Ευρετήρια

Maria K. Krommyda

Εθνικό Μετσόβιο Πολυτεχνείο

Εργαστήριο Βάσεων Γνώσεων &Δεδομένων

B tree

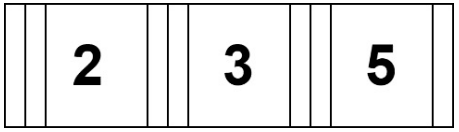
Κατασκευάστε ένα Β δέντρο για το παρακάτω σύνολο από τιμές κλειδιών:

{2,3,5,7,11,17,19,23,29,31}

Το δέντρο είναι αρχικά κενό και οι τιμές προστίθενται σε αύξουσα σειρά. Ο αριθμός δεικτών σε ένα κόμβο είναι n= 4.

Στο Β δέντρο όλοι οι κόμβοι, εκτός από την ρίζα, πρέπει να έχουν τουλάχιστον

$\lfloor (n-1)/2 \rfloor$ κλειδιά.



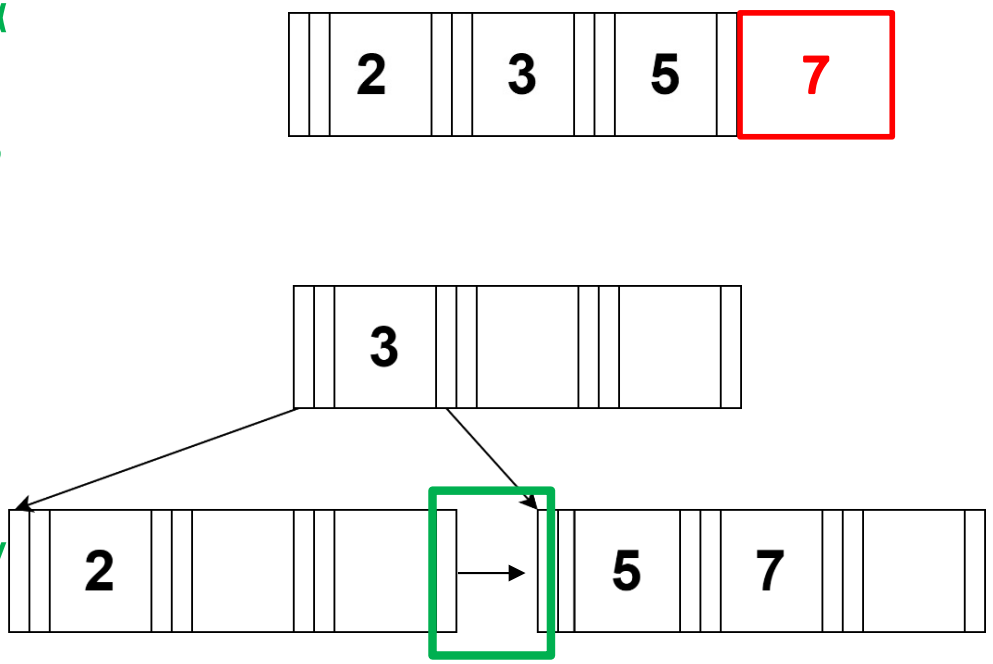
Άρα εδώ πρέπει να έχουν τουλάχιστον

$\lfloor (4-1)/2 \rfloor$ = 1 κλειδί.

B tree

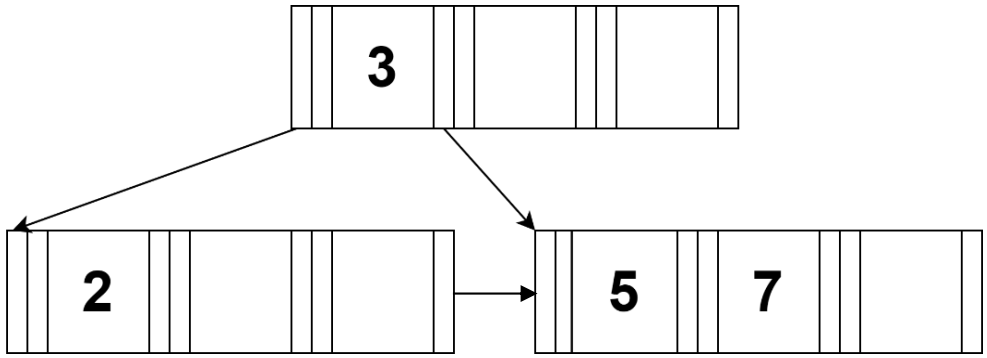
{~~2,3,5~~,7,11,17,19,23,29,31}

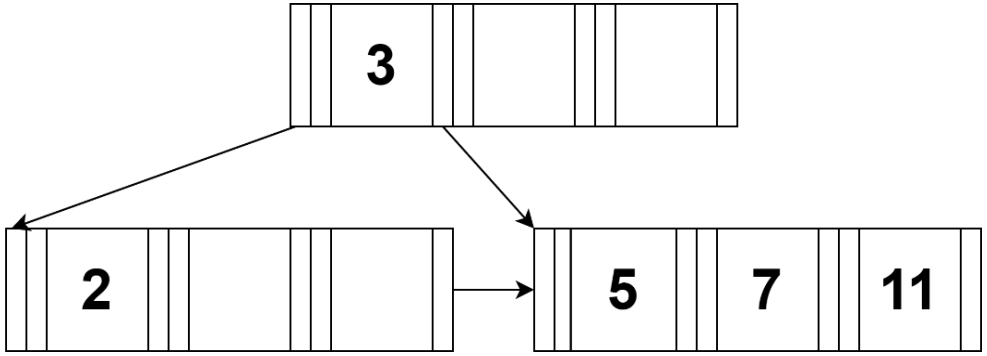
Οι δείκτες ανάμεσα στα φύλλα εμφανίζονται στα B δέντρα για λόγους ομοιογένειας.
ΠΡΟΣΟΧΗ: Δεν μπορούν να χρησιμοποιηθούν για να απαντήσουν ερωτήματα διαστήματος!



B tree

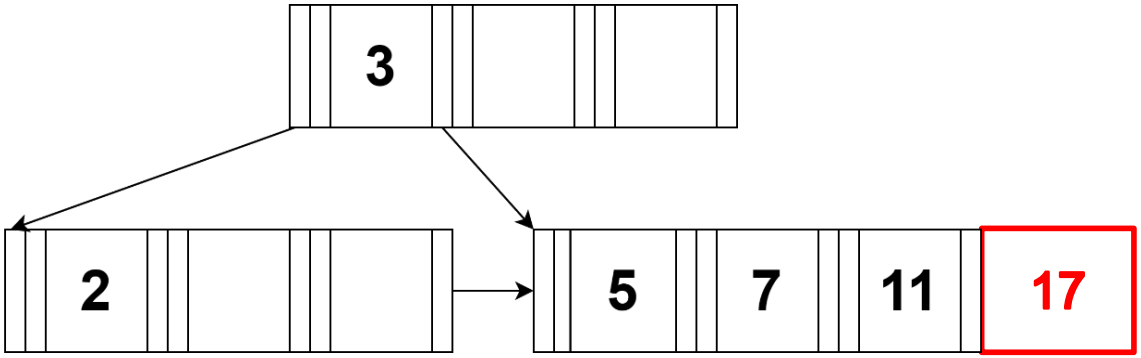
{~~2,3,5,7~~,11,17,19,23,29,31}

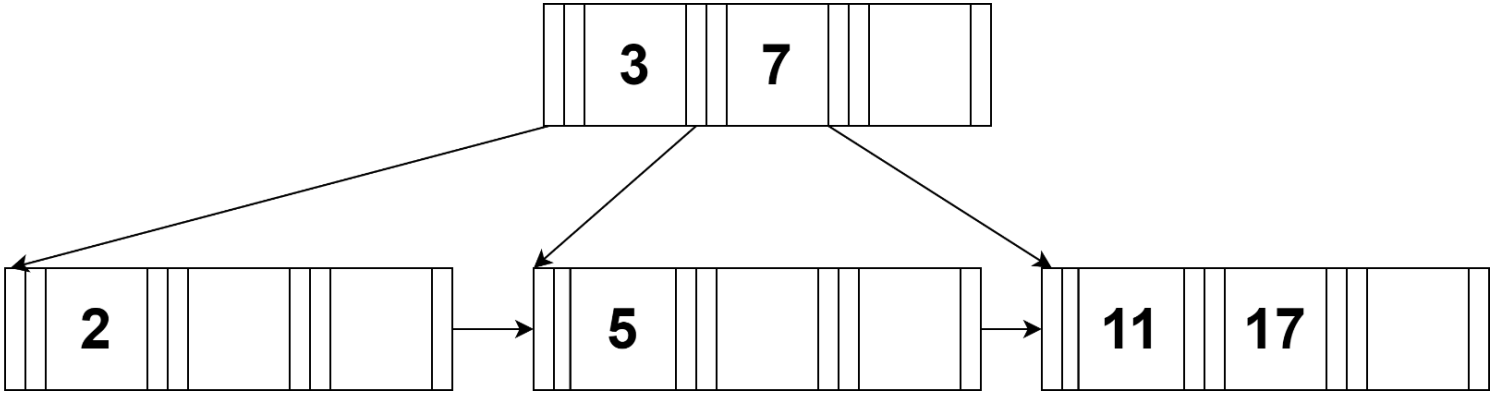




B tree

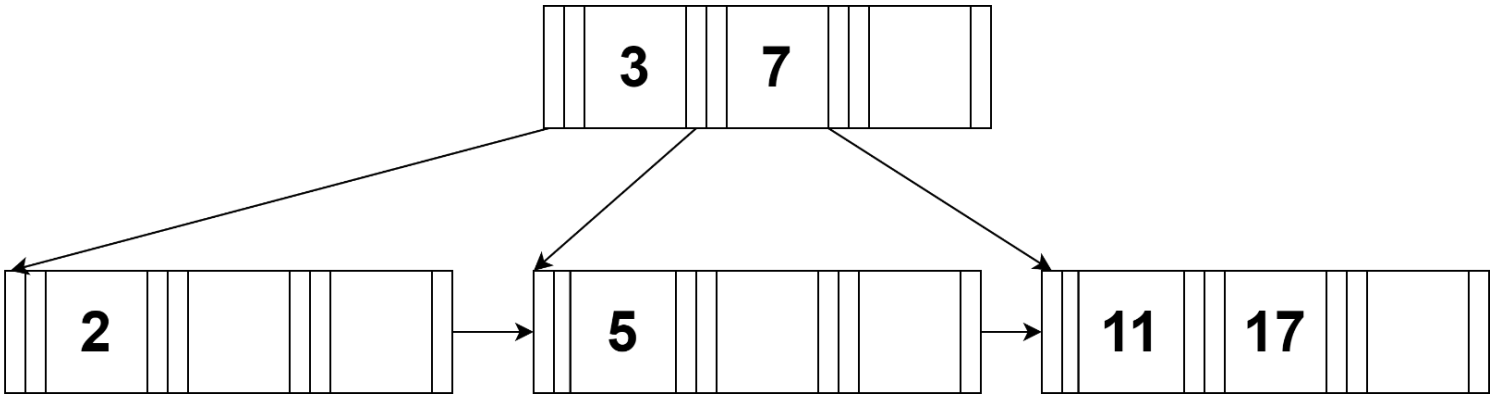
{~~2,3,5,7,11~~,17,19,23,29,31}

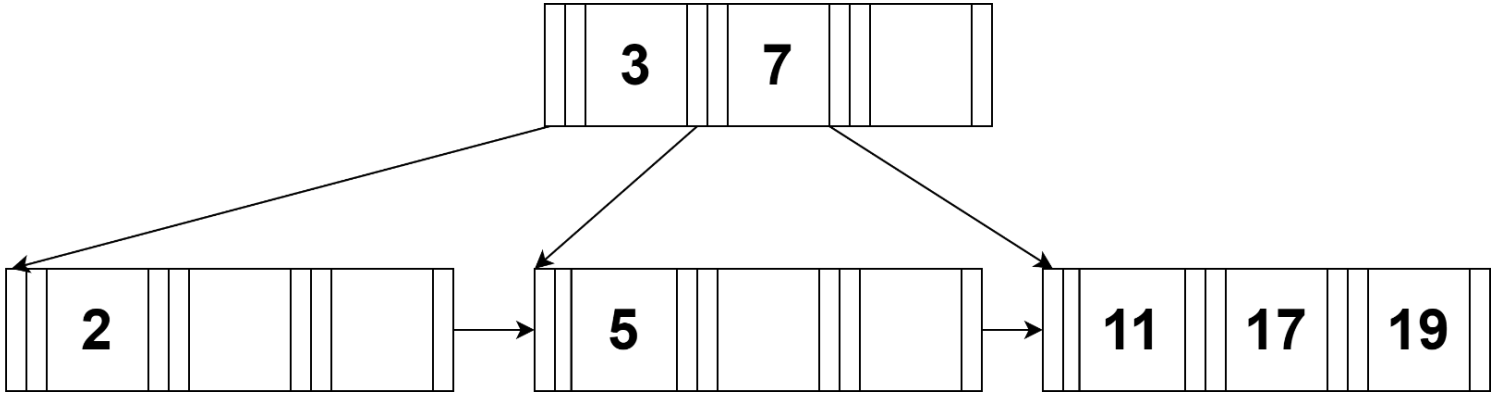




B tree

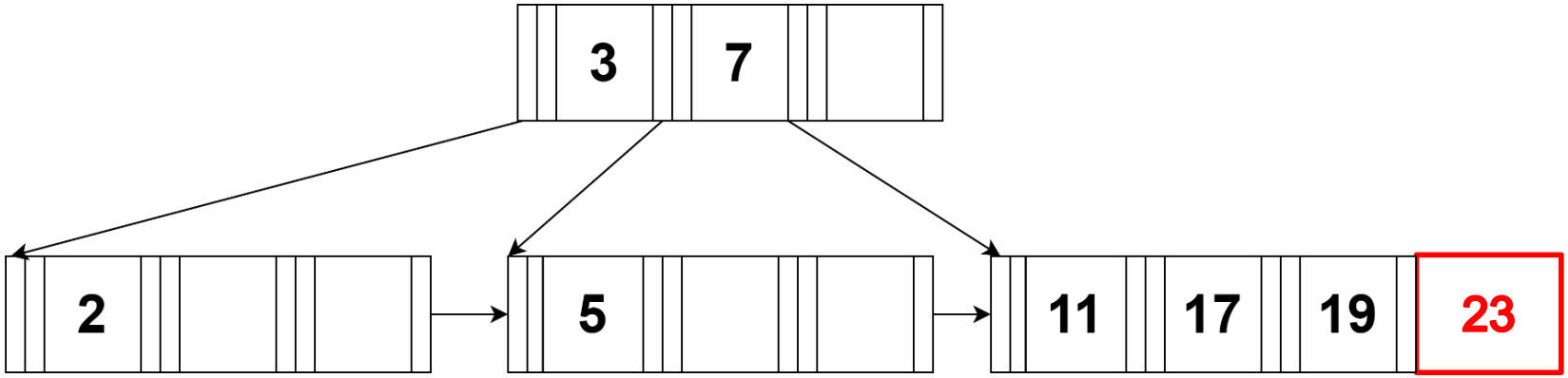
{~~2,3,5,7,11,17,~~19,23,29,31}

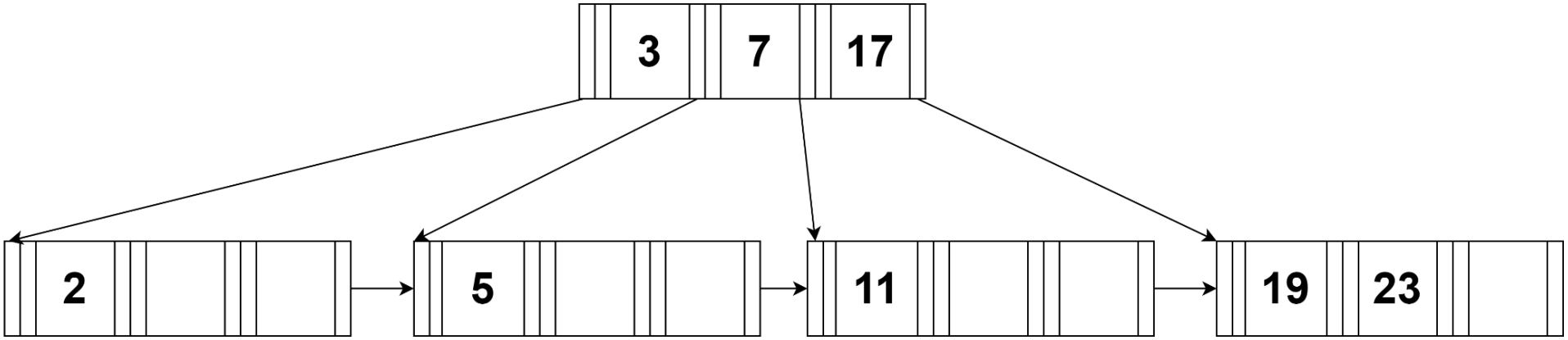




B tree

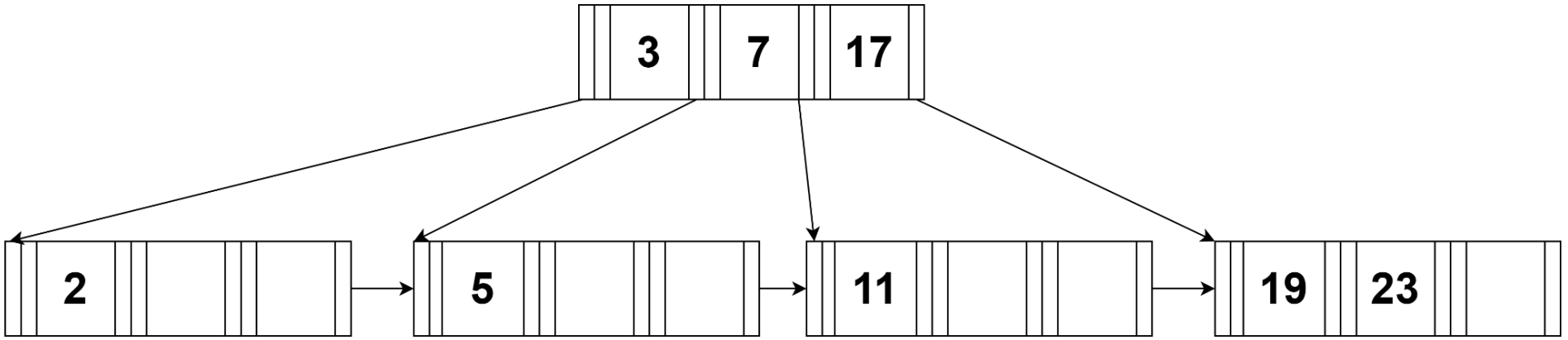
{~~2,3,5,7,11,17,19~~,23,29,31}

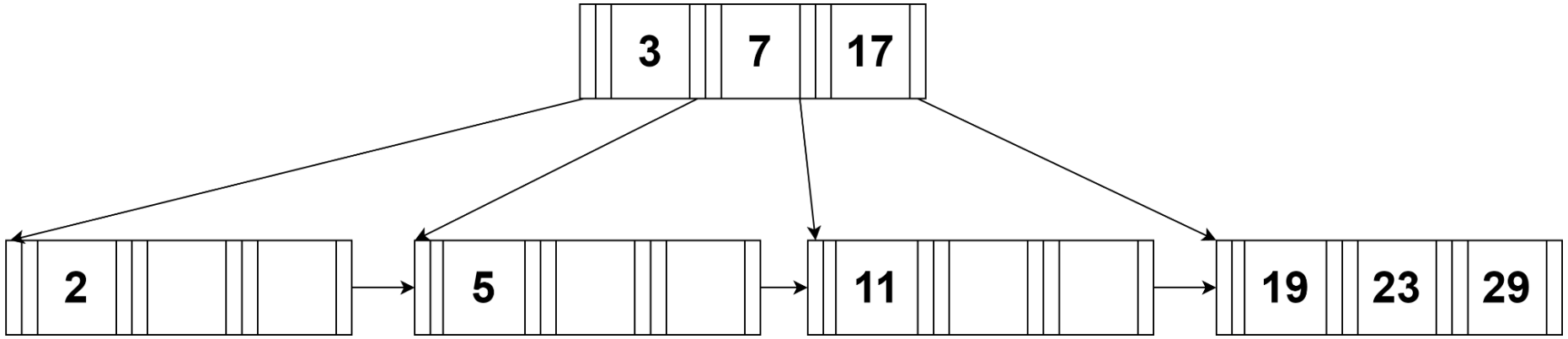




B tree

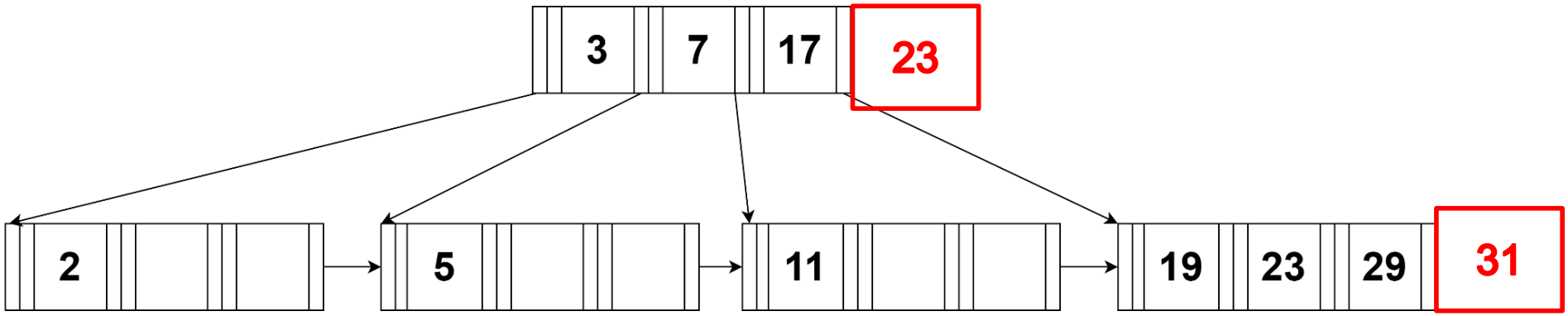
{~~2,3,5,7,11,17,19,23~~,29,31}





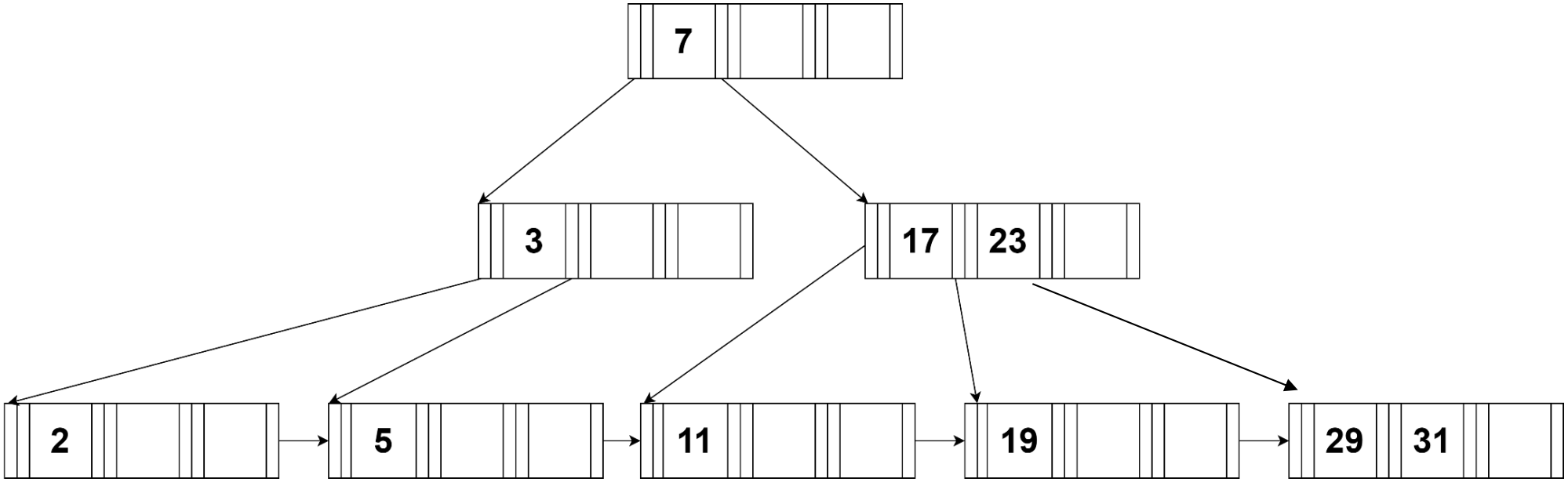
B tree

{~~2,3,5,7,11,17,19,23,29~~,31}



B tree

{~~2,3,5,7,11,17,19,23,29,31~~}



B+ tree

Κατασκευάστε ένα B+ δέντρο για το παρακάτω σύνολο από τιμές κλειδιών:

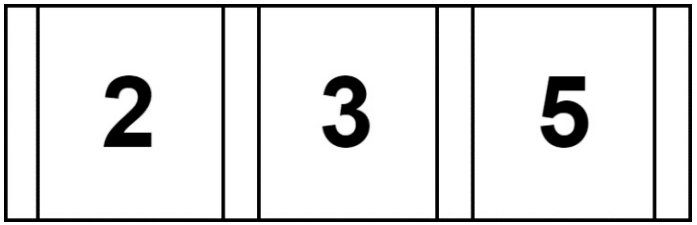
{2,3,5,7,11,17,19,23,29,31}

Το δέντρο είναι αρχικά κενό και οι τιμές προστίθενται σε αύξουσα σειρά. Ο αριθμός δεικτών σε ένα κόμβο είναι n= 4.

Στο B+ δέντρο όλοι οι κόμβοι/φύλλα πρέπει να έχουν τουλάχιστον

⌈(n-1)/2⌉

κλειδιά.

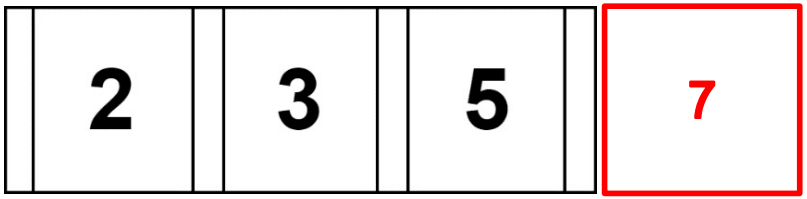


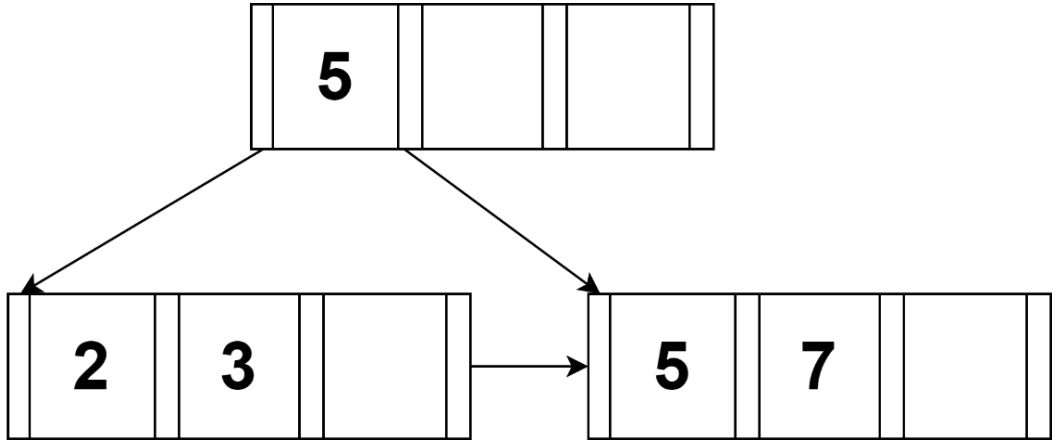
Άρα εδώ πρέπει να έχουν τουλάχιστον

2 κλειδιά.

B+ tree

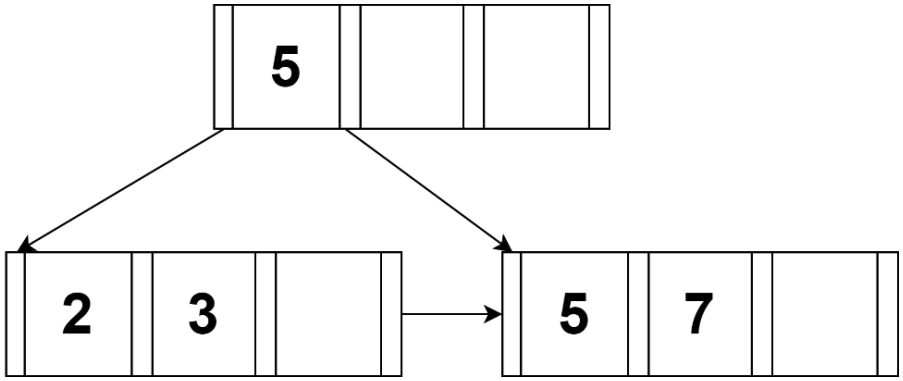
{~~2,3,5~~,7,11,17,19,23,29,31}

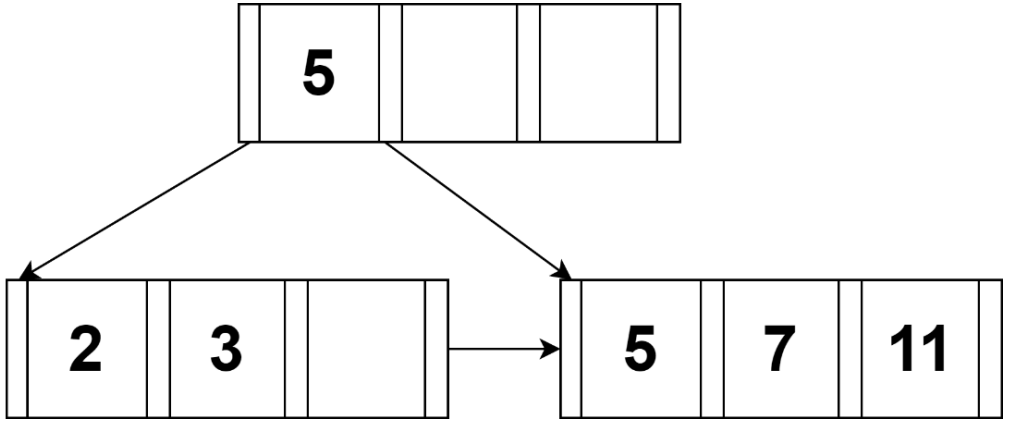




B+ tree

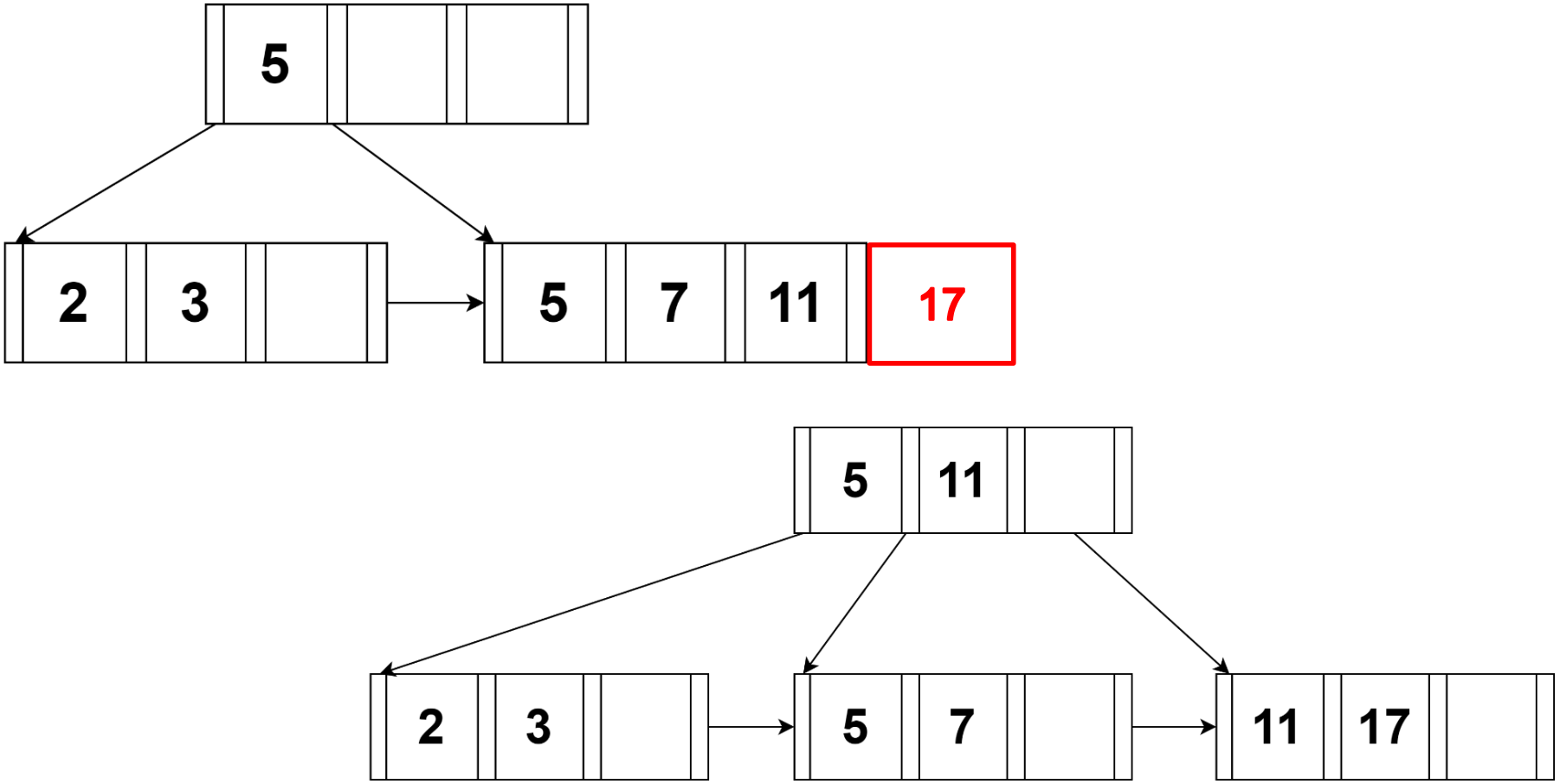
{~~2,3,5,7~~,11,17,19,23,29,31}





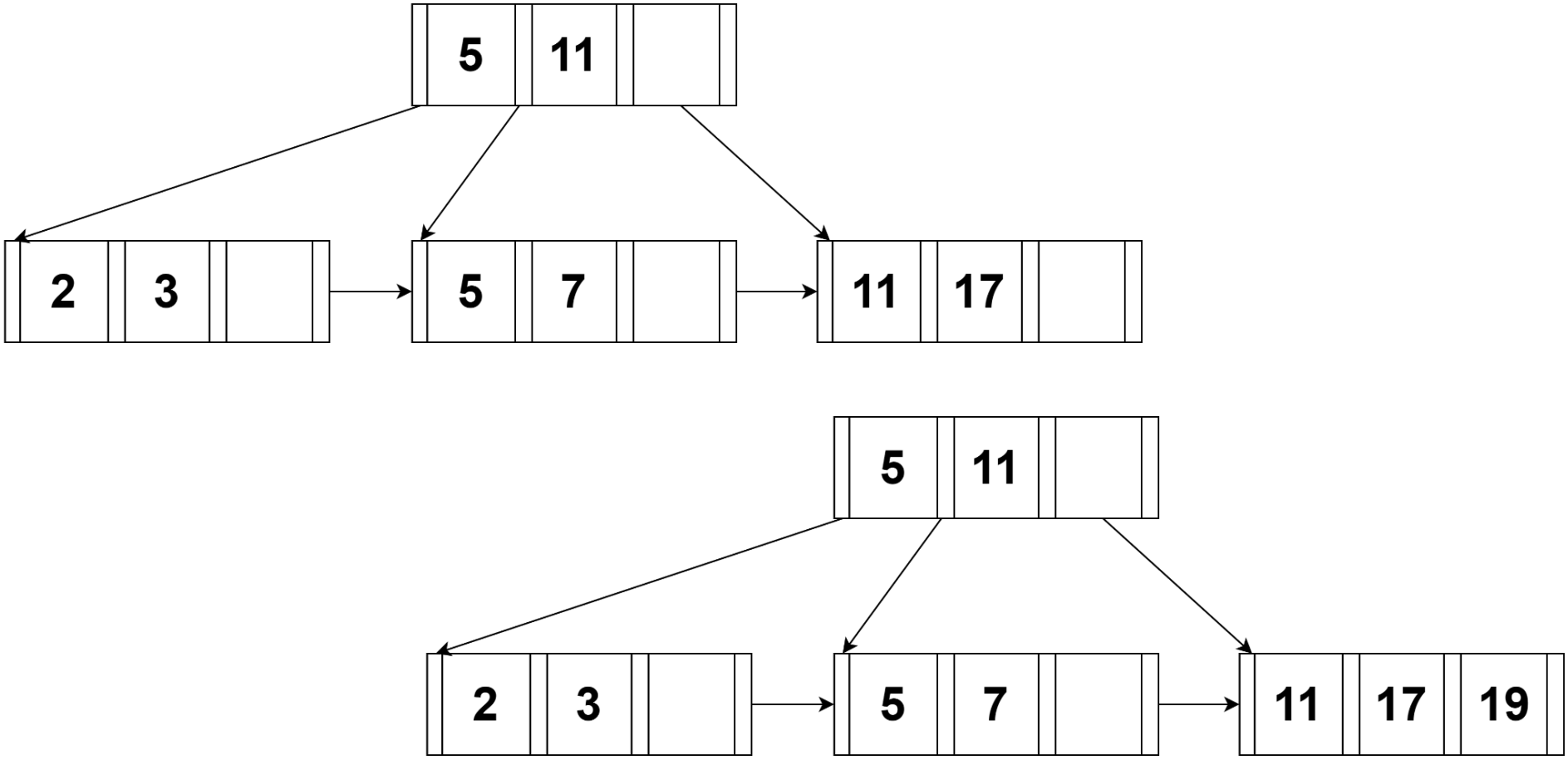
B+ tree

{~~2,3,5,7~~,~~11~~,17,19,23,29,31}



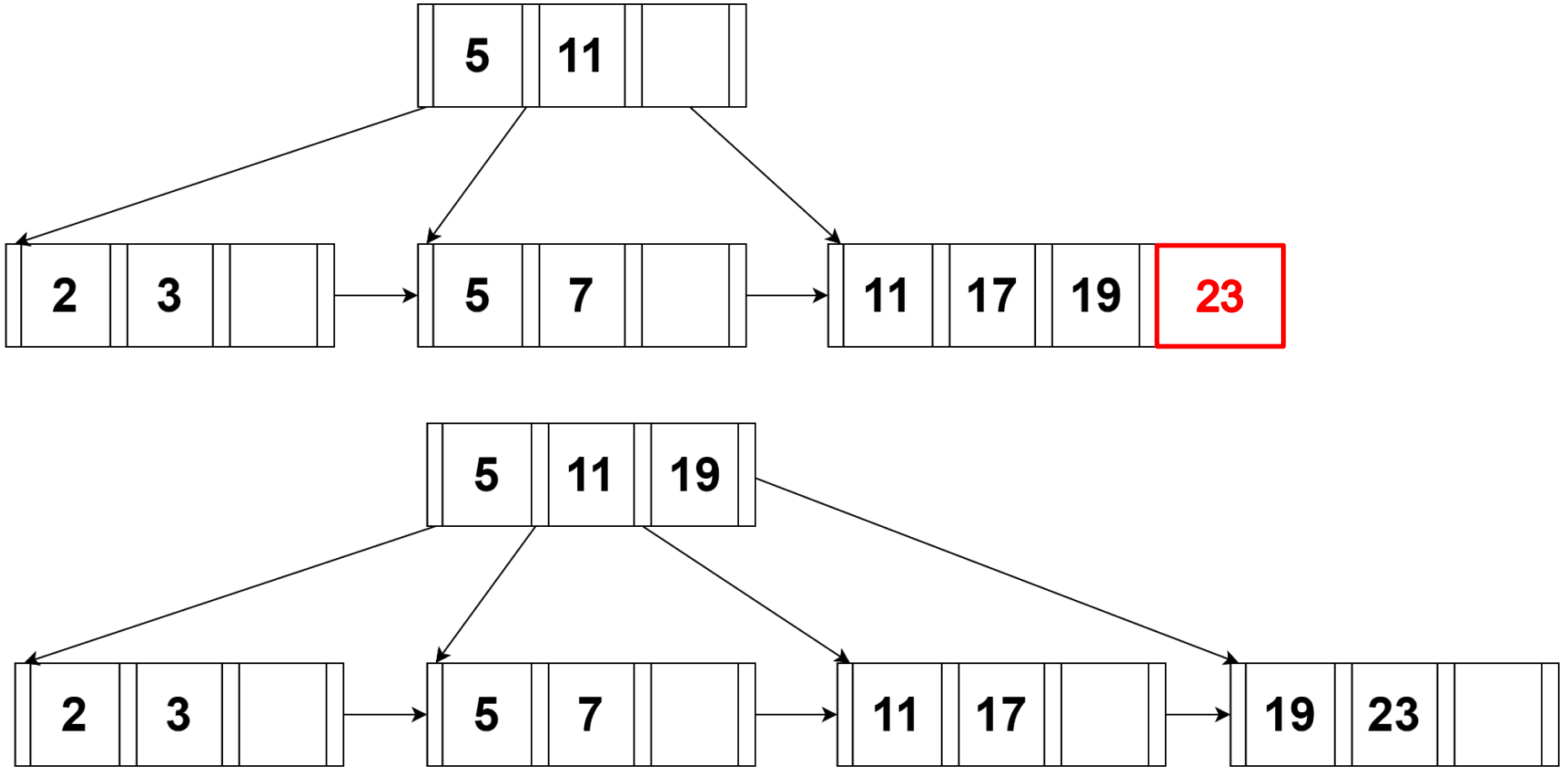
B+ tree

{~~2,3,5,7,11,17,~~19,23,29,31}



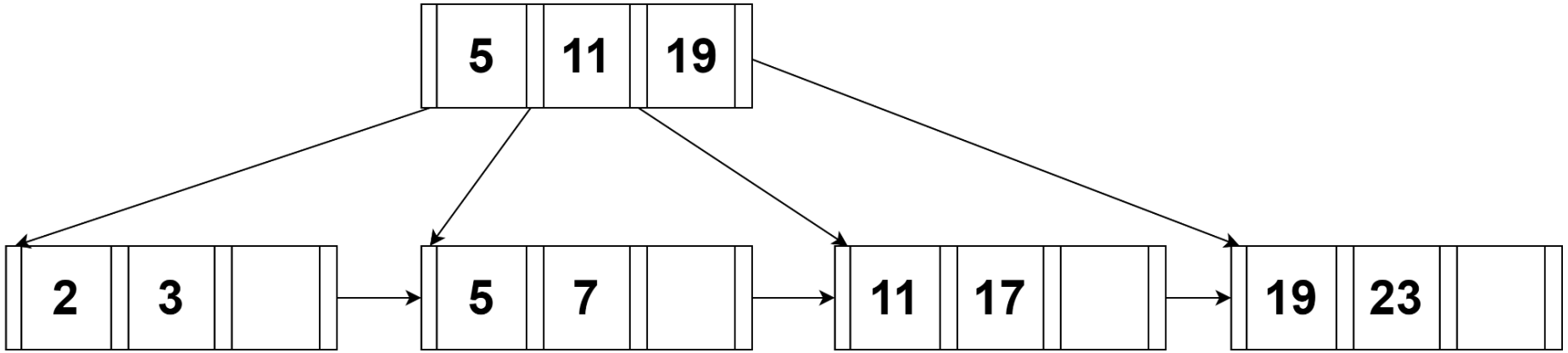
B+ tree

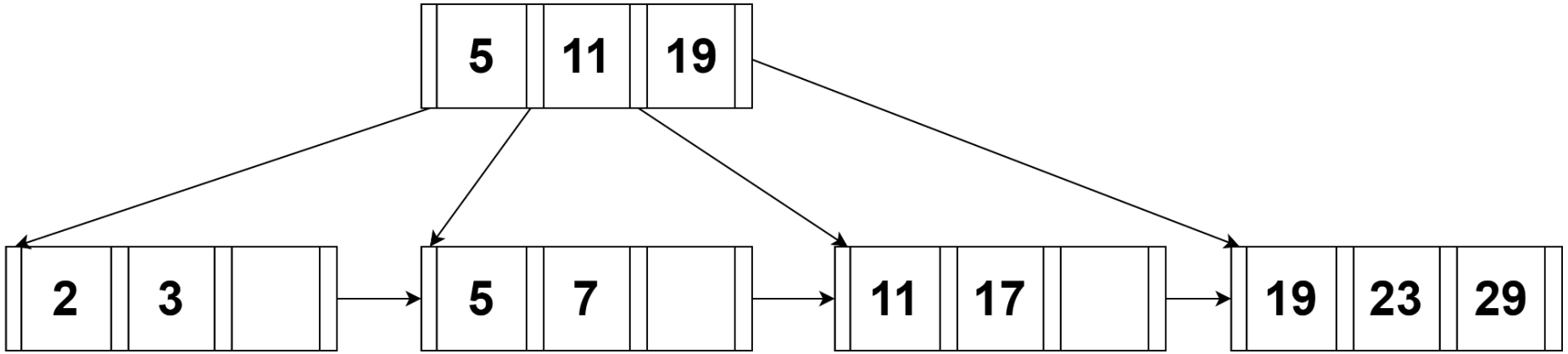
{~~2,3,5,7,11,17,19~~,23,29,31}



B+ tree

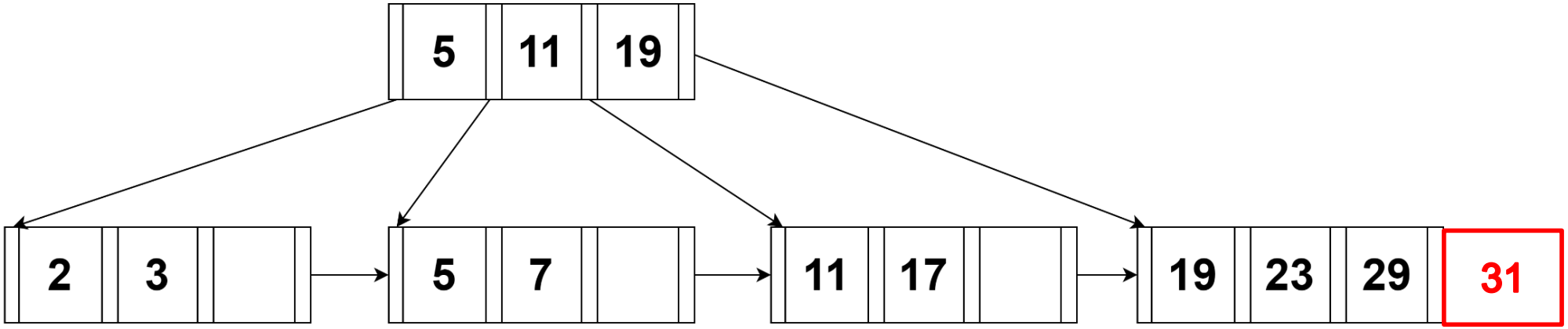
{~~2,3,5,7,11,17,19,23~~,29,31}





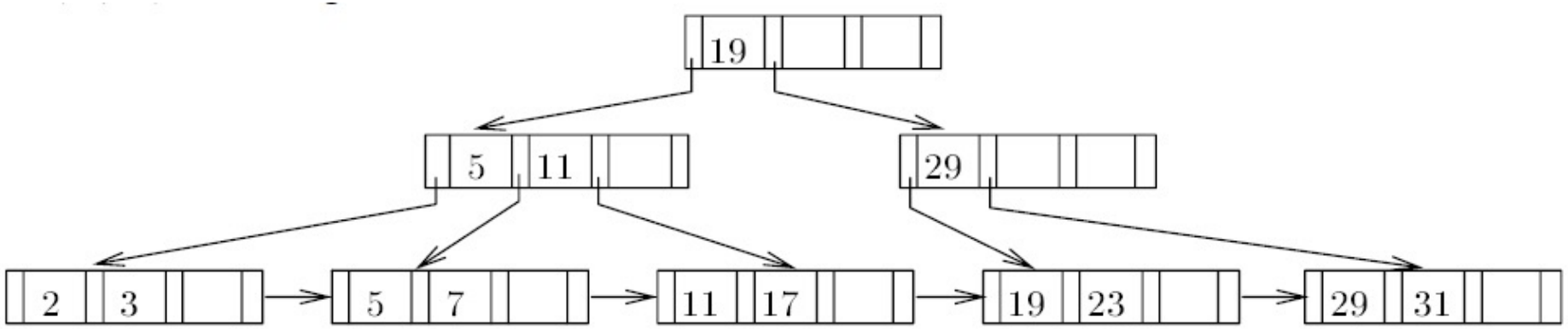
B+ tree

{~~2,3,5,7,11,17,19,23,29~~,31}



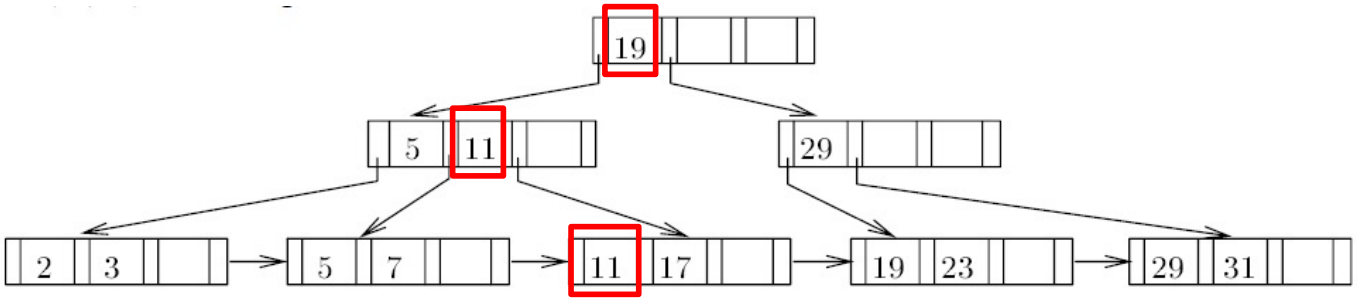
B+ tree

{~~2,3,5,7,11,17,19,23,29,31~~}



B+ tree

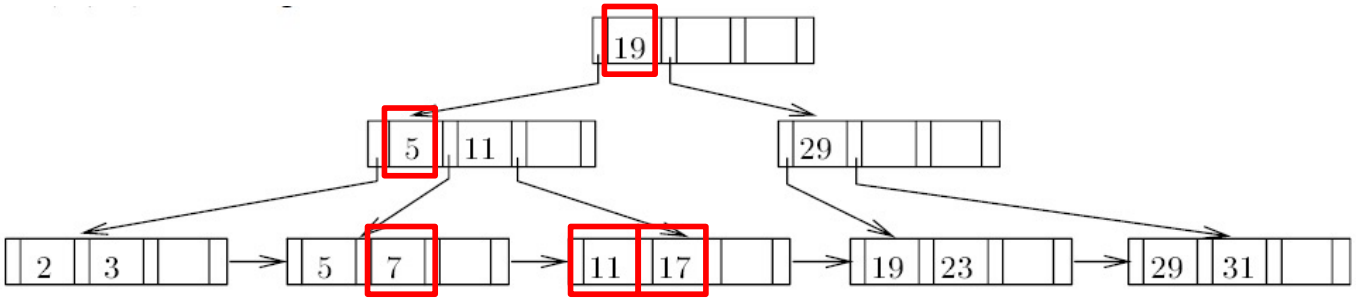
Βρείτε εγγραφές με τιμή κλειδιού αναζήτησης το 11.
Παρουσιάστε αναλυτικά τα βήματα.



B+ tree

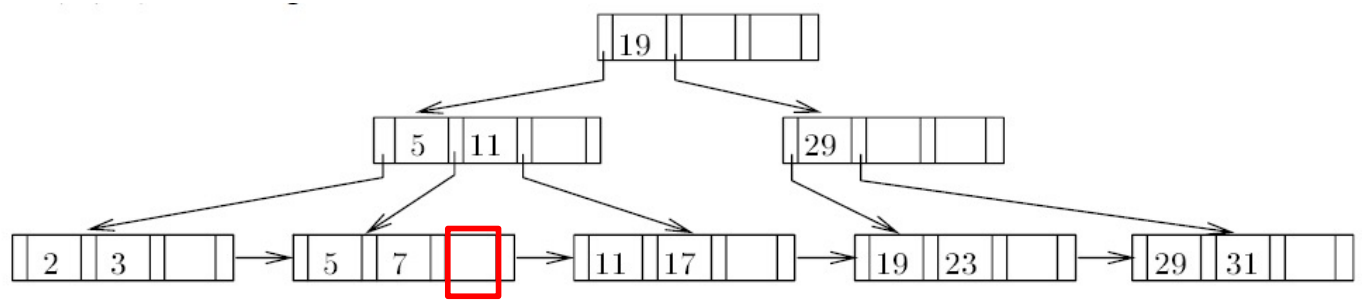
Βρείτε εγγραφές με τιμή κλειδιού αναζήτησης από το 7 μέχρι και το 17, συμπεριλαμβανομένου και των ακραίων τιμών.

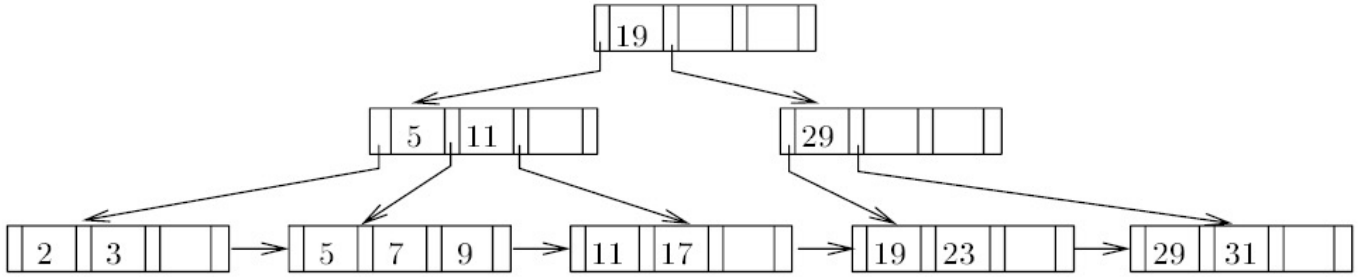
Παρουσιάστε αναλυτικά τα βήματα.



B+ tree

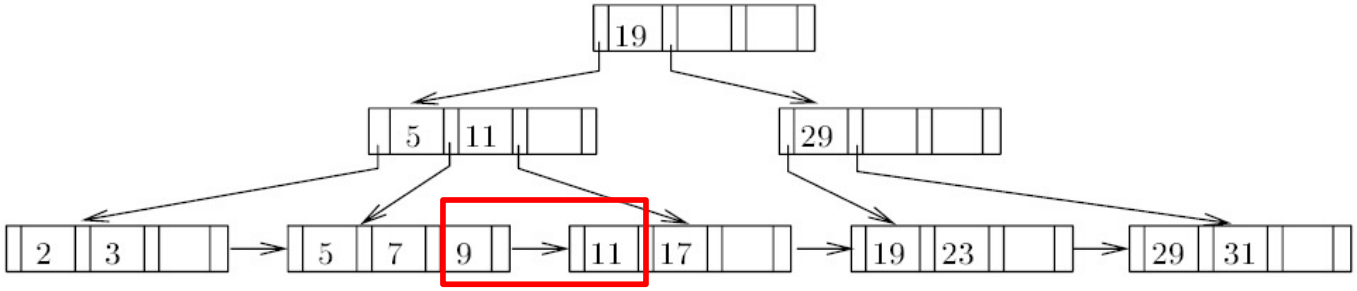
Εισάγετε το 9

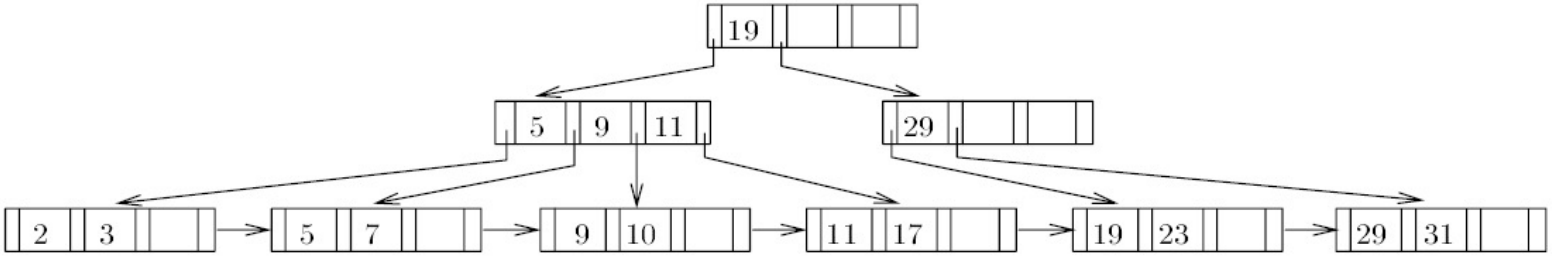




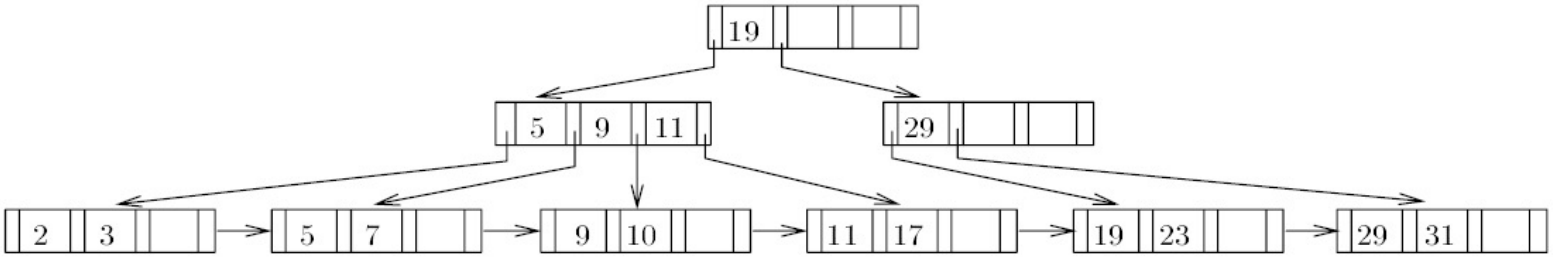
B+ tree

Εισάγετε το 10





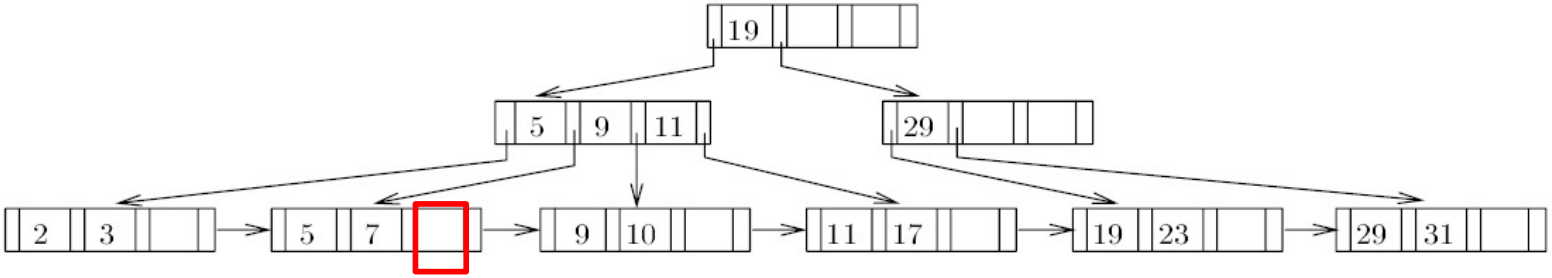
B+ tree

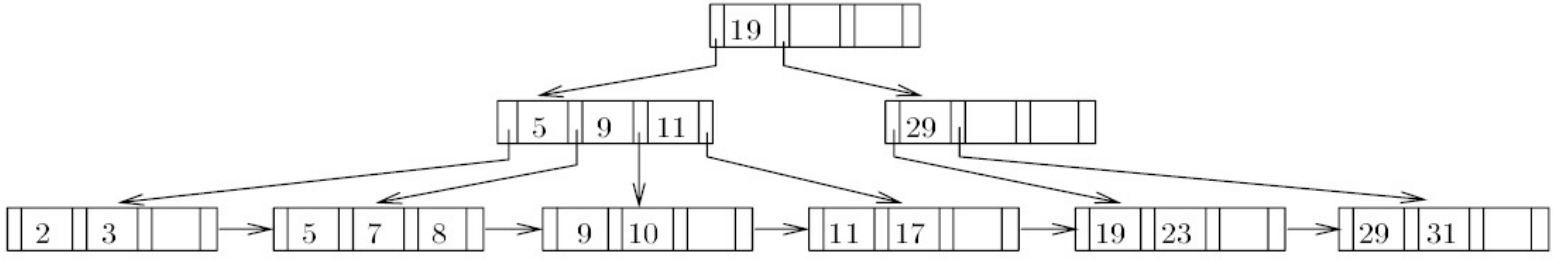


Κανόνας συνένωσης κόμβων/φύλλων:
Αν στο παραπάνω δέντρο διαγράφαμε το 9 τότε θα έπρεπε να ενώσουμε το φύλλο [10] με το **αριστερό** φύλλο [5,7]. Η επιλογή του αριστερού φύλλου είναι σύμβαση, το δέντρο θα υπάκουε όλους τους κανόνες του B+ δέντρου ακόμα και αν η συνένωση γινόταν με το δεξί.
ΠΡΟΣΟΧΗ: Αν διαγράφαμε το 19 τότε το φύλλο [23] θα συνενωνόταν με το [29,31] και όχι με το [11,17]. Πρώτη προτεραιότητα της συνένωσης είναι τα φύλλα να έχουν τον ίδιο πατέρα. (Μια τέτοια διαγραφή θα προκαλούσε αλλαγές και στο δεύτερο επίπεδο του δέντρου {Διαφάνεια 26})

B+ tree

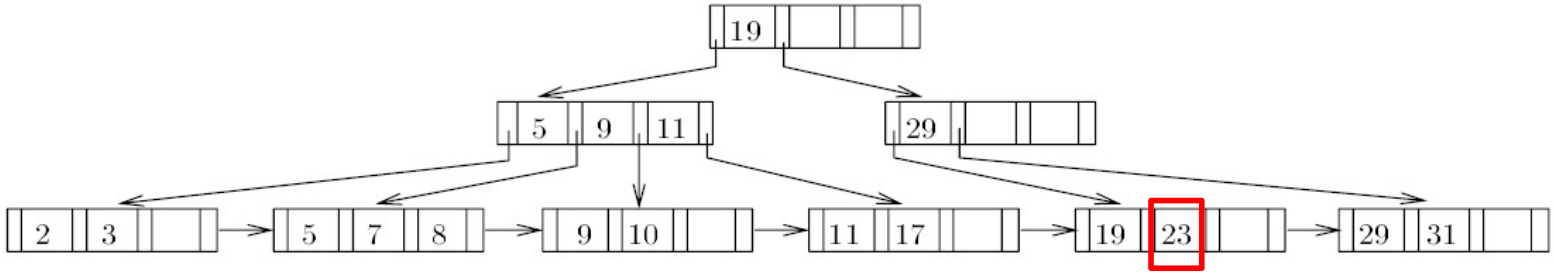
Εισάγετε το 8

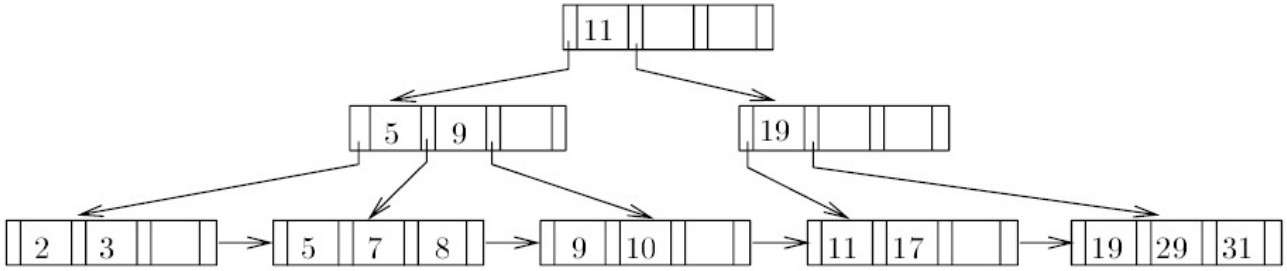




B+ tree

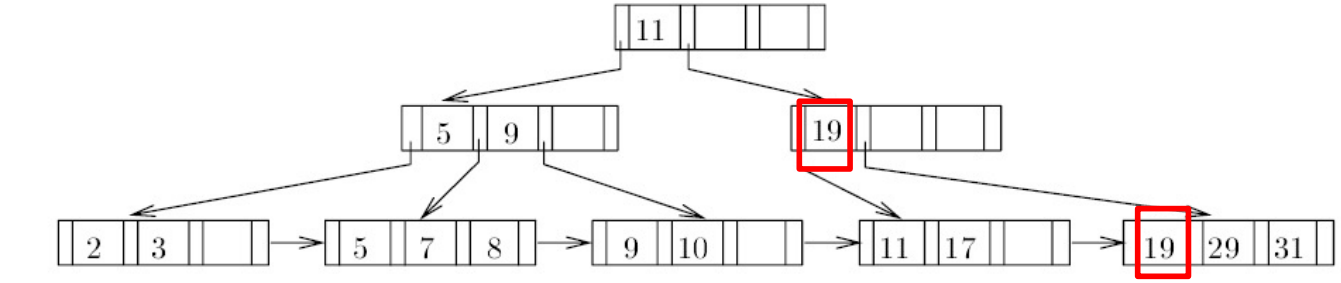
Διαγράψτε το 23

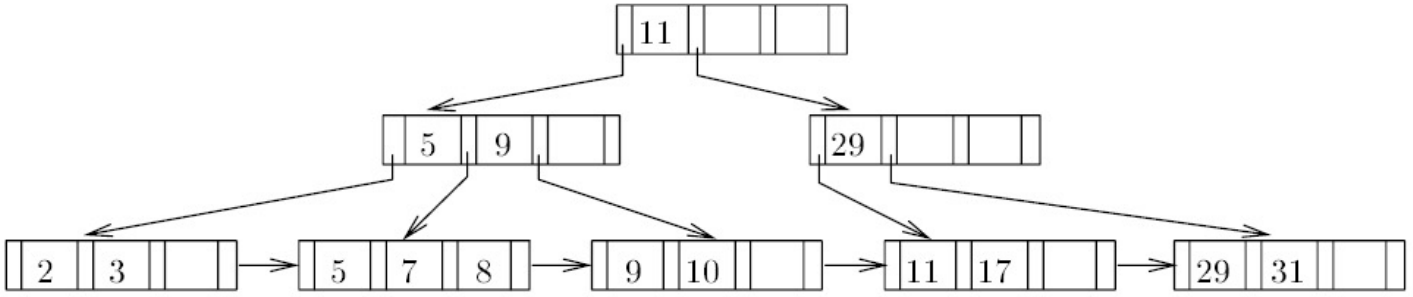




B+ tree

Διαγράψτε το 19





Επεκτάσιμο Hash

Υποθέστε ότι χρησιμοποιείτε επεκτάσιμο hash σε ένα αρχείο και σας δίνει τα παρακάτω αποτελέσματα:

{2,3,5,7,11,17,19,23,29,31}

Δείξτε την δομή hash για αυτές τις τιμές και κάθε bucket μπορεί να περιέχει τρεις εγγραφές.

Χρησιμοποιείστε τα λιγότερα σημαντικά ψηφία.

Επεκτάσιμο Hash

{2,3,5,7,11,17,19,23,29,31}

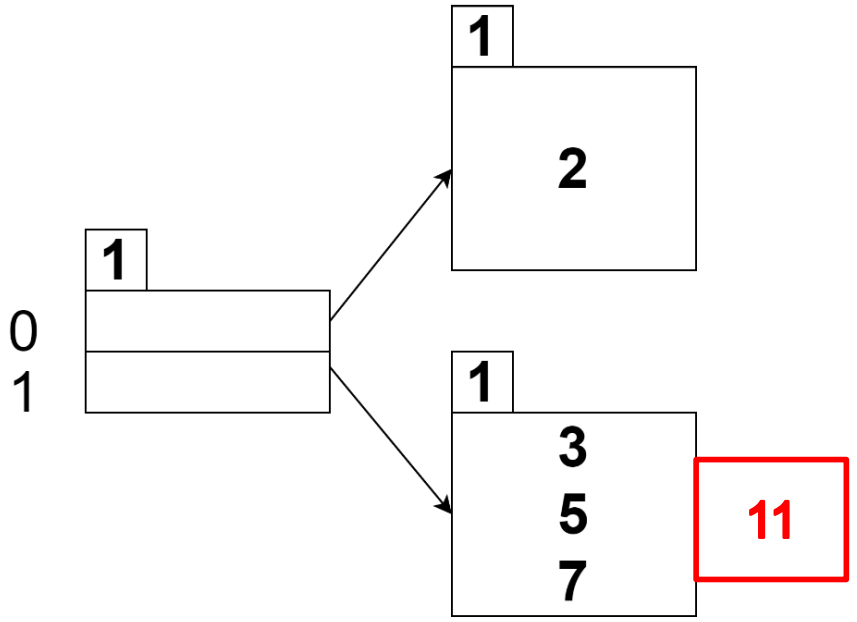
{00010,00011,00101,00111,01011,10001,10011,10111,11101,11111}

2	**00010**
3	**00011**
5	**00101**
7	**00111**
11	**01011**
17	**10001**
19	**10011**
23	**10111**
29	**11101**
31	**11111**

Επεκτάσιμο Hash

{~~2,3,5,7~~,11,17,19,23,29,31}

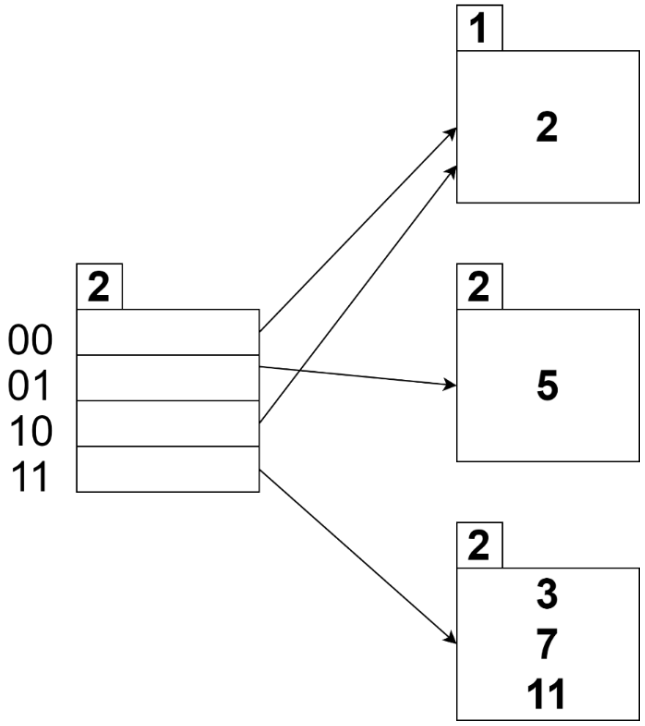
{~~00010,00011,00101,00111~~,01011,10001,10011,10111,11101,11111}



Επεκτάσιμο Hash

{~~2,3,5,7,11~~,17,19,23,29,31}

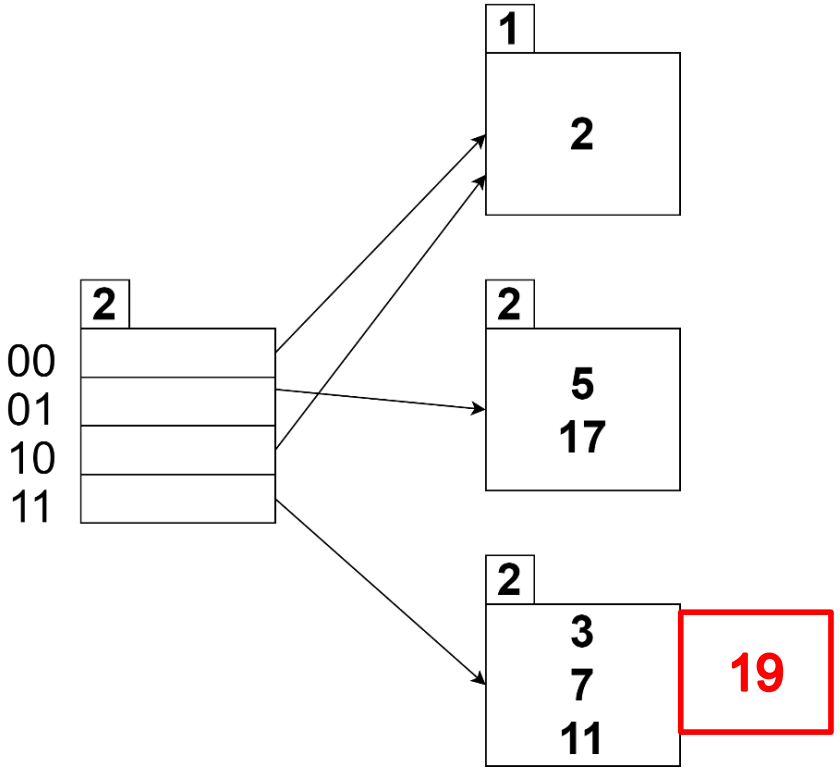
{~~00010,00011,00101,00111,01011~~,10001,10011,10111,11101,11111}



Επεκτάσιμο Hash

{~~2,3,5,7,11,17,~~19,23,29,31}

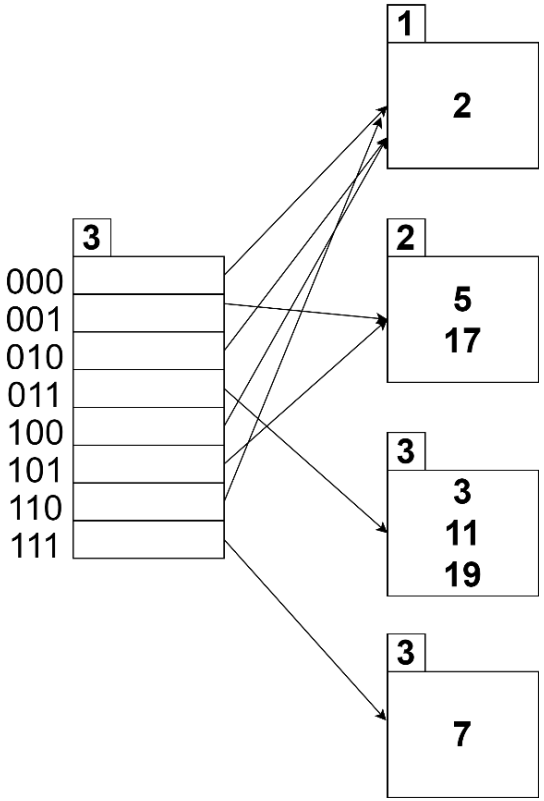
{~~00010,00011,00101,00111,01011,10001,~~10011,10111,11101,11111}



Επεκτάσιμο Hash

{~~2,3,5,7,11,17,19~~,23,29,31}

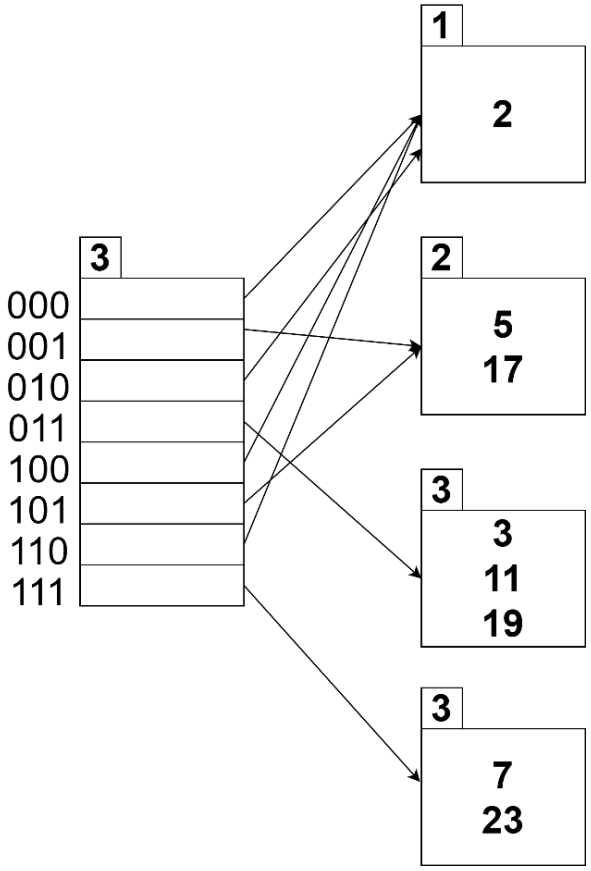
{~~00010,00011,00101,00111,01011,10001,10011,~~10111,11101,11111}



Επεκτάσιμο Hash

{~~2,3,5,7,11,17,19,23,~~29,31}

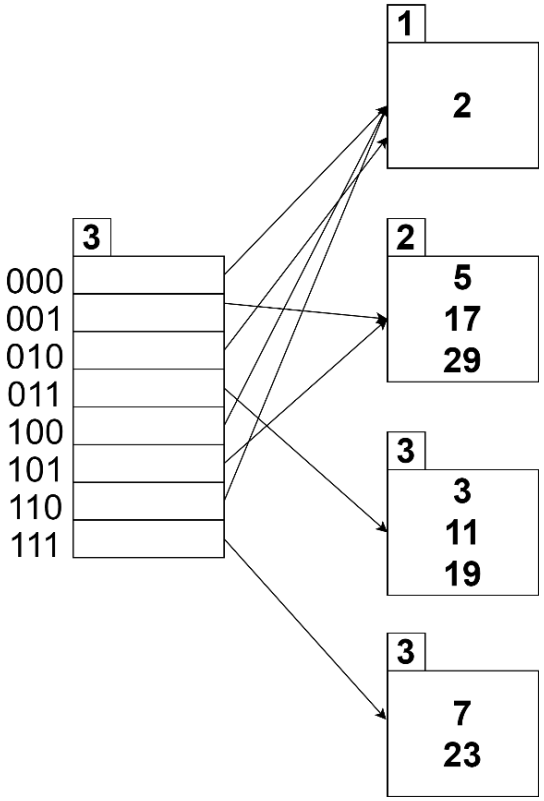
{~~00010,00011,00101,00111,01011,10001,10011,10111,~~11101,11111}



Επεκτάσιμο Hash

{~~2,3,5,7,11,17,19,23,29~~,31}

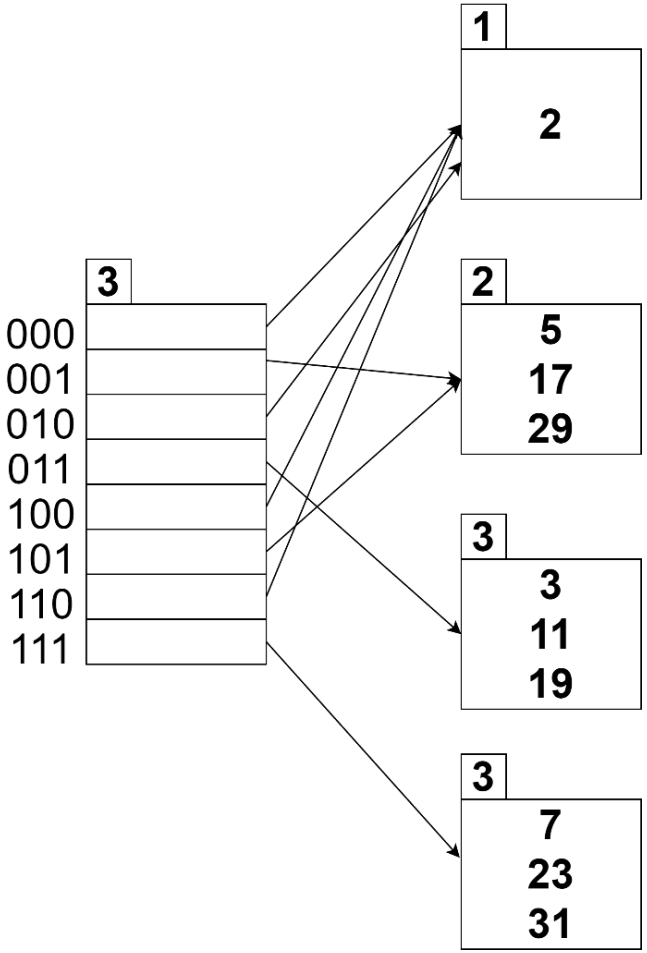
{~~00010,00011,00101,00111,01011,10001,10011,10111,11101~~,11111}



Επεκτάσιμο Hash

~~{2,3,5,7,11,17,19,23,29,31}~~

~~{00010,00011,00101,00111,01011,10001,10011,10111,11101,11111}~~



Επεκτάσιμο Hash

Υποθέστε ότι χρησιμοποιείτε επεκτάσιμο hash σε ένα αρχείο που περιέχει εγγραφές με τις παρακάτω τιμές:

{2,3,5,7,11,17,19,23,29,31}

Δείξτε την δομή hash για αυτό το αρχείο, αν η συνάρτηση είναι h(x)=x mod 8, και κάθε bucket μπορεί να περιέχει τρεις εγγραφές.

Χρησιμοποιείστε τα πιο σημαντικά ψηφία.

Επεκτάσιμο Hash

{2,3,5,7,11,17,19,23,29,31}

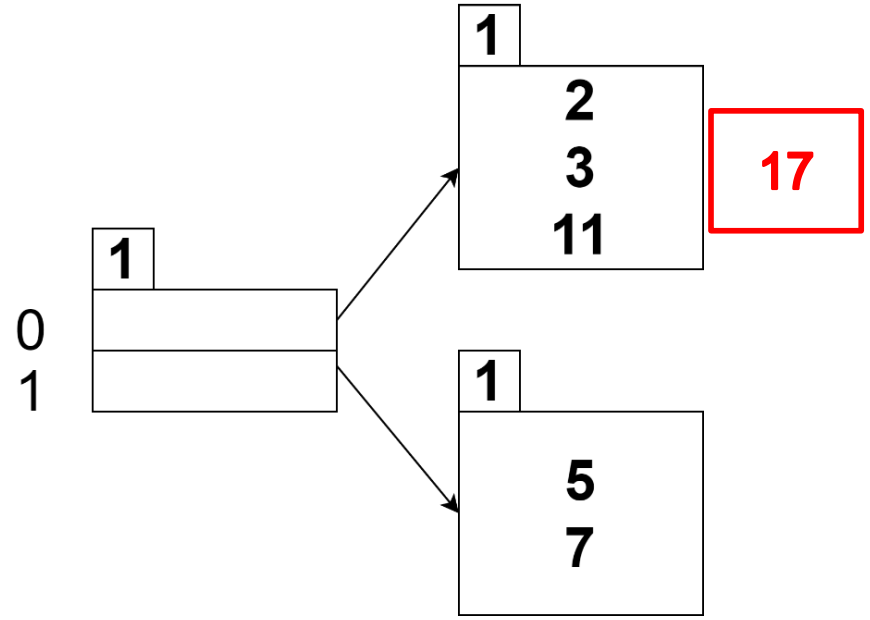
{010,011,101,111,011,001,011,111,101,111}

2 mod 8	**010**
3 mod 8	**011**
5 mod 8	**101**
7 mod 8	**111**
11 mod 8	**011**
17 mod 8	**001**
19 mod 8	**011**
23 mod 8	**111**
29 mod 8	**101**
31 mod 8	**111**

Επεκτάσιμο Hash

{~~2,3,5,7,11~~,17,19,23,29,31}

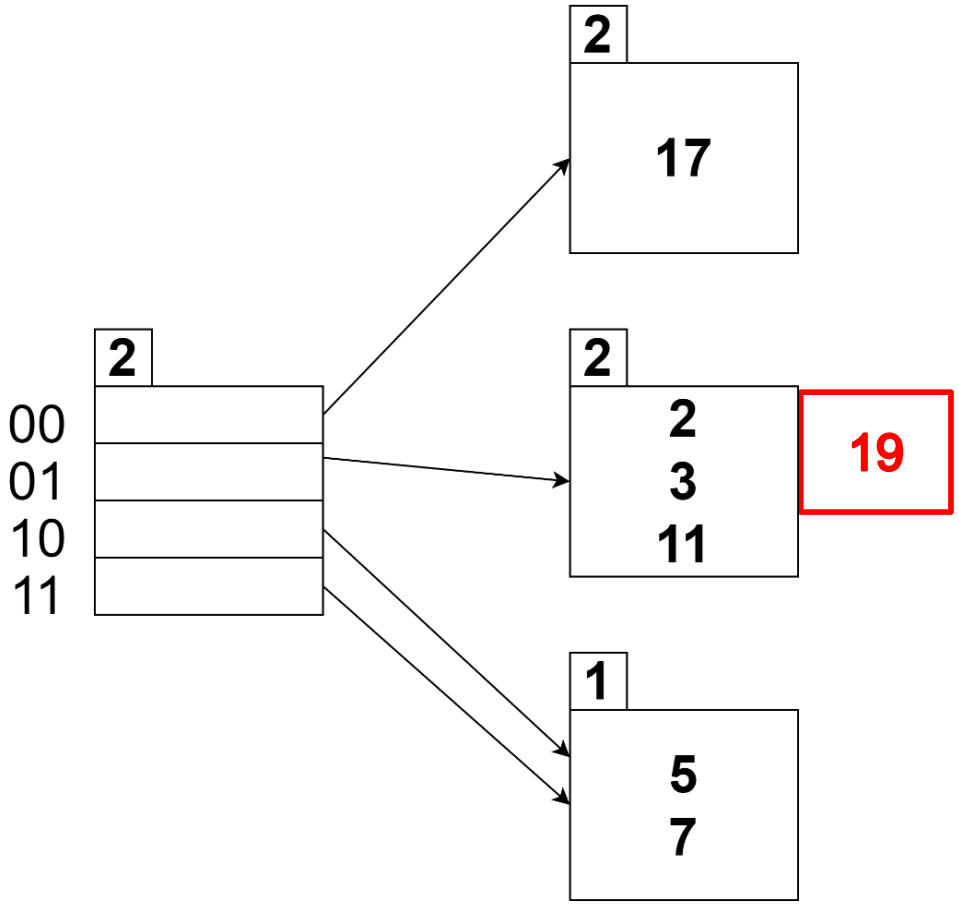
{~~010,011,101,111,011~~,001,011,111,101,111}



Επεκτάσιμο Hash

{~~2,3,5,7,11,17,~~19,23,29,31}

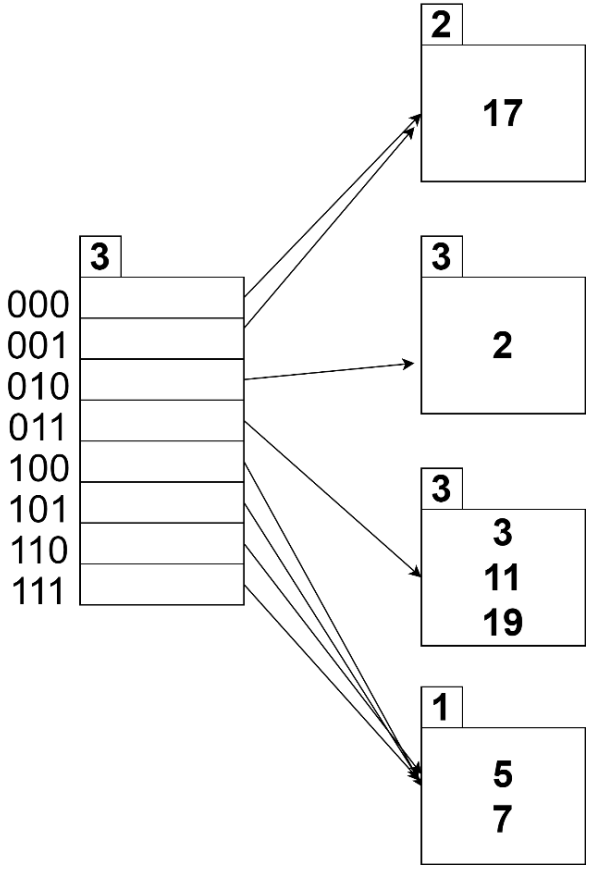
{~~010,011,101,111,011,001,~~011,111,101,111}



Επεκτάσιμο Hash

{~~2,3,5,7,11,17,19~~,23,29,31}

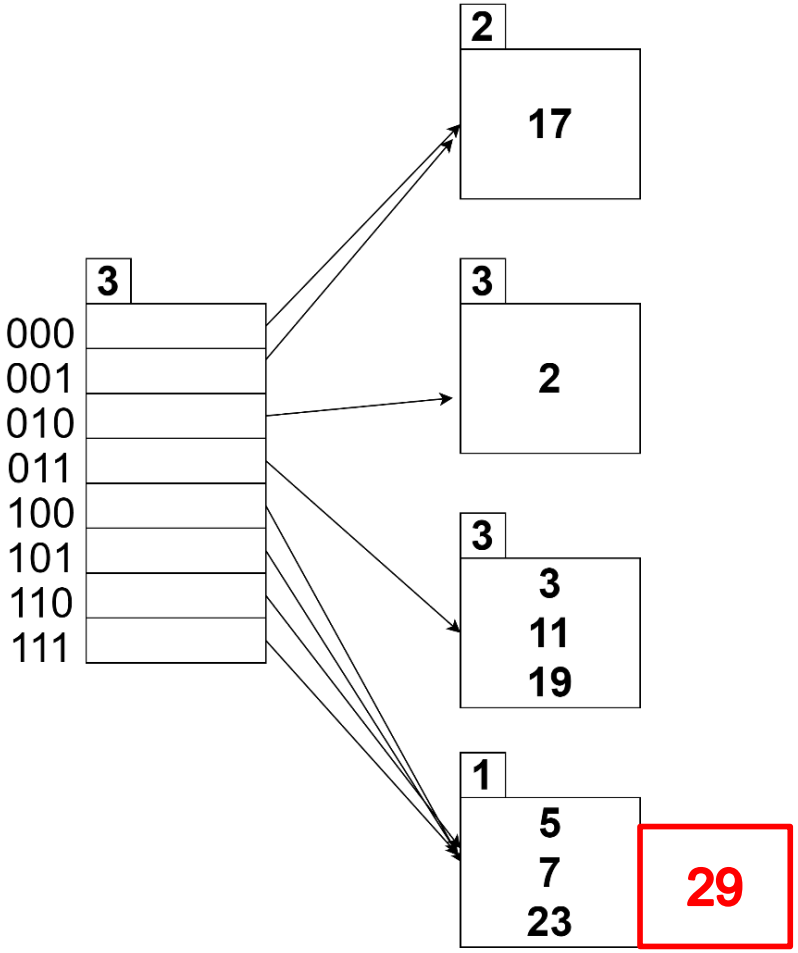
{~~010,011,101,111,011,001,011~~,111,101,111}



Επεκτάσιμο Hash

{~~2,3,5,7,11,17,19,23~~,29,31}

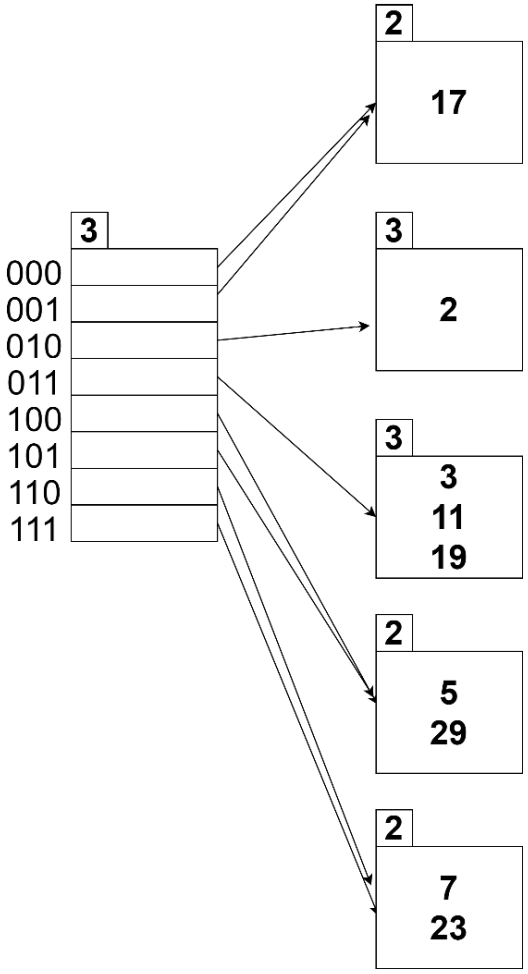
{~~010,011,101,111,011,001,011,111~~,101,111}



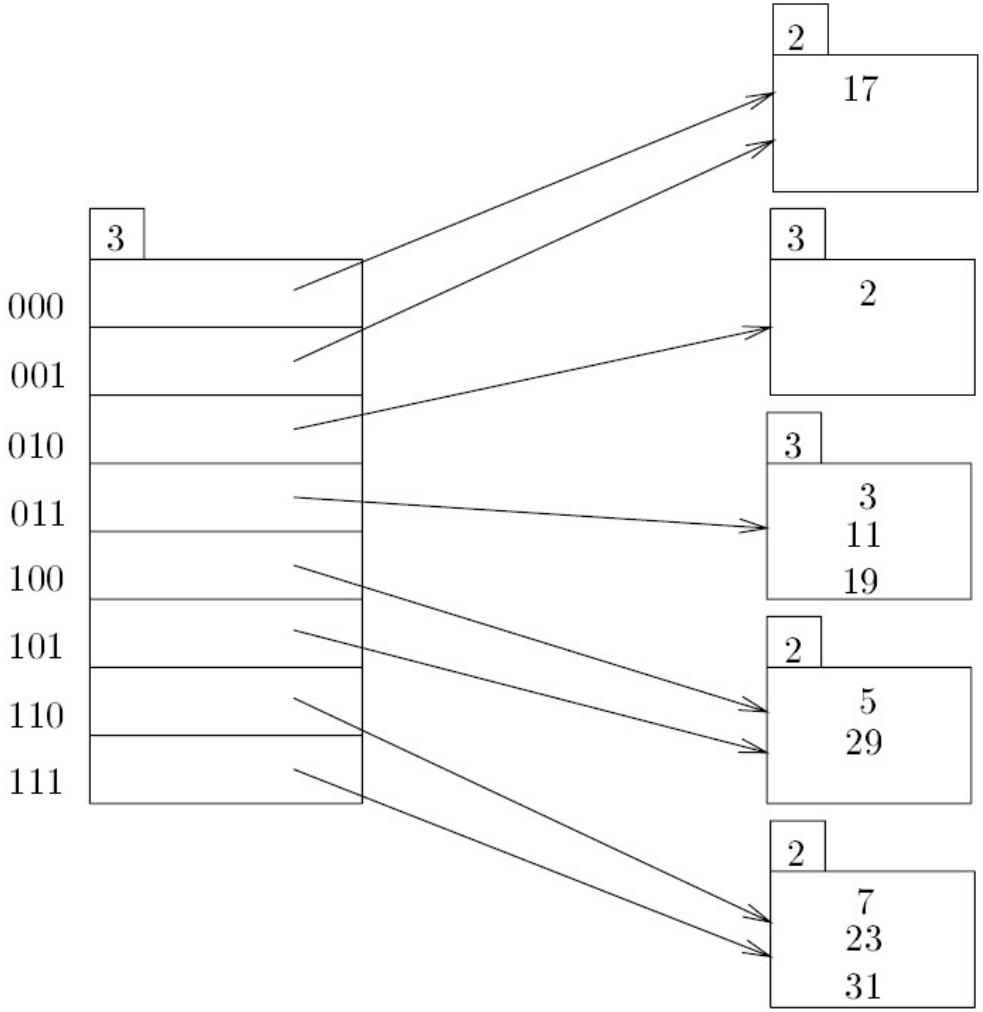
Επεκτάσιμο Hash

{~~2,3,5,7,11,17,19,23,29~~,31}

{~~010,011,101,111,011,001,011,111,101~~,111}



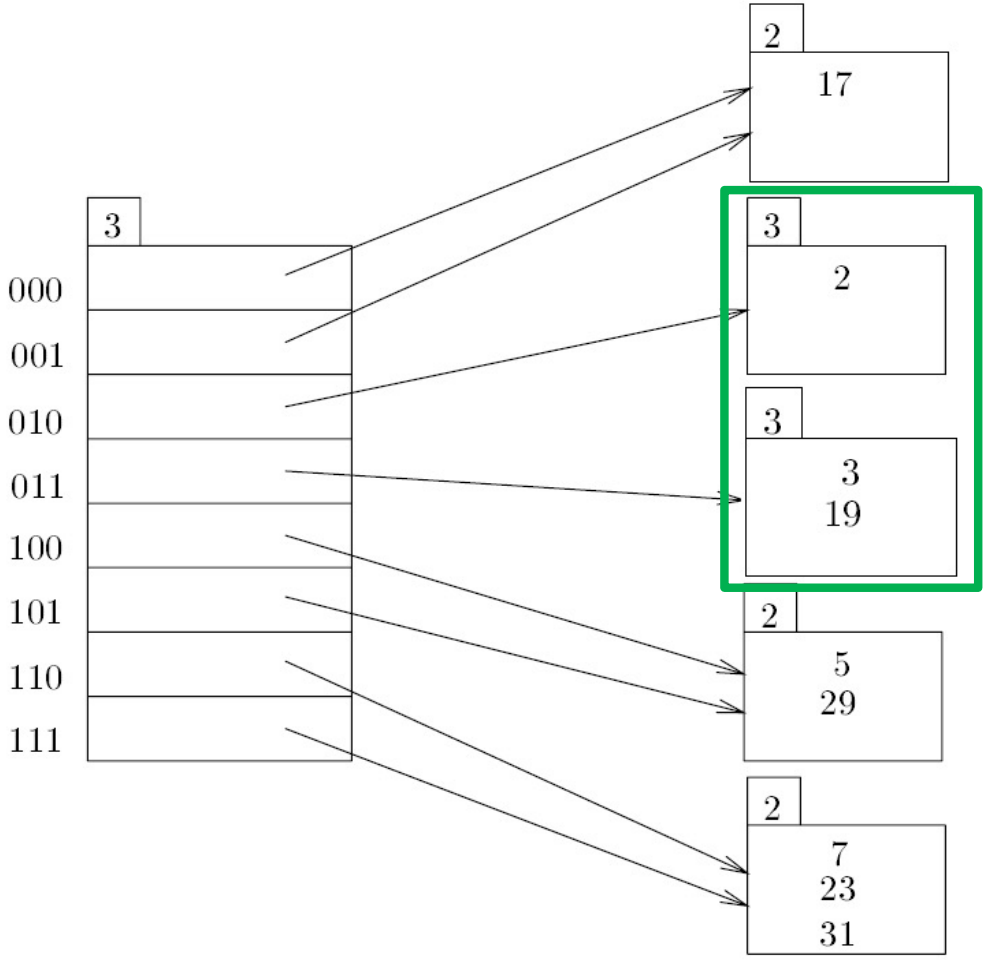
Επεκτάσιμο Hash



Επεκτάσιμο Hash

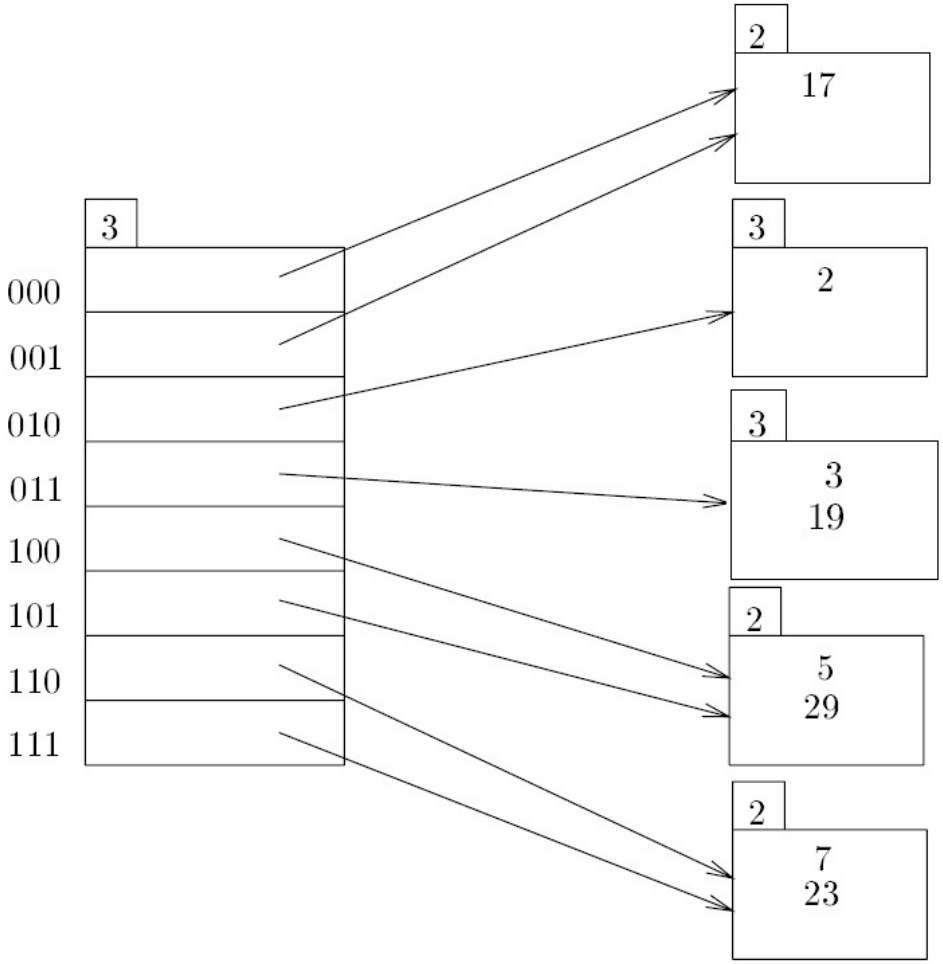
Διαγραφή του 11

Στο επεκτάσιμο Hash δεν έχουμε συνένωση των buckets όταν αυτά αδειάζουν!



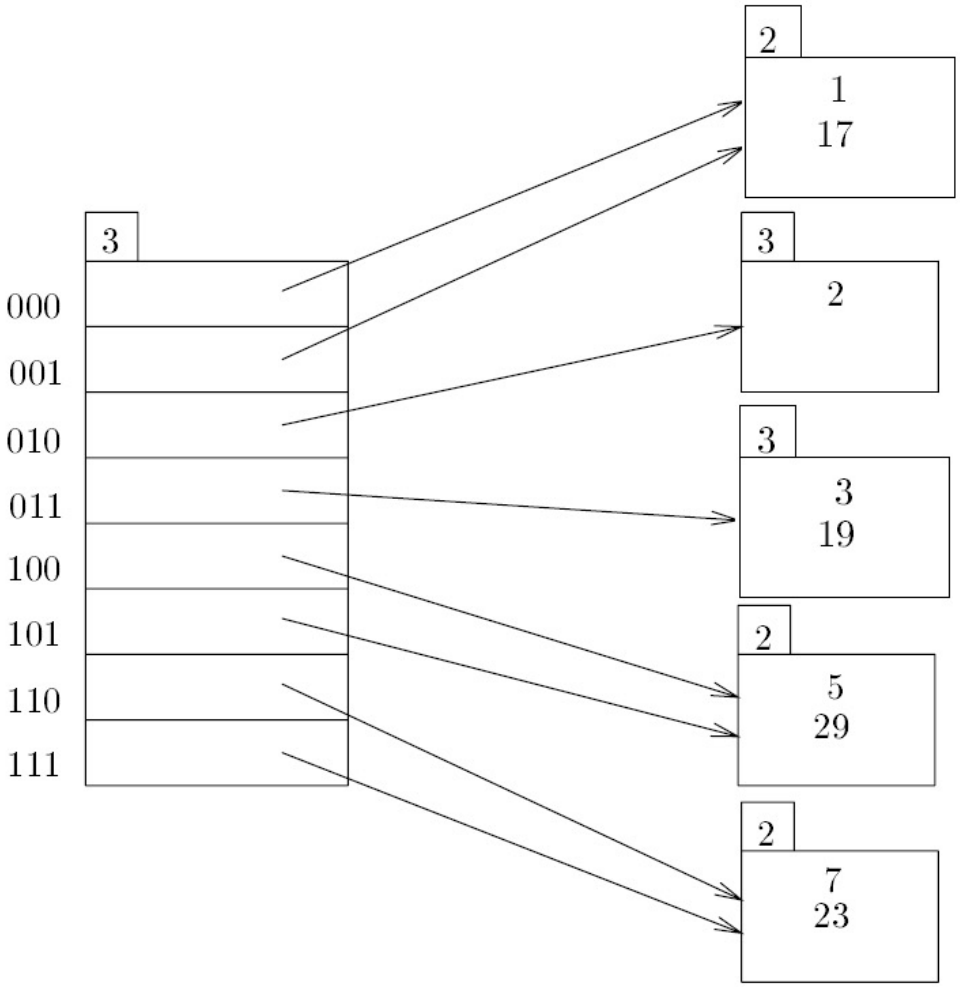
Επεκτάσιμο Hash

Διαγραφή του 31



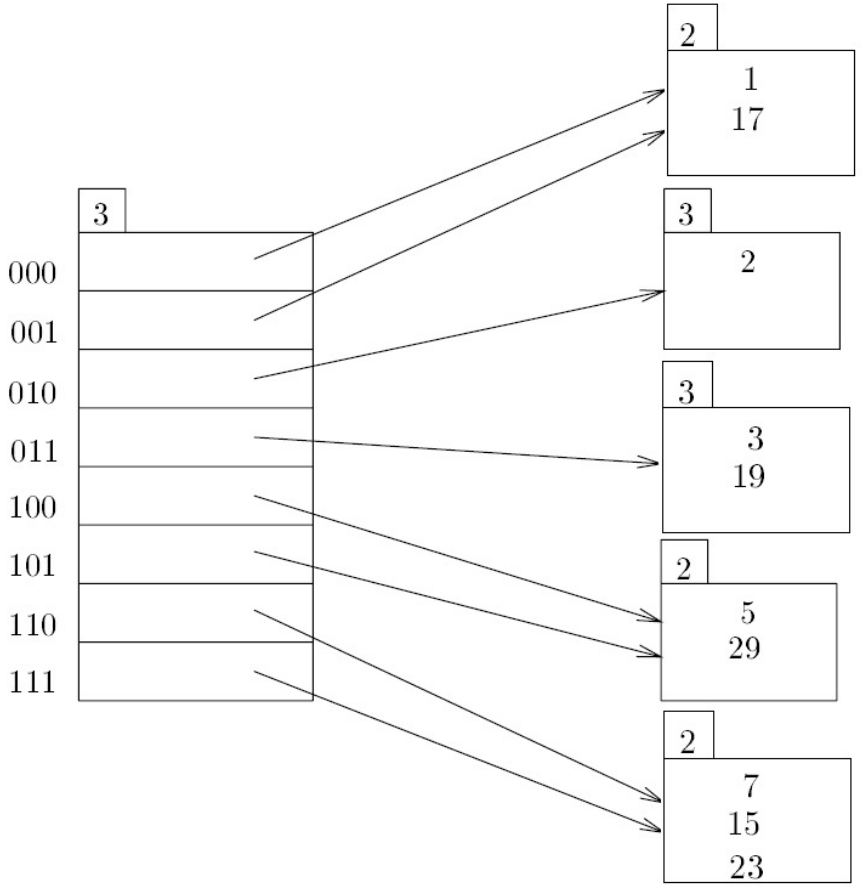
Επεκτάσιμο Hash

Εισαγωγή του 1 (001)



Επεκτάσιμο Hash

Εισαγωγή του 15 (111)



Υπερχείλιση bucket

Αιτίες:

- Η αρχική εκτίμηση για τον αριθμό των εγγραφών της σχέσης ήταν πολύ μικρή με αποτέλεσμα να έχουν δωθεί λιγότεροι buckets από τους αναγκαίους
- Ανομαλίες στην κατανομή των εγγραφών στους buckets
  - Πολλές εγγραφές με την ίδια τιμή κλειδιού αναζήτησης
  - Λανθασμένη επιλογή της συνάρτησης hash, χωρίς τις κατάλληλες ιδιότητες για ομοιομορφία και τυχαιότητα.

Λύσεις:

- Προσεκτική επιλογή της συνάρτησης hash με βάση τις ιδιότητες του γνωρίσματος
- Προσεκτική εκτίμηση του μεγέθους της σχέσης
- Δέσμευση περισότερου χώρου αρχικά μέχρι και 20% για να περιοριστούν οι συνέπειες των ανωμαλιών στην κατανομή και να περιοριστούν τα overflows

Κλειδιά ανά block

Block δίσκου: 500 bytes

R(A, B, C), ευρετήριο στο A

A 23 bytes

Δείκτης 15 bytes

Κλειδιά και δείκτες ανά block?

Έστω n κλειδιά:

(n+1)*15 + n*23 = 500

n*38=485

n=12.76

n=12

Άρα έχω 12 κλειδιά και 13 δείκτες ανά block δίσκου

Φύλλα B+ tree

1000 εγγραφές

R(A, B, C, D)

Ευρετήριο στο C

14 κλειδιά ανά block δίσκου

$$\left\lceil \frac{1000}{14} \right\rceil = \lceil 71.4 \rceil = 72$$

Χρειάζονται 72 φύλλα για να αποθηκευτεί το ευρετήριο

΄Υψος B+ tree

83 φύλλα

6 κλειδιά

Επίπεδα στο δέντρο?

6 κλειδιά -> 7 δείκτες

$$\left\lceil \frac{83}{7} \right\rceil = \lceil 11.85 \rceil = 12$$ blocks στο επίπεδο πάνω από τα φύλλα

$$\left\lceil \frac{12}{7} \right\rceil = \lceil 1.71 \rceil = 2$$ blocks στο επόμενο επίπεδο

1 block η ρίζα, άρα το δέντρο έχει 4 επίπεδα

Linear Hashing

Idea:

Use a family of hash functions h_0, h_1, h_2, ...

– $h_i(key) = h(key) \bmod (2^i N)$; N = initial # buckets

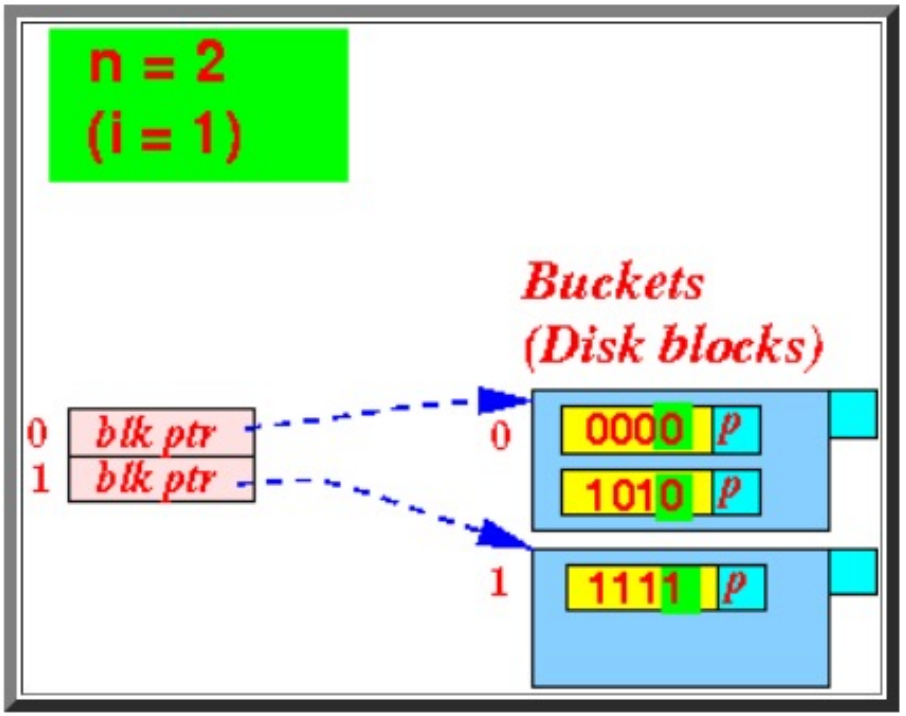
– h is some hash function (range is 0 to 2|MachineBitLength|)

– If $N = 2^{d0}$, for some d_0, h_i consists of applying h and looking at the last d_i bits, where $d_i = d_0 + i$.

– h_{i+1} doubles the range of h_i (similar to directory doubling)

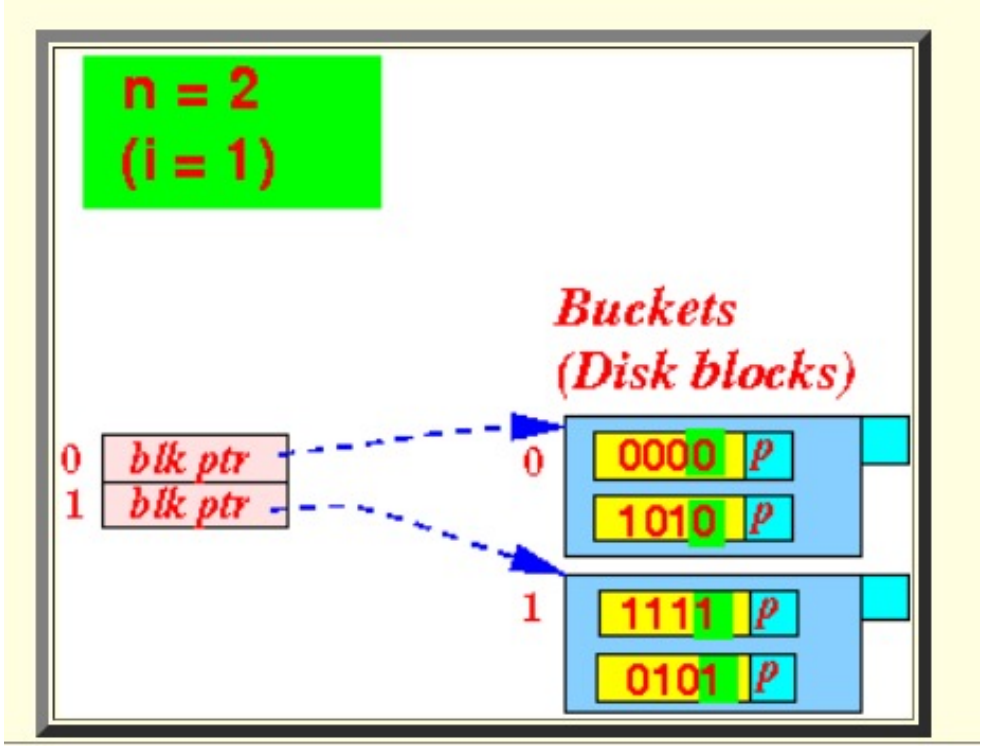
Linear Hashing

85%
πληρότητα
διάσπασης

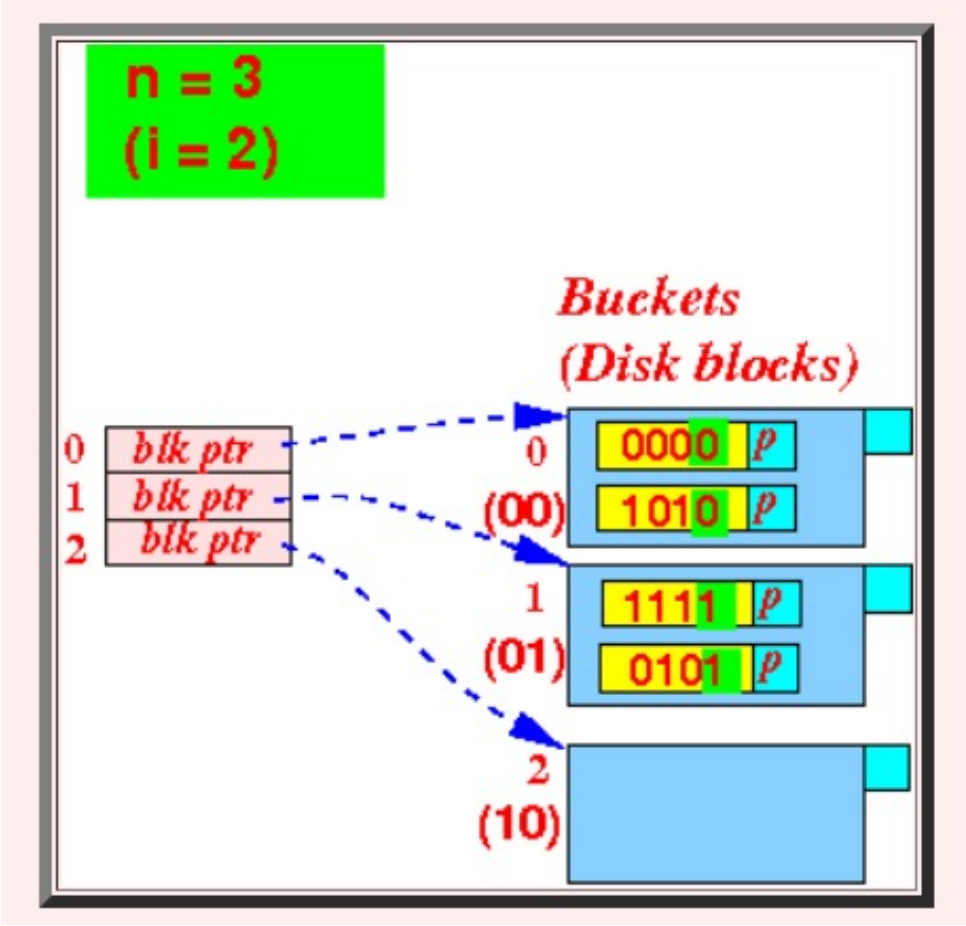


Linear Hashing

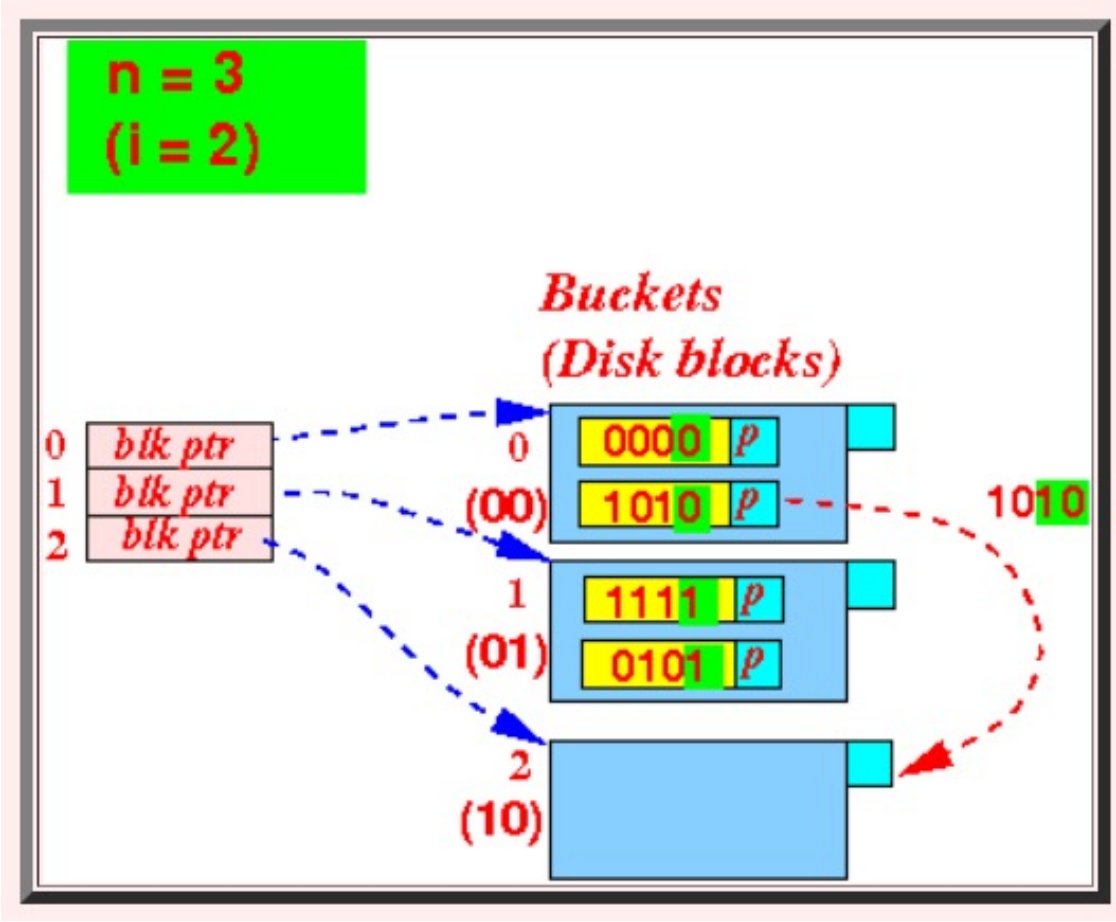
Εισάγετε το 0101



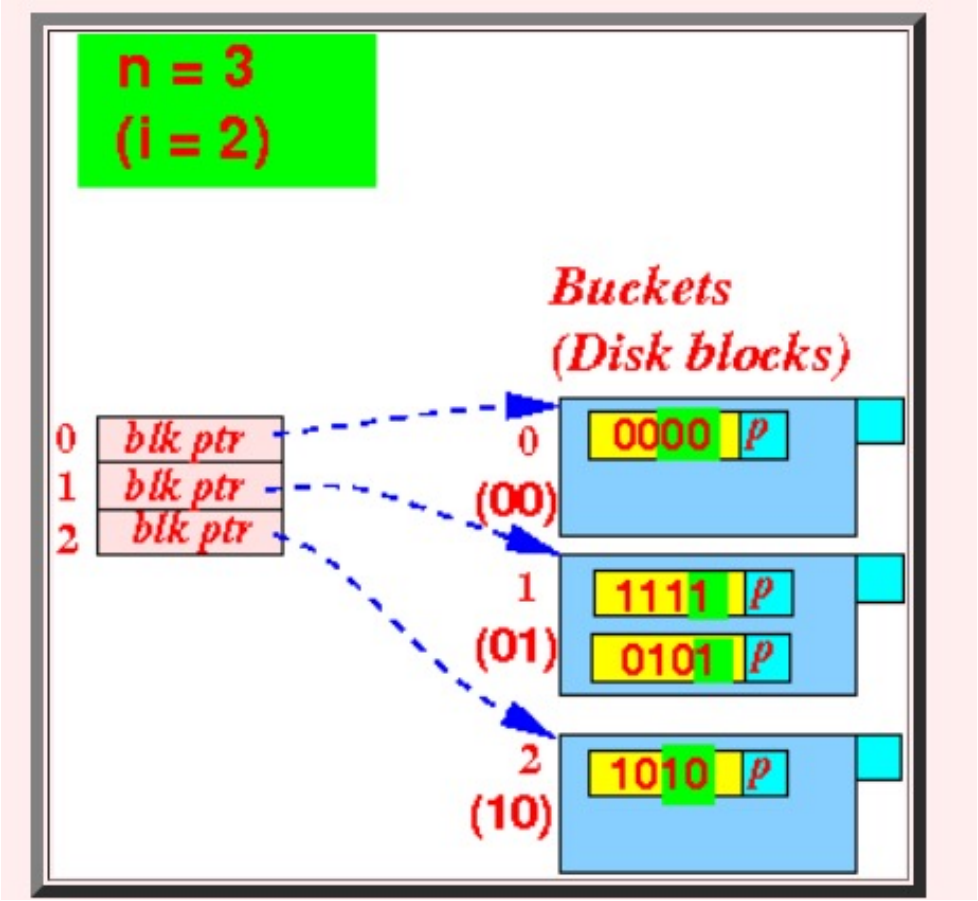
Linear Hashing



Linear Hashing

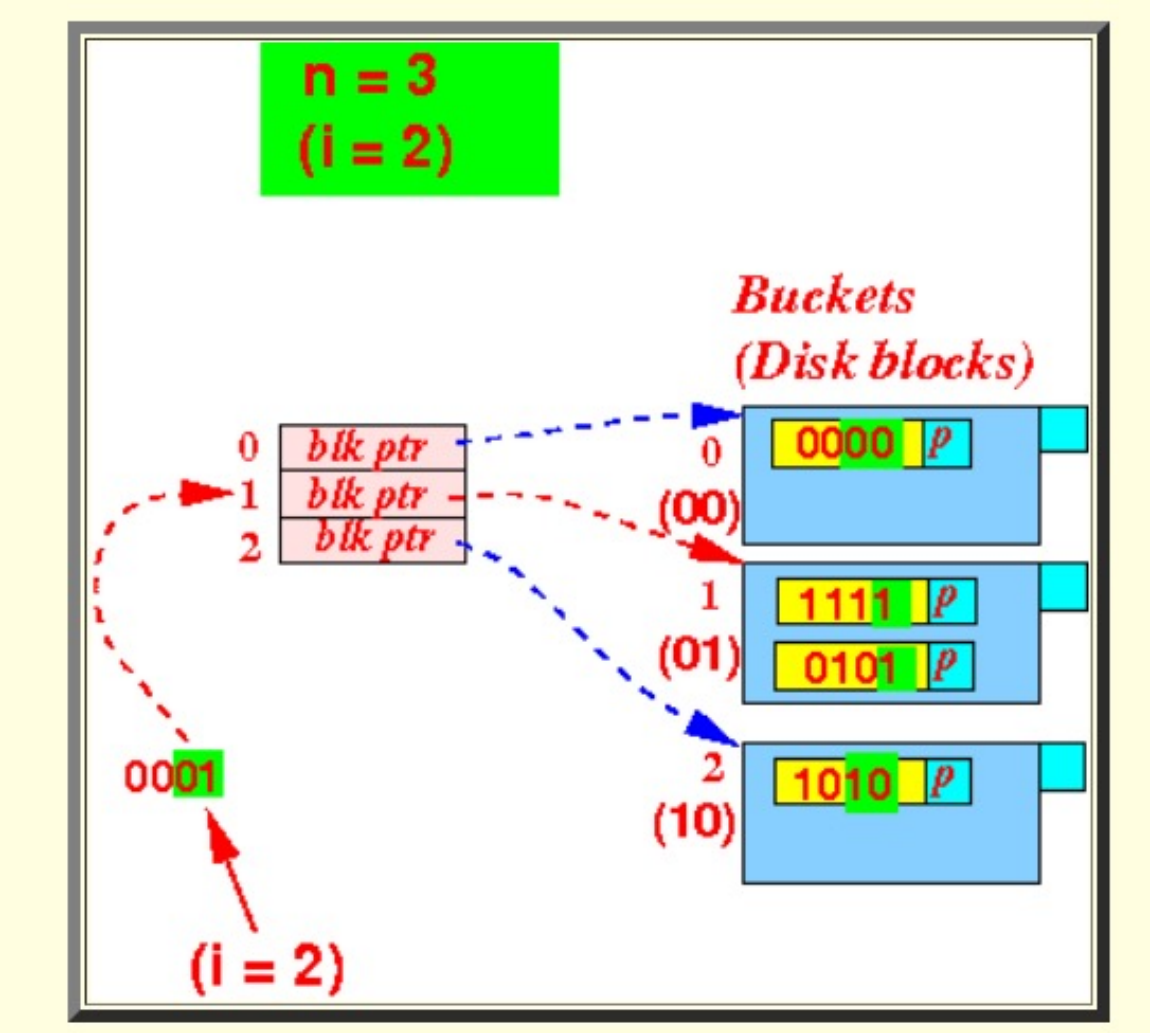


Linear Hashing

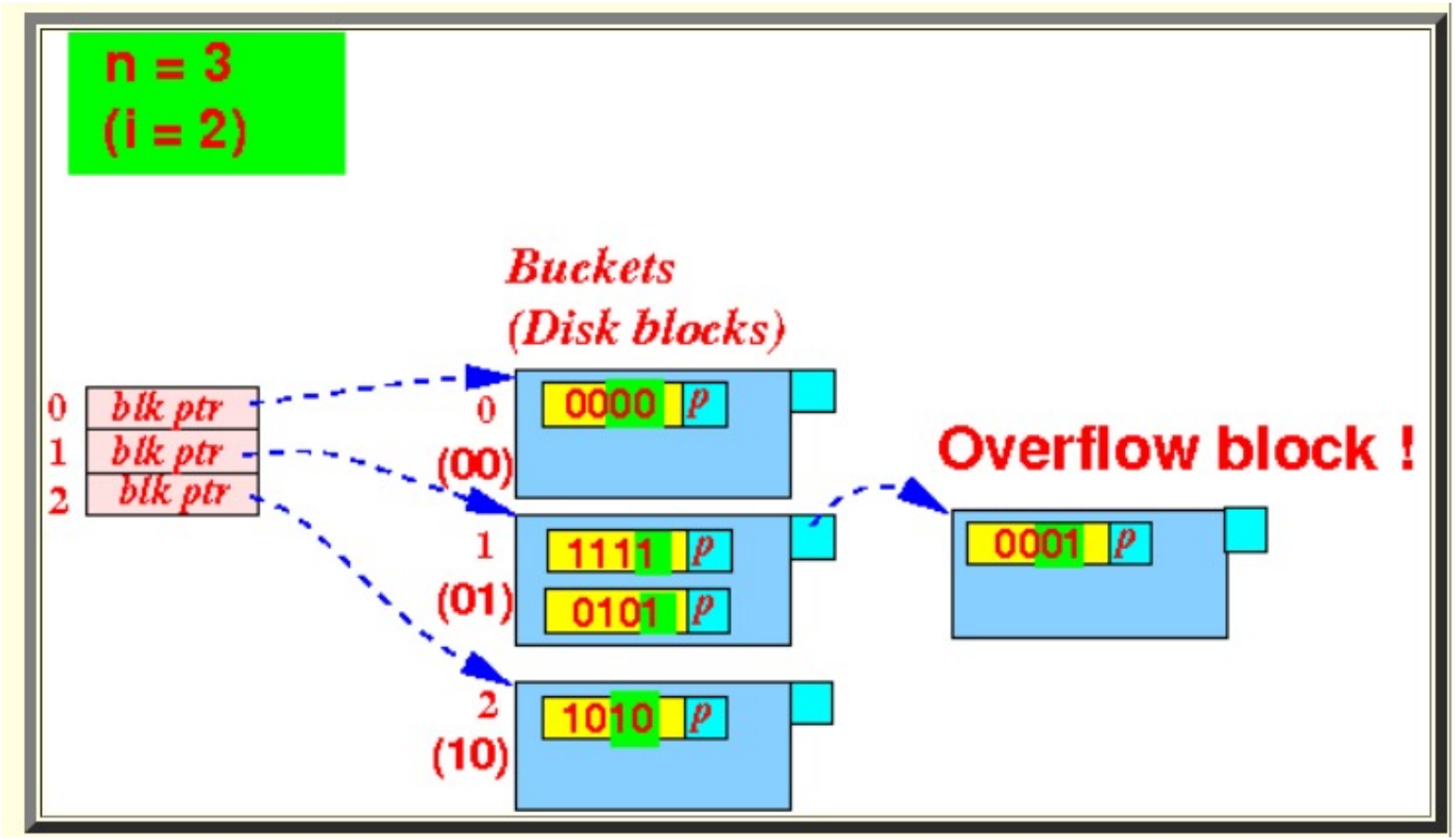


Linear Hashing

Εισάγετε το 0001

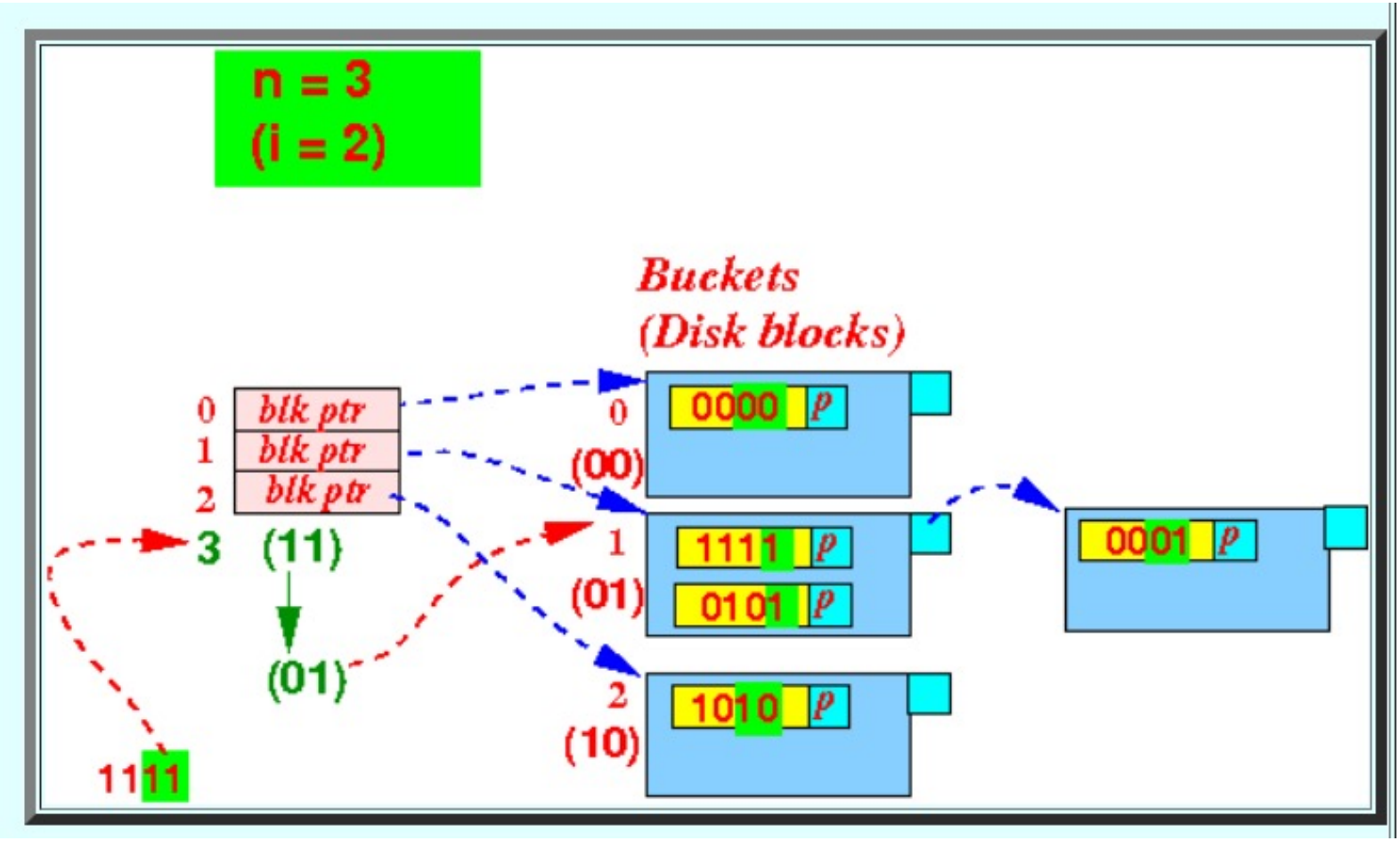


Linear Hashing

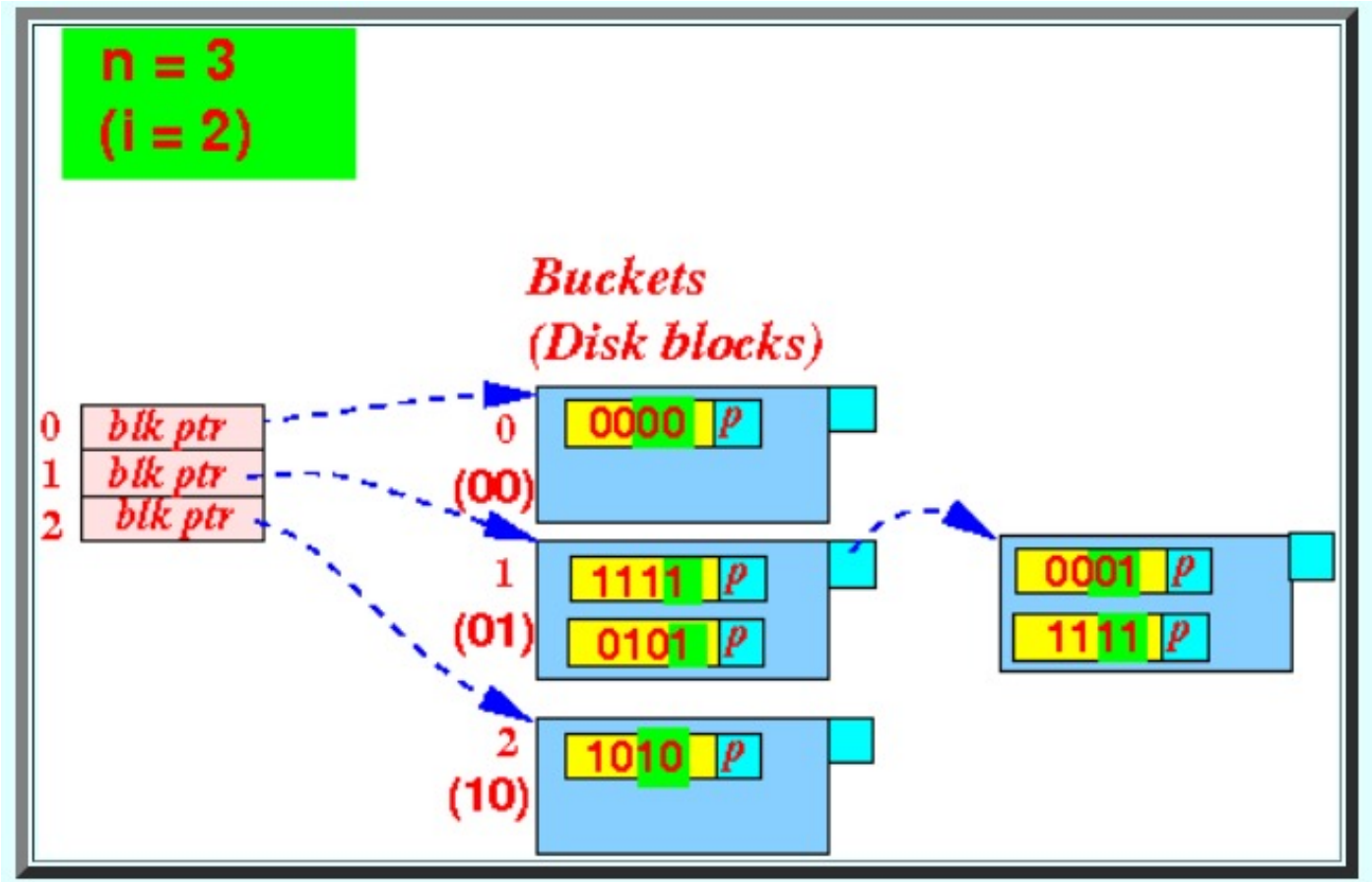


Linear Hashing

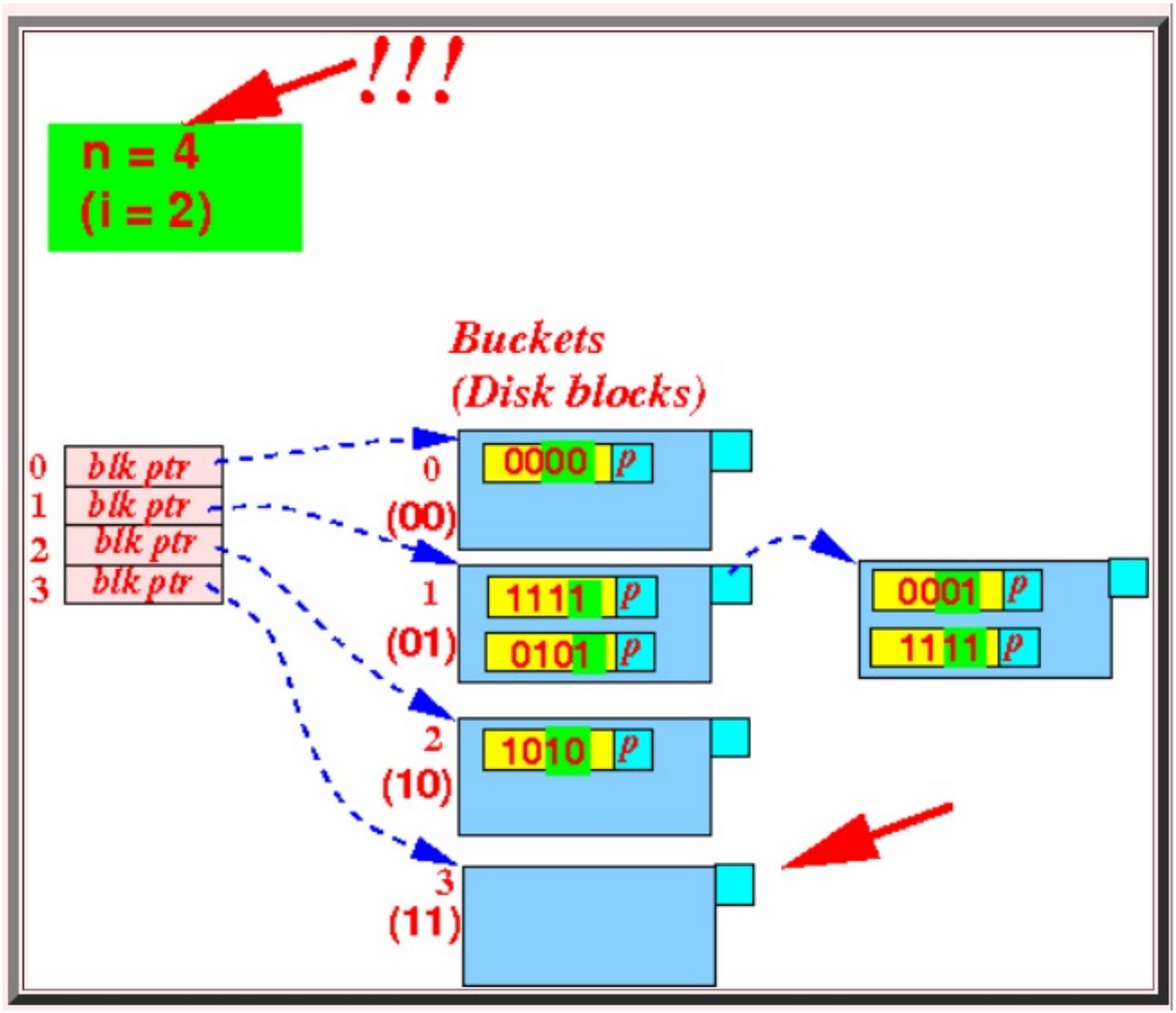
Εισάγετε το 1111



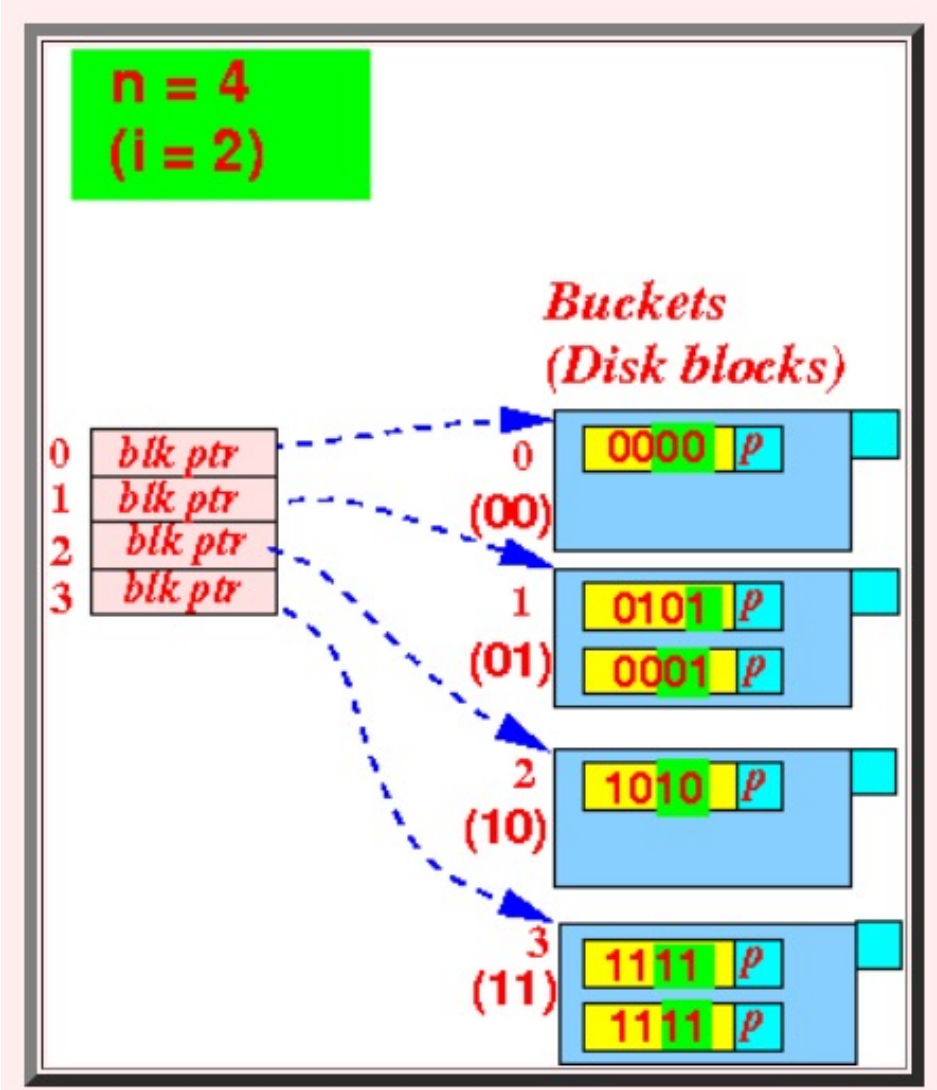
Linear Hashing



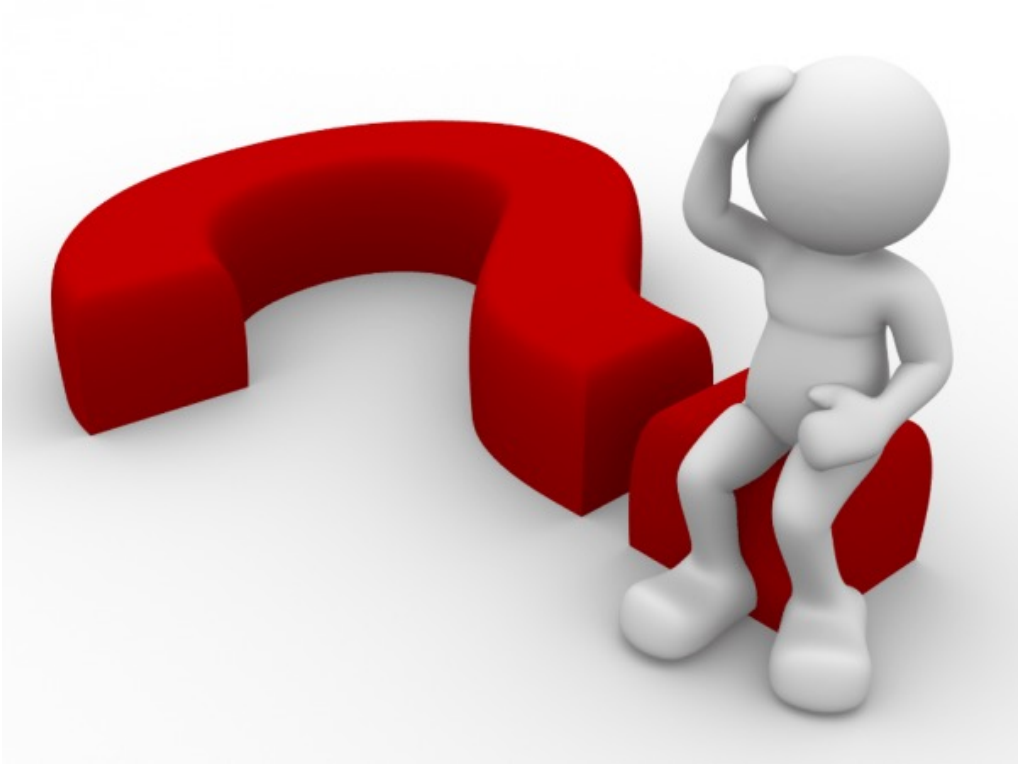
Linear Hashing



Linear Hashing



Questions



Thank you

Maria K. Krommyda

Εθνικό Μετσόβιο Πολυτεχνείο

Εργαστήριο Βάσεων Γνώσεων &Δεδομένων