Σειρά Ασκήσεων 1

Πέτσα Γεωργία 3200155

Άσκηση 1

1)

Με βάση την δομή του μαγνητικού δίσκου για να βρούμε την συνολική του χωρητικότητα έχουμε:

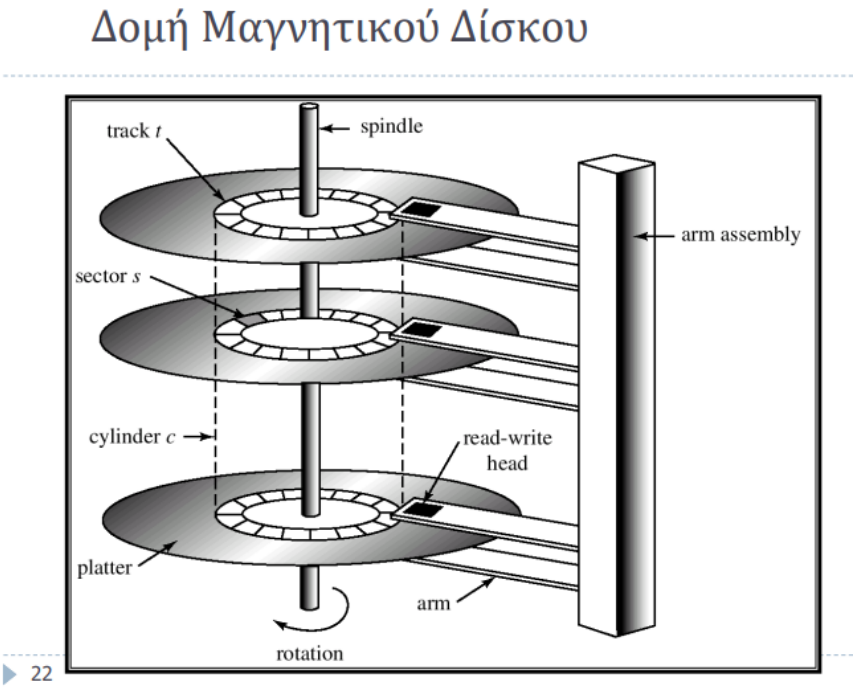
10.000 (κύλινδροι) * 1000 (sectors/track) * 10 (tracks αφού έχουμε 10 platters) * 512 bytes (το μέγεθος του κάθε sector)

=

100.000.000*512 bytes

=

51.200.000.000 bytes = 50.000.000 KB



2)

Για να βρούμε την μέση καθυστέρηση περιστροφής (Average rotation latency) γνωρίζουμε ότι κατά μέσο όρο περιμένουμε μισή περιστροφή, ότι το rpm είναι 10.000 οπότε

Έστω R ο χρόνος για μισή περιστροφή

Ξέρω ότι σε ένα λεπτό (60 sec) θα έχω 10.000 περιστροφές,

R ½ περιστροφή

60sec/R = 10.000/(1/2) => R = 60sec/20.000 => R = 0,003 sec

3)

Για να βρούμε τον χρόνο μεταφοράς ενός block (Block Transfer Time) ξέρουμε ότι εξαρτάται από την ταχύτητα περιστροφής και το μέγεθος του block, οπότε

10.000 RPM, 1.000 sectors/track. Γνωρίζουμε επίσης ότι το μέγεθος ενός block είναι 4096 bytes, άρα για να βρούμε πόσα sectors «χωράνε» σε ένα block έχουμε : 512bytes * number_of_sectors = 4096 bytes => number_of_sectors = 4096 bytes / 512 bytes => number_of_sectors = 8.

Σε 1 περιστροφή (= 60 / 10.000 secs) διαβάζω 1.000 sectors/track

Για να διαβάσω 1 block = 8 διαδοχικά sectors θέλω (8/1000) * (60/10.000) = 48/1.000.000 secs = 0,048 msec

4)

Κάθε εγγραφή καταλαμβάνει ακριβώς ένα block, άρα θα χρησιμοποιούνται 1.000.000 συνεχόμενα blocks δίσκου.

Έχουμε υπολογίσει από το προηγούμενο ερώτημα ότι χωράνε 8 sectors σ ένα block

1 block = 8 sectors είναι τα 8/1.000 ενός track

1 περιστροφή παίρνει 60/10.000= 6ms }

Transfer time = 6 * 8 /1000 = 0,048 * 1.000.000 (blocks δίσκου) = 48 secs

Average Seek time = 8ms (από εκφώνηση) = 0,008 sec

Στη συνέχεια έχουμε και το Average rotational delay = (60/10.000)/2 = 0,003s

Total time = Transfer time + **Average** Seek time + Average rotational delay = 48secs + 0,008sec + 0,003 =48,011 sec

5)

Εφόσον το ευρετήριο B+ δέντρου θα δημιουργηθεί στο πρωτεύον κλειδί του πίνακα, γνωρίζουμε ότι δεν θα περιέχει διπλές εγγραφές. Προκειμένου να ανακτήσουμε N τυχαίες εγγραφές θα έχοουμε :

(Average Seek time + Average rotational delay + Transfer Time) *(3+1) *N =

(0,008s+0,003s + 48 sec) * 4 N = 48,011 * 4 N

= 192,044 * N sec

Εξήγηση : Για να ανακτήσουμε N τυχαίες εγγραφές που δεν είναι σε συνεχόμενα block στον δίσκο θα πρέπει να υπολογίσουμε τον μέσο χρόνο που χρειαζόματε για να βρούμε μια εγγραφή στον δίσκο, μαζί με τη μέση καθυστέρυση που επέρχεται από την περιστροφή του δίσκου μαζί με τον χρόνο μεταφοράς από τον δίσκο στη μνήμη, και όλο αυτό επί 3 (διότι έχουμε 3 επίπεδα (κόμβους που πρέπει να διασχιστούν) διάσχισης του B+ δέντρου) + 1 (εφόσον το B+ δέντρο περιέχει τις αναφορές προς τις σελίδες στον δίσκο όπου βρίσκονται τα δεδομένα/ εγγραφές)

Άσκηση 2

Ευρετήριο συστάδων (clustering/clustered index). Η τιμή του γνωρίσματος ευρετηριοποίησης καθορίζει τη θέση της εγγραφής. Εδώ έχουμε ότι το γνώρισμα ευρετηριοποίησης είναι το empid άρα το clustered index είναι και primary index (Δεν έχουμε διπλοεγγραφές).

Non- clustered index: Εδώ έχουμε ότι το γνώρισμα ευρετηριοποίησης είναι το age (Πιθανή ύπαρξη διπλοεγγραφών).

1. UPDATE Employees SET age = age + 1 -> **ΕΠΙΒΡΑΔΥΝΕΤΑΙ** επειδή έχουμε ευρετήριο στην τιμή age, θα πρέπει να γίνει ενημέρωση ΟΛΟΥ του ευρετηρίου οπότε θα επέλθει κάποια καθυστέρηση.

2. UPDATE Employees SET salary = salary * 1.10 WHERE empid >=1 and empid <=100 -> **ΕΠΙΤΑΧΥΝΕΤΑΙ**

Εδώ ουσιαστικά πρέπει να γίνει ένα range query για να βρούμε τα empid που είναι ανάμεσα στο 1 και το 100 και έπειτα να κάνουμε το update του μισθού. Για να γίνει το πρώτο φιλτράρισμα είναι αποδοτική η χρήση του clustered index εφόσον το φιλτράρισμα θα γίνει στο πρωτεύον κλειδί του πίνακα. (Αποδοτικότερη η χρήση του B+ δέντρου για να απαντήσουμε σε range query)

Το non-clustered index δεν επηρεάζει το συγκεκριμένο ερώτημα διότι δεν έχουμε αναφέρει καθόλου age.

3. UPDATE Employees SET salary=salary * 1.10 WHERE departmentid = 10 ->

Εδώ πρώτα θα πρέπει να γίνει το φιλτράρισμα ώστε να βρεθούν τα departments με id = 10. Ωστόσο δεν υπάρχει κάποιο ευρετήριο στο departmentid οπότε κανένα από τα 2 ευρετήρια που έχουμε

ΔΕΝ ΘΑ ΕΠΗΡΕΑΣΕΙ την ταχύτητα του query

Άσκηση 4

Για να υπολογίσουμε το πόσα blocks δίσκου θα χρειαστούν εργαζόμαστε ως εξής:

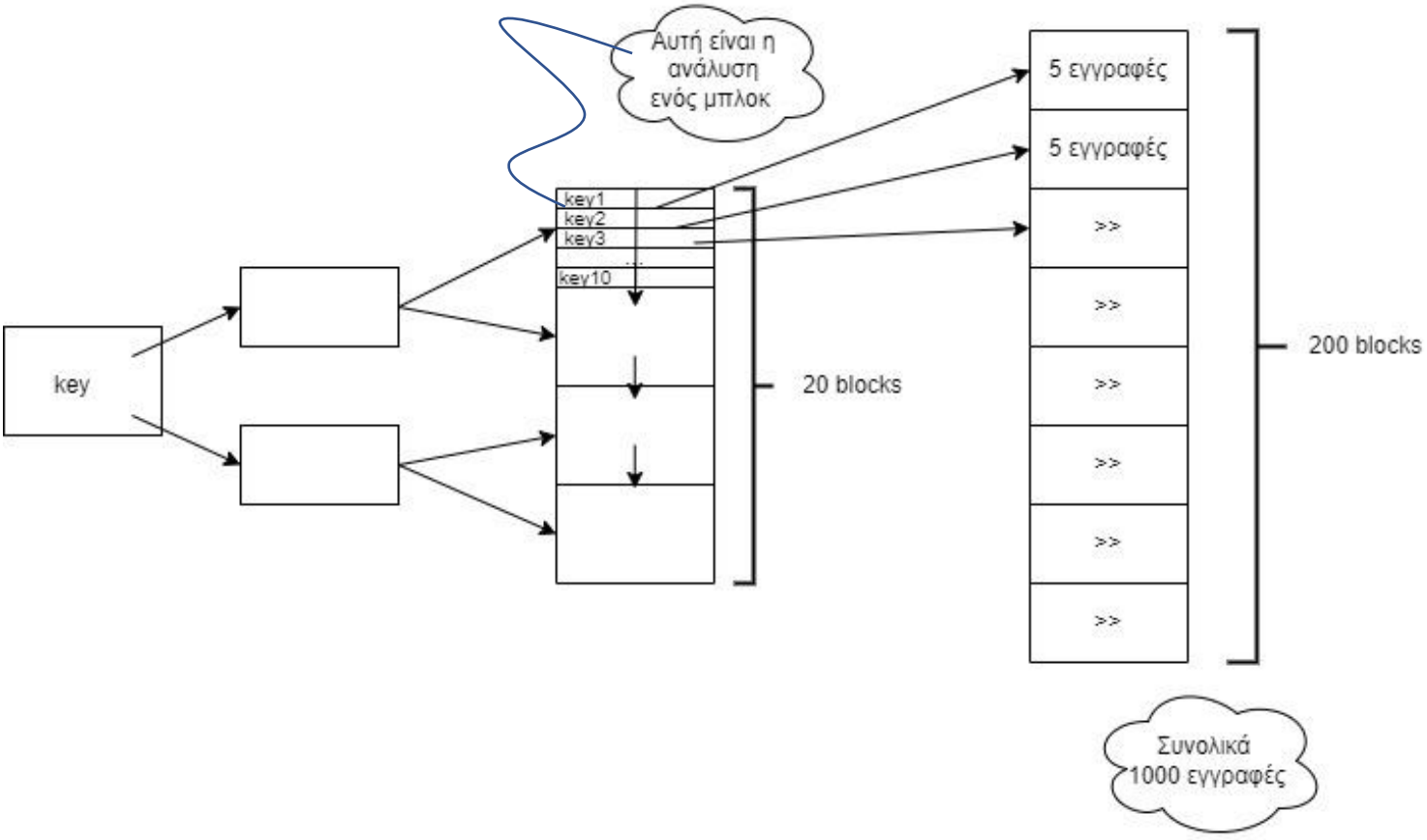
Αρχικά γνωρίζουμε ότι έχουμε 1000 εγγραφές, και ότι κάθε μπλοκ χωράει από 5 εγγραφές, οπότε έχουμε αρχικά 1000 / 5 = 200 blocks στο δίσκο.

Φτιάχνουμε αραιό ευρετήριο οπότε θα έχουμε καταχωρήσεις μόνο για μερικές από τις τιμές αναζήτησης (Καταγράφω την πρώτη τιμή του κλειδιού σε κάθε σελίδα) οπότε σύμφωνα με τους κανόνες δημιουργία ευρετηρίου θα έχουμε

Σε αυτά τα 200 block θέλουμε να δείχνουν pointers από τα keys των ευρετηρίων, συνεπώς θα έχουμε 200 (blocks) / 10 keys (που χωράει κάθε μπλοκ) = 20 leaf nodes. Κάθε ένα από τα 20 φύλλα περιέχει 10 keys και 10 pointers προς τα μπλοκς + 1 pointer προς το επόμενο φύλλο (αφού είμαστε σε leaf nodes). Οπότε έχουμε άλλα 20 μπλοκ για το ευρετήριο. Τώρα, πρέπει να βρούμε και πόσους ενδιάμεσους κόμβους θα έχουμε. Έχουμε 10 κλειδια και 11 pointers μέσα σε κάθε μπλοκ, οπότε θα χρειαστούμε άλλα 2 μπλοκ για να έχουμε pointers προς τα 20 φύλλα. Τέλος, θα πρέπει να υπάρχει και μία ρίζα (ένα μπλοκ) η οποία περιέχει 1 κλειδί και 2 pointers.

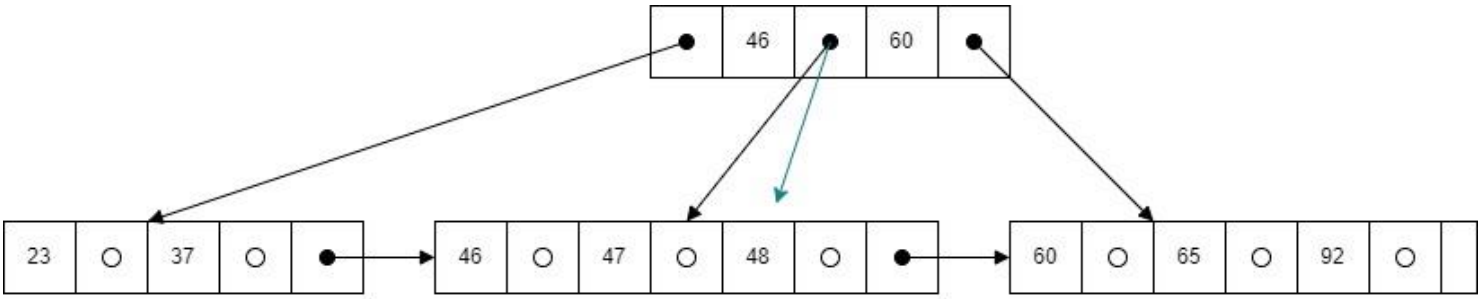
Επομένως συνολικά έχουμε 200 + 20 + 2 + 1 = 223 blocks

Σχηματικά μπορούμε να το δούμε ως εξής



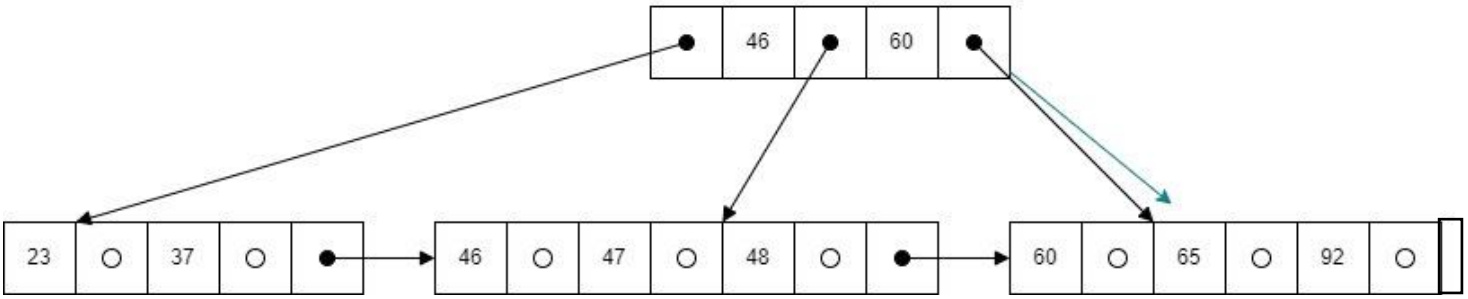
Άσκηση 3

1)

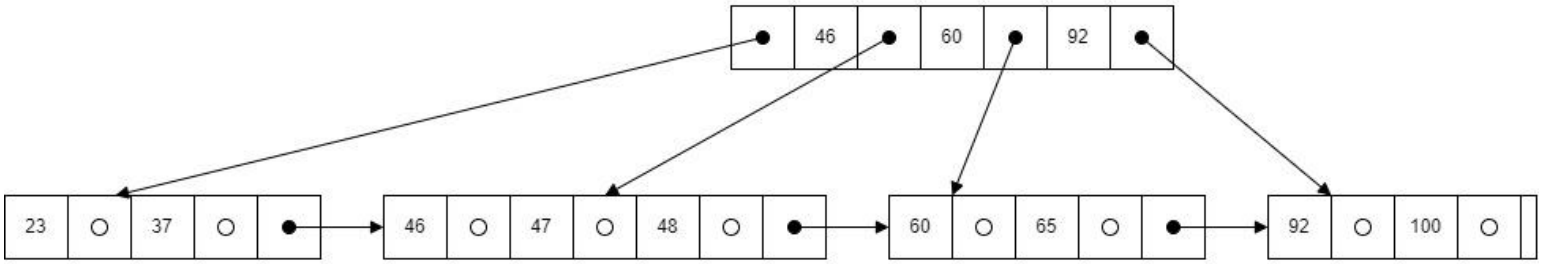


Για την εισαγωγη του 47 έχουμε:
Αρχικά κάνουμε αναζήτηση του 47 (εμφανίζονται με γαλαζια βέλη τα βήματα)
Καταλήγουμε σε κόμβο φύλλο, βλέπουμε ότι 46<47<60, οπότε εισαγάφουμε το 47 εφόσον υπάρχει χώρος

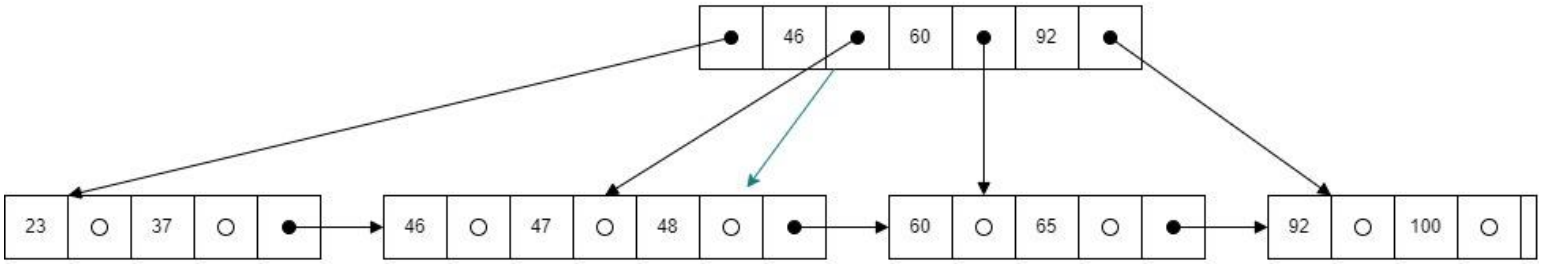
Για την εισαγωγη του 100 έχουμε:
Αρχικά κάνουμε αναζήτηση του 100 (εμφανίζονται με γαλαζια βέλη τα βήματα)



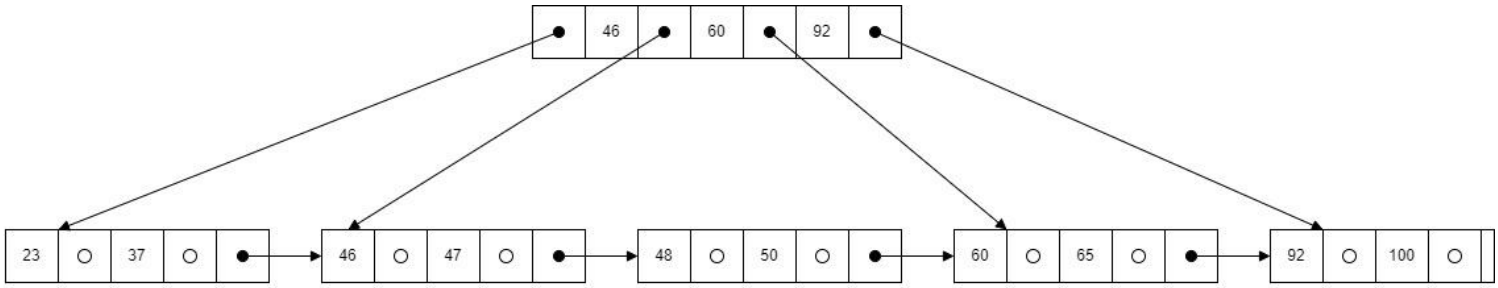
Καταλήγουμε σε κόμβο φύλλο, και παρατηρούμε ότι είναι γεμάτος. Άρα έχουμε leaf-overflow
Οπότε φτιάχνουμε έναν καινούριο κόμβο και μοιράζουμε τα στοιχεία στους δύο για να δημιουργηθεί χώρος.
Επίσης ανενημερώνουμε το καινούριο κλειδί στη ρίζα και τοποθετούμε pointer προς τον κόμβο που φτιάξαμε

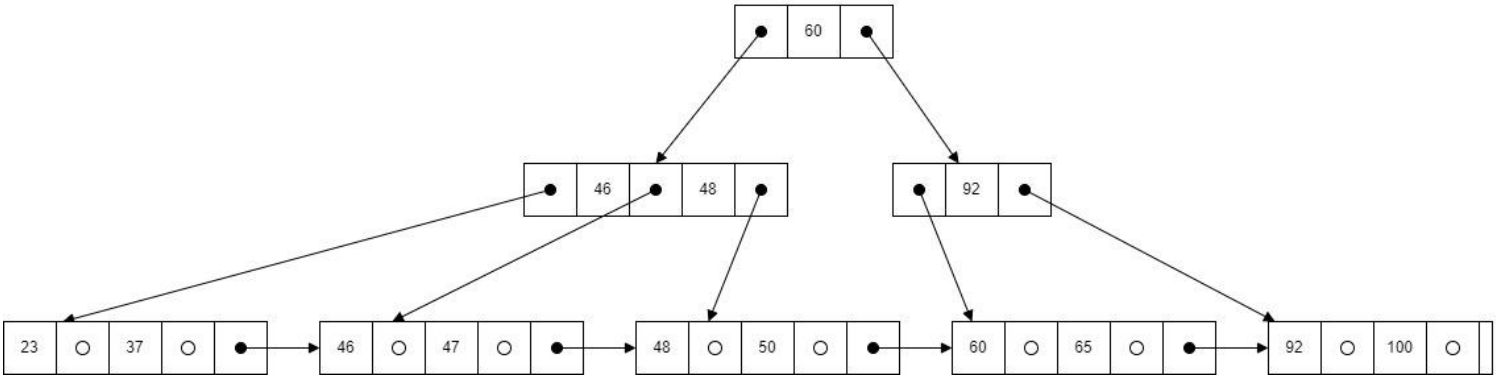


Για την εισαγωγη του 50 έχουμε:
Αρχικά κάνουμε αναζήτηση του 50(εμφανίζονται με γαλαζια βέλη τα βήματα)

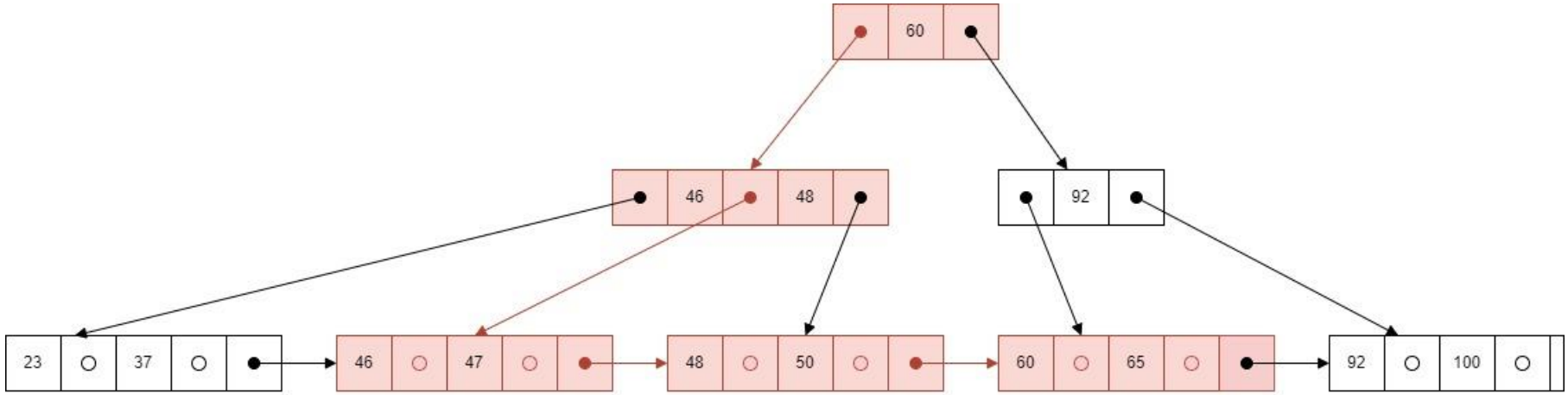


Καταλήγουμε σε κόμβο φύλλο, και παρατηρούμε ότι είναι γεμάτος. Άρα έχουμε leaf-overflow
Εδώ δεν μπορούμε να δημιουργήσουμε καινούριο κόμβο, διότι έχουμε ήδη ότι n=3 που είναι το όριο.
Επομένως θα διασπάσουμε τον προηγούμενο λόμβο, που στην συγκεκριμένη περίπτωση είναι η ρίζα.
Οπότε έχουμε:


46
60
92
23
37
46
47
48
50
60
65
92
100


60
46
48
92
23
37
46
47
48
50
60
65
92
100

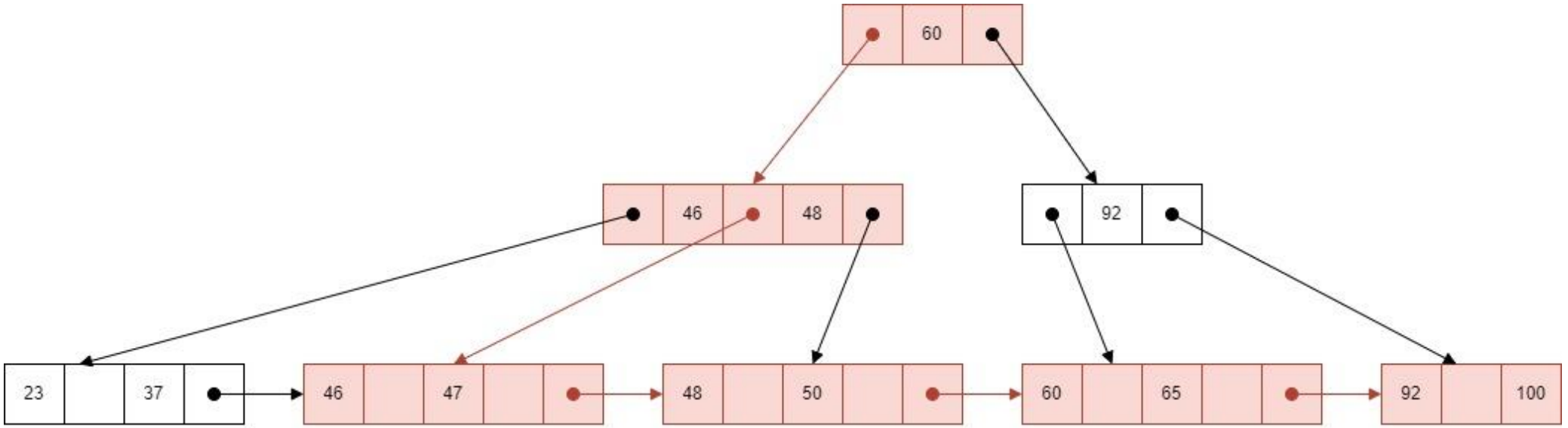
2) α



2α)
Το κλειδί αναζήτησης είναι μοναδικό, οπότε για να απαντήσουμε το ερώτημα εύρους πόσοι και ποιοι κόμβοι πρέπει να προσπελαστούν για να ανακτηθούν όλες οι εγγραφές με κλειδί αναζήτησης key>=46 AND key<=65 κάνουμε το εξής:
Αναζητούμε την τιμή 46. (46<60 -> 46<=46 -> 46 [3 κόμβοι]) Φτάνουμε στην τιμή 46. Από εκεί και μετά, διασχίζουμε τους δεξιότερους κόμβους μέσα από τους δείκτες μέχρι να φτάσουμε σε τιμή >=65. Βλέποπυμε ότι η τιμή 65 βρίσκεται 2 κόμβους αργότερα, οπότε και σταματάμε εκεί την αναζήτηση.

Γενικά η διασικασία αναζήτησης έχει επισημανθεί με ροζ χρώμα. Συνολικά παρατηρούμε ότι διασχίσαμε 3+2 = 5 κομβους.

2) β



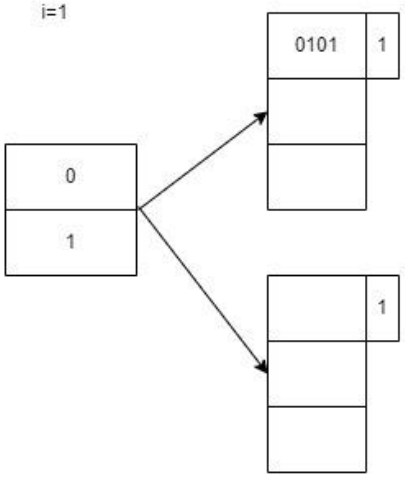
2β)
Το κλειδί αναζήτησης ΔΕΝ είναι μοναδικό, οπότε για να απαντήσουμε το ερώτημα εύρους πόσοι και ποιοι κόμβοι πρέπει να προσπελαστούν για να ανακτηθούν όλες οι εγγραφές με κλειδί αναζήτησης key>=46 AND key<=65 κάνουμε το εξής:
Αναζητούμε την τιμή 46. (46<60 -> 46<=46 -> 46 [3 κόμβοι]) Φτάνουμε στην τιμή 46. Από εκεί και μετά, διασχίζουμε τους δεξιότερους κόμβους μέσα από τους δείκτες μέχρι να φτάσουμε σε τιμή >65. Βλέποπυμε ότι η τιμή 65 βρίσκεται 2 κόμβους αργότερα [+2 κόμβοι], όμως επειδή το κλειδί αναζήτησης δεν είναι μοναδικό, θα μπορούσε να υπάρχει και παρακάτω κι άλλος κόμβος με τιμή 65. Συνεπώς, δεν σταματάμε εκεί την αναζήτηση αλλά διασχίζουμε και τον επόμενο κόμβο [+1 κόμβος], όπου και βλέπουμε ότι υπαρχει η τιμή 92 > 65, Συνεπώς σταματάμε

Γενικά η διασικασία αναζήτησης έχει επισημανθεί με ροζ χρώμα. Συνολικά παρατηρούμε ότι αυτή τη φορά διασχίσαμε 3+2+1 = 6 κομβους.

Σειρά Ασκήσεων 1

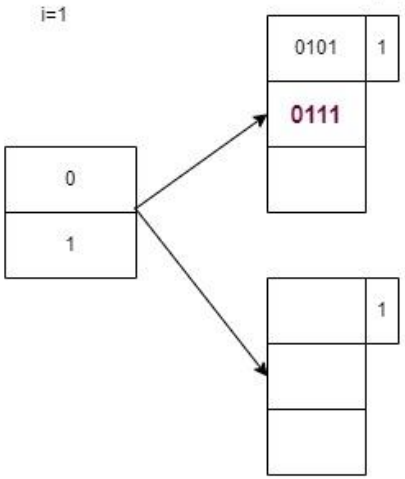
Πέτσα Γεωργία 3200155

Άσκηση 5 *1)*



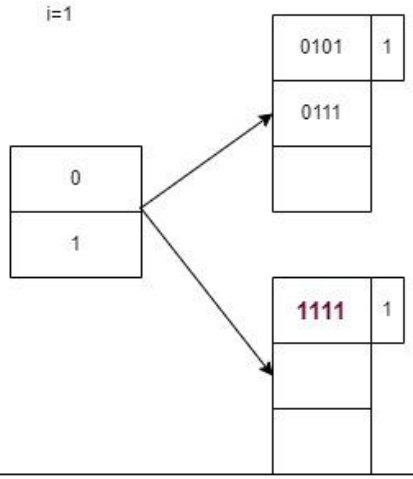
Παρατηρούμε ότι i=1, άρα κοιτάμε το πρώτο ψηφίο του αριθμού, άρα θα πάει στον κάδο νούμερο 0

Για το 0111



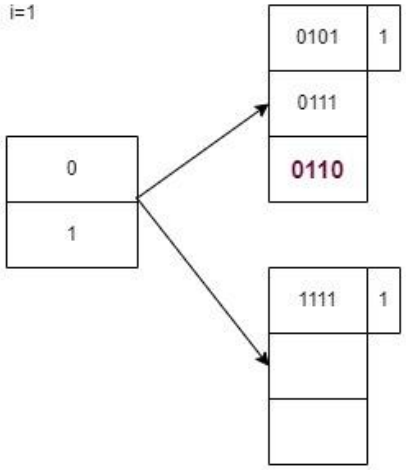
Παρατηρούμε ότι i=1, άρα κοιτάμε το πρώτο ψηφίο του αριθμού, άρα θα πάει στον κάδο νούμερο 1

Για το 1111



Για το 0110,

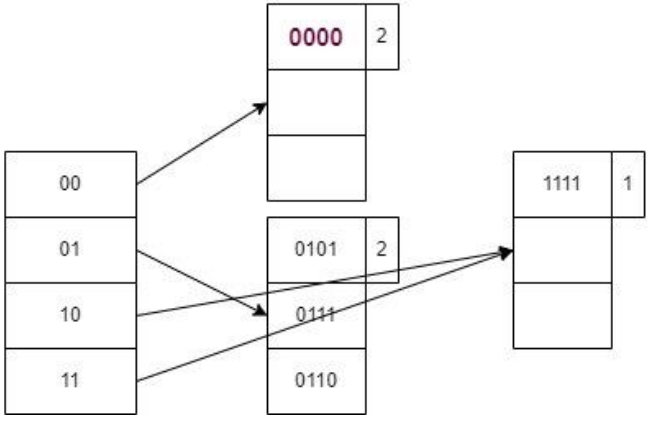
Παρατηρούμε ότι i=1, άρα κοιτάμε το πρώτο ψηφίο του αριθμού, άρα θα πάει στον κάδο νούμερο 0



Για το 0000,

Παρατηρούμε ότι i=1, άρα κοιτάμε το πρώτο ψηφίο του αριθμού, άραθα πάει στον κάδο νούμερο 0, αυτός όμως είναι γεμάτος, οπότε θα έχουμε διασπαση. Αυξάνεται το τοπικό βάθος κατά 1. Παρατηρούμε ότι το νέο τοπικό βάθος είναι 2>1 οπότε θα αυξηθεί και το ολικό, δηλαδή διπλασιάζεται και το ευρετήριο και αυξάνεται το ολικό βάθος κατά ένα. Δηλαδή τώρα i = 2.

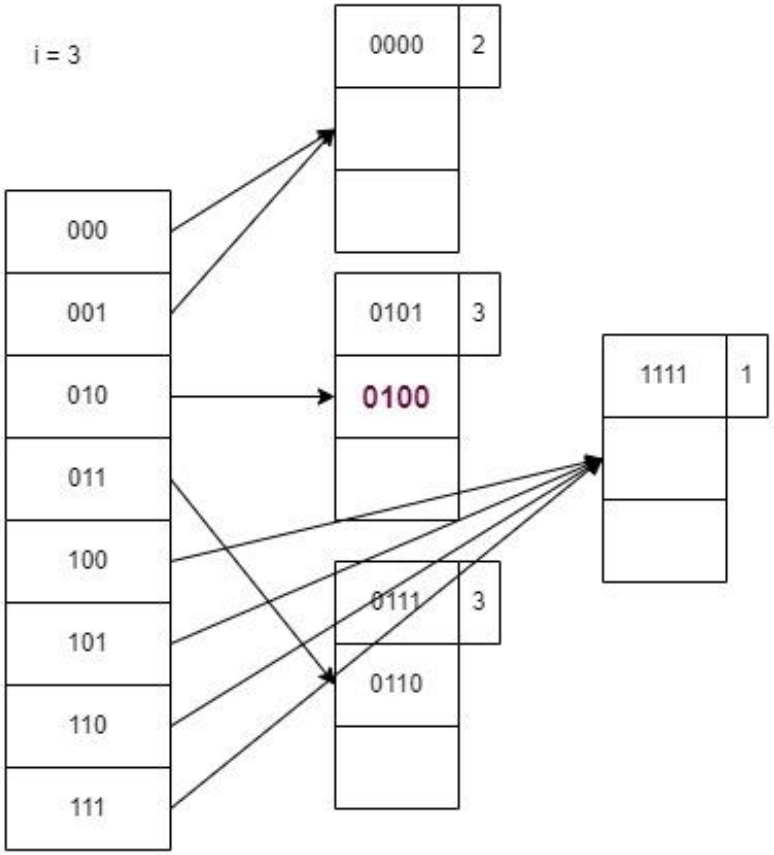
0000



Για το 0100,

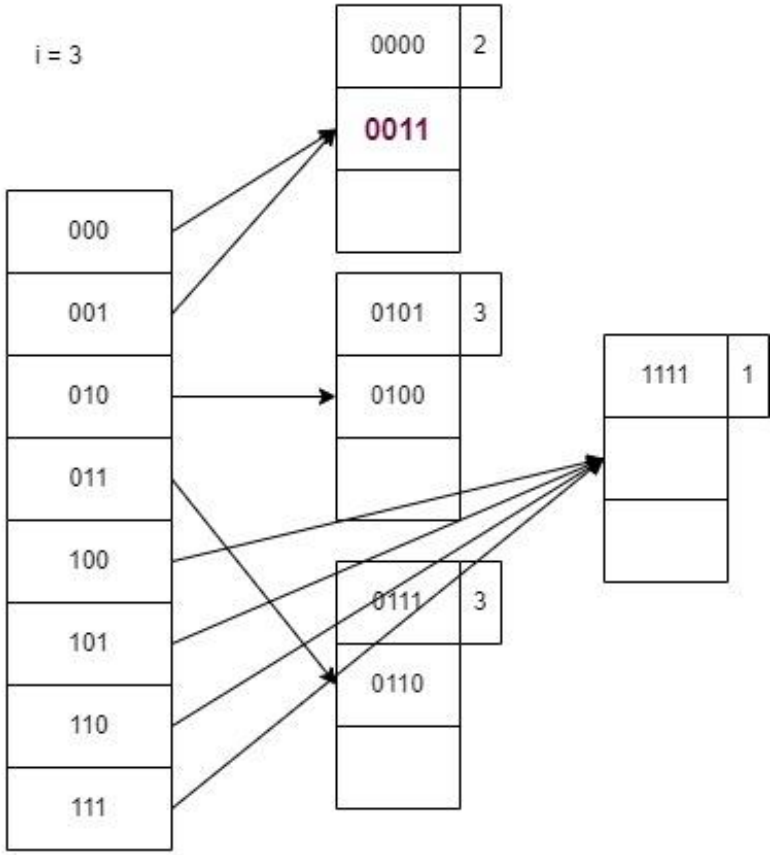
Παρατηρούμε ότι i=2, άρα κοιτάμε τα πρώτα
2 ψηφία του αριθμού, άρα θα πάει στον κάδο νούμερο 01, αυτός όμως είναι γεμάτος, οπότε θα έχουμε διασπαση. Αυξάνεται το τοπικό βάθος κατά 1. Παρατηρούμε ότι το νέο τοπικό βάθος είναι 3>2 οπότε θα αυξηθεί και το ολικό, δηλαδή
διπλασιάζεται και το ευρετήριο και αυξάνεται το ολικό βάθος κατά ένα. Δηλαδή τώρα i = 3.

0100,



Για το 0011,

Παρατηρούμε ότι i=3, άρα κοιτάμε τα πρώτα
3 ψηφία του αριθμού, άρα
θα πάει στον κάδο νούμερο 001 (που είναι ίδιος με τον 000)

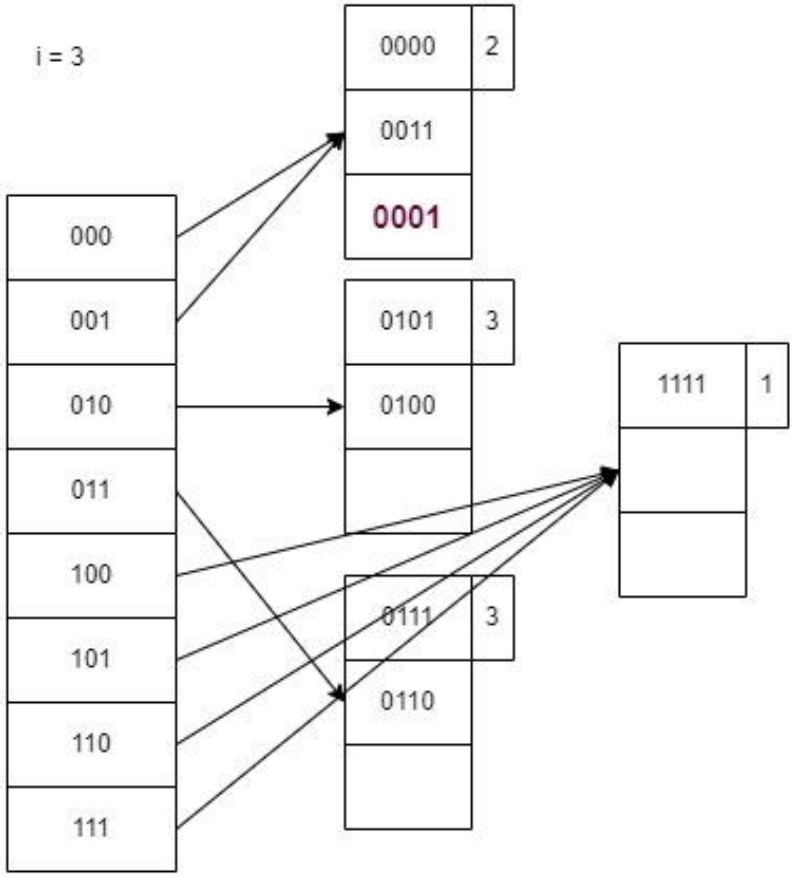


Σειρά Ασκήσεων 1

Πέτσα Γεωργία 3200155

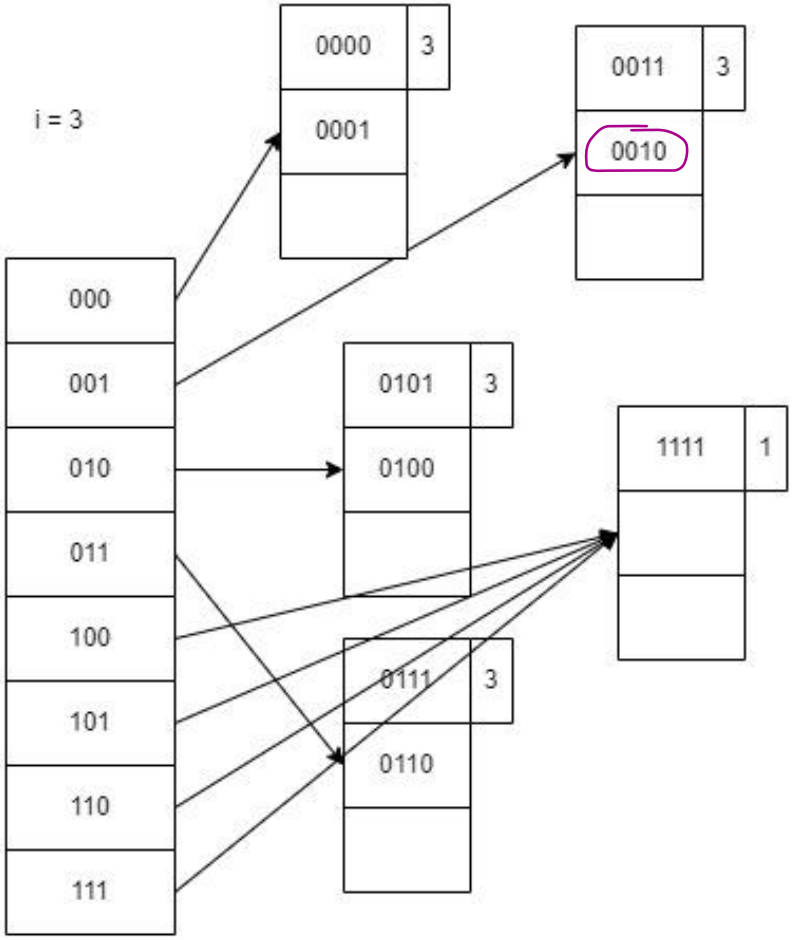
Για το 0001,

Παρατηρούμε ότι i=3, άρα κοιτάμε τα πρώτα 3 ψηφία του αριθμού, άρα θα πάει στον κάδο νούμερο 000 (που είναι ίδιος με τον 001)



Για το 0010,

Παρατηρούμε ότι i=3, άρα κοιτάμε τα πρώτα 3 ψηφία του αριθμού, άρα θα πάει στον κάδο νούμερο 001, αυτός όμως είναι γεμάτος, οπότε θα έχουμε διασπαση. Αυξάνεται το τοπικό βάθος κατά 1.

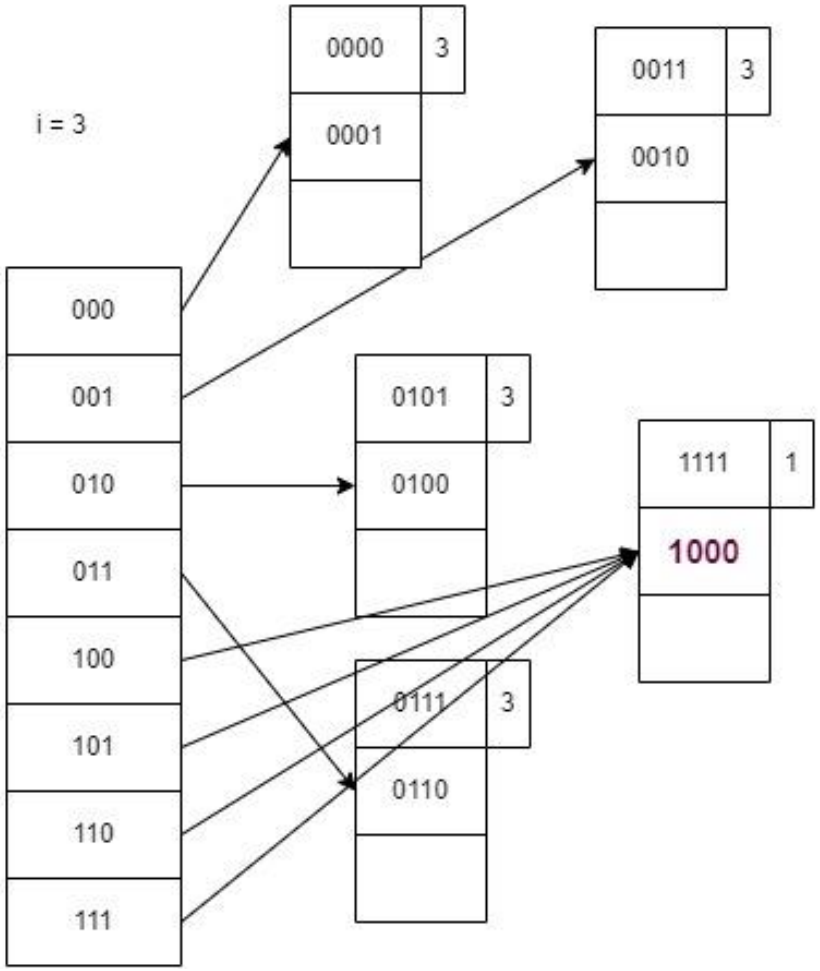


Σειρά Ασκήσεων 1

Πέτσα Γεωργία 3200155

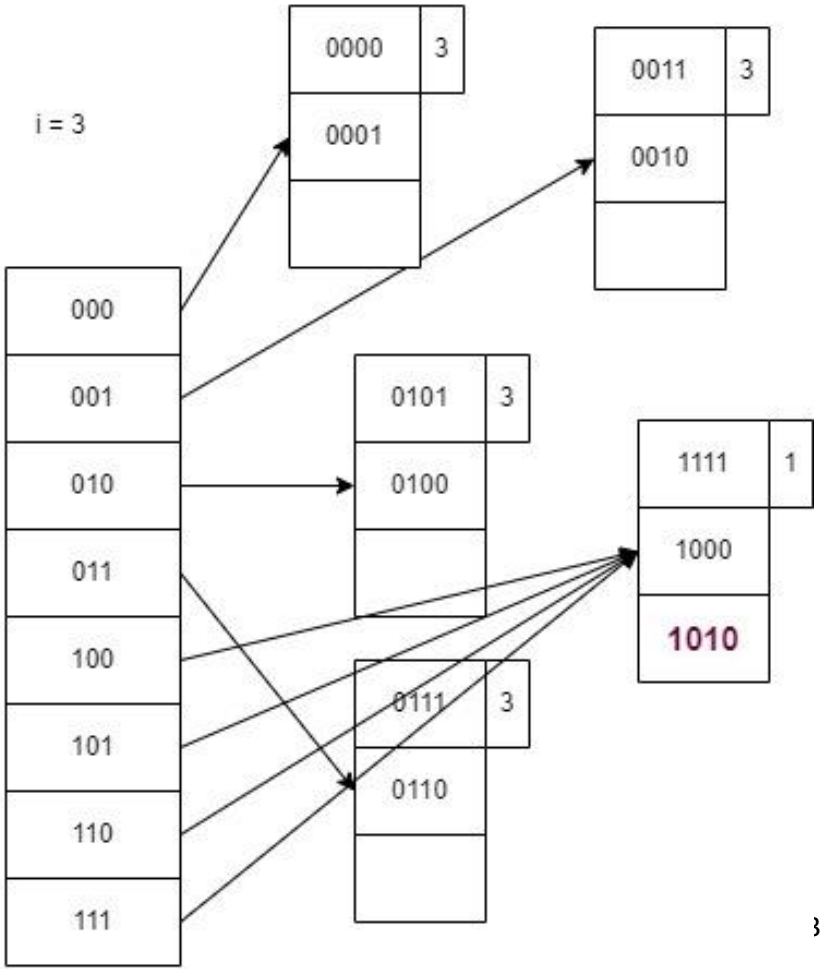
Για το 1000,

Παρατηρούμε ότι i=3, άρα κοιτάμε τα πρώτα 3 ψηφία του αριθμού, άρα θα πάει στον κάδο νούμερο 100 (που είναι ίδιος με τον 101, 110, 111)



Για το 1010,

Παρατηρούμε ότι i=3, άρα κοιτάμε τα πρώτα 3 ψηφία του αριθμού, άρα θα πάει στον κάδο νούμερο 101 (που είναι ίδιος με τον 101, 110, 111)



2)

Ο ελάχιστος αριθμός τιμών που θα οδηγούσε σε διπλασιασμό του ευρετηρίου, θα ήταν 2. Αυτό συμβαίνει γιατί για να οδηγηθούμε σε διπλασιασμό του ευρετηρίου θα πρέπει να γεμίσει κάποιος από τους κάδους με τοπικό βάθος ίσο με το ολικό (δλδ 3), και να έρθει άλλη μία τιμή που η τιμή του hash της θα ανήκει στον ίδιο κάδο, ώστε να οδηγηθούμε σε διάσπαση και κατά συνέπεια σε διπλασιασμό του ευρετηρίου.

Κάποιο παράδειγμα τιμών θα μπορούσε να είναι οι τιμές 0101 και 0100 (που παρόλο που υπάρχουν ήδη θα μπορούσε να δημιουργηθεί σύγκρουση και να οδηγηθούμε σε διπλασιασμό)

Σε αυτήν την περίπτωση, έστω πως η πρώτη εισαγωγή είναι του 0101, η οποία θα γέμιζε τον κάδο 010 και στη συνέχεια η δεύτερη εισαγωγή του 0100 θα οδηγούσε σε διάσπασή του, αυξάνοντας αρχικά το τοπικό βάθος σε 4 και μετά και το ολικό (εφόσον 4 > 3 άρα νέο i = 4).

Άσκηση 6

Γνωρίζουμε ότι έχουμε κάδους με N εγγραφές, χρησιμοποιώντας επεκτατικό κατακερματισμό. Για να συμβεί υπερχείλιση σε κάποιον κάδο, θα πρέπει εκείνος να έχει N+1 εγγραφές, δηλαδή να έρθει μία καινούρια εγγραφή όπου η τιμή του h(key) να είναι ίδια με τον κάδο με τις N εγγραφές. Εφόσον θα συμβεί υπερχείλιση γνωρίζουμε ότι θα αυξηθεί το τοπικό βάθος του κάδου κατά ένα. Οπότε, για να υπολογίσουμε την πιθανότητα ένας κάδος να χρειαστεί να αντιμετωπιστεί αναδρομικά θα πρέπει τα πρώτα i+1 bits να είναι ίδια. Γνωρίζουμε ωστόσο εκ των προτέρων ότι τα πρώτα i bits είναι ήδη ίδια, οπότε έχοντας αυτό σαν δεδομένο ψάχνουμε την πιθανότητα το i-οστό + 1 bit να είναι ίδιο σε κάθε μία από τις n+1 εγγραφές. Αυτό λοιπόν ισούται με $(½)^{n+1}$