Εργασία 1

Σχεδιασμός Βάσεων Δεδομένων

Καρράς Κωνσταντίνος 3180076

❖ Άσκηση 1

Δεδομένα

Μέγεθος τομέα (sector) 4096 bytes

65536 ίχνη (tracks) ανά επιφάνεια (surface)

8 πλακέτες (platters) διπλής όψης

256 τομείς ανά ίχνος

Μέσος Χρόνος Μετακίνησης Κεφαλής 10 ms

Ταχύτητα περιστροφής 7200 rpm

Λύση

a) **Χωρητικότητες**
   - Χωρητικότητα ίχνους: Κάθε ίχνος έχει 256 τομείς και κάθε τομέας έχει μέγεθος 4096 bytes. Άρα κάθε ίχνος έχει **4096 bytes * 256 = 1.048.576 bytes = 1 MB**.
   - Χωρητικότητα επιφάνειας: Κάθε επιφάνεια έχει 65536 ίχνη και κάθε ίχνος έχει 1.048.576 bytes. Άρα κάθε επιφάνεια έχει **1.048.576 bytes * 65536 = 68.719.476.736 bytes = 64 GB.**
   - Χωρητικότητα ολόκληρου του δίσκου: Ο δίσκος έχει 8 πλακέτες διπλής όψης, όπου η κάθε όψη(επιφάνεια) έχει 68.719.476.736 bytes. Άρα ο δίσκος έχει συνολικά **68.719.476.736 bytes * 8 * 2 = 1.099.511.627.776 bytes = 1 TB.**

b) Ο δίσκος είναι διπλής όψης, ωστόσο ισχύει ότι και στους δίσκους με 1 όψη, δηλαδή ο αριθμός των κυλίνδρων είναι ίσος με τον αριθμό τον αριθμό των ιχνών. Άρα ο αριθμός των κυλίνδρων ισούται με **65536.**

c) **<u>Καθυστερήσεις Περιστροφής</u>**

- <u>Μέγιστη καθυστέρηση περιστροφής:</u> Πρόκειται για την καθυστέρηση, όπου ο δίσκος βρίσκεται στο σωστό ίχνος, αλλά δεν είναι στο σωστό τομέα. Συγκεκριμένα, βρίσκεται στον αμέσως επόμενο τομέα από αυτόν που θέλει να έχει πρόσβαση. Επειδή, λοιπόν, υπάρχουν 256 τομείς ανά ίχνος θα πρέπει να περιμένει μια πλήρη περιστροφή. Από τα δεδομένα ισχύει ότι ο δίσκος έχει ταχύτητα περιστροφής 7200 rotations per minute. Άρα χρειάζεται να γίνει μία πλήρης περιστροφή η οποία διαρκεί χρόνο ίσο με **60 / 7200 = 8,33 ms.**
- <u>Μέση Καθυστέρηση περιστροφής:</u> Πρόκειται για την καθυστέρηση, όπου ο δίσκος βρίσκεται στο σωστό ίχνος, αλλά δεν είναι στο σωστό τομέα. Συγκεκριμένα, βρίσκεται στον τομέα ακριβώς «απέναντι» από αυτόν που θέλει. Άρα τώρα θα πρέπει να περιμένει μισή περιστροφή, δηλαδή να περιμένει τα μισά sectors. Έτσι, λοιπόν, θα χρειαστεί χρόνος ίσος με **128 / 256 * 60 / 7200 = 4,16 ms.**

d) Ο δίσκος έχει ταχύτητα περιστροφής 7200 rpm, δηλαδή σε 60 seconds κάνει 7200 περιστροφές. Σε 1 second κάνει 7200/60 = 120 περιστροφές. Η χωρητικότητα του κάθε ίχνους είναι από το πρώτο ερώτημα 1 MB. Άρα σε 1 second ο δίσκος μπορεί να διαβάσει και να μεταφέρει 120 MB. Σε αυτό όμως θα πρέπει να προστεθεί και ο χρόνος μετακίνησης της κεφαλής ανάμεσα στα sectors. Οι μετακινήσεις είναι συνολικά 119 (υποθέτω ότι βρίσκεται στο σωστό track κατά την πρώτη προσπέλαση, οπότε δεν θα χρειαστεί να μετακινηθεί η κεφαλή). Άρα σε χρόνο **119 * 10 ms + 1 second = 1,19 s + 1 s = 2,19 seconds ο δίσκος διαβάζει και μεταφέρει 120 MB.** Επομένως, το **transfer rate του δίσκου είναι ίσο με 120 MB / 2,19 sec = 54,7945 MBps = 438,356 Mbps.**

❖ Άσκηση 2

R

#a	b	c	d	e
10	50	30	18	20

Pointer
6

<u>Ευρετήρια</u>

a) Το dense index έχει έναν pointer για κάθε εγγραφή της σχέσης R στο πρωτεύων κλειδί. Δεδομένου ότι υπάρχουν N εγγραφές και κάθε pointer καταλαμβάνει 6 bytes στο δίσκο σημαίνει ότι θα χρειαστούν **16 * N** bytes για την αποθήκευση των δεικτών και του πρωτεύοντος κλειδιού (εγγραφές ευρετηρίου). Άρα τα συνολικά μπλοκ θα είναι **128 * N / 1024 + 16 * N bytes / 1024 bytes = 2^7 * N / 2^{10} + 2^4 * N / 2^{10} = N / 2^3 + N / 2^6 = 9N / 64 blocks.**

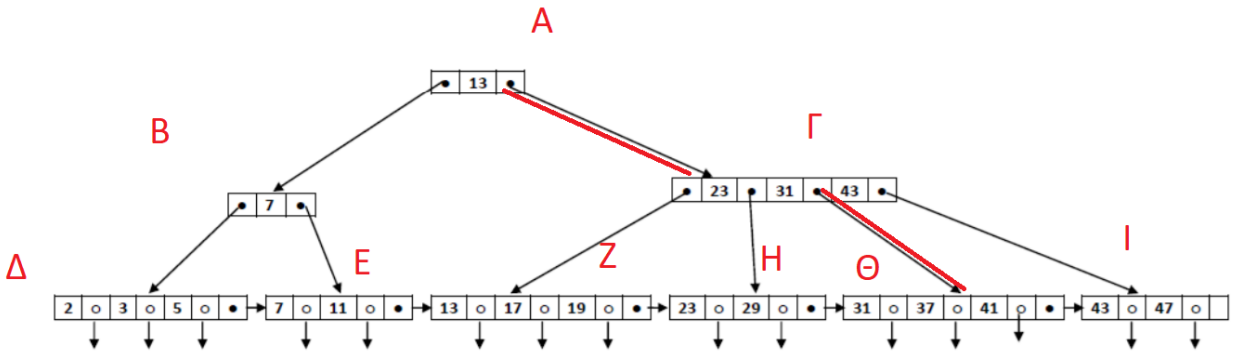
b) Το sparse index έχει έναν pointer για την πρώτη εγγραφή κάθε νέας σελίδας που χρησιμοποιεί η σχέση R για την αποθήκευση των δεδομένων. Από τη στιγμή που το συνολικό πλήθος μιας εγγραφής είναι 128 bytes (10 + 50 +30 +18 +20 = 128) και κάθε σελίδα έχει χωρητικότητα 1024 bytes σημαίνει ότι κάθε σελίδα θα περιέχει 8 εγγραφές. Pointer, όμως, θα υπάρχει μόνο για την πρώτη εγγραφή κάθε σελίδας. Άρα για την αποθήκευση των δεικτών και του πρωτεύοντος κλειδιού, αφού εκεί θα εφαρμοστεί το ευρετήριο χρειαζόμαστε **1 / 8 * 16 * N bytes = 2 * N bytes**. Άρα τα συνολικά μπλοκ που θα χρειαστούν είναι **128 * N / 1024 + 2 * N bytes / 1024 bytes = N / 2^3 + N / 512 = 65 * N / 512 blocks.**

*<u>(Και στις 2 περιπτώσεις το 128 * N / 1024 είναι ο χώρος που καταλαμβάνει η σχέση R.)</u>*

❖ Άσκηση 3

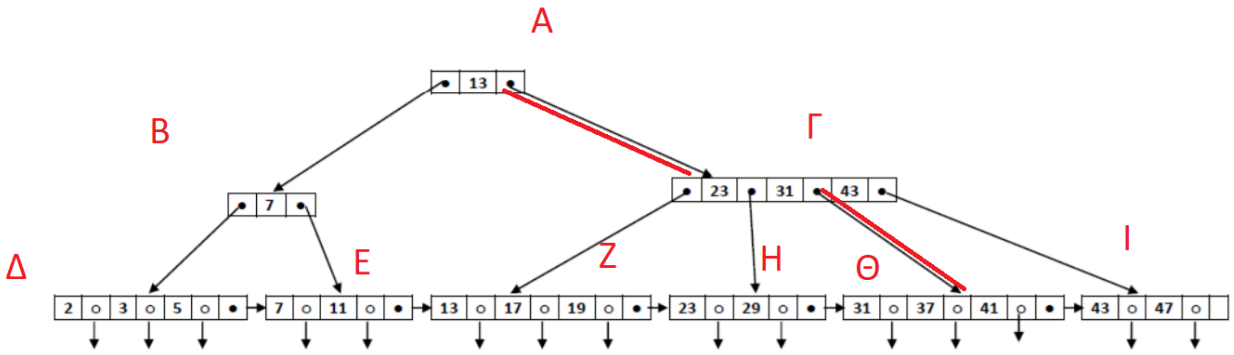
B+ Δέντρα

1)



Θα ξεκινήσει από τον κόμβο **Α**. Θα διαπιστώσει ότι το 41 είναι μεγαλύτερο από το 13 και θα πάει στον κόμβο **Γ**. Θα δει ότι το 41 είναι μεγαλύτερο του 31 αλλά μικρότερο από το 43 και έτσι θα ακολουθήσει το 3^{ο} κλαδί και θα μεταβεί στον κόμβο **Θ**. Προσπελαύνοντάς τον από την αρχή, θα δει την τιμή 41 και θα πάρει τον pointer του 41, που βρίσκεται δεξιά του και δείχνει στην εγγραφή με κλειδί 41.
(Η διαδρομή φαίνεται και με την κόκκινη γραμμή που είναι δίπλα από την μαύρη, ώστε να τονίζεται ο κύριος δρόμος που θα ακολουθηθεί)

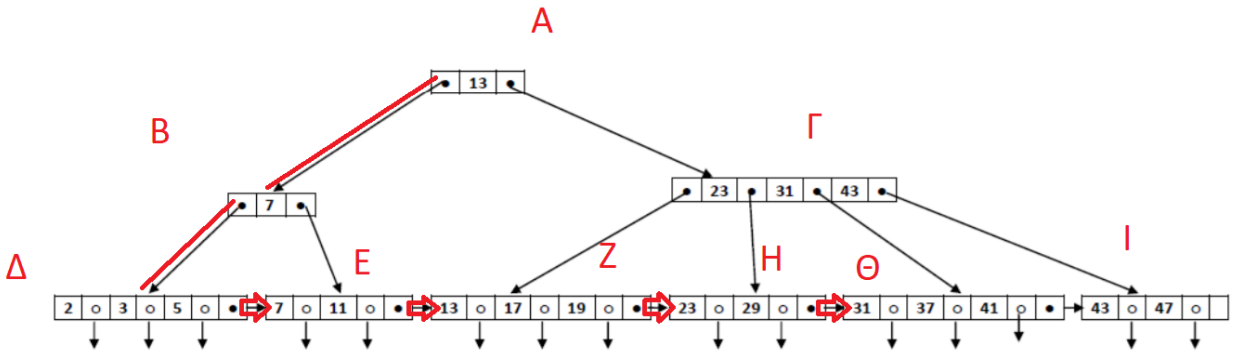
2)



Ομοίως με πριν θα ξεκινήσει από τον κόμβο **Α**. Θα διαπιστώσει ότι το 40 είναι μεγαλύτερο από το 13 και θα πάει στον κόμβο **Γ**. Θα δει ότι το 40 είναι μεγαλύτερο του 31 αλλά μικρότερο από το 43 και έτσι θα ακολουθήσει το 3^{ο} κλαδί και θα μεταβεί στον κόμβο **Θ**. Προσπελαύνοντάς

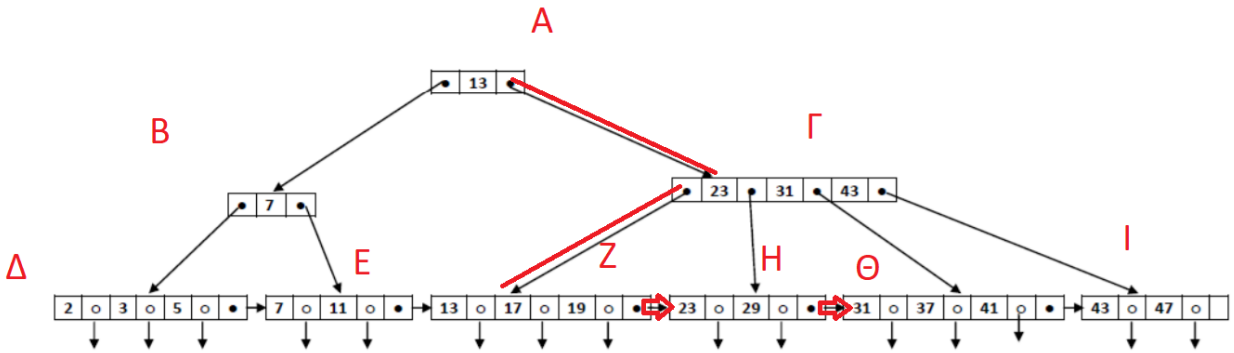
τον από την αρχή, δεν θα δει την τιμή 40. Συγκεκριμένα, θα διαπιστώσει ότι το 40 είναι μεγαλύτερο από το 37 αλλά μικρότερο από το 41 και επειδή ο κόμβος Θ είναι φύλλο σημαίνει ότι δεν υπάρχει ο pointer 40 που δείχνει στη συγκεκριμένη εγγραφή που αναζητούμε.
(Η διαδρομή φαίνεται και με την κόκκινη γραμμή που είναι δίπλα από την μαύρη, ώστε να τονίζεται ο κύριος δρόμος που θα ακολουθηθεί)

3)



Αφού πρόκειται για κλειδιά μικρότερα του 30 θα ακολουθήσει το αριστερό κλαδί από κάθε κόμβο. Έτσι από τον **Α** θα μεταβεί στον **Β** και από εκεί στον **Δ**. Προσπελαύνοντας τον κόμβο **Δ** θα έχει βρει pointers των εγγραφών με κλειδιά μικρότερα του 30 και όσο δεν θα βλέπει τιμή μεγαλύτερη ή ίση του 30 θα συνεχίζει και στον διπλανό κόμβο. Με αυτόν τον τρόπο, λοιπόν, θα συνεχίσει και στον **Ε** και στον **Ζ** και στον **Η** και τέλος στον **Θ** όπου θα σταματήσει. Και αυτό διότι θα συναντήσει τιμή μεγαλύτερη του 30.
(Η διαδρομή φαίνεται και με την κόκκινη γραμμή που είναι δίπλα από την μαύρη, αλλά και με τα κόκκινα βελάκια ώστε να τονίζεται ο κύριος δρόμος που θα ακολουθηθεί)

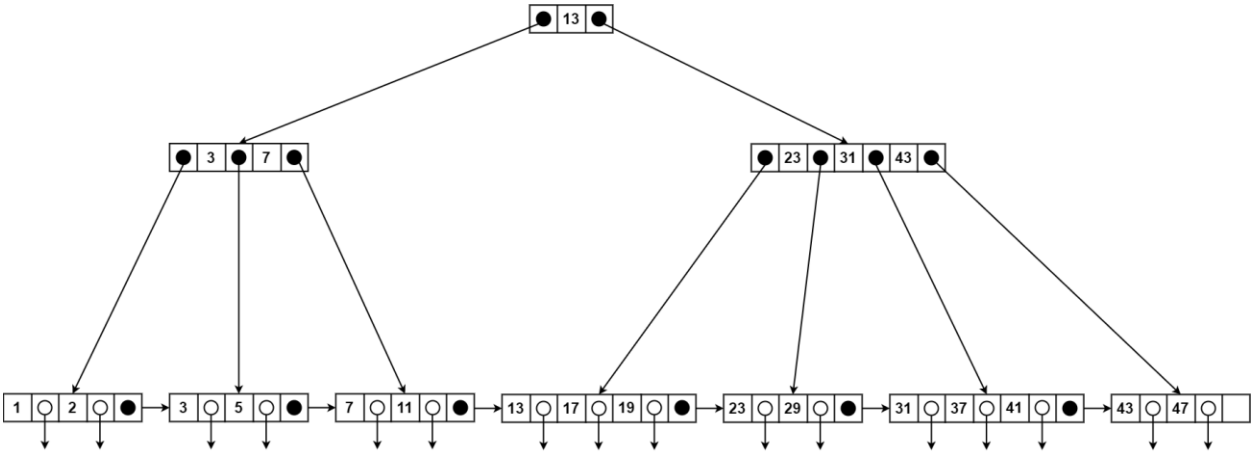
4)



Αφού πρόκειται για κλειδιά μεγαλύτερα ή ίσα του 20 και μικρότερα ή ίσα του 35 θα ακολουθήσει τους δείκτες ώστε να βρει το 20 (που είναι το αριστερό άκρο του διαστήματος), αν υπάρχει, και στη συνέχεια θα ψάχνει ακολουθιακά στους διπλανούς κόμβους, οι οποίοι λόγω της δομής του B+ δέντρου, θα έχουν τιμές σίγουρα μεγαλύτερες από 20. Έτσι λοιπόν, θα ξεκινήσει από τη ρίζα όπως πάντα και θα δει ότι το 20 είναι μεγαλύτερο από το 13. Άρα θα πάει από τον κόμβο **Α** στον κόμβο **Γ**. Εκεί θα διαπιστώσει ότι το 20 είναι μικρότερο από το 23 και θα ακολουθήσει το αριστερό κλαδί μεταβαίνοντας στον κόμβο **Z**. Εκεί όλα τα κλειδιά είναι μικρότερα του 20, οπότε δε θα τα συμπεριλάβει και θα συνεχίσει στον επόμενο κόμβο, τον **Η**. Οι τιμές αυτού του κόμβου βρίσκονται μέσα στο διάστημα της αναζήτησής του. Μετά και αφού βρίσκεται ακόμη μέσα στα όρια, θα συνεχίσει στον επόμενο κόμβο, τον **Θ**. Από αυτόν τον κόμβο θα επιλέξει μόνο το κλειδί 31 και στη συνέχεια θα δει ότι το επόμενο κλειδί είναι το 37 και θα σταματήσει.
(Η διαδρομή φαίνεται και με την κόκκινη γραμμή που είναι δίπλα από την μαύρη, αλλά και με τα κόκκινα βελάκια ώστε να τονίζεται ο κύριος δρόμος που θα ακολουθηθεί)

5) Για την εισαγωγή του κλειδιού 1 και για τη διατήρηση των κανόνων του B+ δέντρου το κλειδί 1 πρέπει να τοποθετηθεί στην 1η θέση, ώστε να παραμένουν ταξινομημένα τα φύλλα, αλλά και γιατί το 1 είναι μικρότερο από οποιοδήποτε άλλο κλειδί στο δέντρο. Όμως στο δοθέν σχήμα υπάρχουν ήδη 3 κλειδιά στο αριστερότερο φύλλο. Επομένως, θα πρέπει να το σπάσω. Κατά το σπάσιμο το μεσαίο στοιχείο του αριστερότερου κόμβου ανεβαίνει ένα επίπεδο. Έτσι, λοιπόν, το 3 ανεβαίνει στο πάνω επίπεδο μαζί

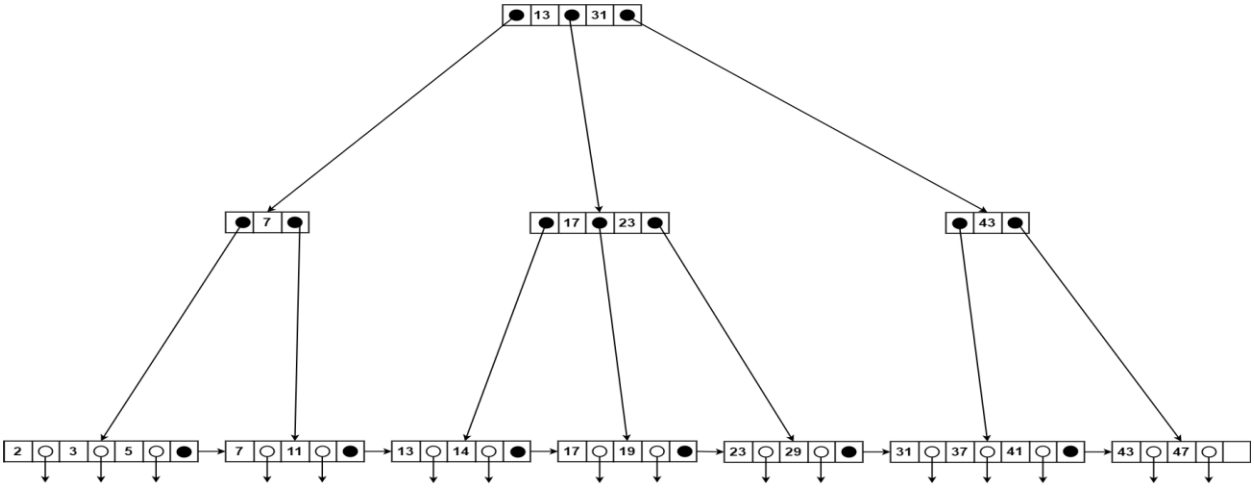
με το 7 δίνοντας τη δυνατότητα για ένα νέο εύρος τιμών(πριν ήταν <7 και >=7, ενώ τώρα είναι < 3, [3,7) και >=7). Το 1 θα μπει πλέον στον αριστερότερο κόμβο μαζί με το 2, ικανοποιώντας την ιδιότητα. Στο διάστημα [3,7) θα μπει το 3 και το 5, ενώ τέλος στο διάστημα >=7 θα παραμείνει το 7 και το 11. Όλοι οι άλλοι κόμβοι παραμένουν απείραχτοι.



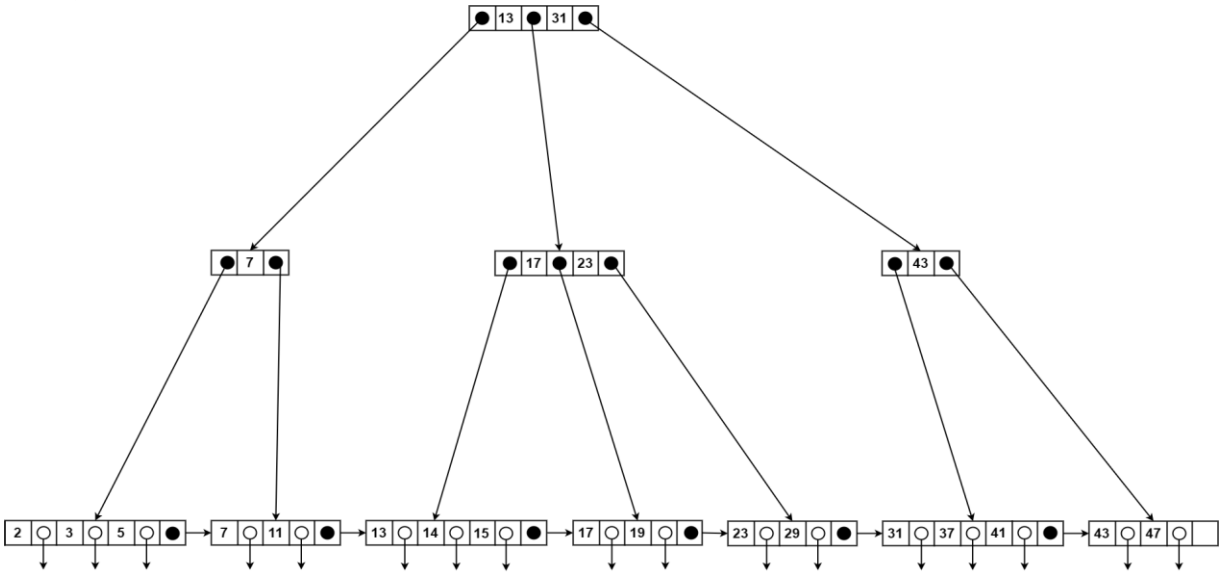
6)

- <u>Εισαγωγή του 14:</u>
  Για την εισαγωγή του κλειδιού 14 και για τη διατήρηση των ιδιοτήτων του B+ δέντρου θα πρέπει να σπάσω ο 3^{ο} φύλλο κατά σειρά, μετρώντας από αριστερά, γιατί αυτή είναι η θέση του κλειδιού 14 και γιατί ο κόμβος είναι γεμάτος. Με το ίδιο σκεπτικό όπως και στο ερώτημα 5 παίρνω το μεσαίο κλειδί και το ανεβάζω ένα επίπεδο. Το κλειδί αυτό είναι το 17. Ωστόσο, βλέπω ότι και το από πάνω επίπεδο είναι γεμάτο. Ακριβώς με το ίδιο σκεπτικό ανεβάζω στο επίπεδο της ρίζας τον μεσαίο δείκτη του επιπέδου 1 (τον δείκτη 31). Έτσι η ρίζα «προσθέτει» ένα ακόμη εύρος τιμών (πριν ήταν <13 και >=13 ενώ τώρα είναι <13,[13,31) και >=31). Κατ' αυτόν τον τρόπο ο δείκτης 17 ανεβαίνει στο επίπεδο 1 και είναι στον ίδιο κόμβο με τον δείκτη 23(λόγω του ότι βρίσκονται στο διάστημα [13,31)). Μέσω αυτών των 2 δεικτών τα κλειδιά 13 και 14 πάνε πριν το 17, τα κλειδιά 17 και 19 πάνε στο διάστημα [17,23), ενώ τέλος τα

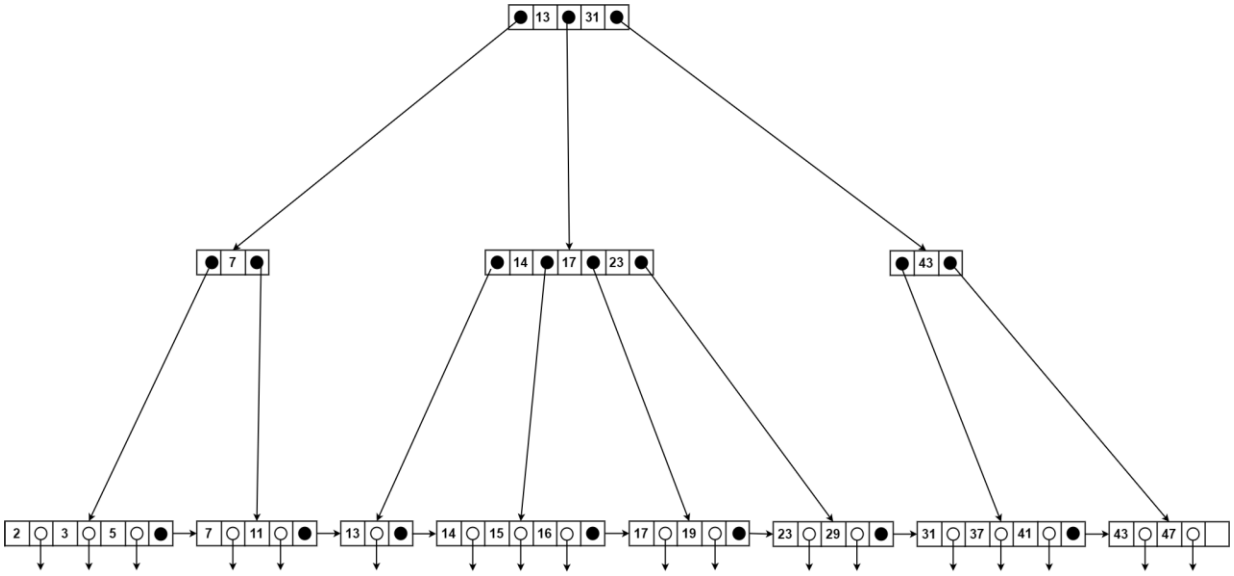
κλειδιά 23 και 29 πάνε στο >=23.Όλοι οι άλλοι κόμβοι παραμένουν απείραχτοι. Εν ολίγοις, το γεγονός ότι κατά την άνοδο του 17 αναγκάστηκα να σπάσω και ένα κόμβο στο 1ο επίπεδο και ουσιαστικά από 2 κόμβους να έχω 3, μου έδωσε την δυνατότητα περισσότερων διαστημάτων σε πλήθος όπου κατανέμουν καλύτερα (ομοιόμορφα) τα κλειδιά.



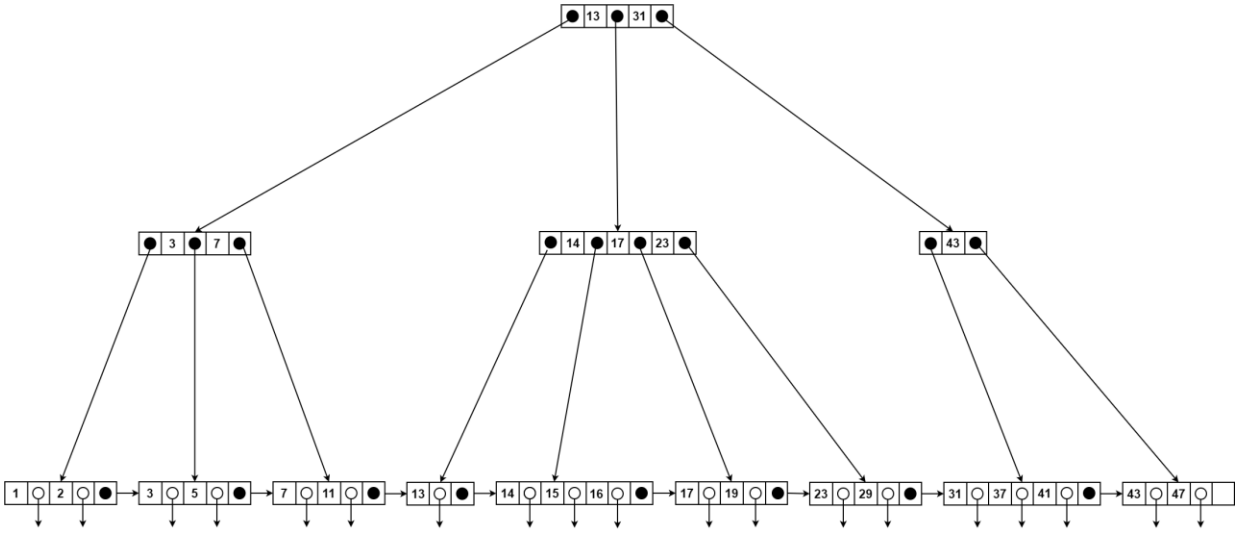
- <u>Εισαγωγή του 15:</u>
  Κατά την εισαγωγή του 15 όλα είναι πιο απλά. Το μόνο που πρέπει να γίνει είναι να μπει στον κόμβο με το 13 και το 14 (συγκεκριμένα μετά το 14) και έτσι ικανοποιούνται όλοι οι περιορισμοί του B+ δέντρου.



- <u>Εισαγωγή του 16:</u>
  Για την εισαγωγή του κλειδιού 16 θα πρέπει να σπάσω το 3^{ο} φύλλο, προκειμένου να ισχύουν οι περιορισμοί του B+ δέντρου. Με το ίδιο σκεπτικό όπως και στις προηγούμενες εισαγωγές θα ανεβάσω το μεσαίο κλειδί (το 14) ένα επίπεδο. Με αυτόν τον τρόπο θα προκύψει ένα επιπλέον φύλλο. Το ένα φύλλο θα είναι αυτό που έχει κλειδιά μικρότερα του 14 και το άλλο που έχει κλειδιά μεγαλύτερα ή ίσα του 14. Έτσι το 13 θα πάει σε έναν κόμβο μόνο του και τα κλειδιά 14, 15 και 16 θα πάνε στο διάστημα που ορίζουν οι δείκτες 14 και 17, δηλαδή στο διάστημα [14,17).

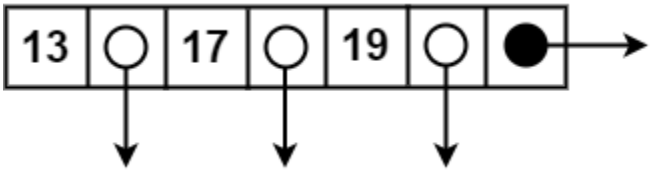


- <u>Τελική μορφή Δέντρου:</u>



❖ Άσκηση 4

Στην περίπτωση όπου είχαμε ένα B+ δέντρο με n=3, τότε ένα φύλλο του (γεμάτο) θα είχε την μορφή που φαίνεται στην παρακάτω φωτογραφία. Αυτό σημαίνει ότι έχει 4 δείκτες (3 προς εγγραφές δεδομένων και 1 προς άλλο κόμβο). Επίσης, έχει και 3 κλειδιά. Άρα στη γενική περίπτωση τα κλειδιά είναι n, οι pointers προς κόμβους είναι 1 και οι pointers προς δεδομένα είναι n.



Άρα ισχύει, αφού χρησιμοποιείται όλη η σελίδα , το εξής:

12 * n + 12 + 8 * n = 2048 ⇔12 * n + 12 + 8 * n = 2048⇔20 * n = 2036 ⇔ n = 101.8 => n = 101, αφού το $n \epsilon \mathbb{N}$.

Άρα, ο συνολικός αριθμών εγγραφών της σχέσης θα είναι

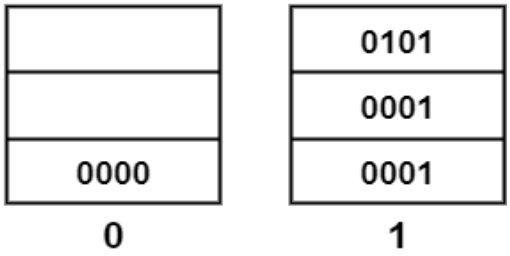
102 * 102 * 101 = 1.050.804

(Οι pointers στο επίπεδο της ρίζας προς το επόμενο επίπεδο είναι 102. Για κάθε κόμβο στο αμέσως επόμενο επίπεδο (από τη ρίζα) οι pointers προς τα φύλλα είναι επίσης 102. Για κάθε έναν από τα φύλλα, τα κλειδιά είναι 101. Άρα έτσι προέκυψε το 102 * 102 * 101)

❖ Άσκηση 5

1η Μορφή:

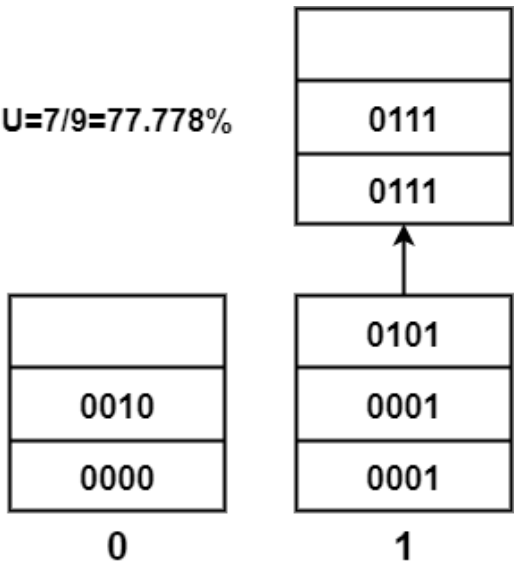
U=4/6=66.667%



Στην αρχική μορφή τα buckets είναι κενά. Αρχίζω και εισάγω κλειδιά και παρατηρώ ταυτόχρονα ότι δεν υπερβαίνουν το 70% της χωρητικότητας των κάδων. Κατά την 5η εισαγωγή (5ο κλειδί είναι το 0111) δεν υπάρχει αρκετός χώρος. Τότε δημιουργείται σελίδα υπερχείλισης στο bucket με bit 1 και δεν αυξάνεται ο αριθμός των κάδων, καθώς θα έχουμε συνολικά 5 εγγραφές για 9 θέσεις.

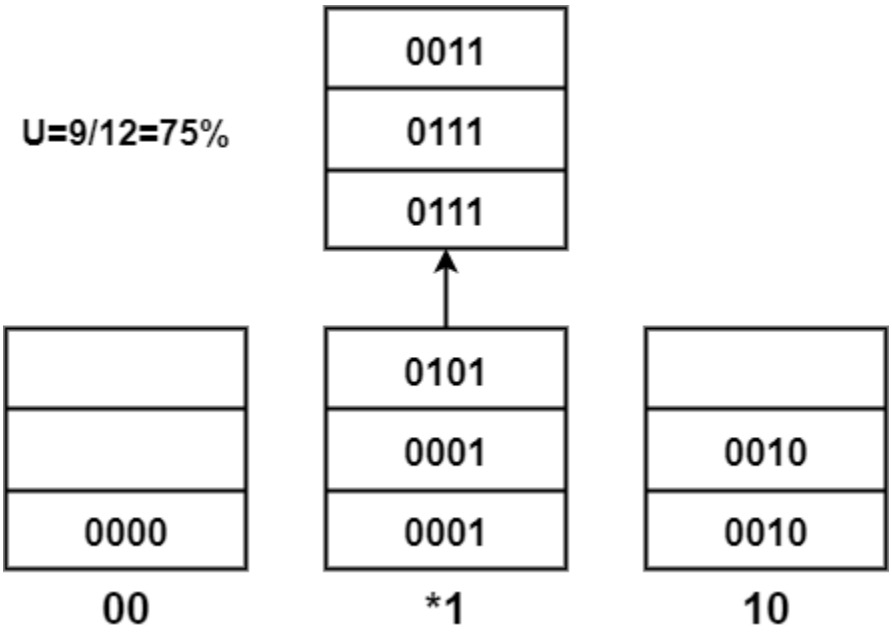
2η Μορφή:

U=7/9=77.778%



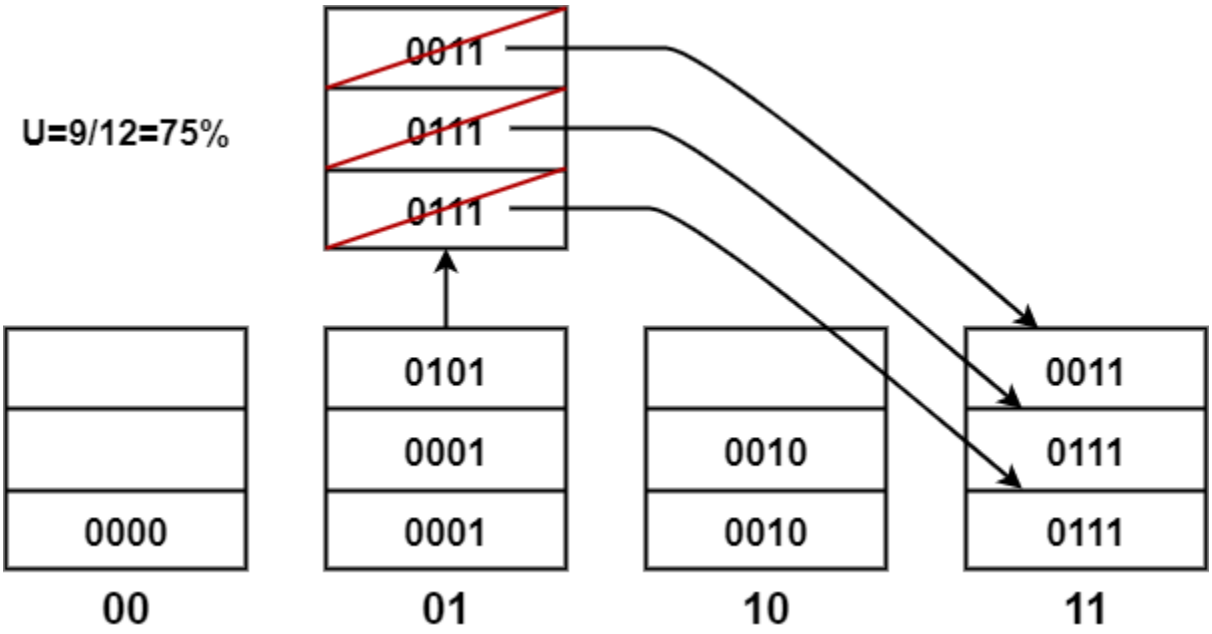
Έχει ήδη δημιουργηθεί η σελίδα υπερχείλισης από την εισαγωγή του 5ου κλειδιού και στη συνέχεια εισάγονται όσο υπάρχει χώρος και άλλες εγγραφές. Παρατηρώ, όμως, ότι κατά την εισαγωγή του 7ου κλειδιού ο χώρος που καταλαμβάνουν όλα τα κλειδιά σε σχέση με τα buckets που υπάρχουν είναι μεγαλύτερος από το επιτρεπτό όριο. Έτσι στο επόμενο βήμα οφείλω να αυξήσω τους κάδους.

3η Μορφή:



Δημιουργώ όπως υποσχέθηκα ένα νέο bucket με τιμή 10. Έτσι όποια bucket τελειώνουν σε 0 δεν αρκεί πλέον να ξέρω μόνον το τελευταίο bit τους αλλά θα πρέπει να ξέρω και το προτελευταίο. Αυτά που τελειώνουν σε 1 πάνε στο μοναδικό bucket που τους αντιστοιχεί. Συνεχίζω, λοιπόν, και εισάγω και άλλα κλειδιά. Κατά την εισαγωγή του 9ου κλειδιού παρατηρώ και πάλι ότι έχω υπερβεί το επιτρεπτό όριο και κάνω ακριβώς την ίδια διαδικασία. Δημιουργώ, δηλαδή, ένα νέο bucket με τιμή 11 και μεταφέρω σε αυτό τις τιμές από το bucket *1 που του αντιστοιχούν (ακολουθεί φωτογραφία).

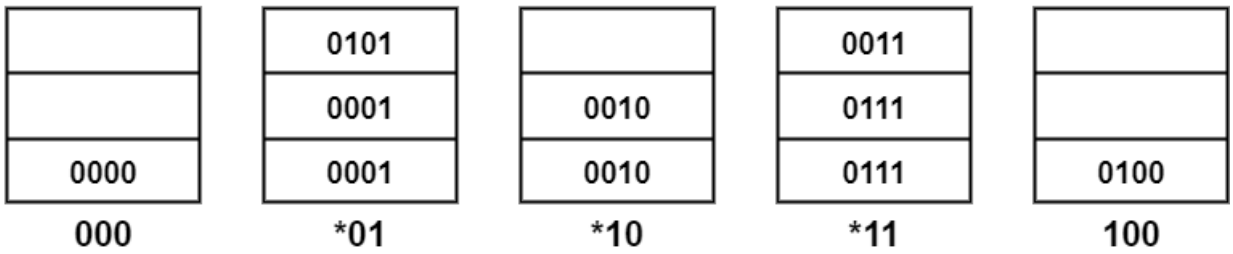
4η Μορφή:



Δημιουργώντας, λοιπόν, αυτό το bucket «φεύγει» όλη η σελίδα υπερχείλισης και αντιστοιχίζεται στο νέο bucket. Η σελίδα υπερχείλισης διαγράφεται και επομένως το πρόβλημα υπέρβασης του επιτρεπτού ορίου παραμένει. Τέλος, δημιουργώ όπως θα δείτε παρακάτω ένα νέο bucket me με τιμή 100, ώστε να το διαχωρίσω από το 1ο και σε όλα τα υπόλοιπα bucket ενδιαφέρομαι μόνο για τα 2 τελευταία bits οπότε τοποθετώ το * σαν μπαλαντέρ (για 0 ή 1).

5η μορφή:

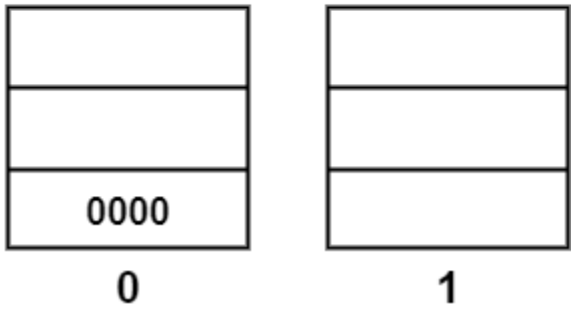
U=10/15=66.667%



Ακολουθούν και φωτογραφίες με την εισαγωγή κάθε κλειδιού ξεχωριστά και χωρίς επεξήγηση, διότι αναφέρθηκε ήδη το σκεπτικό δημιουργίας buckets και τοποθέτησης κλειδιών

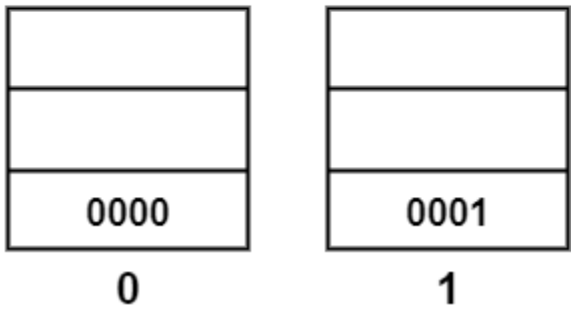
1ο κλειδί:

U=1/6=16.667%



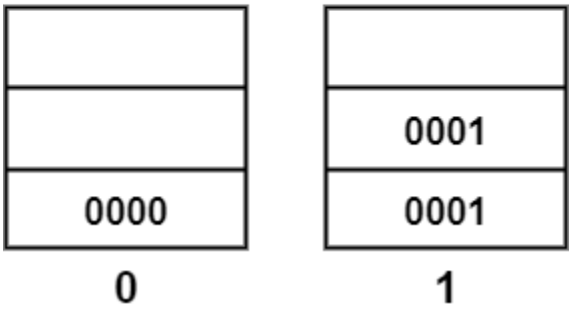
2ο κλειδί:

U=2/6=33.333%



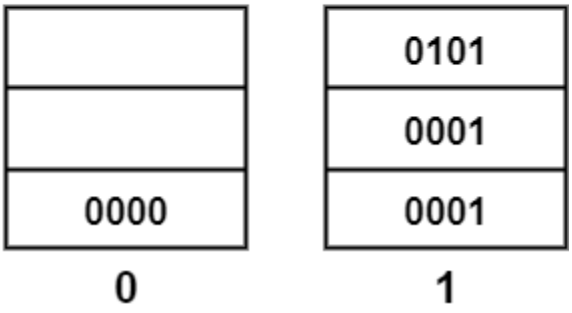
3ο κλειδί:

U=3/6=50%



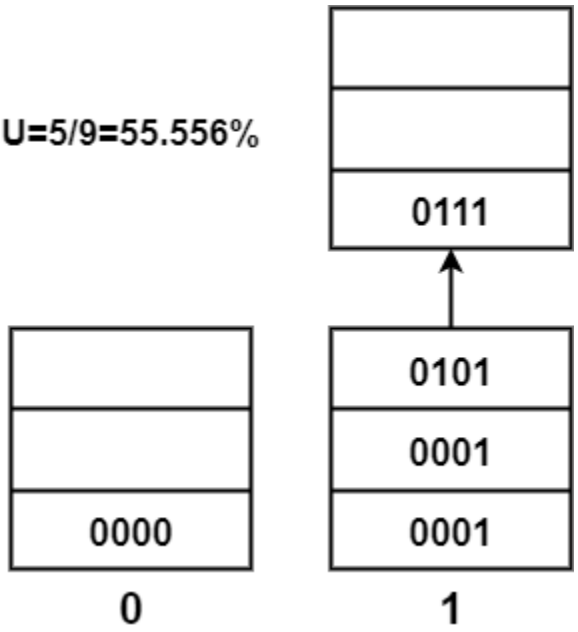
4ο κλειδί:

U=4/6=66.667%



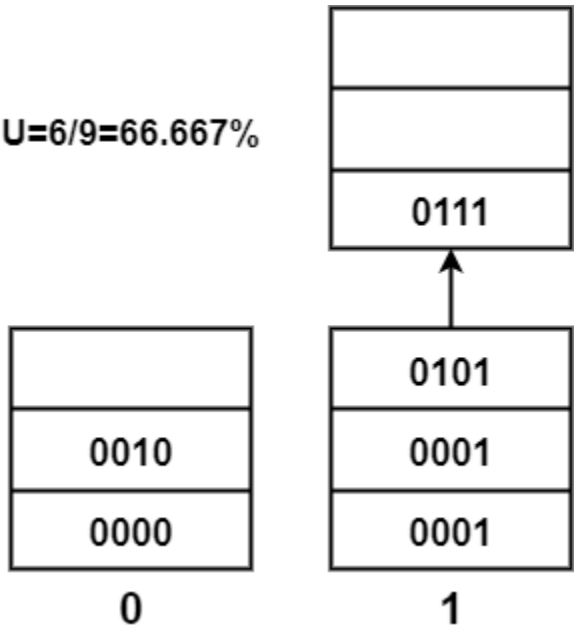
5ο κλειδί:

U=5/9=55.556%

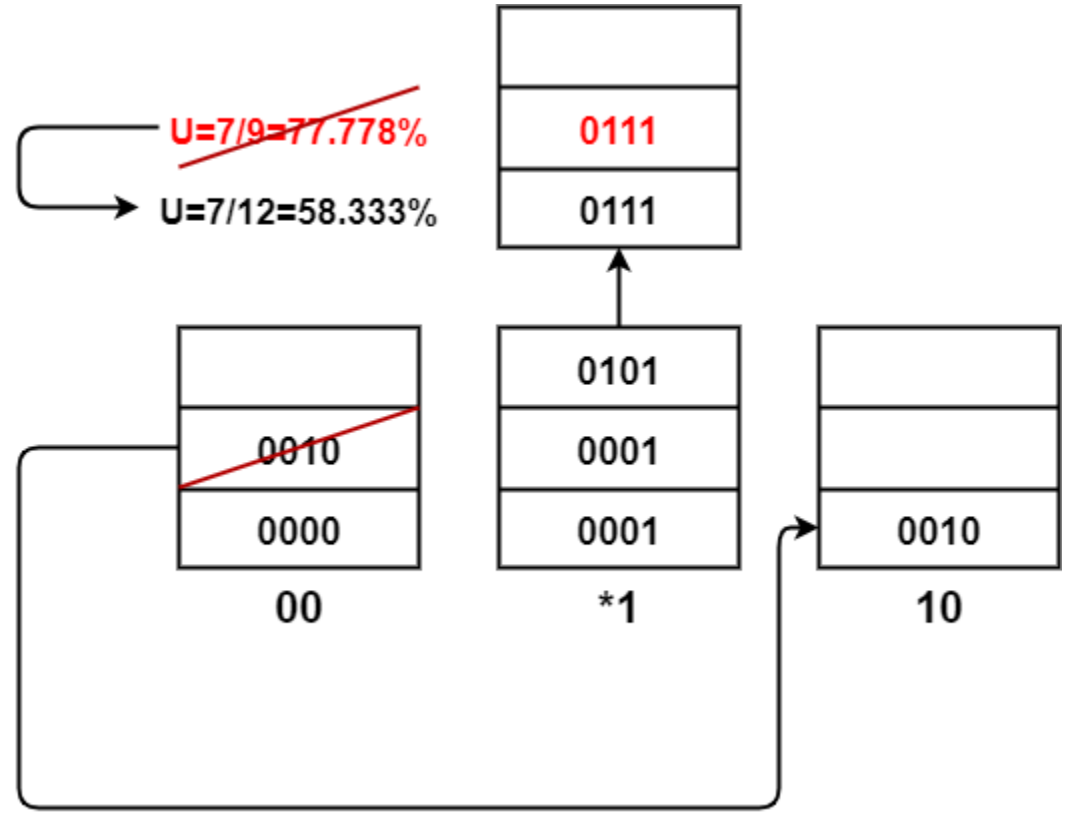


6ο κλειδί:

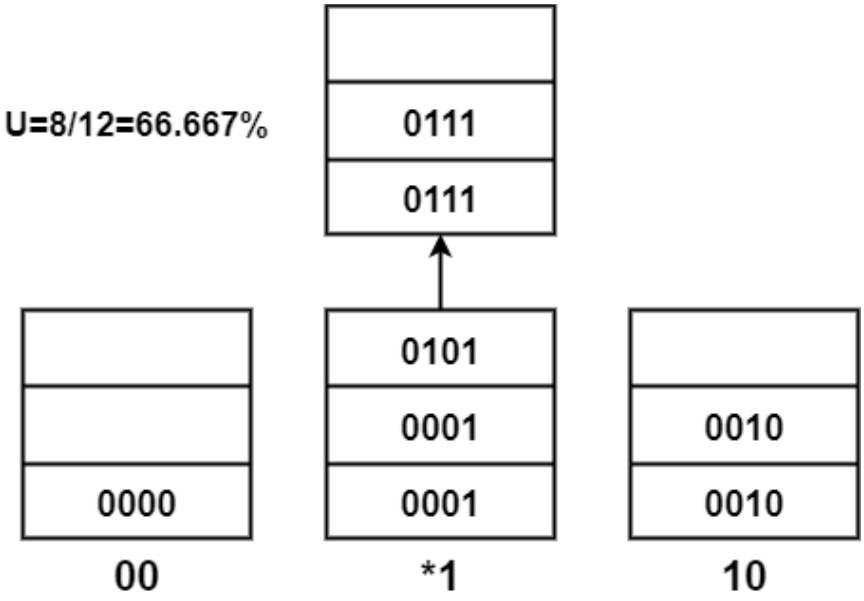
U=6/9=66.667%



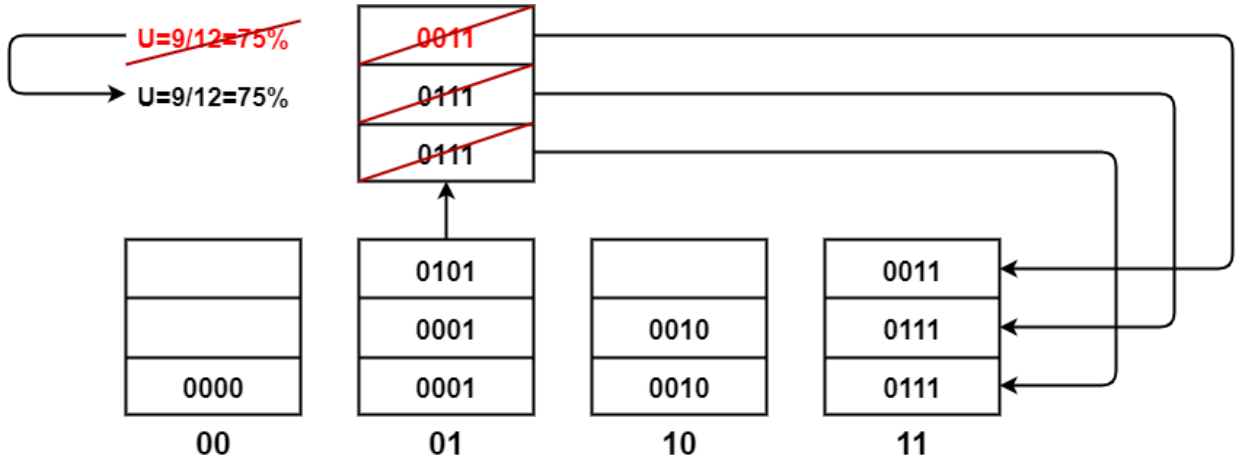
7ο κλειδί:



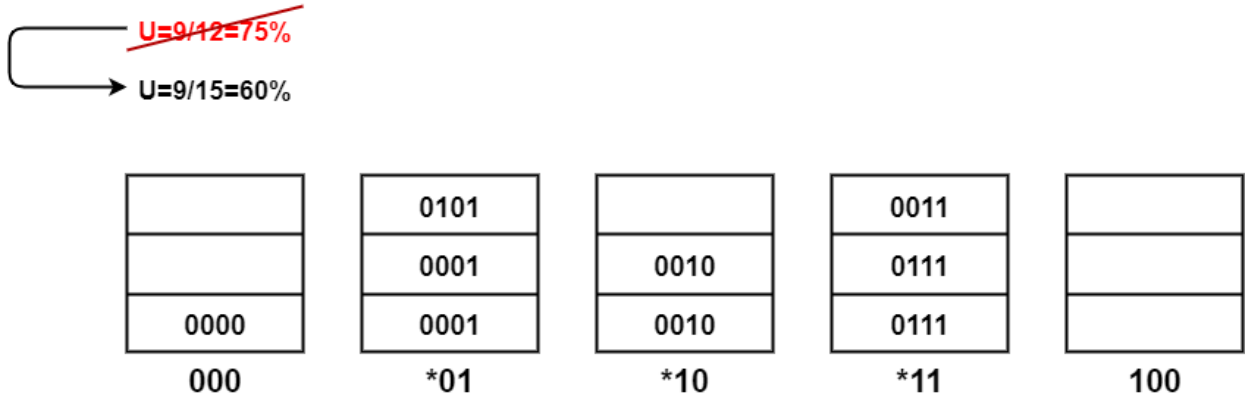
8ο κλειδί:



9ο κλειδί (1η εικόνα) :

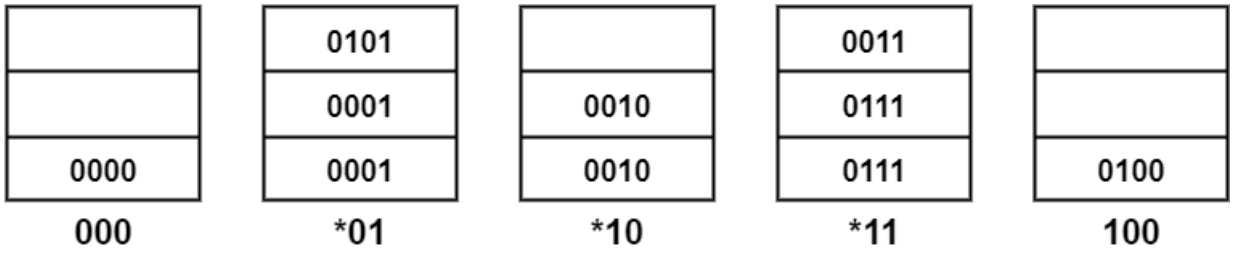


9ο κλειδί (2η εικόνα) :



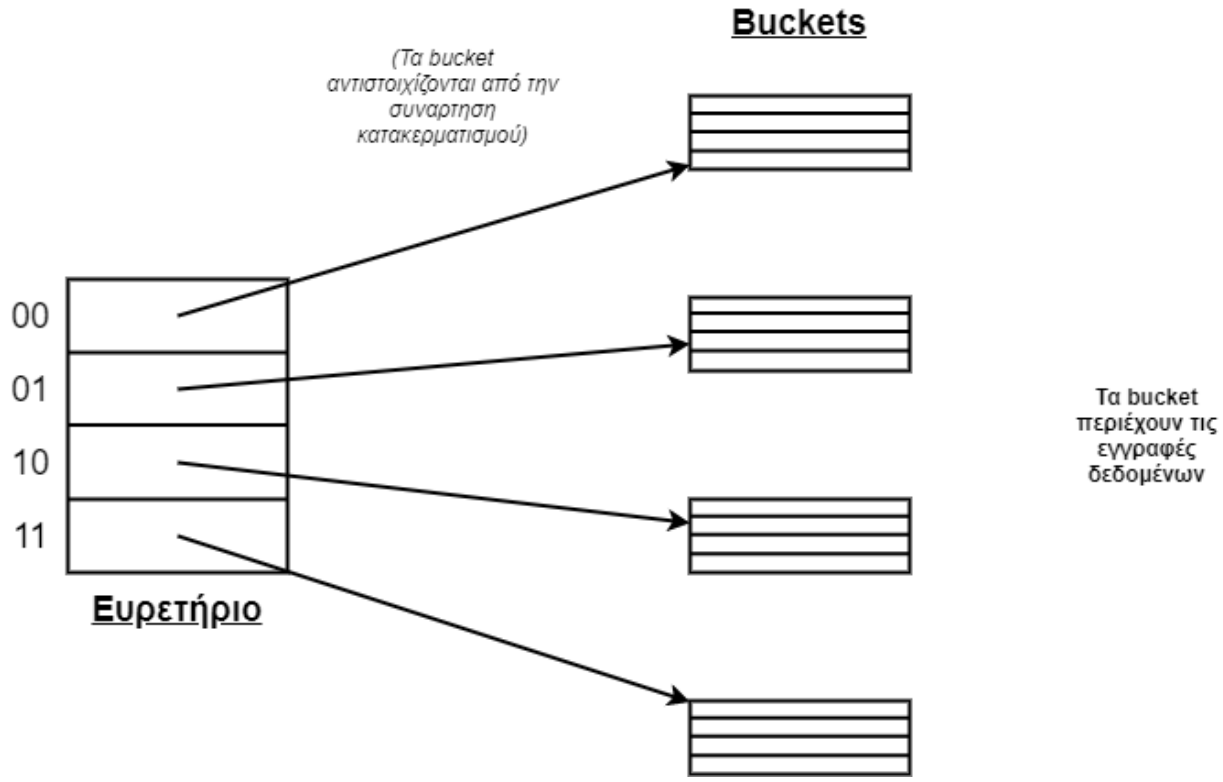
10ο κλειδί:

U=10/15=66.667%



❖ Άσκηση 6

Σε ένα περιβάλλον επεκτατικού κατακερματισμού υπάρχει μία συνάρτηση κατακερματισμού, η οποία οδηγεί σε ένα ευρετήριο. Στο ευρετήριο υπάρχουν οι εγγραφές, που δηλώνουν σε ποιο bucket είναι αποθηκευμένη η εγγραφή. Άρα κάθε εγγραφή του ευρετηρίου είναι 4 bytes και κάθε εγγραφή στα buckets είναι 400 bytes.



παράδειγμα όπου χρησιμοποιούνται 2 bits (ολικό βάθος) και προκύπτουν 4 buckets

Έτσι προκύπτουν τα εξής:

a) Από τη στιγμή που χρησιμοποιούνται 10 bits (ολικό βάθος) σημαίνει ότι το ευρετήριο έχει 2^{10} θέσεις (στο παράδειγμα με 2 bits είχαμε $2^2 = 4$ θέσεις). Άρα οι συνολικές θέσεις του ευρετηρίου είναι **1024.**

b) Κάθε εγγραφή ευρετηρίου είναι 4 bytes. Επομένως, **1024 θέσεις * 4 bytes = 4096 bytes = 4KB.**

c) Τα συνολικά buckets είναι 1024 όσες και οι θέσεις του ευρετηρίου (άλλωστε κάθε θέση δείχνει και σε ένα bucket). Έτσι έχουμε **(1024 buckets * 2400 bytes) / 400 bytes = 6.144 εγγραφές δεδομένων συνολικά σε όλα τα buckets** (αφού το πολλαπλασίασα με το 1024, που είναι το πλήθος τους)**.**

❖ Άσκηση 7

Στην περίπτωση που οι μισθοί είναι ομοιόμορφα κατανεμημένοι συμφωνώ με τον σχεδιαστή. Στην γενική περίπτωση, όμως, οι μισθοί δεν είναι ομοιόμορφα κατανεμημένοι (σε καμία χώρα, πόσο μάλλον στην Ελλάδα), αλλά κανονικά. Αυτό σημαίνει ότι κάποια buckets θα επιβαρυνθούν με πολλές εγγραφές, ενώ κάποια άλλα θα έχουν πολύ λιγότερες. Ιδανικό είναι να τα buckets να είναι γεμάτα από 50-80%(Rule of thumb). Στην προκειμένη περίπτωση, λοιπόν, δεν συμφωνώ με τον σχεδιαστή, καθώς θα έπρεπε να ορίσει περισσότερους κάδους στις περιοχές μισθών όπου ανήκουν οι περισσότεροι εργαζόμενοι.