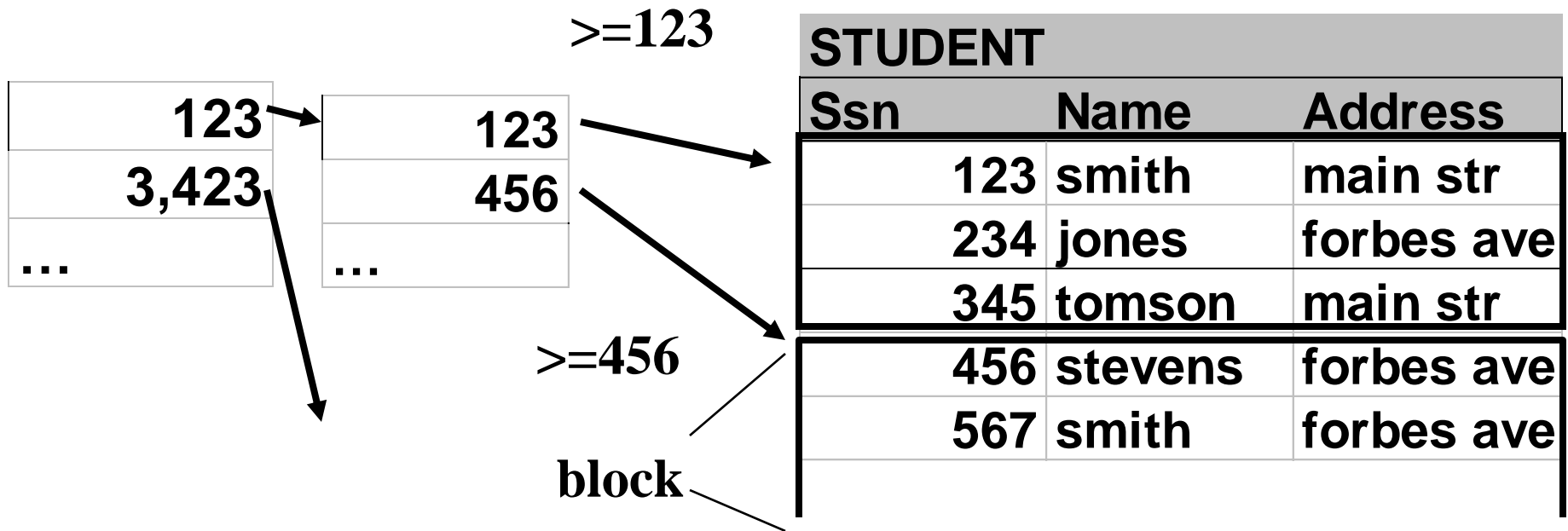
- Πρωτεύοντα / δευτερεύοντα ευρετήριο
- **Σειριακή μέθοδος προσπέλασης (ISAM)**
- B - trees, B+ - trees
- Κατακερματισμός
 - Στατικός κατακερματισμός
 - Δυναμικός κατακερματισμός



ISAM

- Τι θα συμβεί εάν το ευρετήριο είναι υπερβολικά μεγάλο για ψάξουμε σ'αυτό σειριακά;
 - Θα χρησιμοποιήσουμε ένα πολυεπίπεδο ευρετήριο...

ISAM



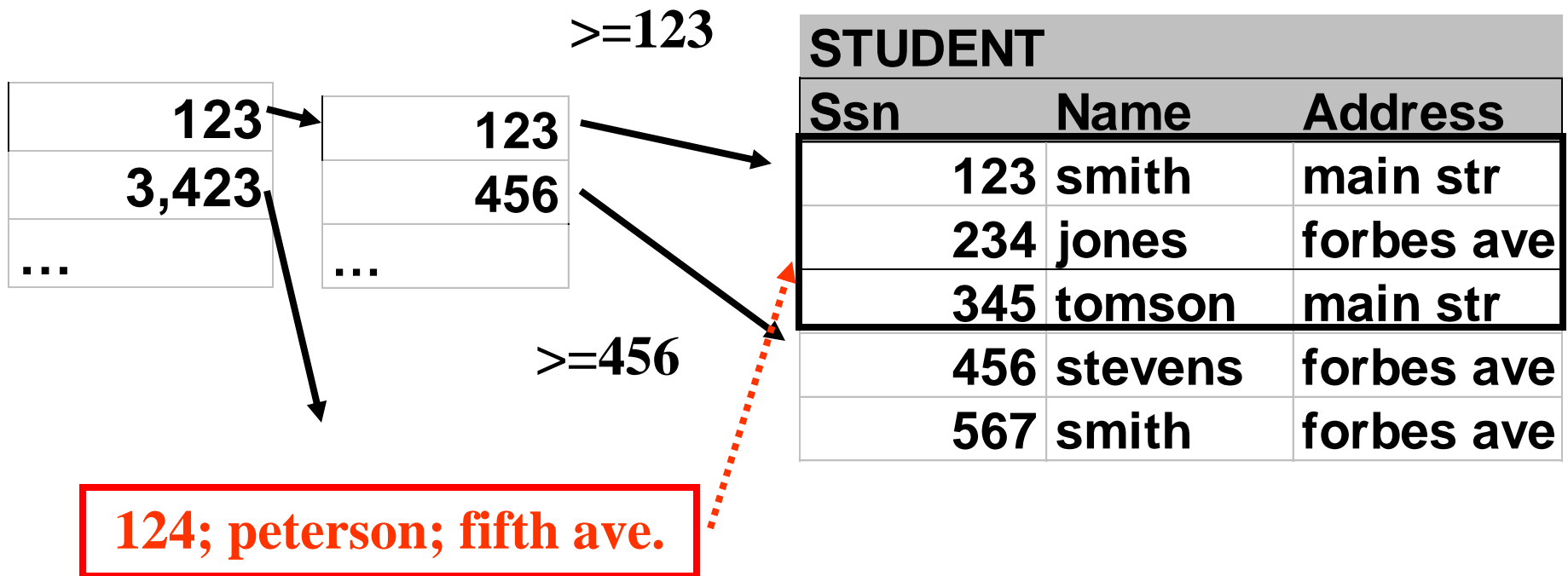


ISAM - Παρατηρήσεις

- Εάν το ευρετήριο είναι πολύ μεγάλο
 - το αποθηκεύουμε στον δίσκο
 - διατηρούμε ευρετήριο για το ευρετήριο στην μνήμη
- Συνήθως δύο επίπεδα δεικτών
 - το πρώτο επίπεδο θα περιέχει τις πρώτες εγγραφές κάθε block του δίσκου (γιατί;)

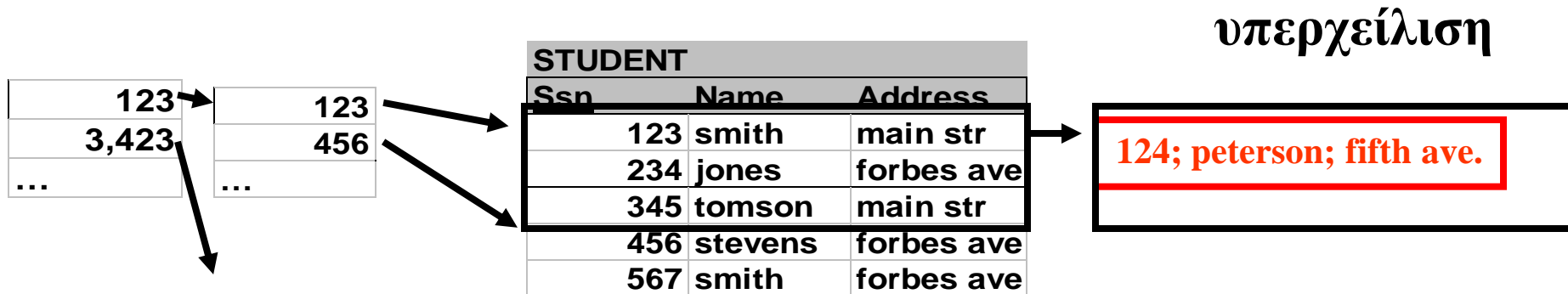
ISAM - Παρατηρήσεις

- Τι γίνεται με τις εισαγωγές / διαγραφές;



ISAM - Παρατηρήσεις

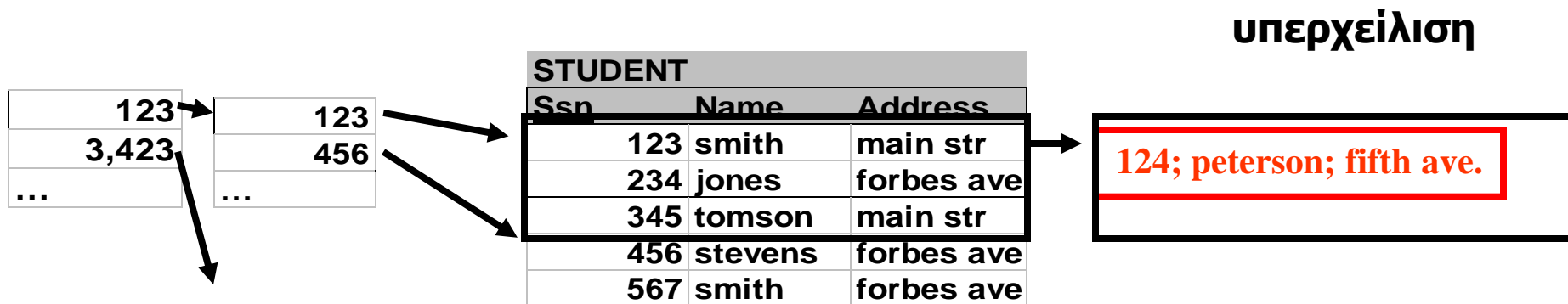
- Τι γίνεται με τις εισαγωγές / διαγραφές;



Πρόβλημα;

ISAM - Παρατηρήσεις

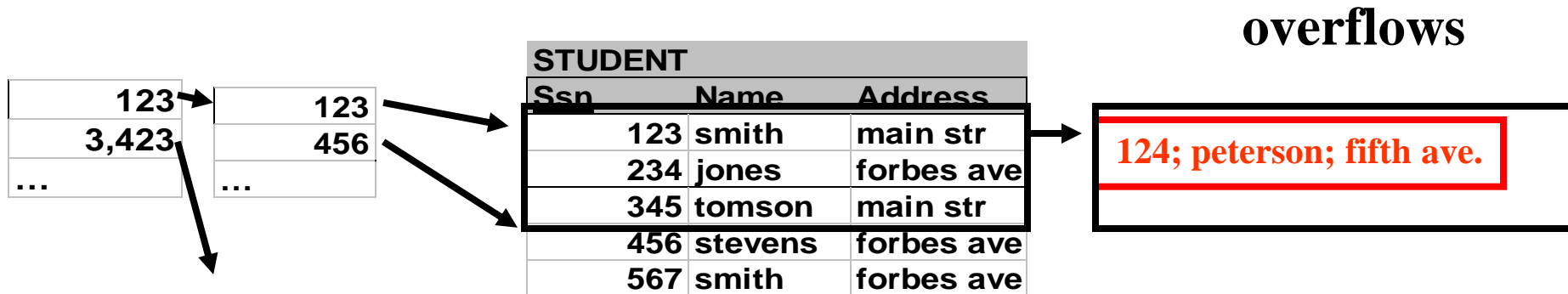
- Τι γίνεται με τις εισαγωγές / διαγραφές;



- Οι αλυσίδες υπερχείλισης μπορεί να μεγαλώσουν υπερβολικά – **Τι μπορούμε να κάνουμε;**

ISAM - Παρατηρήσεις

- Τι γίνεται με τις εισαγωγές / διαγραφές;



- Οι αλυσίδες υπερχείλισης μπορεί να μεγαλώσουν υπερβολικά - Έτσι:

- σταμάτα την διαδικασία & αναδιοργάνωσε τις εγγραφές
- άρχισε όταν η χρησιμοποίηση είναι στο ~80%



Μέχρι τώρα

- ... δείκτες (όπως ISAM) πάσχουν όταν υπάρχει συχνή αναγκαιότητα για ανανεώσεις δεδομένων
- Μία σειριακή σάρωση δεδομένων σε ένα πρωτεύον ευρετήριο είναι αποδοτική, αλλά μία σειριακή σάρωση σε ένα δευτερεύον ευρετήριο είναι ακριβή
 - Κάθε προσπάθεια ανάκτησης εγγραφής μπορεί να απαιτήσει την ανάκτηση ενός block από τον δίσκο
- Εναλλακτικές δομές ευρετηρίου : **B - trees**



Ευρετηριοποίηση (Indexing)- Περίληπτικά

- Πρωτεύοντα / δευτερεύοντα ευρετήρια
- Σειριακή μέθοδος προσπέλασης (ISAM)
- **B - trees, B+ - trees**
- Κατακερματισμός
 - Στατικός κατακερματισμός
 - Δυναμικός κατακερματισμός



B-trees

- Η οικογένεια των **πιο χρήσιμων** σχημάτων ευρετηρίων (B-trees, B⁺-trees, B^{*}-trees)
- Μπορούν να χρησιμοποιηθούν για την υλοποίηση
 - Πρωτευόντων/ δευτερευόντων ευρετηρίων
 - Πυκνών / αραιών ευρετηρίων
- Είναι ισοζυγισμένα (**balanced**) “n-way” δέντρα αναζήτησης



B-trees

Μειονεκτήματα ευρετηριοποιημένων σειριακών αρχείων (ISAM):

- Η απόδοση μειώνεται καθώς το αρχείο μεγαλώνει
- Δημιουργία πολλών Block υπερχείλισης
- Απαιτείται περιοδική αναδιοργάνωση ολόκληρου του αρχείου

Πλεονεκτήματα των B⁺-tree ευρετηρίων αρχείων:

- Αυτόματη αναδιοργάνωση με μικρές τοπικές αλλαγές μετά από εισαγωγές και διαγραφές
- Δεν απαιτείται αναδιοργάνωση ολόκληρου του αρχείου

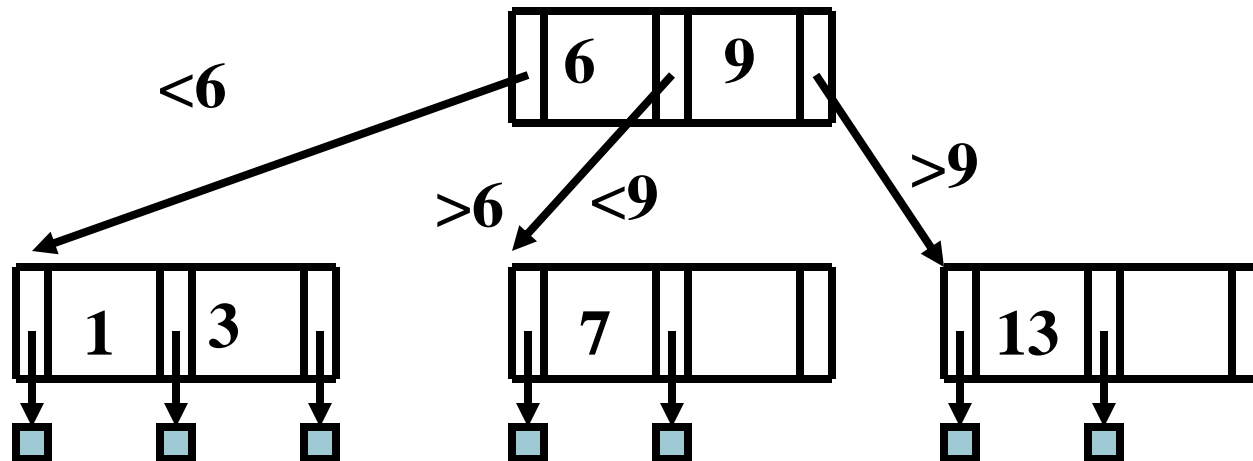
Μειονεκτήματα των B⁺-tree ευρετηρίων αρχείων:

- Επιπλέον επιβάρυνση χώρου και χρόνου για εισαγωγές και διαγραφές

Τα πλεονεκτήματα των B⁺trees υπερκεράζουν τα μειονεκτήματά τους και χρησιμοποιούνται ευρέως

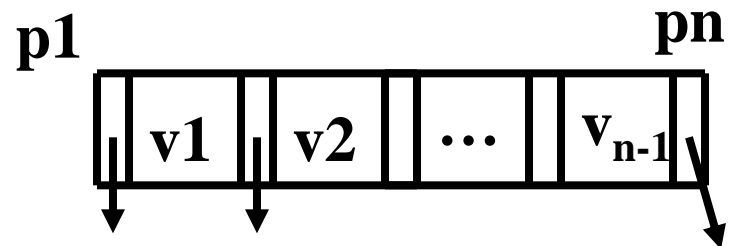
B-trees

Π.χ., B-tree τάξης 3 (δηλ: το πολύ 3 δείκτες από κάθε κόμβο):



B-tree ιδιότητες:

- Κάθε κόμβος, σε ένα B-tree τάξης n :
 - Ταξινομημένος με βάση το κλειδί
 - Το πολύ n δείκτες
 - Το ελάχιστο $n/2$ δείκτες (εκτός από την ρίζα)
 - Όλα τα φύλλα στο ίδιο επίπεδο
 - Εάν ο αριθμός των δεικτών είναι k τότε ο κάθε κόμβος έχει ακριβώς $k-1$ κλειδιά



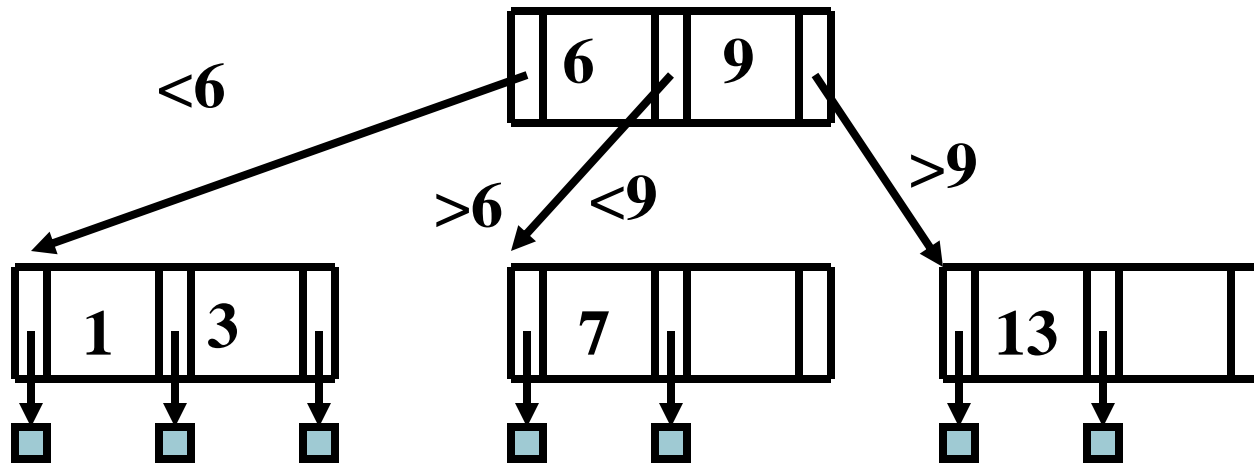


Ιδιότητες

- Κόμβοι “block aware” : κάθε κόμβος → σελίδα δίσκου
- $O(\log(N))$ για τα πάντα! (ins/del/search)
- Τυπικά, εάν $N = 50 - 100$, τότε 2 – 3 επίπεδα
- Χρησιμοποίηση $\geq 50\%$, εγγυημένα, κατά μέσο όρο 69%

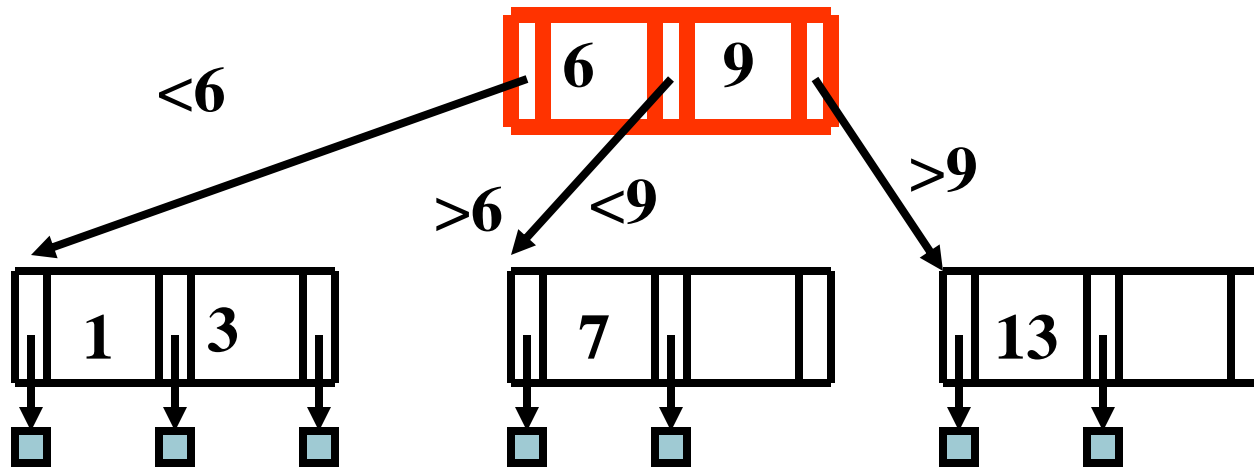
Ερωτήματα- Queries

- Αλγόριθμος για ερώτημα ακριβούς ταιριάσματος (exact match query);
(Π.χ., ssn=8?)



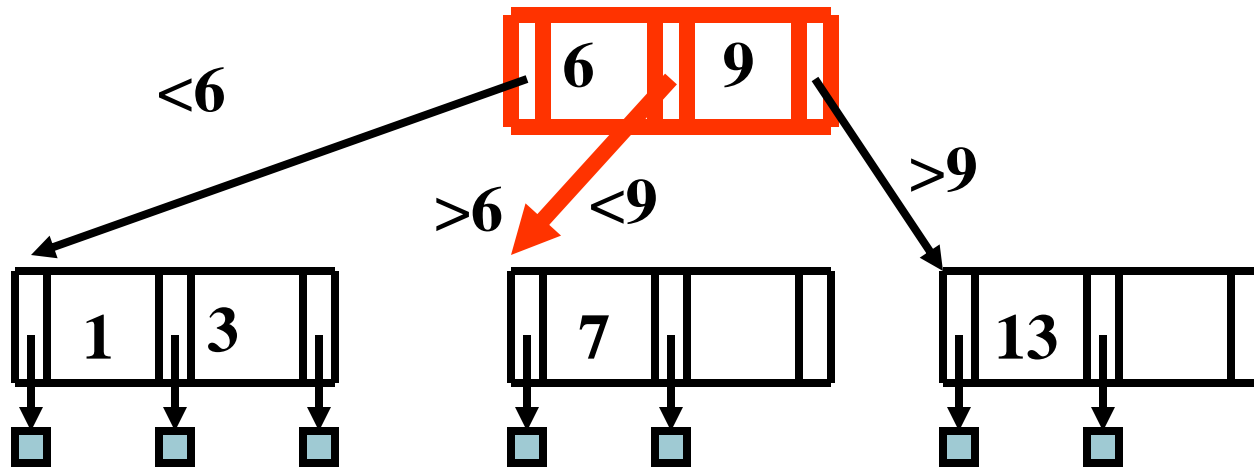
Ερωτήματα- Queries

- Αλγόριθμος για ερώτημα ακριβούς ταιριάσματος (exact match query);
(Π.χ., ssn=8?)



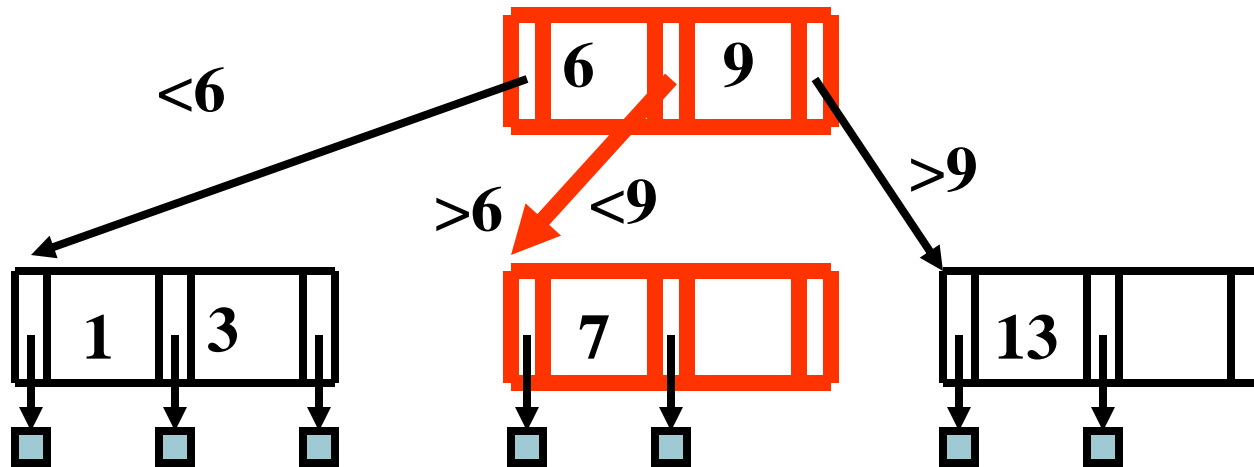
Ερωτήματα- Queries

- Αλγόριθμος για ερώτημα ακριβούς ταιριάσματος (exact match query);
(Π.χ., ssn=8?)



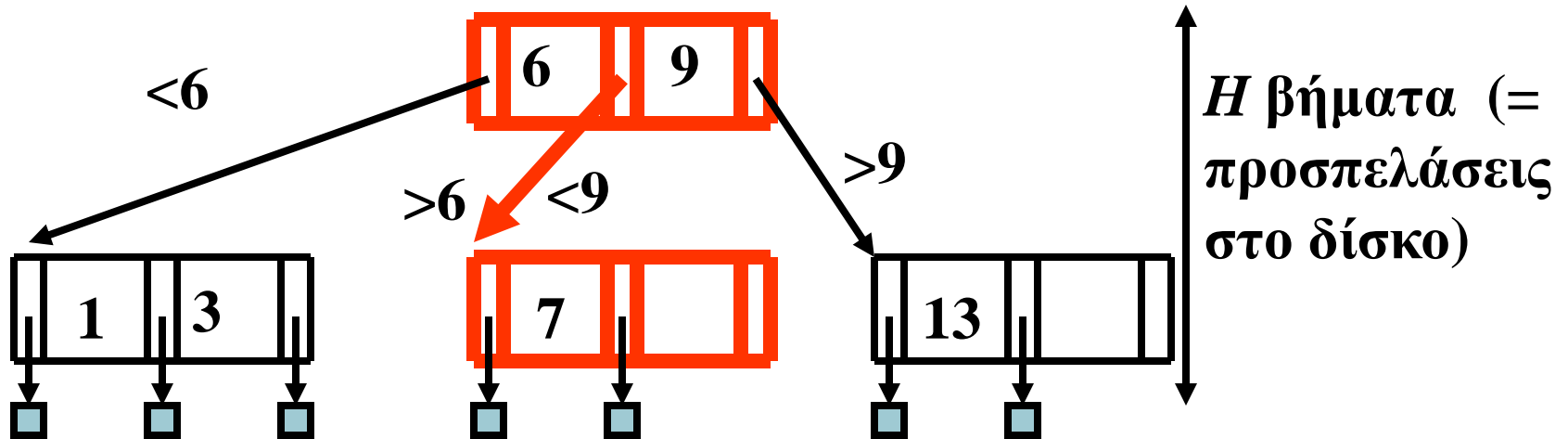
Ερωτήματα- Queries

- Αλγόριθμος για ερώτημα ακριβούς ταιριάσματος (exact match query);
(Π.χ., ssn=8?)



Queries

- Αλγόριθμος για ερώτημα ακριβούς ταιριάσματος (exact match query);
(Π.χ., ssn=8?)



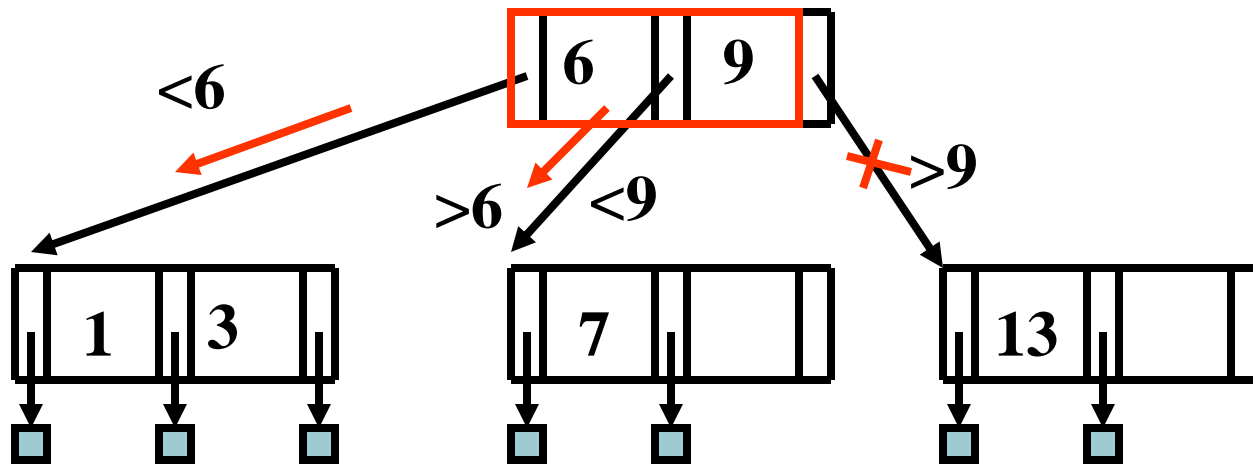


Ερωτήματα- Queries

- Τι γίνεται για ερωτήματα διαστήματος (range queries); (Π.χ., $5 < salary < 8$)
- Εγγύτητα/ αναζητήσεις κοντινότερου γείτονα; (Π.χ., $salary \sim 8$)

Ερωτήματα- Queries

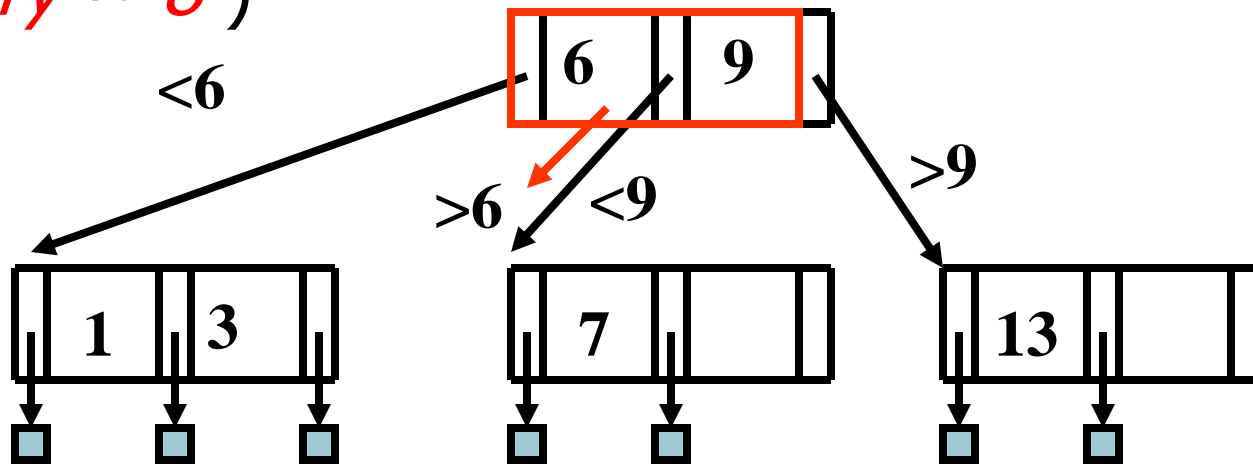
- Τι γίνεται για ερωτήματα διαστήματος (Π.χ., $5 < salary < 8$)



Ερωτήματα- Queries

- Τι γίνεται για ερωτήματα διαστήματος ;(Π.χ., $5 < salary < 8$)

- Εγγύτητα/ αναζητήσεις κοντινότερου γείτονα; (Π.χ., $salary \sim 8$)



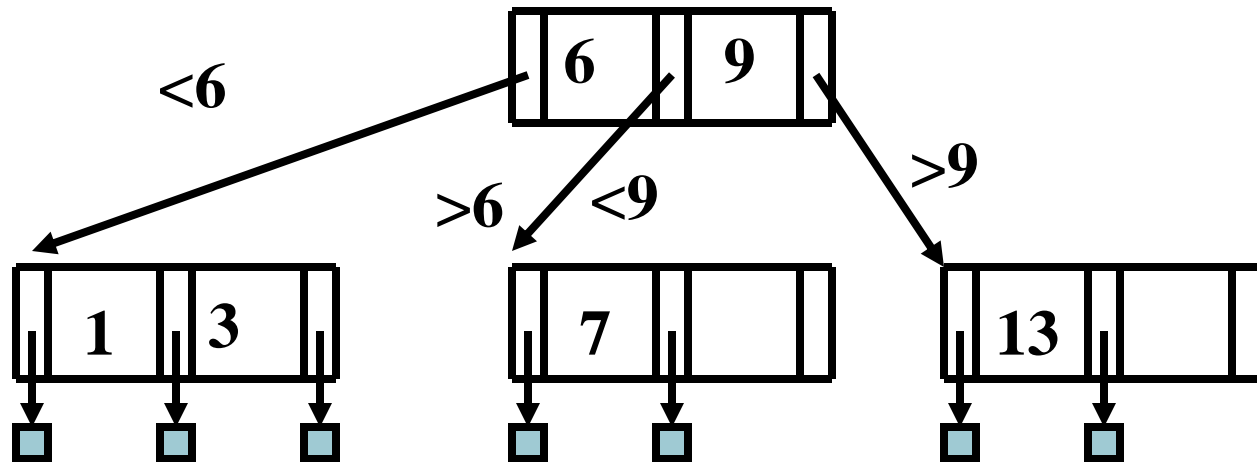


B-trees: Εισαγωγές

- Εισαγωγή σε φύλλο (leaf)
σε περίπτωση υπερχείλισης προώθησε τον μέσο προς τα πάνω (αναδρομικά)
- Διέσπασε: ώστε να διατηρούνται οι ιδιότητες ενός B-tree

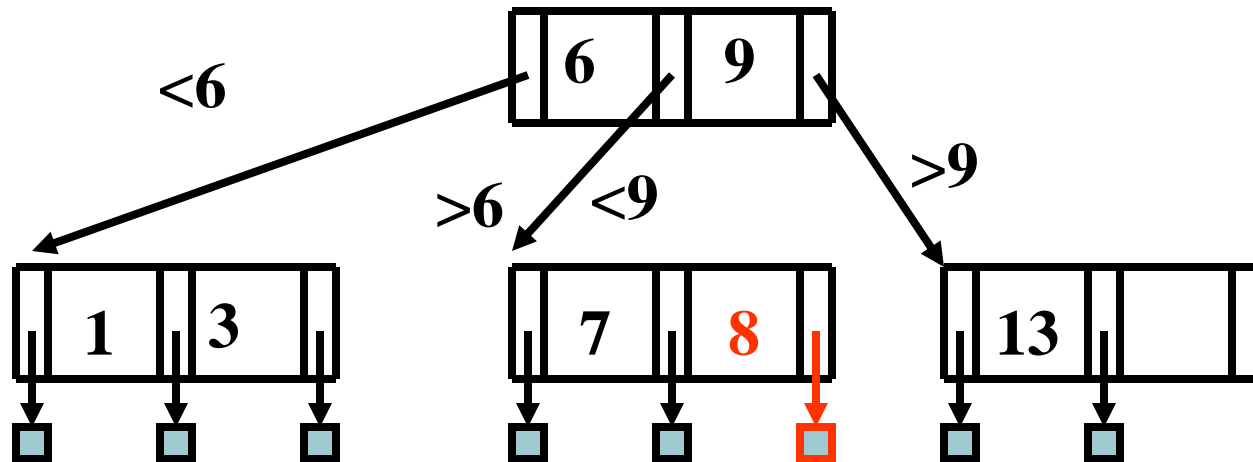
B-trees

Εύκολη περίπτωση: Tree T0; Εισαγωγή του '8'



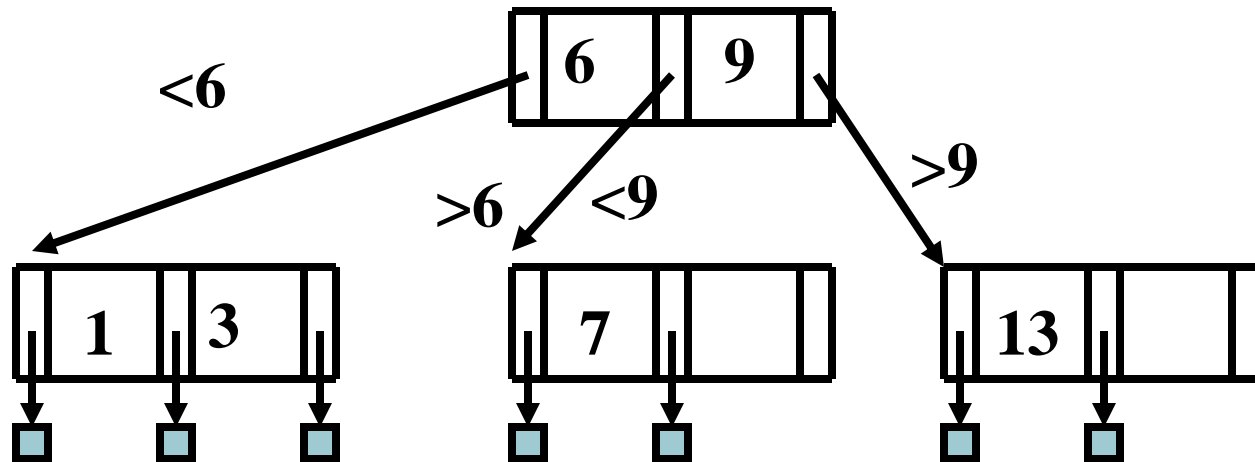
B-trees

Tree T0; Εισαγωγή του '8'



B-trees

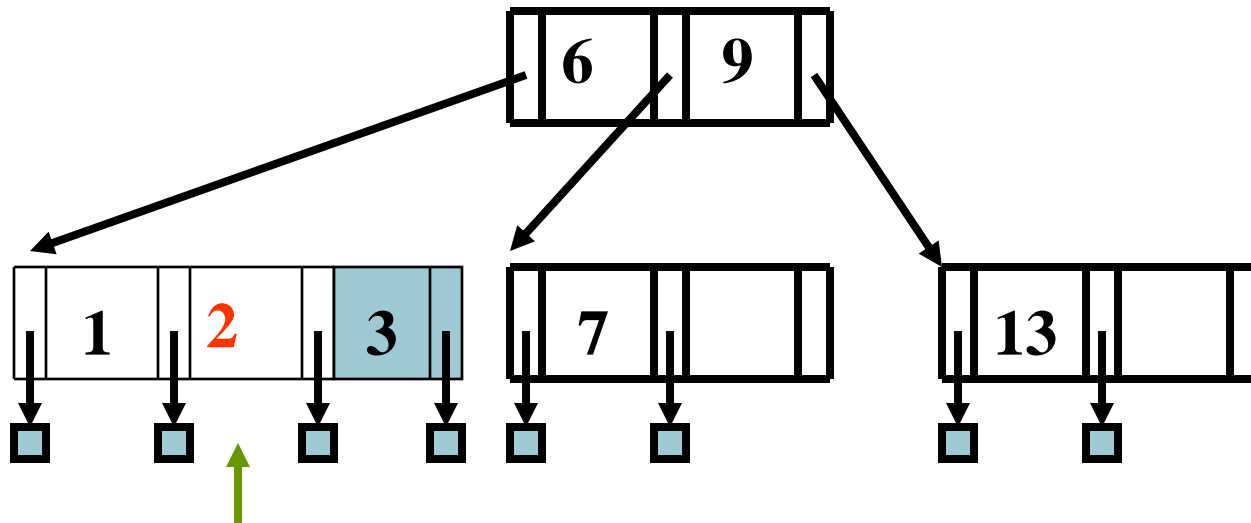
Η δυσκολότερη περίπτωση : Tree T0; Εισαγωγή του '2'



2

B-trees

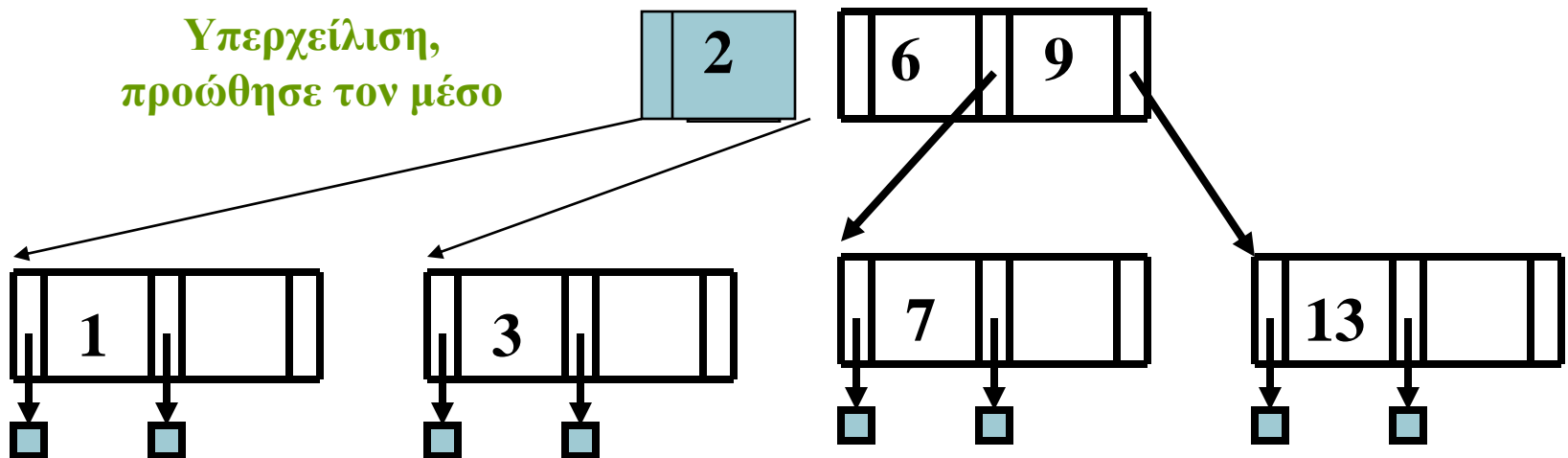
Η δυσκολότερη περίπτωση : Tree T0; Εισαγωγή του '2'



Πρώθησε τον μέσο
προς τα πάνω

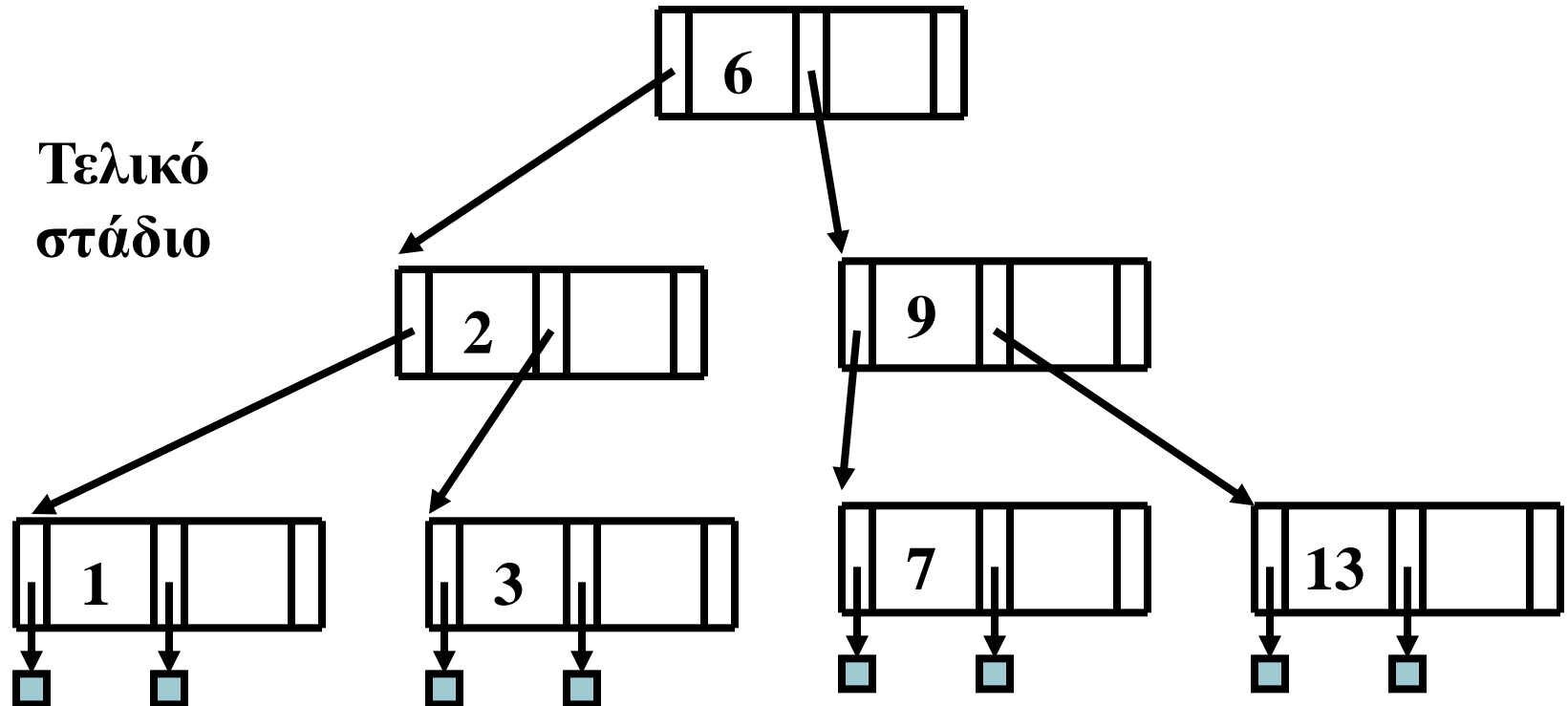
B-trees

Η δυσκολότερη περίπτωση : Tree T0; Εισαγωγή του '2'



B-trees

Η δυσκολότερη περίπτωση : Tree T0; Εισαγωγή του '2'





B-trees - Εισαγωγή

- Ερώτηση: τι θα γίνει εάν υπάρχουν 2 μέσοι; (π.χ., δέντρο τάξης 4)
- Απάντηση: λειτουργεί οτιδήποτε από τα δύο



B-trees - Εισαγωγή

- Εισαγωγή σε φύλλο - σε υπερχείλιση, προώθησε το μέσο πάνω (αναδρομικά – ‘διέδωσε την διάσπαση’)
- **Διάσπαση**: για την διατήρηση όλων των ιδιοτήτων ενός B-tree (!!)
- Παρατηρήστε πως αναπτύσσεται: το ύψος αυξάνεται όταν υπάρξει υπερχείλιση και διασπαστεί η ρίζα
- **Αυτόματη** αύξηση και αναδιοργάνωση (συγκριτικά με ISAM!)

Ψευδοκώδικας

```
INSERTION OF KEY 'K'
```

```
    find the correct leaf node 'L';
```

```
    if ( 'L' overflows ){
```

```
        split 'L', by pushing the middle key upstairs to  
parent node 'P';
```

```
        if ('P' overflows){
```

```
            repeat the split recursively;
```

```
        }
```

```
    else{
```

```
        add the key 'K' in node 'L'; /* Διατήρησε την  
διάταξη των κλειδιών στον κόμβο 'L' */
```

```
    }
```



Ευρετηριοποίηση (Indexing)- Περίληπτικά

- Πρωτεύοντα / δευτερεύοντα ευρετήρια
- Σειριακή μέθοδος προσπέλασης (ISAM)
- B - trees,
 - Ορισμός, αναζήτηση, εισαγωγή, **διαγραφή**
- B+ - trees
- Κατακερματισμός
 - Στατικός κατακερματισμός
 - Δυναμικός κατακερματισμός



Διαγραφή

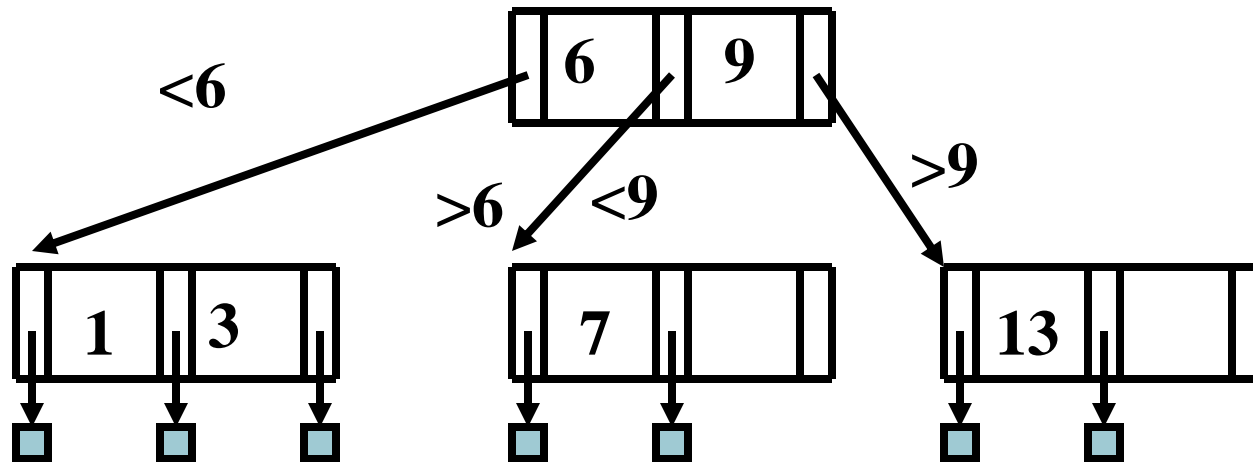
Ο αλγόριθμος συνοπτικά :

- Διέγραψε κλειδί
- Σε περίπτωση **υπερχείλισης** μπορεί να προκληθεί **συγχώνευση**

Στην πράξη κάποιοι σχεδιαστές απλά αφήνουν να συμβεί υπερχειλίση ...

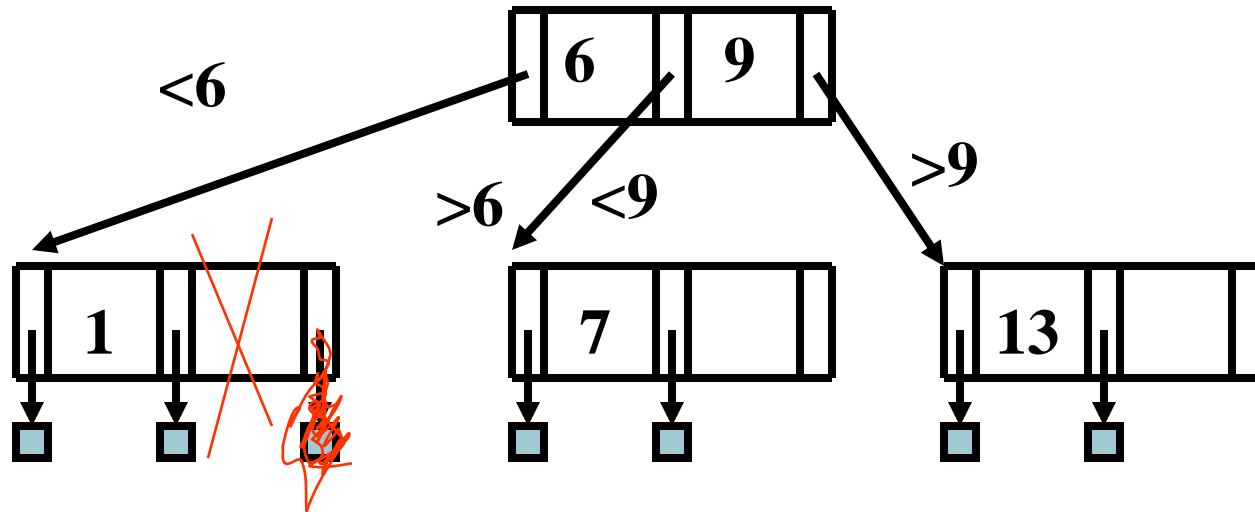
B-trees – Διαγραφή

Η ευκολότερη περίπτωση: Tree T0; Διαγραφή του '3'



B-trees – Διαγραφή

Η ευκολότερη περίπτωση: Tree T0; Διαγραφή του '3'



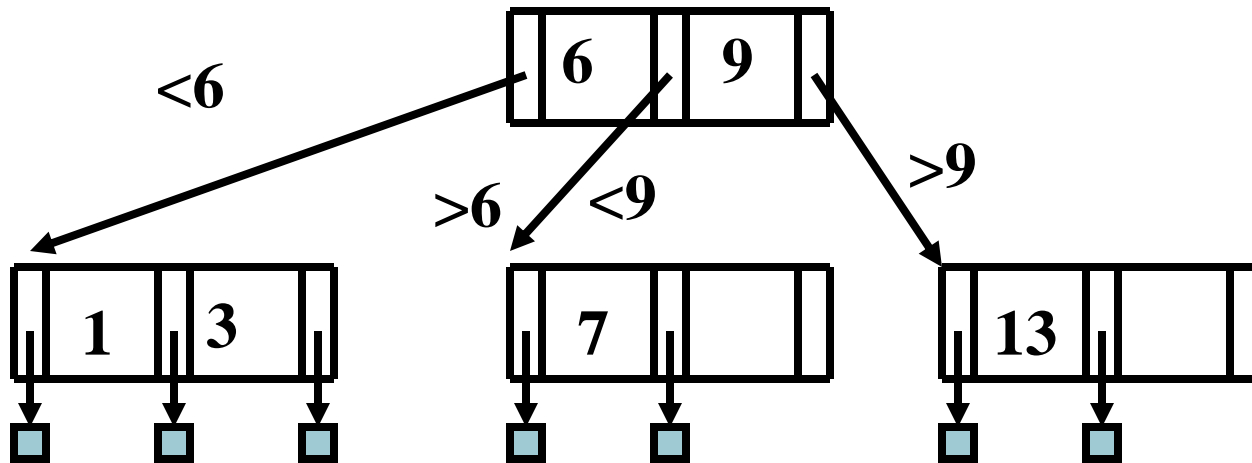


B-trees – Διαγραφή

- ➔ **1^η Περίπτωση:** διαγραφή κλειδιού από κόμβο φύλο - χωρίς υπερχείλιση
- **2^η Περίπτωση:** διαγραφή κλειδιού από εσωτερικό κόμβο - χωρίς υπερχείλιση
- **3^η Περίπτωση:** διαγραφή κλειδιού από κόμβο φύλο - υποχείλιση και “πλούσιος” γειτονικός κόμβος (συγγενής)
- **4^η Περίπτωση:** διαγραφή κλειδιού από κόμβο φύλο - υποχείλιση και “φτωχός” γειτονικός κόμβος (συγγενής)

B-trees – Διαγραφή

- **1^η Περίπτωση:** διαγραφή κλειδιού από κόμβο φύλο - χωρίς υπερχειλίση
- Π.χ. Διέγραψε το **3** από το δέντρο from T0



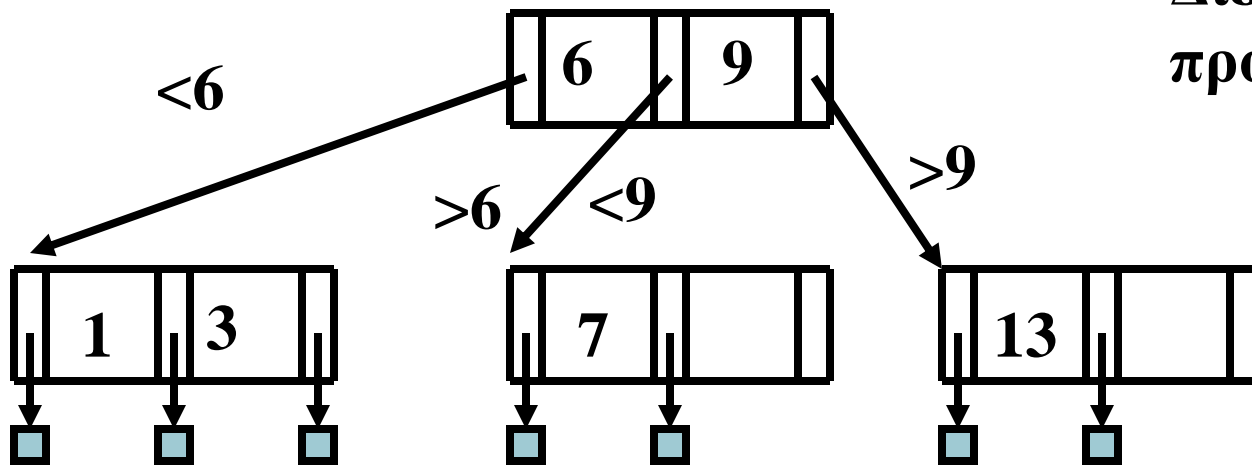


B-trees – Διαγραφή

- **1^η Περίπτωση:** διαγραφή κλειδιού από κόμβο φύλο - χωρίς υπερχείλιση
- ⇒ ■ **2^η Περίπτωση:** διαγραφή κλειδιού από εσωτερικό κόμβο - χωρίς υπερχείλιση
- **3^η Περίπτωση:** διαγραφή κλειδιού από κόμβο φύλο - υποχείλιση και “πλούσιος” γειτονικός κόμβος
- **4^η Περίπτωση:** διαγραφή κλειδιού από κόμβο φύλο - υποχείλιση και “φτωχός” γειτονικός κόμβος

B-trees – Διαγραφή

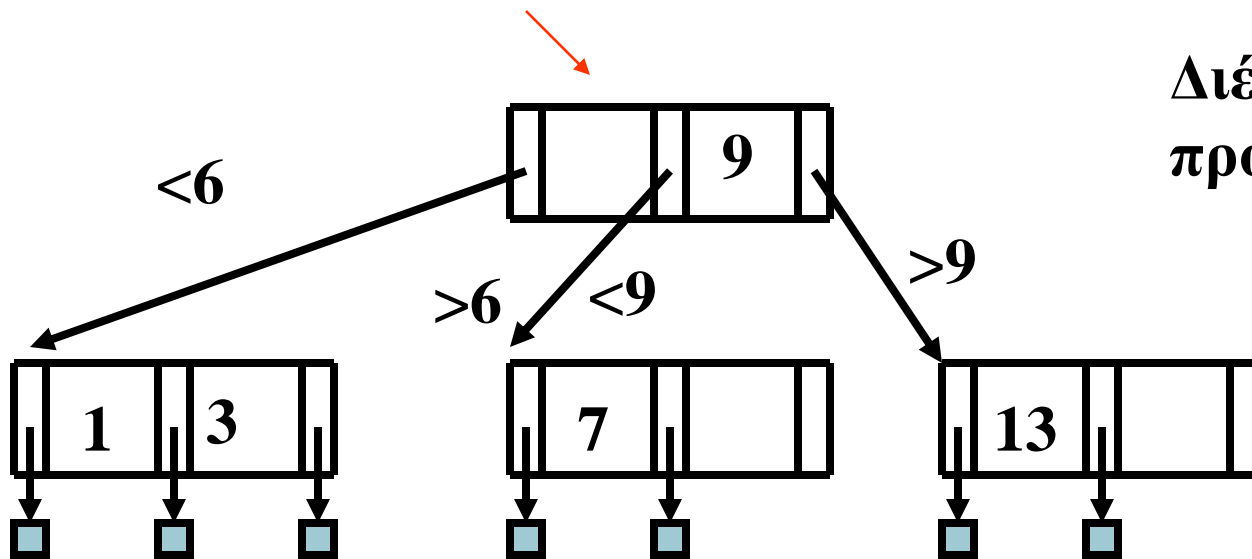
- **2^η Περίπτωση:** διαγραφή κλειδιού από εσωτερικό κόμβο - χωρίς υπερχείλιση
- (Π.χ. , διέγραψε το **6** από το T0)



Διέγραψε & προώθησε:

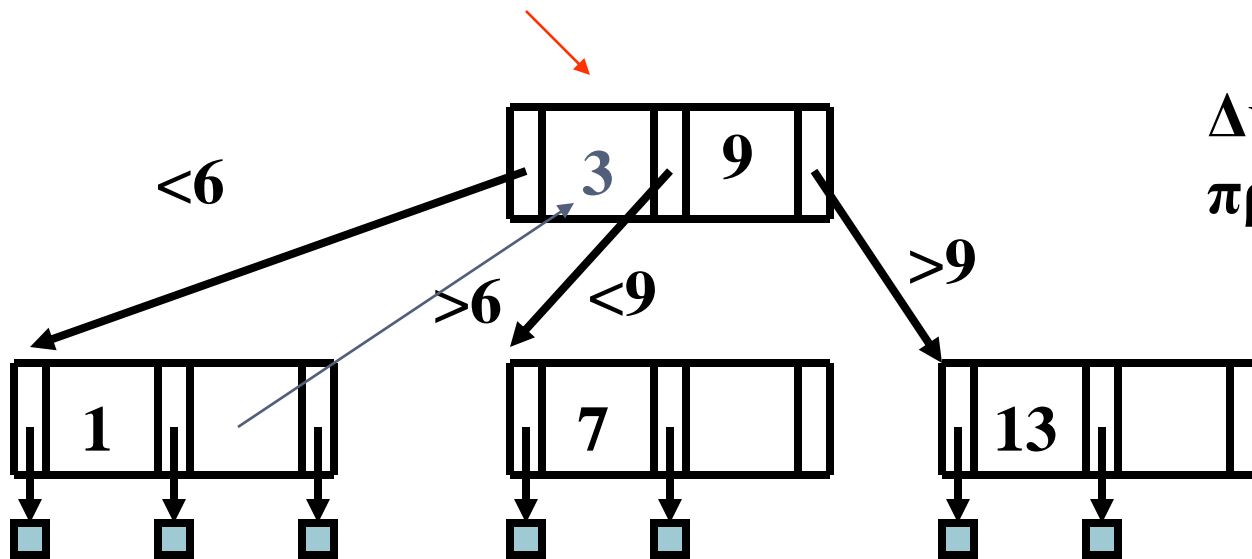
B-trees – Διαγραφή

- **2^η Περίπτωση:** διαγραφή κλειδιού από εσωτερικό κόμβο - χωρίς υπερχειλίση
- (Π.χ. , διέγραψε το **6** από το T0)



B-trees – Διαγραφή

- **2^η Περίπτωση:** διαγραφή κλειδιού από εσωτερικό κόμβο - χωρίς υπερχειλίση
- (Π.χ. , διέγραψε το **6** από το T0)

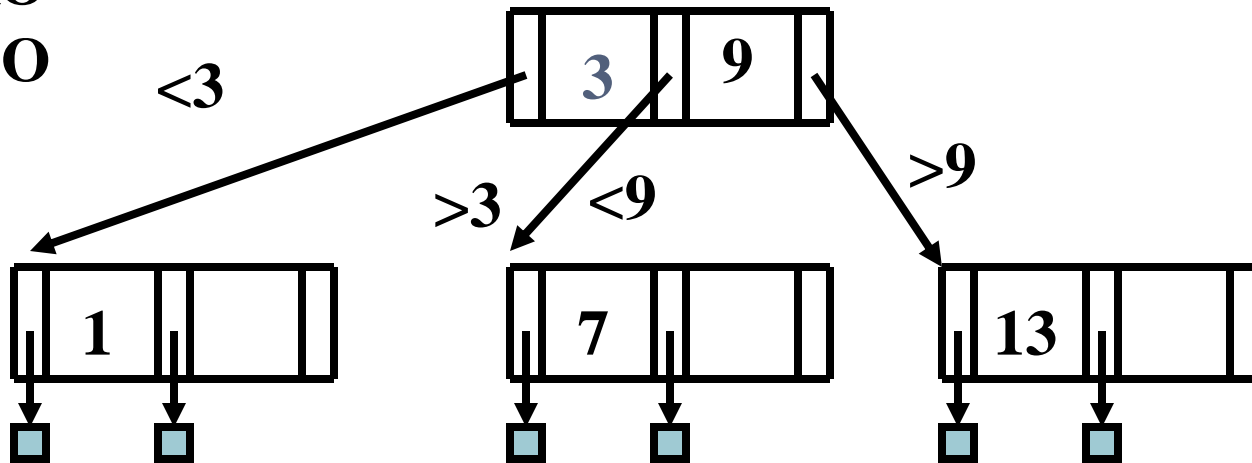


Διέγραψε &
προώθησε:

B-trees – Διαγραφή

- **2^η Περίπτωση:** διαγραφή κλειδιού από εσωτερικό κόμβο - χωρίς υπερχειλίση
- (Π.χ. , διέγραψε το **6** από το T0)

**ΤΕΛΙΚΟ
ΔΕΝΤΡΟ**





B-trees – Διαγραφή

- **2^η Περίπτωση:** διαγραφή κλειδιού από εσωτερικό κόμβο
 - χωρίς υπερχείλιση
 - (Π.χ. , διέγραψε το 6 από το T0)
- Ερώτηση: Πως προωθούμε?
- Απάντηση: επέλεξε το μεγαλύτερο κλειδί από το αριστερότερο υποδέντρο (ή το μικρότερο από δεξιό υπόδεντρο)
- Παρατήρηση:
Κάθε διαγραφή οδηγεί τελικά σε διαγραφή κλειδιού από κάποιο κόμβο φύλλο

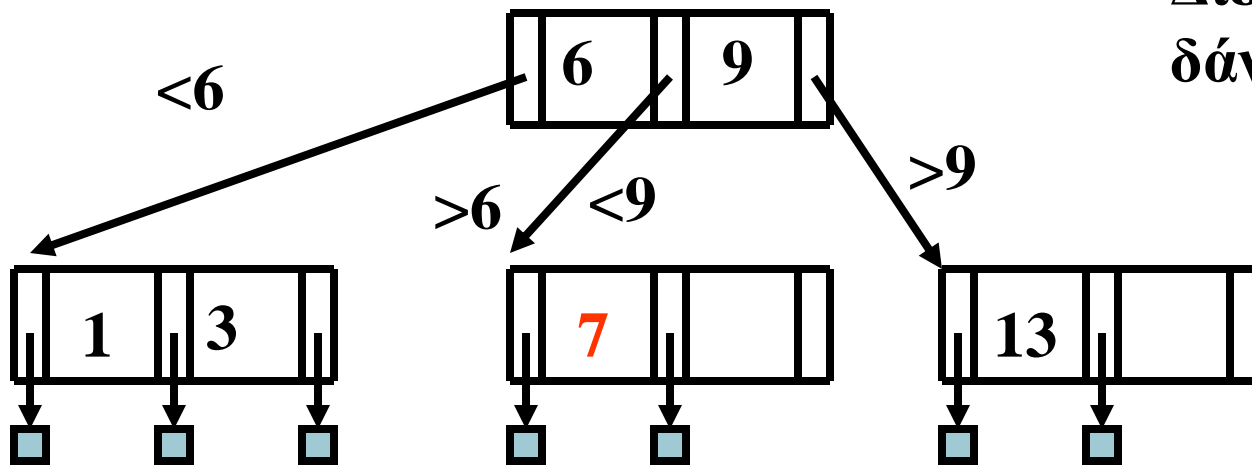


B-trees – Διαγραφή

- **1^η Περίπτωση:** διαγραφή κλειδιού από κόμβο φύλο - χωρίς υπερχείλιση
- **2^η Περίπτωση:** διαγραφή κλειδιού από εσωτερικό κόμβο - χωρίς υπερχείλιση
- ➔ ■ **3^η Περίπτωση:** διαγραφή κλειδιού από κόμβο φύλο - υποχείλιση και “πλούσιος” γειτονικός κόμβος
- **4^η Περίπτωση:** διαγραφή κλειδιού από κόμβο φύλο - υποχείλιση και “φτωχός” γειτονικός κόμβος

B-trees – Διαγραφή

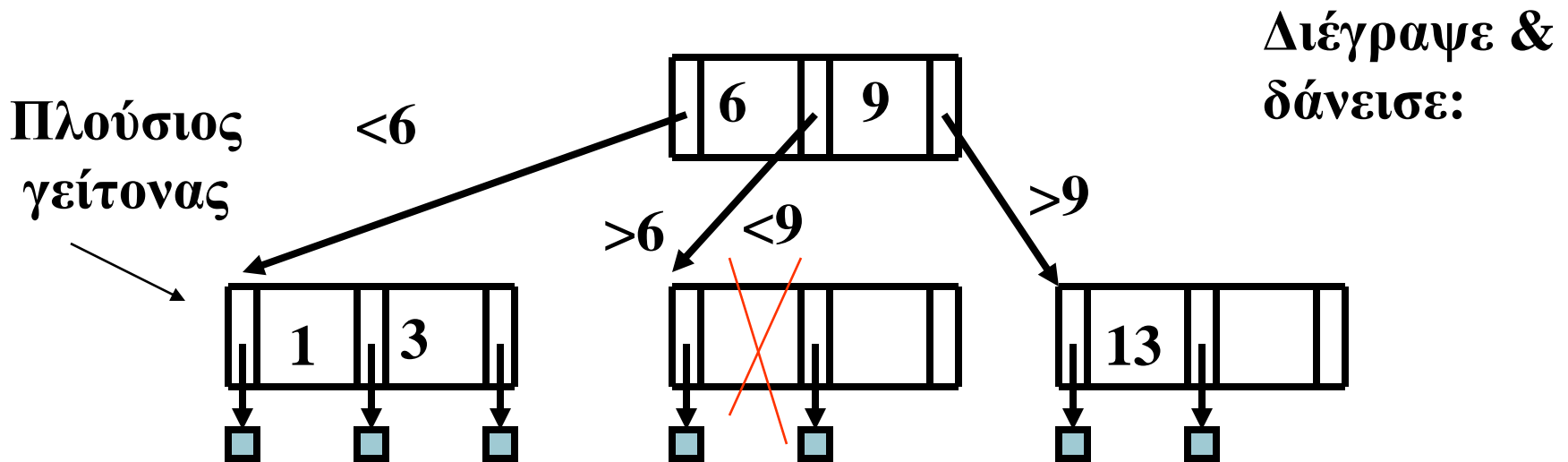
- **3^η Περίπτωση:** διαγραφή κλειδιού από κόμβο φύλο - υποχείλιση και "πλούσιος" γειτονικός κόμβος (Π.χ. , διέγραψε το **7** από το T0)



Διέγραψε &
δάνεισε:

B-trees – Διαγραφή

- **3^η Περίπτωση:** διαγραφή κλειδιού από κόμβο φύλο - υποχείλιση και "πλούσιος" γειτονικός κόμβος
- (Π.χ. , διέγραψε το **7** από το T0)



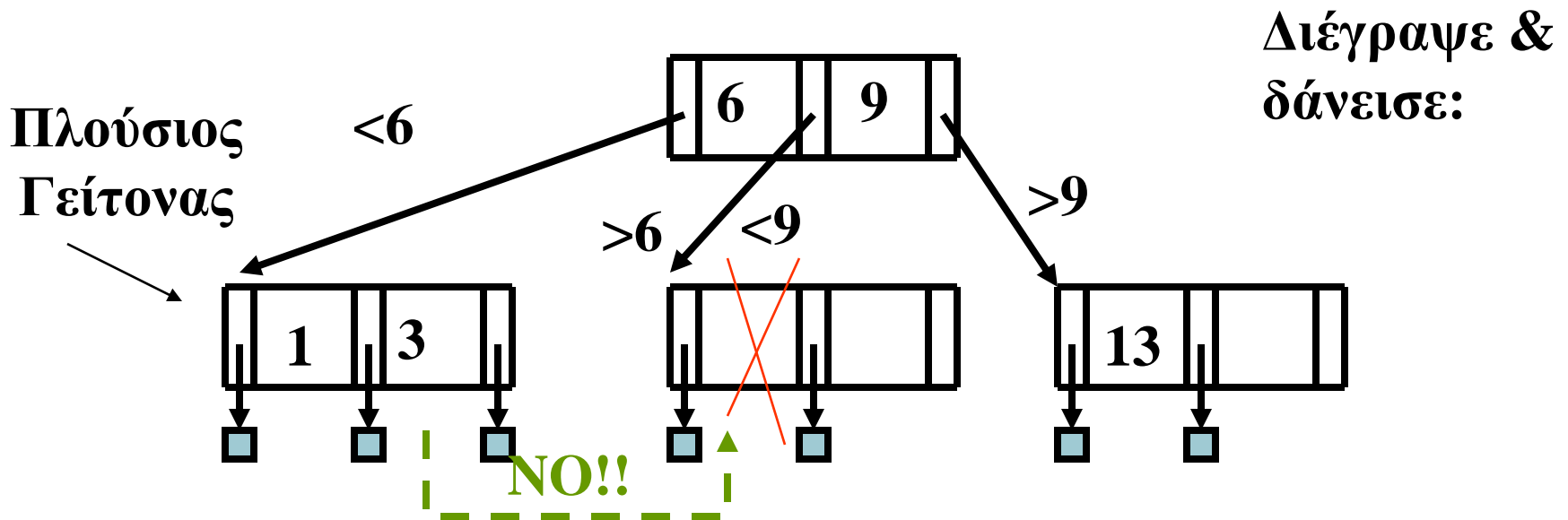


B-trees – Διαγραφή

- **3^η Περίπτωση:** διαγραφή κλειδιού από κόμβο φύλο - υποχείλιση και “πλούσιος” γειτονικός κόμβος
- ‘Πλούσιος γείτονας’ = μπορεί να δώσει ένα κλειδί χωρίς υποχείλιση
- ‘Δανείζει’ ένα κλειδί : πάντα ΔΙΑΜΕΣΟΥ του ΜΗΤΡΙΚΟΥ ΚΟΜΒΟΥ!

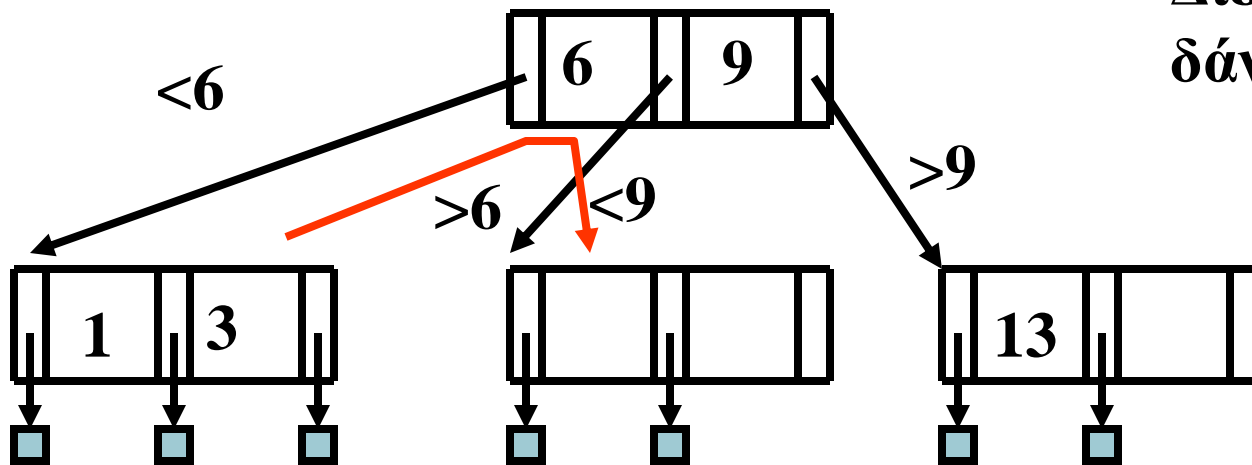
B-trees – Διαγραφή

- **3^η Περίπτωση:** διαγραφή κλειδιού από εσωτερικό κόμβο - υποχείλιση και "πλούσιος" γειτονικός κόμβος (Π.χ. , διέγραψε το **7** από το T0)



B-trees – Διαγραφή

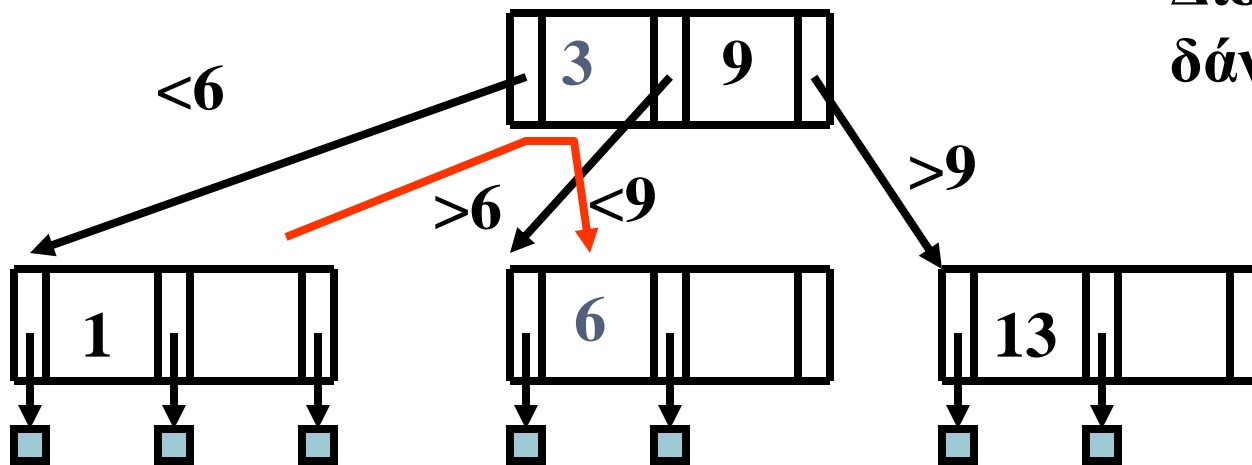
- **3^η Περίπτωση:** διαγραφή κλειδιού από εσωτερικό κόμβο - υποχείλιση και "πλούσιος" γειτονικός κόμβος (Π.χ. , διέγραψε το **7** από το T0)



Διέγραψε &
δάνεισε:

B-trees – Διαγραφή

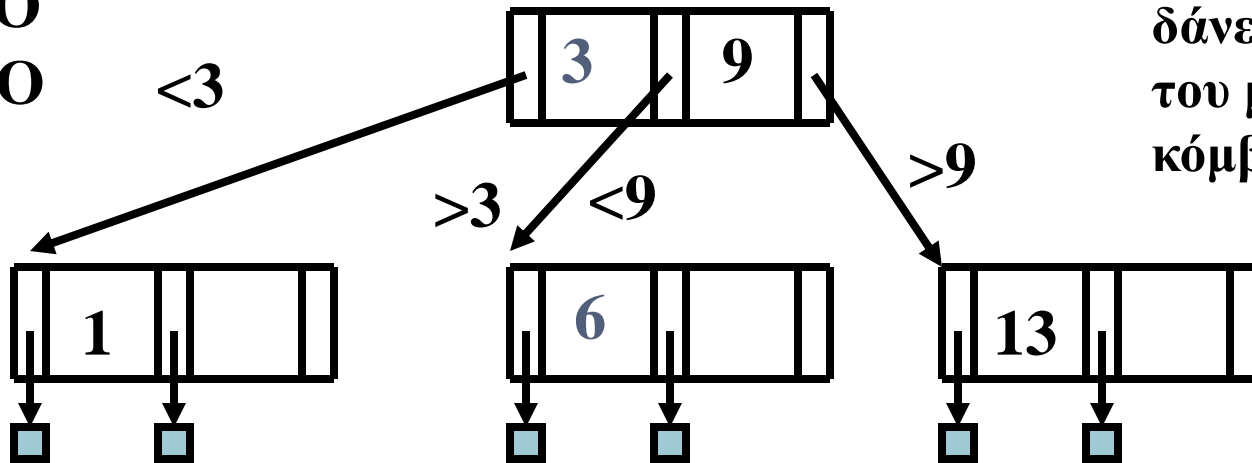
- **3^η Περίπτωση:** διαγραφή κλειδιού από εσωτερικό κόμβο - υπερχείλιση και “πλούσιος” γειτονικός κόμβος (Π.χ. , διέγραψε το 7 από το T0)



B-trees – Διαγραφή

- **3^η Περίπτωση:** διαγραφή κλειδιού από εσωτερικό κόμβο - υποχείλιση και “πλούσιος” γειτονικός κόμβος (Π.χ. , διέγραψε το **7** από το T0)

ΤΕΛΙΚΟ
ΔΕΝΤΡΟ



Διέγραψε &
δάνεισε **διαμέσου**
του μητρικού
κόμβου:

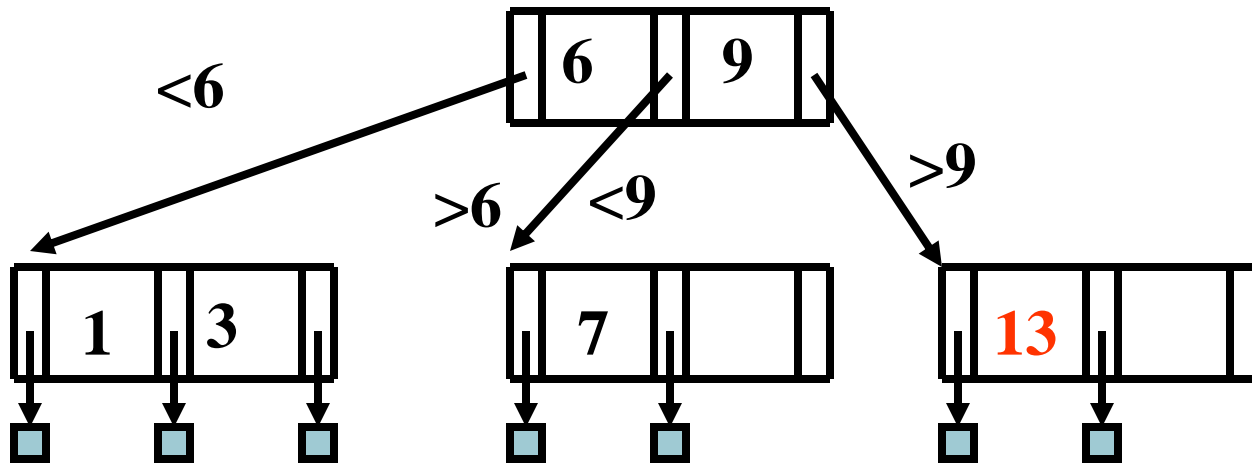


B-trees – Διαγραφή

- **1^η Περίπτωση:** διαγραφή κλειδιού από κόμβο φύλο - χωρίς υπερχείλιση
- **2^η Περίπτωση:** διαγραφή κλειδιού από εσωτερικό κόμβο - χωρίς υπερχείλιση
- **3^η Περίπτωση:** διαγραφή κλειδιού από κόμβο φύλο - υποχείλιση και “πλούσιος” γειτονικός κόμβος
- ⇒ ■ **4^η Περίπτωση:** διαγραφή κλειδιού από κόμβο φύλο - υποχείλιση και “φτωχός” γειτονικός κόμβος

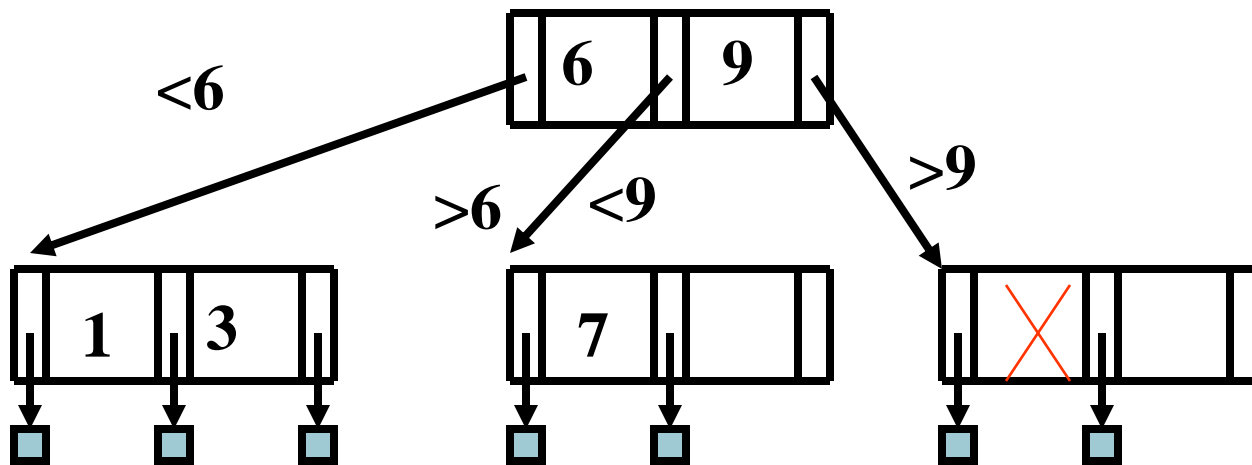
B-trees – Διαγραφή

- **4^η Περίπτωση:** διαγραφή κλειδιού από κόμβο φύλο - υποχείλιση και "φτωχός" γειτονικός κόμβος
- (Π.χ., διέγραψε το **13** από το T0)



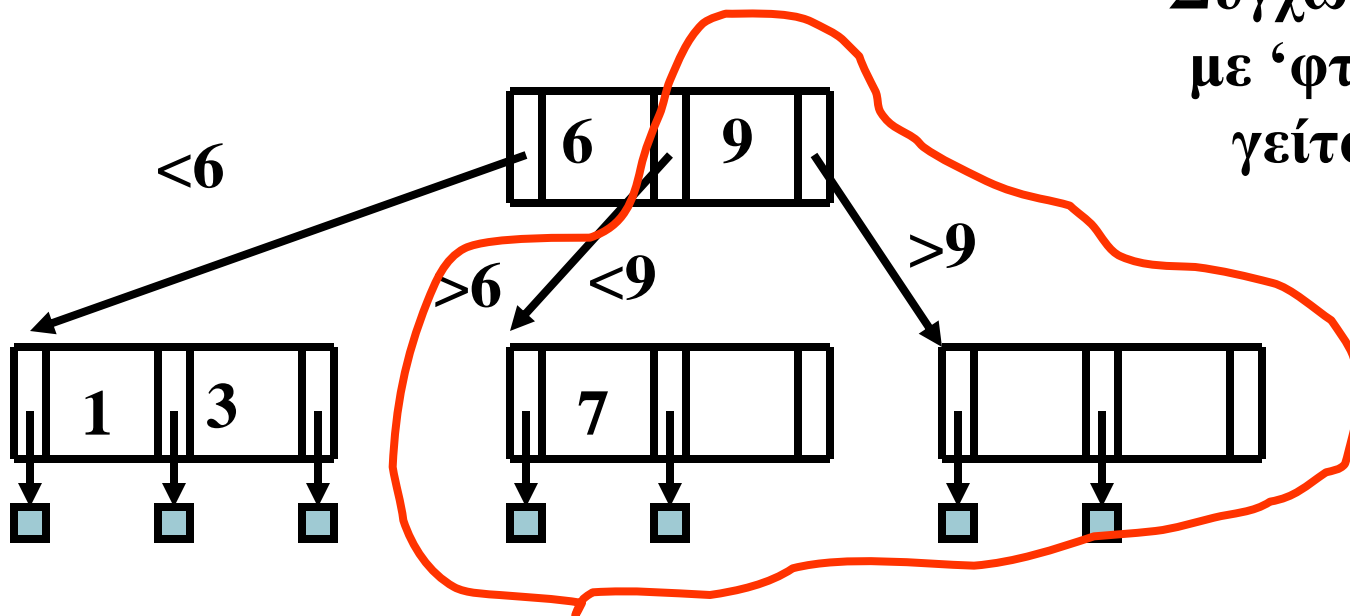
B-trees – Διαγραφή

- **4^η Περίπτωση:** διαγραφή κλειδιού από κόμβο φύλο - υποχείλιση και “φτωχός” γειτονικός κόμβος
- (Π.χ., διέγραψε το **13** από το T0)



B-trees – Διαγραφή

- **4^η Περίπτωση:** διαγραφή κλειδιού από κόμβο φύλο - υποχείλιση και "φτωχός" γειτονικός κόμβος
- (Π.χ., διέγραψε το **13** από το T0)



Συγχώνευσε
με 'φτωχό'
γείτονα

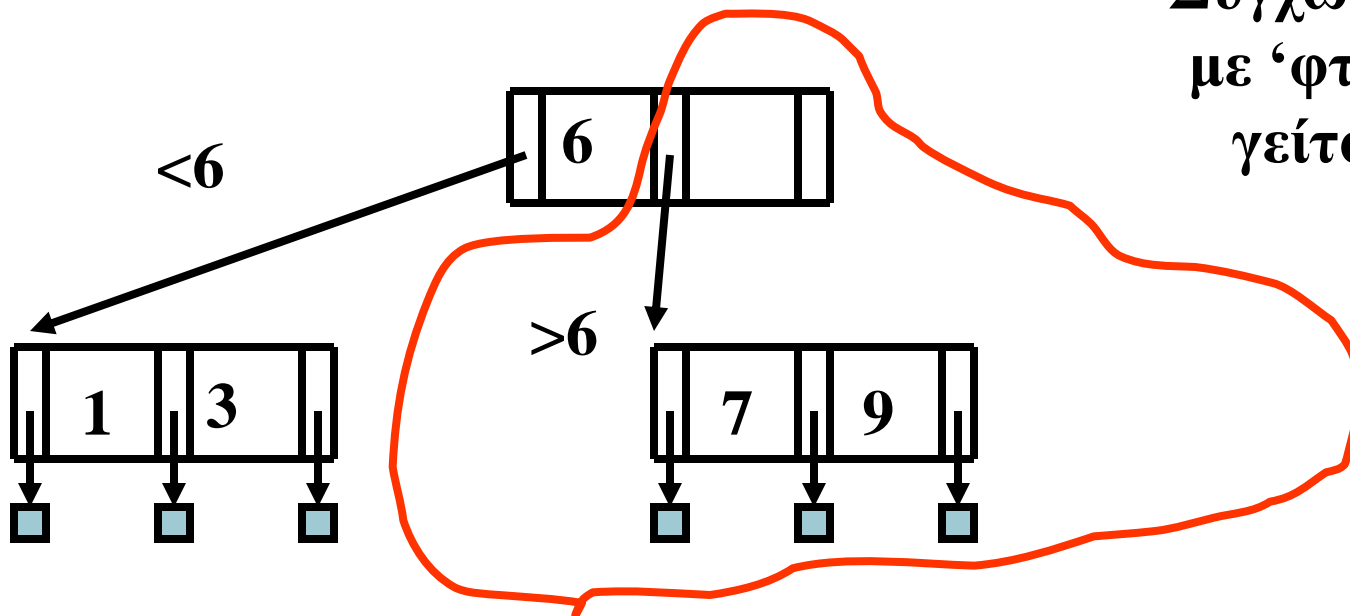


B-trees – Διαγραφή

- **4^η Περίπτωση:** διαγραφή κλειδιού από κόμβο φύλο - υποχείλιση και “φτωχός” γειτονικός κόμβος
(Π.χ., διέγραψε το **13** από το T0)
- Συγχώνευσε, αποσπώντας ένα κλειδί από τον **πατέρα**
- Ακριβώς το αντίθετο από την εισαγωγή: ‘διέσπασε και προώθησε προς τα επάνω’, αντί για ‘συγχώνευσε και προώθησε προς τα κάτω’
- Παράδειγμα:...

B-trees – Διαγραφή

- **4^η Περίπτωση:** διαγραφή κλειδιού από κόμβο φύλο - υποχείλιση και “φτωχός” γειτονικός κόμβος
- (Π.χ., διέγραψε το **13** από το T0)

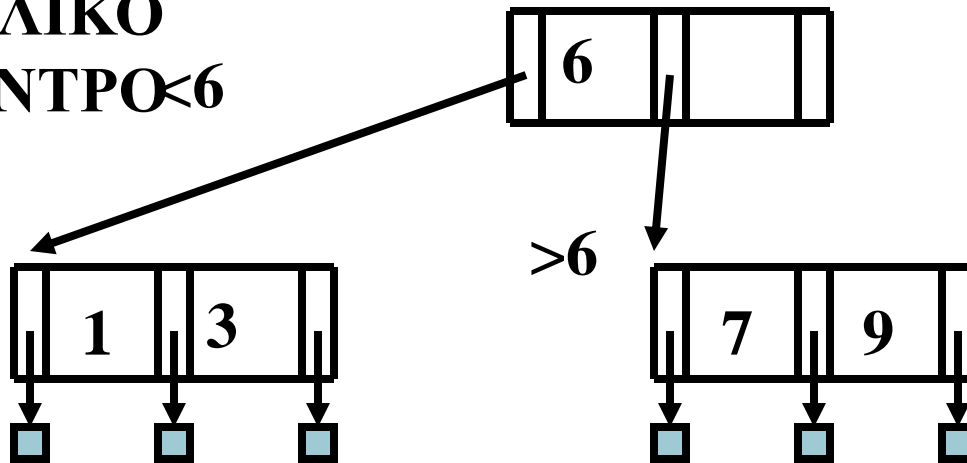


Συγχώνευσε
με ‘φτωχό’
γείτονα

B-trees – Διαγραφή

- **4^η Περίπτωση:** διαγραφή κλειδιού από κόμβο φύλο - υποχείλιση και “φτωχός” γειτονικός κόμβος
- (Π.χ., διέγραψε το **13** από το T0)

ΤΕΛΙΚΟ
ΔΕΝΤΡΟ < 6





B-trees – Διαγραφή

- **4^η Περίπτωση:** διαγραφή κλειδιού από κόμβο φύλο
- υποχείλιση και “φτωχός” γειτονικός κόμβος
- → ‘απέσπασε κλειδί από τον μητρικό κόμβο και συγχώνευσε’

- Ερώτηση: Τι θα συμβεί αν συμβεί υποχείλιση στον μητρικό κόμβο;
- Απάντηση: επανέλαβε αναδρομικά?



Διαγραφή B-tree – Ψευδοκώδικας

DELETION OF KEY 'K'

```
locate key 'K', in node 'N'
if( 'N' is a non-leaf node) {
    delete 'K' from 'N';
    find the immediately largest key 'K1';
    /* which is guaranteed to be on a leaf
node 'L' */
    copy 'K1' in the old position of 'K';
    invoke this DELETION routine on 'K1' from
the leaf node 'L';
else {
/* 'N' is a leaf node */
... (next slide..)
```

Διαγραφή B-tree – Ψευδοκώδικας

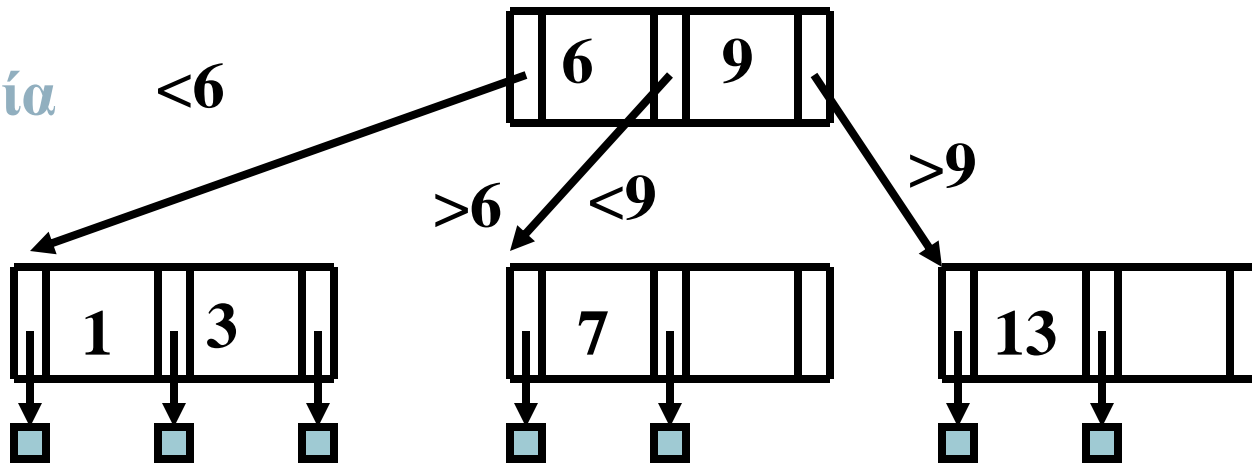
```
/* 'N' is a leaf node */
  if( 'N' underflows ){
    let 'N1' be the sibling of 'N';
    if( 'N1' is "rich"){ /* ie., N1 can lend us
a key */
      borrow a key from 'N1' THROUGH the parent
node;
    }else{ /* N1 is 1 key away from
underflowing */
      MERGE: pull the key from the parent 'P',
and merge it with the keys of 'N' and
'N1' into a new node;
      if( 'P' underflows){ repeat recursively }
    }
  }
```

B-trees πρακτικά:

Στην πράξη:

- Όχι κενά φύλλα
- Δείκτες προς εγγραφές

Θεωρία

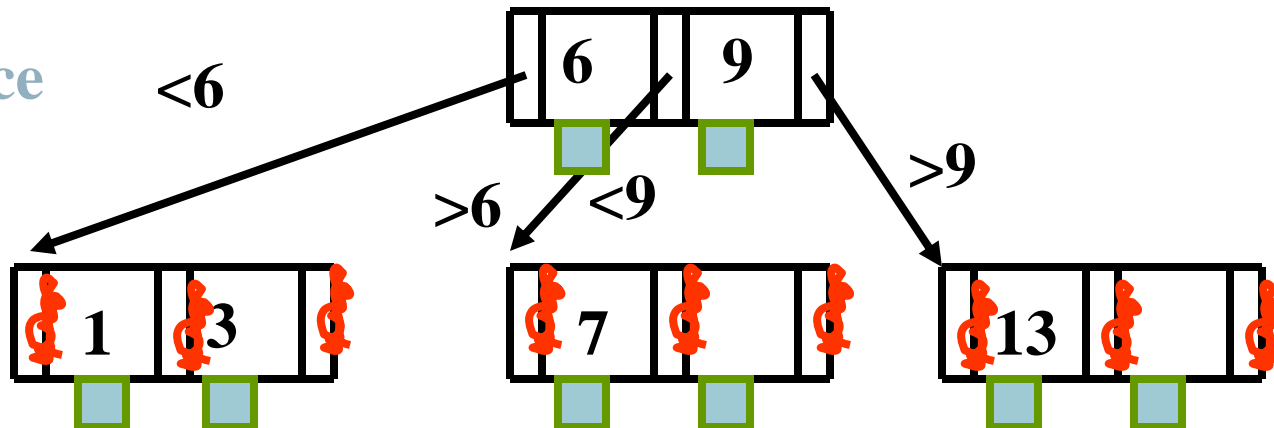


B-trees πρακτικά:

Στην πράξη:

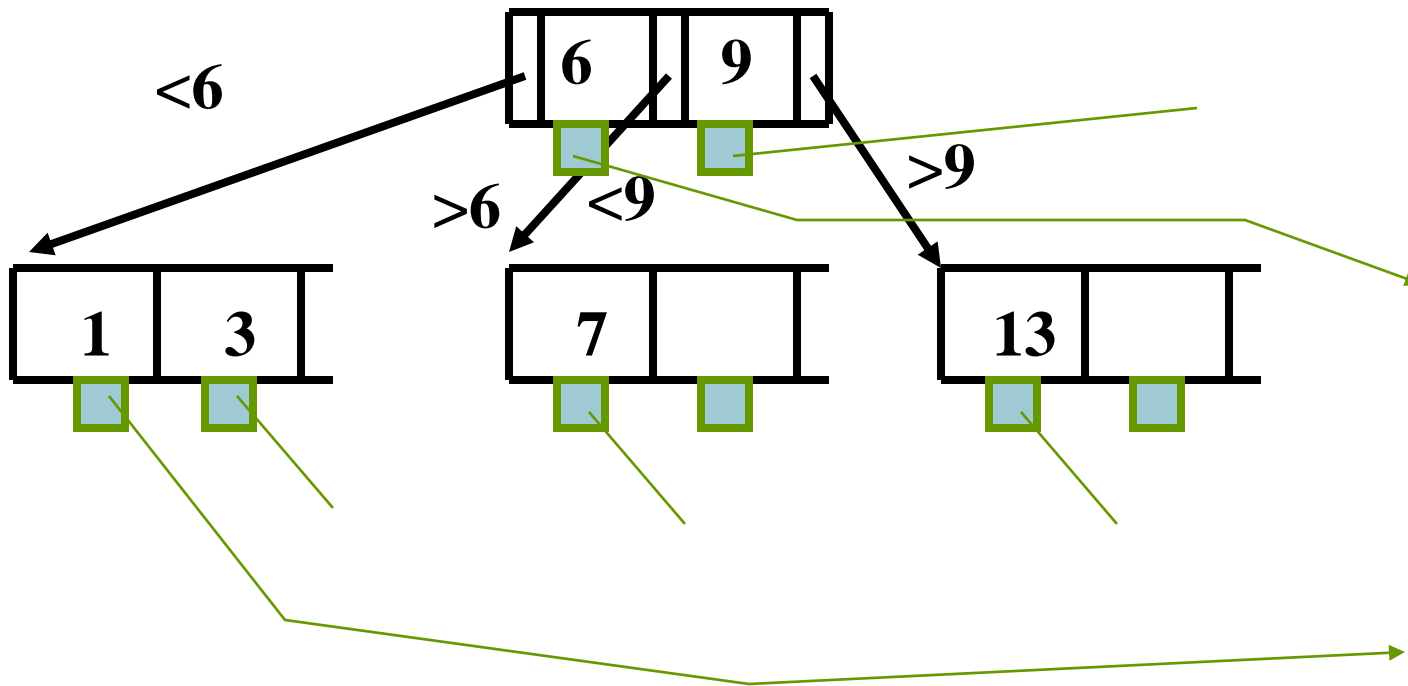
- Όχι κενά φύλλα
- Δείκτες προς εγγραφές

practice



B-trees στην πράξη:

Πρακτικά:

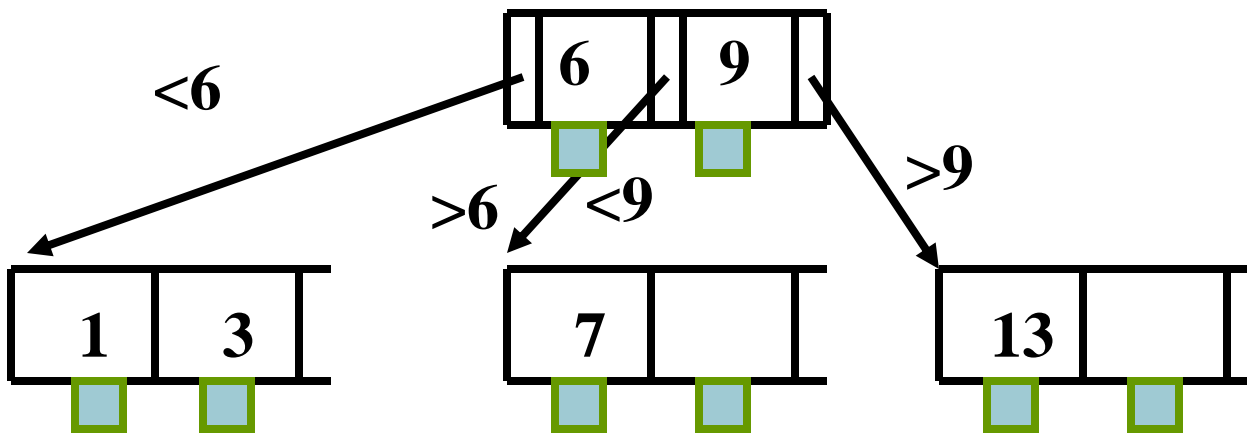


ΑΦΜ
3		
7		
6		
9		
1		

B-trees στην πράξη:

Πρακτικά τα πρότυπα είναι:

- Κόμβοι φύλλα: $(v_1, rp_1, v_2, rp_2, \dots, v_n, rp_n)$
- Εσωτερικοί κόμβοι: $(p_1, v_1, rp_1, p_2, v_2, rp_2, \dots)$



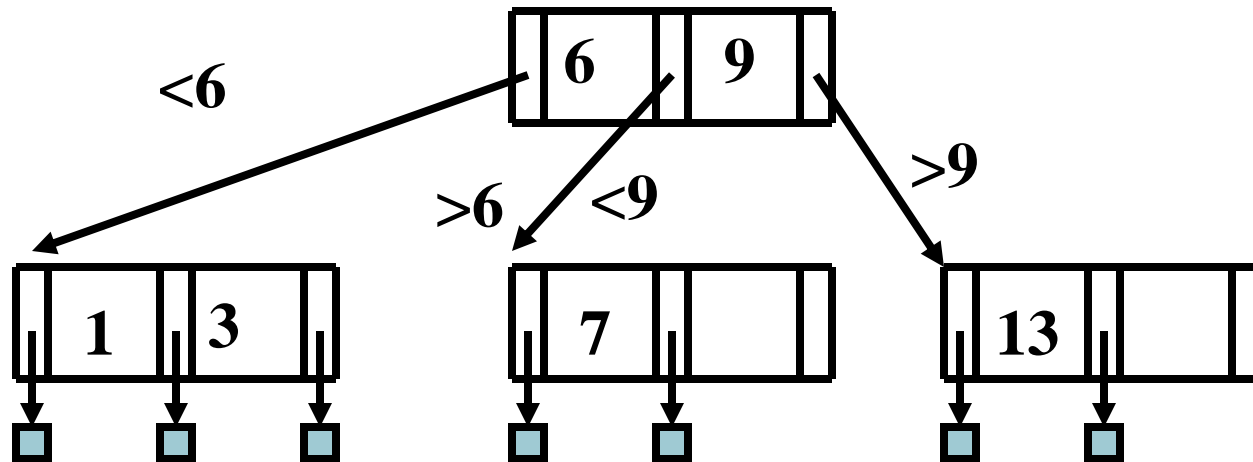


Ευρετηριοποίηση (Indexing)- Περίληπτικά

- Πρωτεύοντα / δευτερεύοντα ευρετήρια
- Σειριακή μέθοδος προσπέλασης (ISAM)
- B - trees,
- **B+ - trees**
- Κατακερματισμός

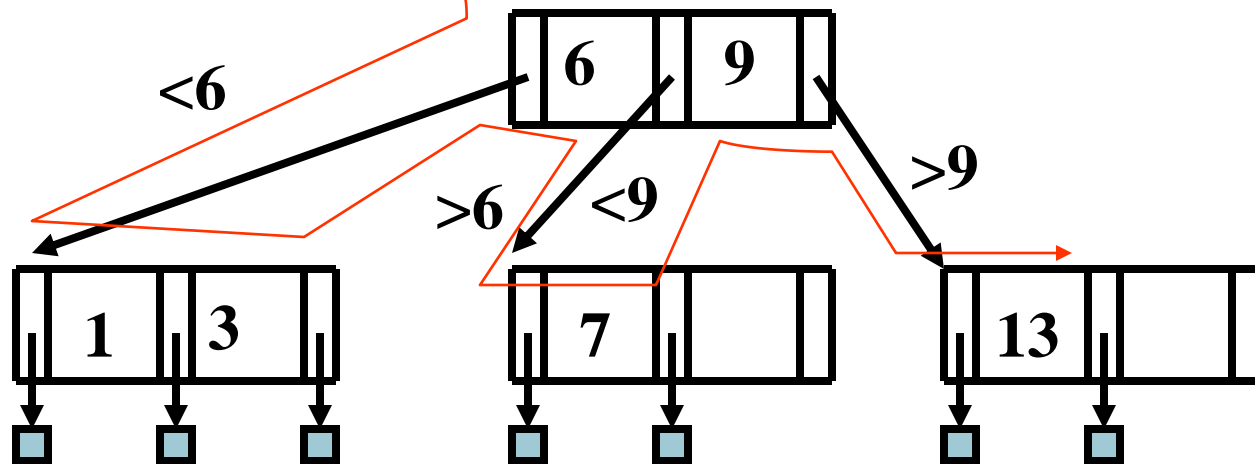
B+ trees - κίνητρο

B-tree – τύπωσε τα κλειδιά σε ταξινομημένη σειρά :



B+ trees - κίνητρο

B-tree: χρειάζεται να επιστρέφει στον μητρικό κόμβο (back-tracking) – Πως θα το αποφύγουμε;

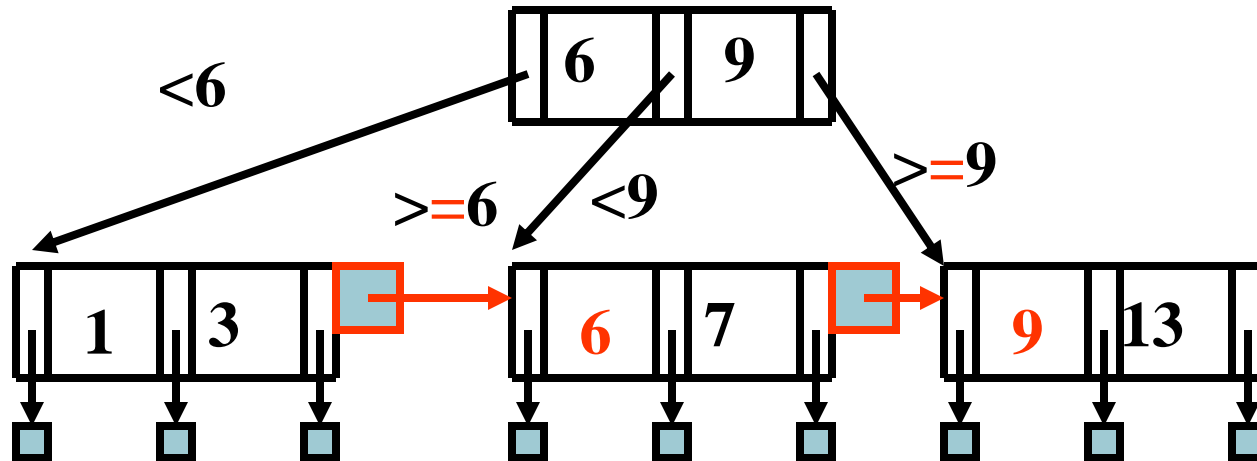




Λύση: B^+ - trees

- Διευκόλυνε σειριακές πράξεις
- Ενώνουν όλους τους κόμβους φύλλα μαζί και ..
- Αντιγράφουν κλειδιά των εσωτερικών κόμβων ώστε να βεβαιωθεί ότι **κάθε κλειδί εμφανίζεται στο επίπεδο των φύλλων**

B+ trees





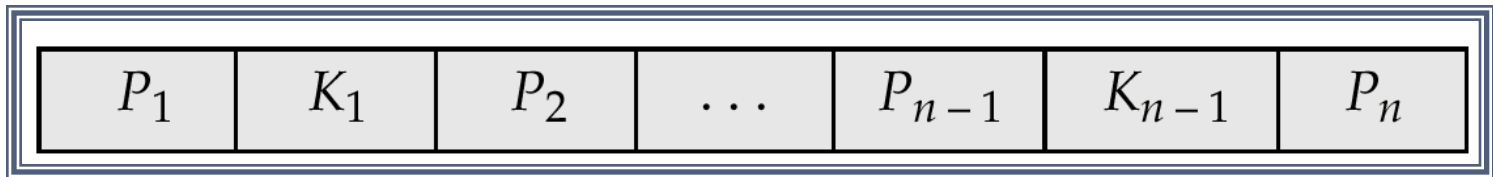
B⁺-Trees (συν.)

Ένα B⁺-tree είναι μια δενδρική δομή (με ρίζα) που ικανοποιεί τις παρακάτω προϋποθέσεις / ιδιότητες :

- Όλα τα μονοπάτια από την ρίζα ως το επίπεδο των φύλλων έχουν το ίδιο μήκος
- Κάθε κόμβος που δεν είναι ρίζα ή φύλλο έχει το ελάχιστο $\lceil n/2 \rceil$ και το μέγιστο n παιδιά
- Ένας κόμβος φύλλο έχει το ελάχιστο $\lceil (n-1)/2 \rceil$ και το μέγιστο $n-1$ τιμές
- Ειδικές περιπτώσεις:
 - Εάν η ρίζα δεν είναι φύλλο θα πρέπει να έχει τουλάχιστον δύο παιδιά
 - Εάν η ρίζα είναι φύλλο (δηλ δεν υπάρχουν άλλοι κόμβοι στο δέντρο), μπορεί να έχει το ελάχιστο 0 και το μέγιστο $(n-1)$ τιμές

B⁺-Tree Δομή κόμβων

- Τυπικός κόμβος

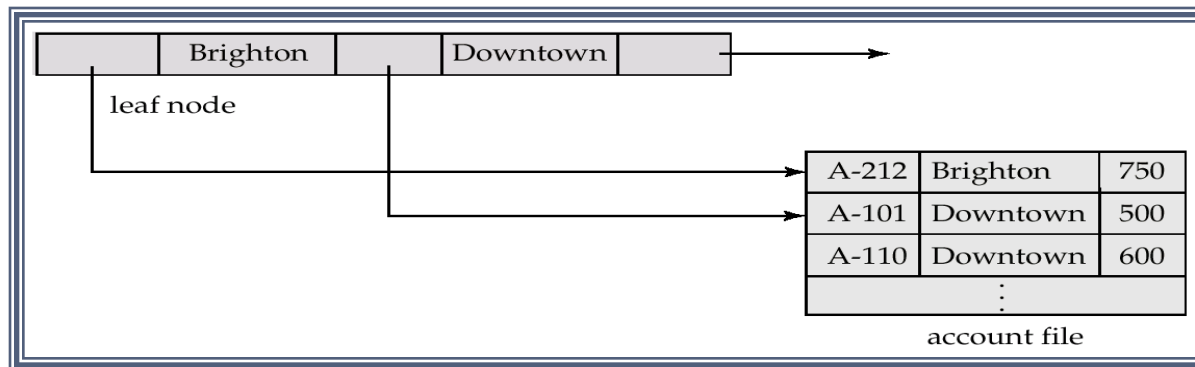


- K_i : οι τιμές των κλειδιών αναζήτησης (search-key values)
- P_i : δείκτες
 - προς τους κόμβους παιδιά (για εσωτερικούς κόμβους) ή
 - προς εγγραφές ή κάδους εγγραφών (για κόμβους φύλλα)
- Τα κλειδιά αναζήτησης σε έναν κόμβο είναι διατεταγμένα

$$K_1 < K_2 < K_3 < \dots < K_{n-1}$$

Κόμβοι φύλλα σε ένα B⁺-Tree – Ιδιότητες

- Για $i = 1, 2, \dots, n-1$, ο δείκτης P_i είτε δείχνει σε εγγραφές αρχείου με τιμή K_i είτε σε κάποιο κάδο δεικτών για εγγραφές αρχείων, όπου κάθε εγγραφή έχει κλειδί αναζήτησης με τιμή K_i .
- Θα χρησιμοποιήσουμε την δομή κάδων μόνο εάν το κλειδί αναζήτησης δεν αποτελεί πρωτεύων κλειδί
- Εάν οι κόμβοι L_i, L_j είναι φύλλα και $i < j$, οι τιμές των κλειδιών αναζήτησης του L_i είναι μικρότερες από τις τιμές των κλειδιών αναζήτησης του L_j
- Ο δείκτης P_n δείχνει στον επόμενο κόμβο φύλλο κατά την φορά διάταξης των κλειδιών αναζήτησης

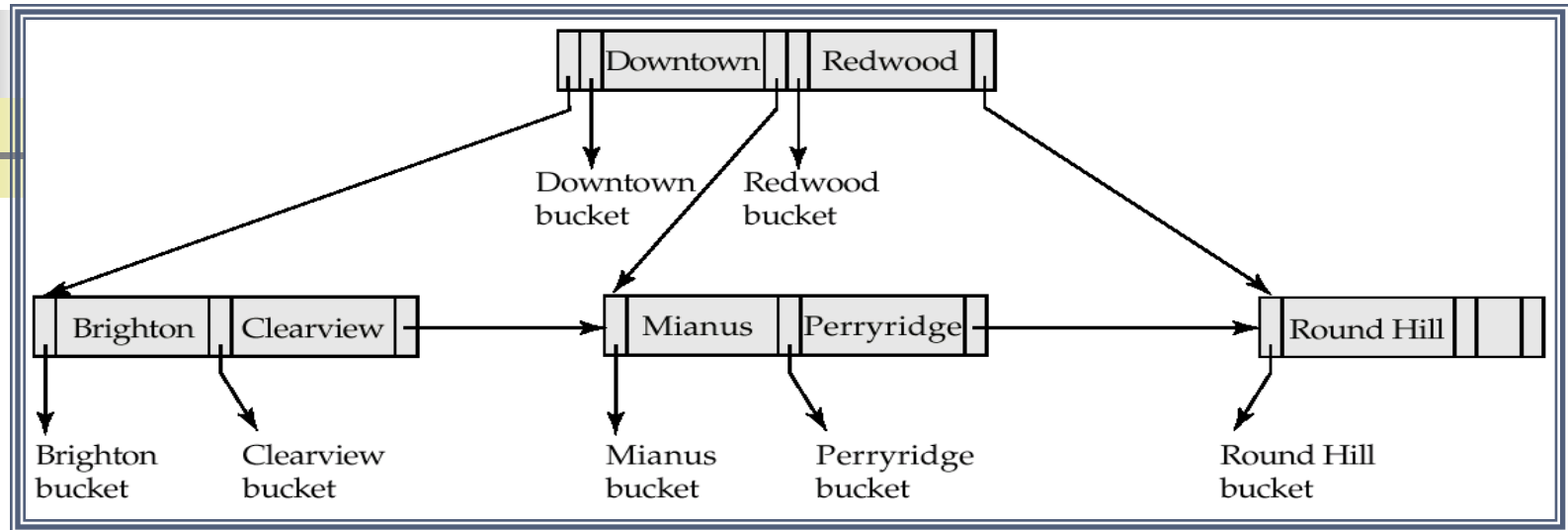


Ιδιότητες εσωτερικών κόμβων των B⁺-Trees

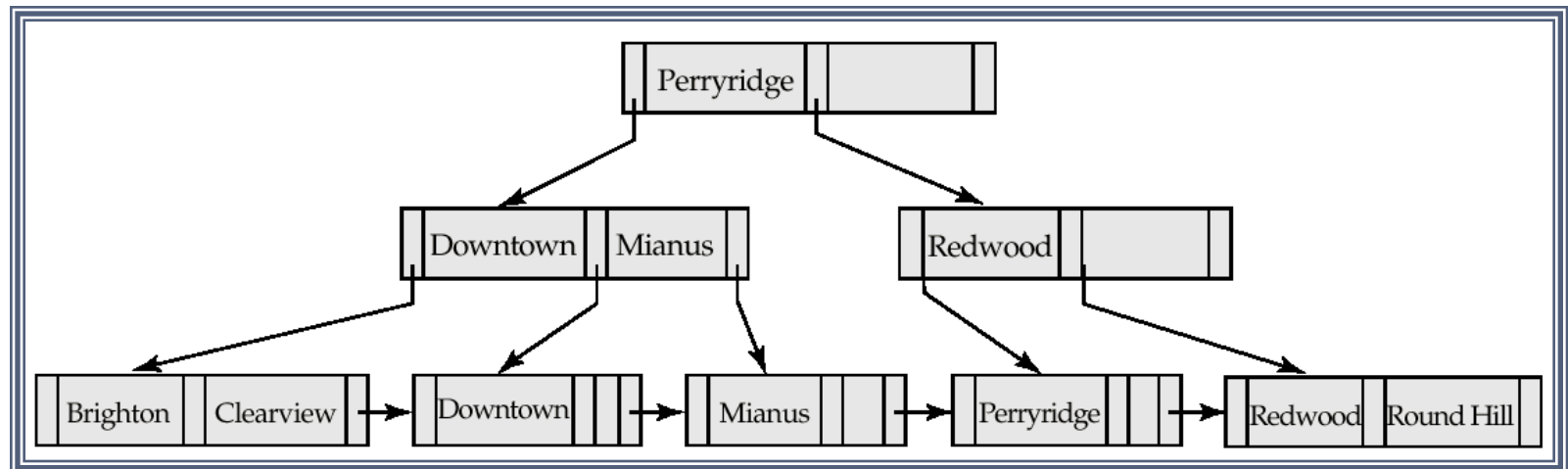
- Οι εσωτερικοί κόμβοι σχηματίζουν ένα πολύ-επίπεδο αραιό ευρετήριο προς τους κόμβους φύλλα. Για έναν εσωτερικό κόμβο με m δείκτες, έχουμε:
 - Όλες οι τιμές των κλειδιών αναζήτησης του υποδέντρου στο οποίο δείχνει ο δείκτης P_1 είναι μικρότερες από K_1
 - Για $2 \leq i \leq n-1$, όλα τα κλειδιά αναζήτησης στο υπόδεντρο στο οποίο δείχνει ο δείκτης P_i θα έχει τιμές μεγαλύτερες ή ίσες του K_{i-1} και μικρότερες του K_{m-1}



B-Tree vs B+-Tree



B-tree (επάνω) και B+-tree (κάτω) για ίδιο σύνολο δεδομένων



B+ tree Εισαγωγή - Ψευδοκώδικας

```
INSERTION OF KEY 'K'
```

```
insert search-key value to 'L' such that the  
keys are in order;
```

```
if ( 'L' overflows) {
```

```
    split 'L' ;
```

```
    insert (ie., COPY) smallest search-key  
value
```

```
    of new node to parent node 'P' ;
```

```
    if ('P' overflows) {
```

```
        repeat the B-tree split procedure  
recursively;
```

```
        /* Παρατήρηση: διάσπαση του B-TREE; Όχι  
του B+ -tree */
```

```
    }
```

```
}
```

B+ tree Εισαγωγή – Ψευδοκώδικας

(συνέχεια)

/* ΠΡΟΣΟΧΗ:

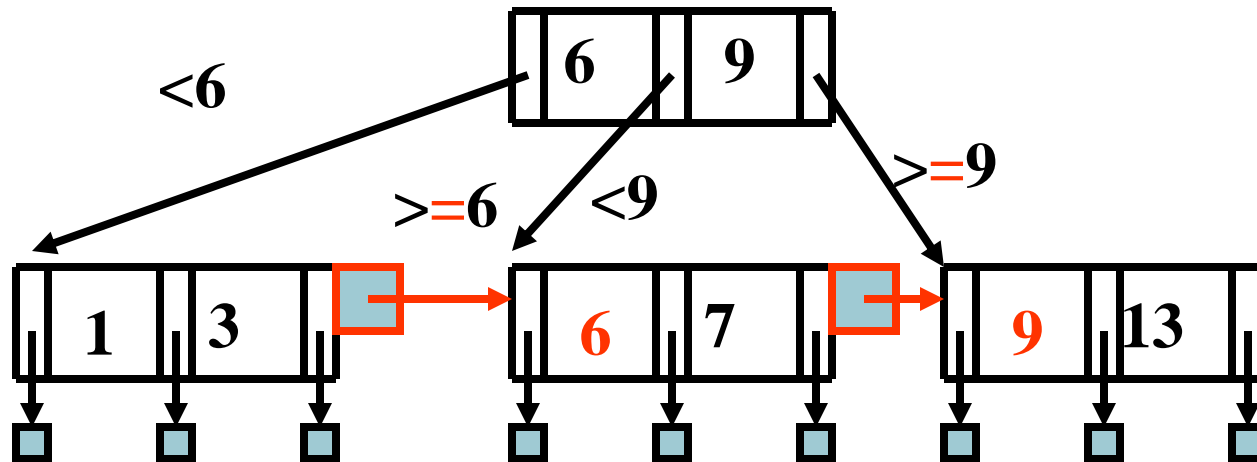
Διαχειριζόμαστε μια διάσπαση στο επίπεδο των κόμβων φύλλων με το να αντιγράψουμε το μέσο κλειδί στον κόμβο του ανώτερου επιπέδου.

Διαχειριζόμαστε μια διάσπαση σε υψηλότερο επίπεδο με το να προωθούμε το μέσο κλειδί προς τα επάνω

*/

B+ trees - Εισαγωγή

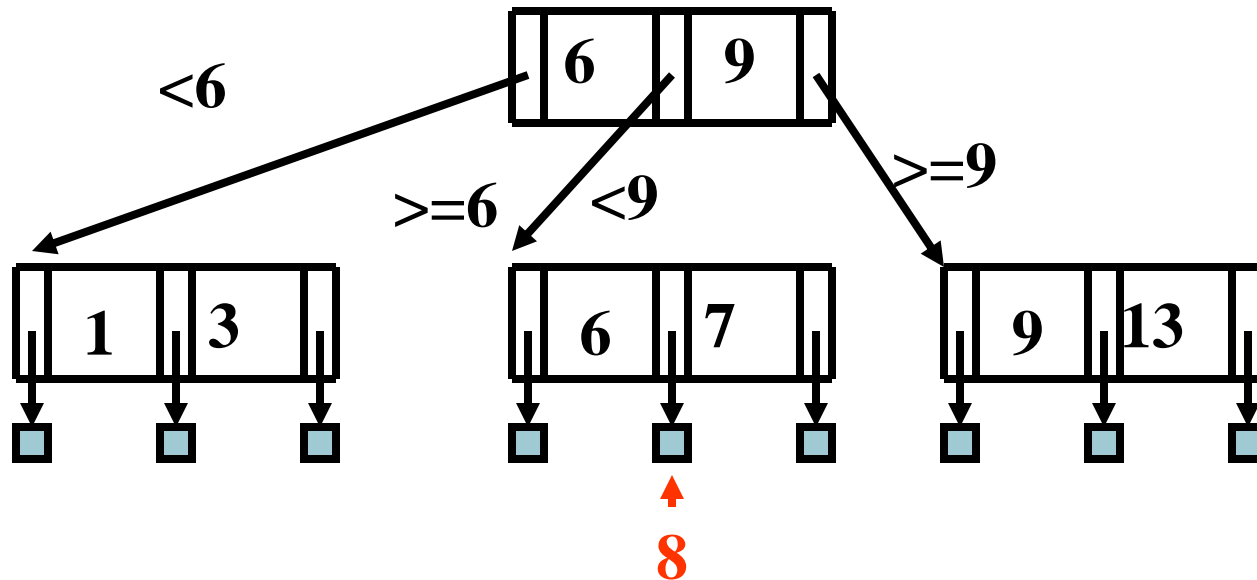
Π.χ., εισαγωγή
'8'



B+ trees - Εισαγωγή

Π.χ., εισαγωγή

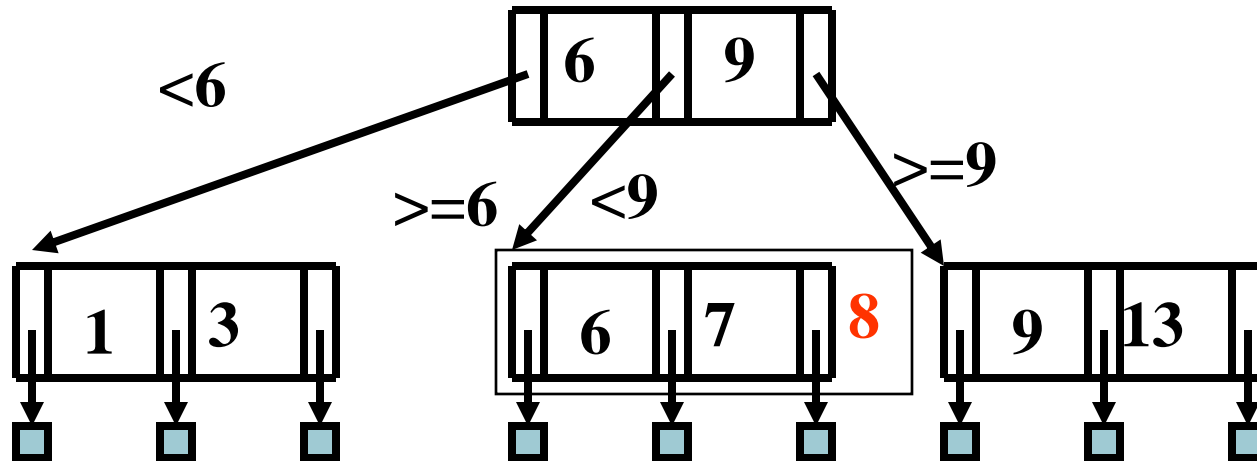
'8'



B+ trees - Εισαγωγή

Π.χ., εισαγωγή

'8'

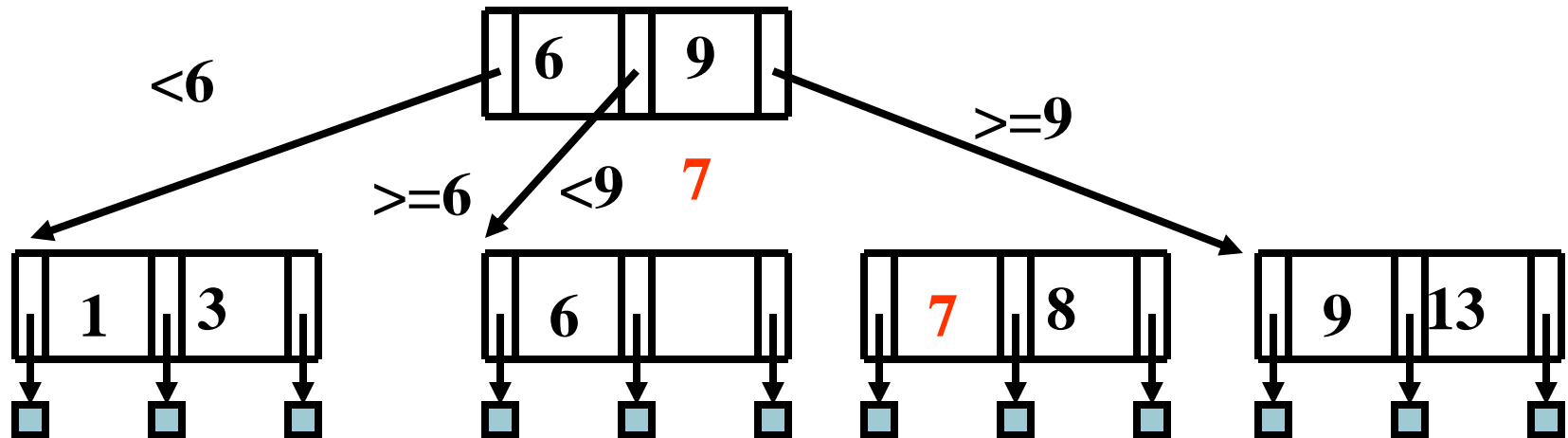


**Αντέγραψε την
μεσαία τιμή επάνω**

B+ trees - Εισαγωγή

Π.χ., εισαγωγή

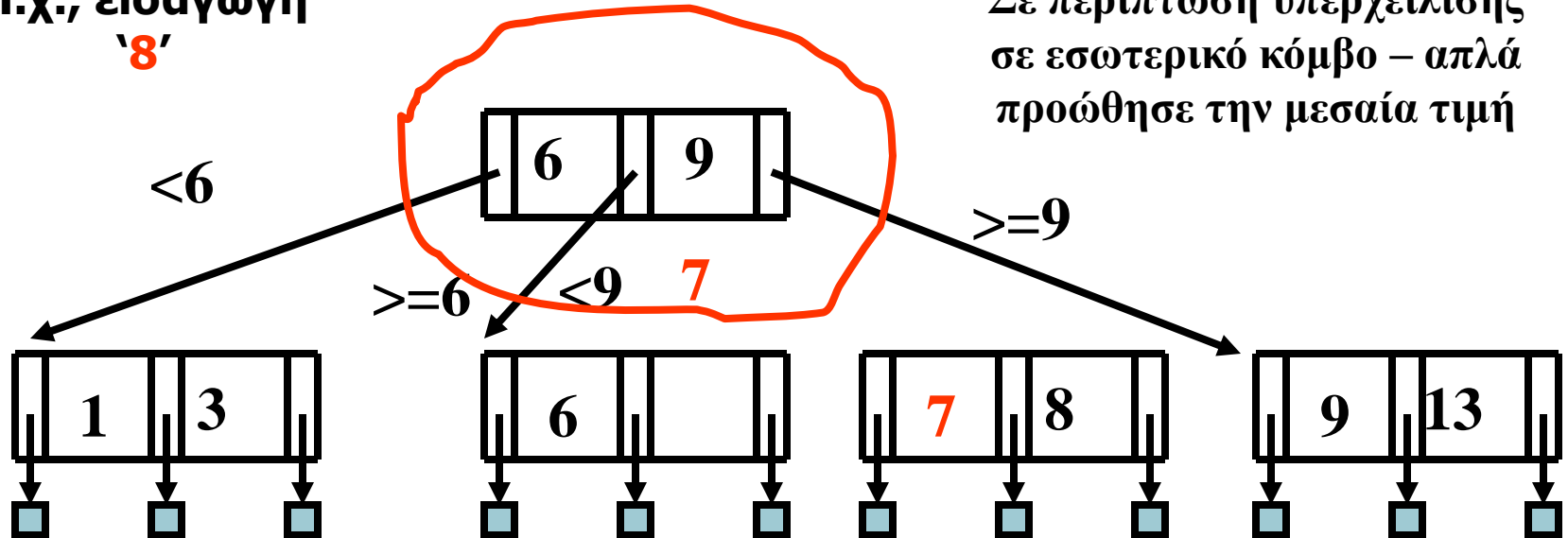
'8'



**Αντέγραψε την
μεσαία τιμή επάνω**

B+ trees - Εισαγωγή

Π.χ., εισαγωγή
'8'

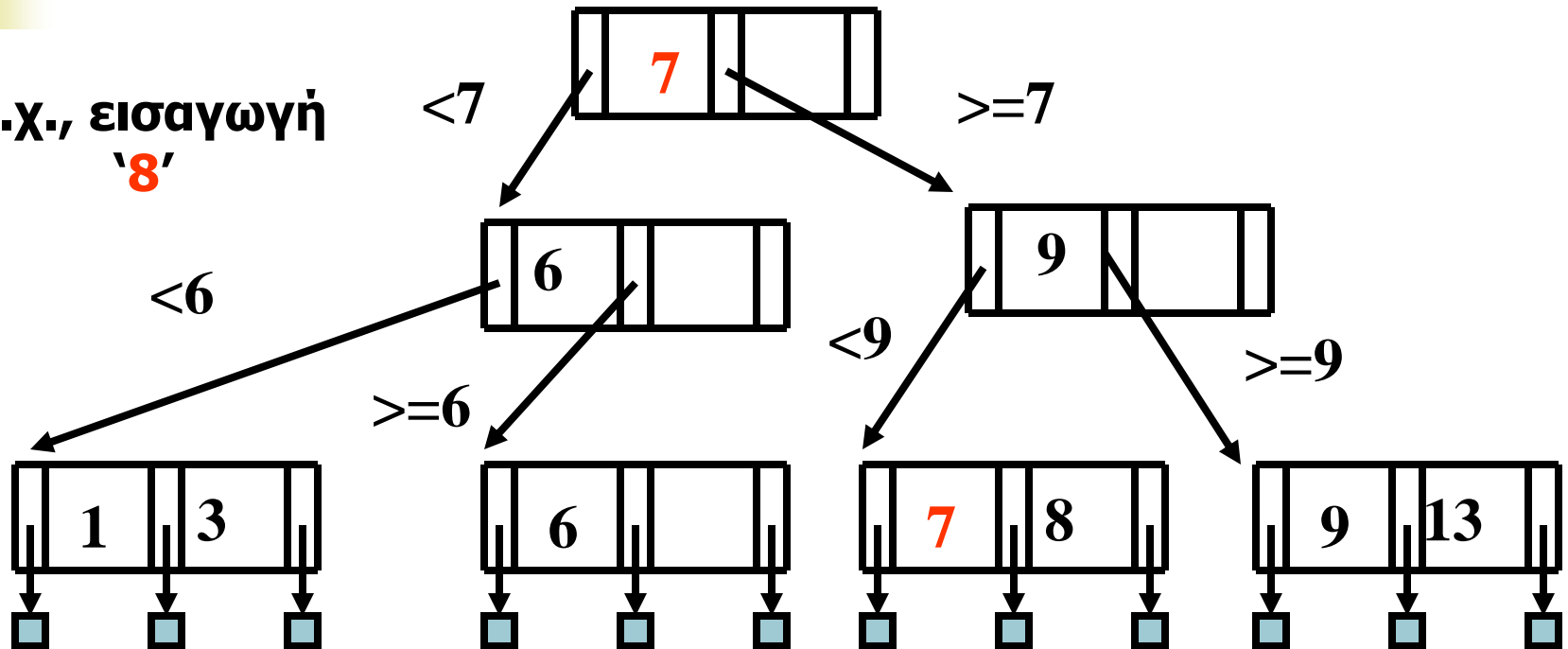


Σε περίπτωση υπερχείλισης
σε εσωτερικό κόμβο – απλά
προώθησε την μεσαία τιμή

**Αντέγραψε την
μεσαία τιμή επάνω**

B+ trees - Εισαγωγή

Π.χ., εισαγωγή
'8'



Τελικό Δέντρο



B-Trees vs B+-Trees

■ Πλεονεκτήματα ευρετηρίων B-Trees:

- Μπορεί να χρησιμοποιηθούν λιγότεροι κόμβοι από το αντίστοιχο B⁺-Tree.
- Μερικές φορές είναι δυνατό να βρούμε την τιμή του κλειδιού αναζήτησης πριν προσπελάσουμε τον κόμβο φύλλο.

■ Μειονεκτήματα ευρετηρίων B-Trees:

- Μόνο ένα μικρό κλάσμα των κλειδιών αναζήτησης μπορεί να βρεθούν νωρίς
- Οι εσωτερικοί κόμβοι είναι συνήθως μεγαλύτεροι, έτσι ο βαθμός διακλάδωσης (fan-out) μειώνεται. Έτσι, τα B-Trees έχουν συνήθως μεγαλύτερο βάθος από τα αντίστοιχα B⁺-Trees.
- Η εισαγωγή και διαγραφή περισσότερο πολύπλοκη απ' ό τι στα B⁺-Trees
- Υλοποίηση δυσκολότερη από ό τι για B⁺-Trees.

■ Τυπικά τα πλεονεκτήματα των B-Trees δεν υπερκαλύπτουν τα μειονεκτήματά τους

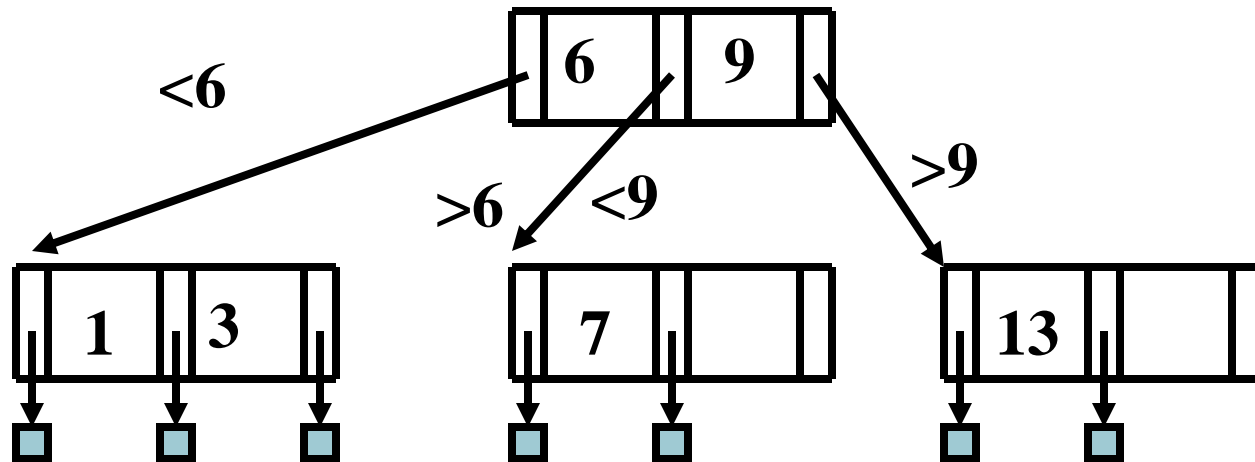


B*-tree

- Σε ένα B-tree, χρησιμοποίηση (utilization) χειρότερης περίπτωσης = 50%, εάν μόλις έχουμε διασπάσει όλες τις σελίδες
- Πως θα αυξήσουμε την χρησιμοποίηση στα B - trees?
- ... με τα B* - trees!

B-trees και B*-trees

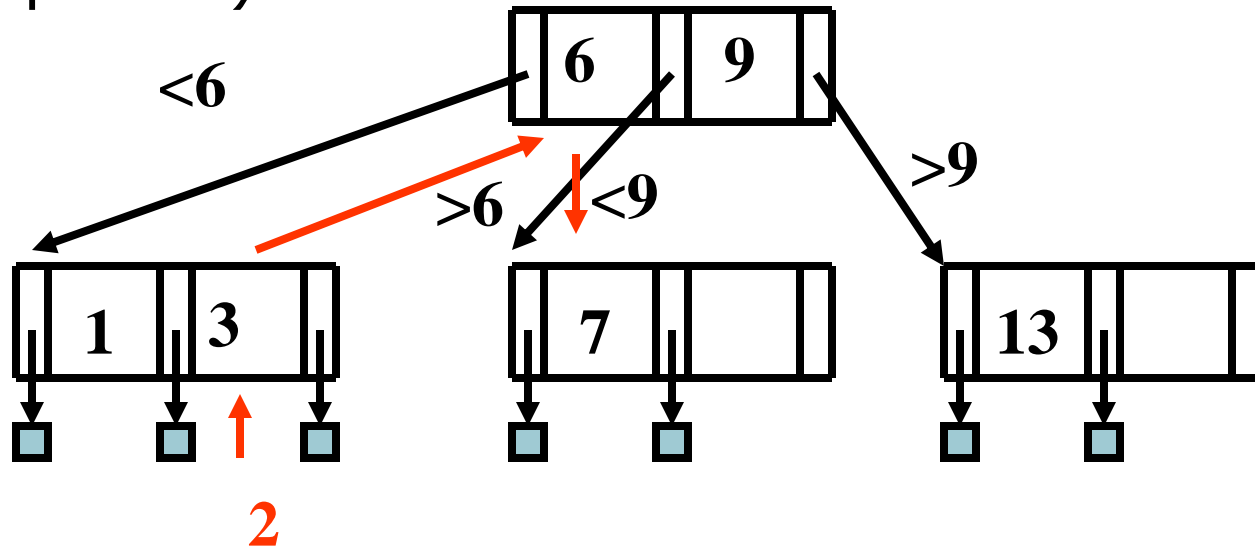
Π.χ., Tree T0; Εισαγωγή του '2'



2

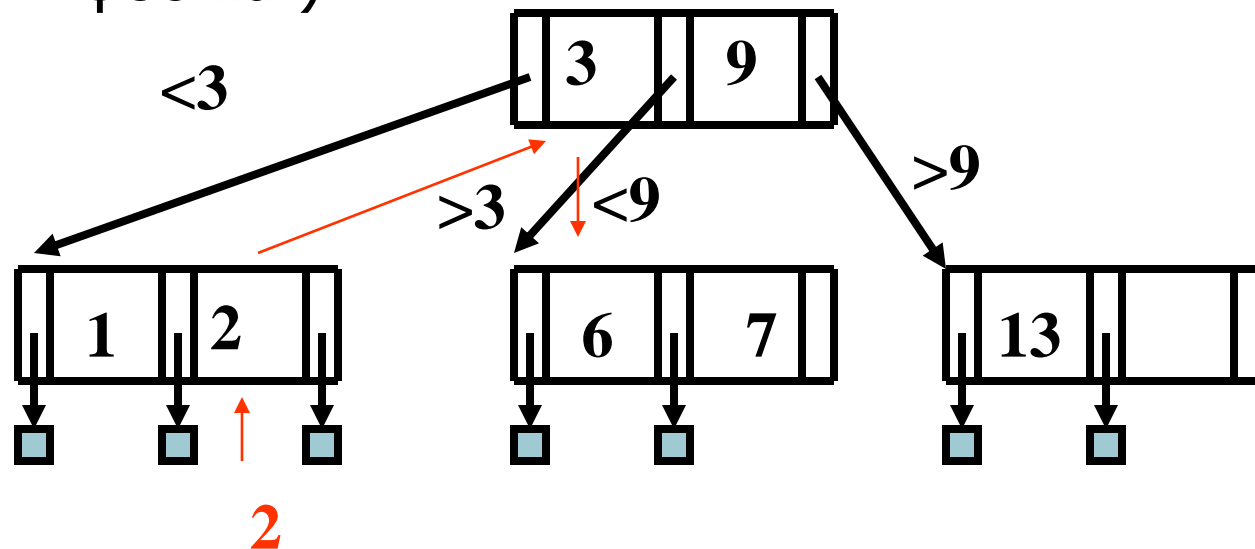
B*-trees: Αναβληθείς διάσπαση!

Αντί να προκληθεί διάσπαση, δάνεισε κλειδί στον γειτονικό κόμβο! (διαμέσου του μητρικού κόμβου, φυσικά!)



B*-trees: Αναβληθείς διάσπαση!

- Αντί να προκληθεί διάσπαση, δάνεισε κλειδί στον γειτονικό κόμβο! (διαμέσου του μητρικού κόμβου, φυσικά!)





B*-trees: Αναβληθείς διάσπαση!

- Παρατηρήσεις: μικρότερο ύψος, περισσότερο συμπαγές, ταχύτερο δέντρο
- Είναι μια από τις σπάνιες περιπτώσεις όπου η χρησιμοποίηση χώρου και η ταχύτητα αυξάνονται ταυτόχρονα
- **Αλλά:** τι γίνεται στην περίπτωση που ο γειτονικός κόμβος δεν έχει χώρο για να μας δανείσει ;



B*-trees: Αναβληθείς διάσπαση!

- **Αλλά:** τι γίνεται στην περίπτωση που ο γειτονικός κόμβος δεν έχει χώρο για να μας δανείσει ;
- **Απάντηση:** Διάσπαση 2-σε-3 :
 - Πάρε τα κλειδιά από τον γειτονικό κόμβο
 - Συγχώνευσέ τα με αυτά του αρχικού κόμβου (και ένα κλειδί από τον πατέρα)
 - Διέσπασέ τα στα τρία
- **Λεπτομέρειες:** αρκετά μπερδεμένο (ακόμα χειρότερο στην περίπτωση της διαγραφής)



Συμπεράσματα

- Όλες οι παραλλαγές ενός B-tree μπορούν να χρησιμοποιηθούν για κάθε είδους ευρετήριο
 - Πρωτεύον/ Δευτερέων
 - Αραιό/ Πυκνό
- Όλα έχουν εξαιρετική απόδοση χειρότερης περίπτωσης
 - $O(\log N)$ για εισαγωγές/διαγραφές/αναζήτηση
- Είναι η δομή ευρετηρίου που συνήθως χρησιμοποιείται



Ευρετηριοποίηση (Indexing)- Περίληπτικά

- Πρωτεύοντα / δευτερεύοντα ευρετήρια
- Σειριακή μέθοδος προσπέλασης (ISAM)
- B - trees, B+ - trees
- **Κατακερματισμός**
 - **Στατικός**
 - **Δυναμικός**