1η Σειρά Ασκήσεων

Δημήτριος Τσιομπίκας AM : 3180223

Άσκηση 1

a)

Έχουμε : 4096 bytes ανά τομέα (sector)

256 τομείς ανά ίχνος

Άρα η χωρητικότητα ενός ίχνους είναι :

4096 * 256 = **1,048,576 bytes**

Η μία επιφάνεια έχει 65536 ίχνη άρα η χωρητικότητα μίας επιφάνειας θα είναι :

1,048,576 * 65536 = **68,719,476,736 bytes**

Ο Δίσκος έχει 8 πλακέτες διπλής όψεως δηλαδή 16 επιφάνειες.

Άρα η χωρητικότητα του δίσκου θα είναι :

68,719,476,736 * 16 = **1,099,511,627,776 bytes**

b)

Οι αριθμοί των κυλίνδρων του δίσκου ισούνται με τον αριθμό των tracks ανά επιφάνεια. Οπότε έχουμε 65536 κυλίνδρους.

c)

<u>Μέση Καθυστέρηση Περιστροφής</u> :

60 secs 7200 περιστροφές

R ½ περιστροφή (μισή)

Άρα 7200 R = 30 => R = 30/7200 secs => R = 0.00417 secs

Οπότε R = 4.17 ms

<u>Μέγιστη Καθυστέρηση Περιστροφής</u>

60 secs 7200 περιστροφές

R 1 περιστροφή (πλήρης)

Άρα 7200R = 60 => R = 60/7200 secs => R = 0.00833 secs

Οπότε R = 8.33 ms

d)

Αρχικά βρίσκουμε πόσες φορές περιστρέφεται ο δίσκος σε 1 δευτερόλεπτο.

60/7200 = 1/120

Άρα περιστρέφεται 120 φορές σε 1 sec.

Άρα το transfer rate ισούται με :

<u>120 * μέγεθος ενός track = 120 * 1,048,576 = 125,829,120 bytes/sec = 120 MB/sec.</u>

Άσκηση 2

a)

Εφόσον η εγγραφή στο Primary Key παίρνει 10 bytes και ο pointer του δέντρου είναι 6 bytes το συνολικό μέγεθος μιας εγγραφής στο γνώρισμα a θα είναι 16 bytes.

Το μέγεθος μιας σελίδας είναι 1024 bytes άρα έχουμε :

1024/16 = 64. Άρα κάθε σελίδα περιέχει 64 εγγραφές.

Παίρνοντας τα υπόλοιπα γνωρίσματα χωρίς τον δείκτη , βλέπουμε ότι το μέγεθος μιας εγγραφής θα είναι : 10 + 50 + 30 + 18 + 20 = 128 άρα:

1024/128 = 8 εγγραφές.

Οπότε ο αριθμός των απαιτούμενων block (σελίδων) θα είναι

N/8 + N/64.

b)

Εφόσον στο Sparse Index πρέπει να παίρνουμε μόνο την πρώτη καταγραφή από κάθε σελίδα αυτό σημαίνει ότι παίρνουμε το 1/8 κάθε φορά (εφόσον κάθε σελίδα έχει 8 εγγραφές).

Οπότε έχουμε σε κάθε σελίδα :

16/8 = 2 εγγραφές (γνωρίζουμε το 16, μέγεθος της εγγραφής με pointer από το προηγούμενο ερώτημα)

Άρα έχουμε :

1024/2 = 512 εγγραφές σε κάθε σελίδα.

Οι εγγραφές χωρίς δείκτη παραμένουν οι ίδιες οπότε ο αριθμός των απαιτούμενων block θα είναι :

N/8 + N/512.

Άσκηση 3

1)

Η αναζήτηση ξεκινά από την ρίζα (που έχει τιμή 13) , παρατηρεί ο αλγόριθμος ότι 41 > 13 και συνεχίζει στο δεξί μέρος του B+ tree. Εδώ παρατηρεί ότι 31 < 41 < 43 οπότε πάει στον κόμβο που περιέχει τις τιμές 31,37,41. Εδώ θα εκτελέσει Binary Search και θα καταλήξει στην τιμή 41 , που είναι η τιμή που θέλαμε και θα επιστρέψει θετικό μήνυμα.

2)

Τα βήματα που θα ακολουθηθούν θα είναι ακριβώς τα ίδια με το παραπάνω υποερώτημα αλλά αυτή τη φορά , αφού τελειώσει η Δυαδική Αναζήτηση ο αλγόριθμος θα επιστρέψει ότι η αναζήτηση **απέτυχε** καθώς το 40 δεν υπάρχει στο B+ Tree.

3)

Εδώ έχουμε ένα range query που θέλει να βρούμε όλες τις τιμές < 30. Αρχικά βρίσκουμε το min του Δέντρου που εδώ είναι το 2. Μόλις ο αλγόριθμος το βρει ξεκινάει από τον πρώτο

κόμβο που περιέχει τις τιμές 2,3,5 , τελειώνει στον κόμβο με τις τιμές 23,29 και επιστρέφει όλες αυτές τις τιμές **(δηλαδή 2,3,5,7,11,13,17,19,23,29).**

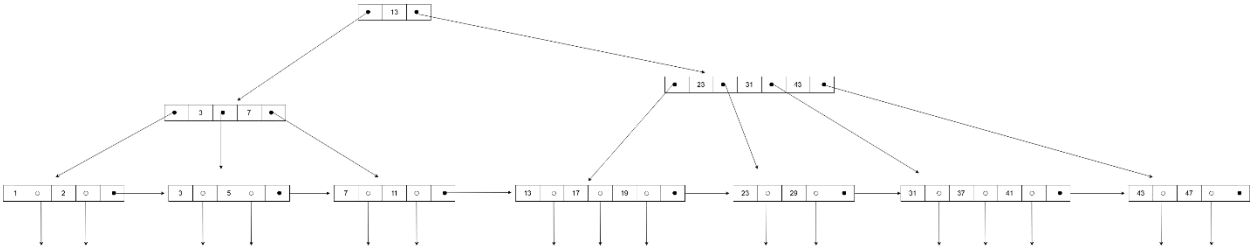
4)

Εδώ έχουμε άλλο ένα range query που ζητάει όλες τις τιμές >= 20 και <= 35. Ξεκινάει ο αλγόριθμος από τη ρίζα και διαπιστώνει ότι 20 > 13 άρα συνεχίζει δεξιά. Εδώ παρατηρεί ότι 20 < 23 άρα συνεχίζει στον κόμβο με τιμές 13,17,19. Όμως εδώ όλες οι τιμές είναι < 20 άρα ο αλγόριθμος συνεχίζει στον επόμενο κόμβο. Εδώ οι τιμές 23,29 είναι >= 20 και <= 35 οπότε τις κρατάει και συνεχίζει στον επόμενο κόμβο. Η τιμή 31 είναι >= 20 και <= 35 αλλά οι τιμές 37,41 είναι > 35 οπότε ο αλγόριθμος σταματάει στο 31 και επιστρέφει τις τιμές **23,29,31.**

5)

Ξεκινάμε με αναζήτηση για το κλειδί με τιμή 1. Ο αλγόριθμος παρατηρεί ότι 1 < 13 οπότε πάει στο αριστερό παιδί της ρίζας. Το 1 < 7 οπότε ο αλγόριθμος πάει στο αριστερό παιδί (που έχει τιμές 2,3,5) . Εδώ εκτελείται Δυαδική αναζήτηση και συμπεραίνει ότι δεν υπάρχει το 1. Επειδή όμως δεν υπάρχει χώρος , ο αλγόριθμος θα τοποθετήσει το 3 στον γονέα κόμβο μαζί με το 7 και θα δημιουργήσει ένα νέο παιδί στα αριστερά

του 3 που θα περιέχει το 1 και το 2 και στη μέση θα υπάρχει το 3 και το 5. Το δέντρο θα είναι έτσι :



6)

InsertKey(14)

Αρχικά εδώ με την εισαγωγή του 14 έχουμε no leaf overflow στον δεξί κόμβο της ρίζας οπότε δημιουργούμε νέα ρίζα με τις τιμές 13,31. Δημιουργείται νέο παιδί ρίζας που περιέχει τις τιμές το 17,23 μέχρι στιγμής και η διάταξη θα είναι ως εξής :

Παιδί < 17 περιέχει : 13,14

Παιδί >= 17 και < 23 περιέχει : 17,19

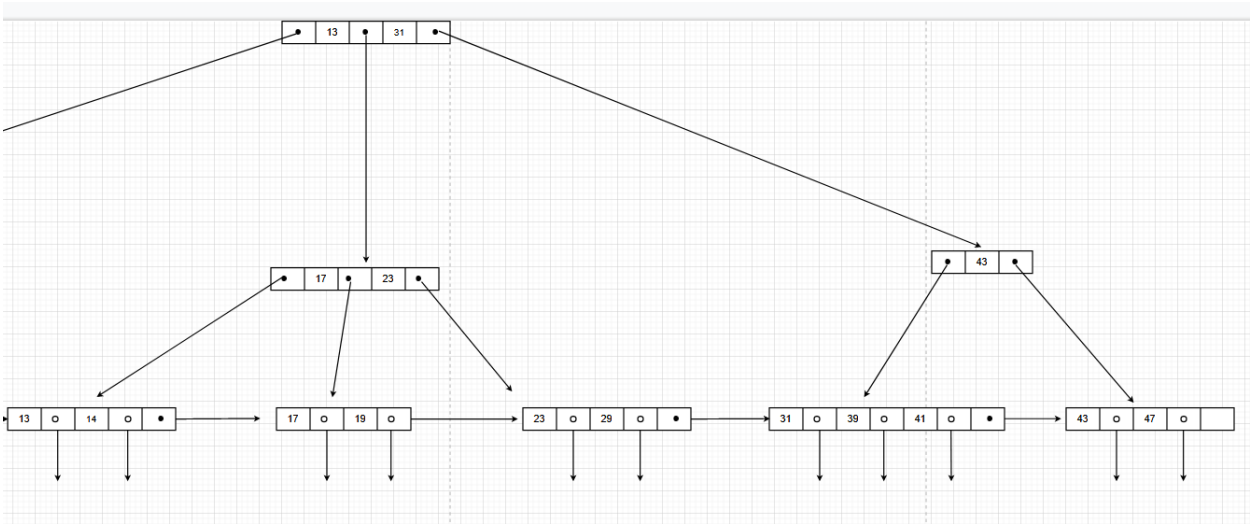
Παιδί > 23 περιέχει : 23,29

Επίσης στα δεξιά της ρίζας (> 31) το παιδί θα έχει την τιμή 43 και:

Παιδί < 43 περιέχει : 31,37,41

Παιδί >= 43 περιέχει : 43

Το αριστερό μέρος του δέντρου παραμένει ίδιο με πριν.



InsertKey(15)

Το 15 πηγαίνει στο μεσαίο παιδί (μέχρι στιγμής έχουμε τιμές 15,17,23 στο μεσαίο παιδί) και έχουμε το εξής δέντρο:

Μεσαίο παιδί ρίζας : 15 17 23

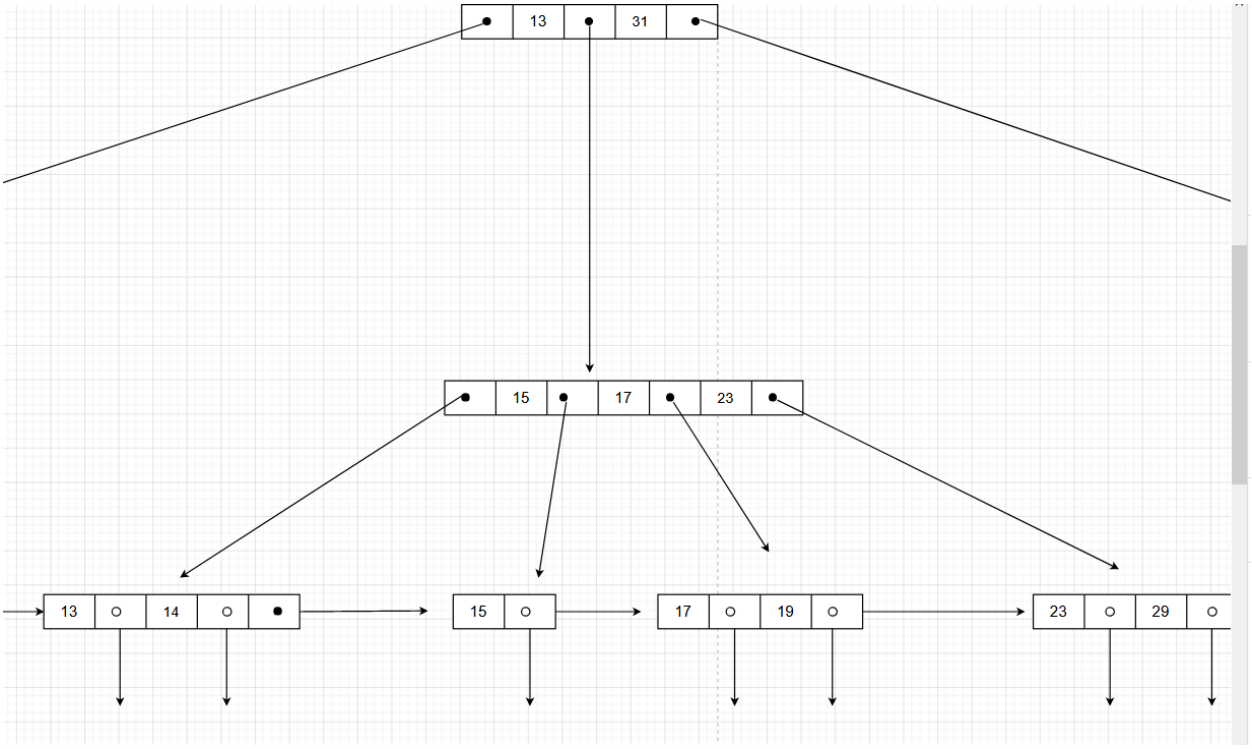
Παιδί < 15 : 13,14

Παιδί >= 15 και < 17 : 15

Παιδί >= 17 και < 23 : 17,19

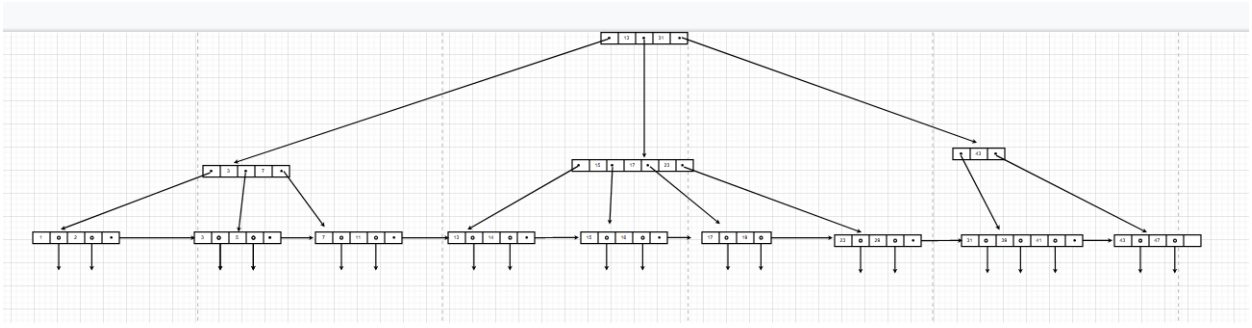
Παιδί > 23 : 23,29

Το υπόλοιπο δέντρο παραμένει ως έχει.



InsertKey(16)

Το 16 θα εισαχθεί στο Παιδί >= 15 και < 17 καθώς υπάρχει χώρος και το τελικό δέντρο που προκύπτει είναι το εξής :



Άσκηση 4

Έστω πως χρησιμοποιούνται οι περισσότεροι δείκτες p.

Έχουμε :

P * (Μέγεθος Δείκτη Κόμβου) + (p-1) * Μέγεθος κλειδιού αναζήτησης <= Μέγεθος σελίδας

Δηλαδή :

p * (12) + (p-1) * 8 <= 2048 => 20p – 8 <= 2048 => p <= 102.8

Άρα **p = 102** περίπου.

Γνωρίζουμε ότι ο αριθμός επιπέδων του δέντρου είναι από 0 εως 2 άρα 3.

Άρα ο μέγιστος αριθμός εγγραφών θα είναι :

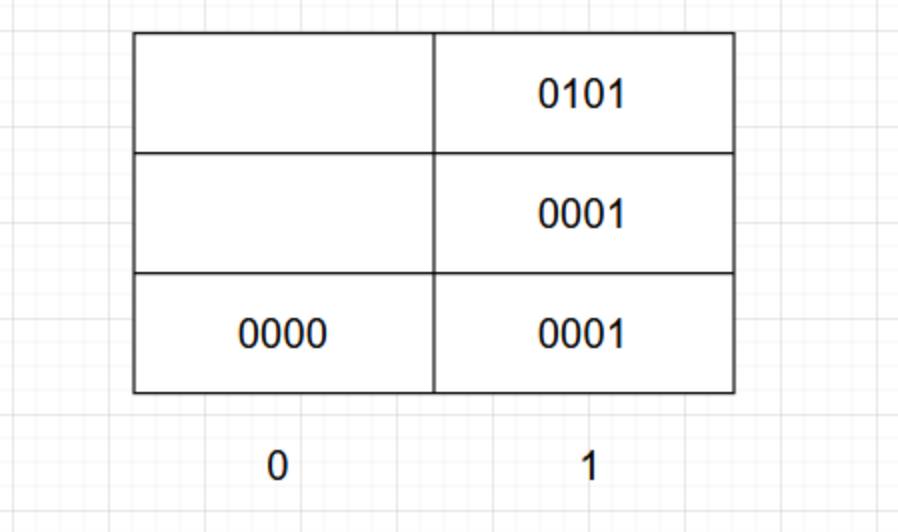
max = p^ΑριθμόςΕπιπέδων – p^(ΑριθμόςΕπιπέδων-1) =

102^3 – 102^2 = 1,061,208 – 10,404 = **1,050,804 εγγραφές**

Άσκηση 5

Εισαγωγή 0001,0001,0101

Μέχρι εδώ έχει χώρο ο κάδος:

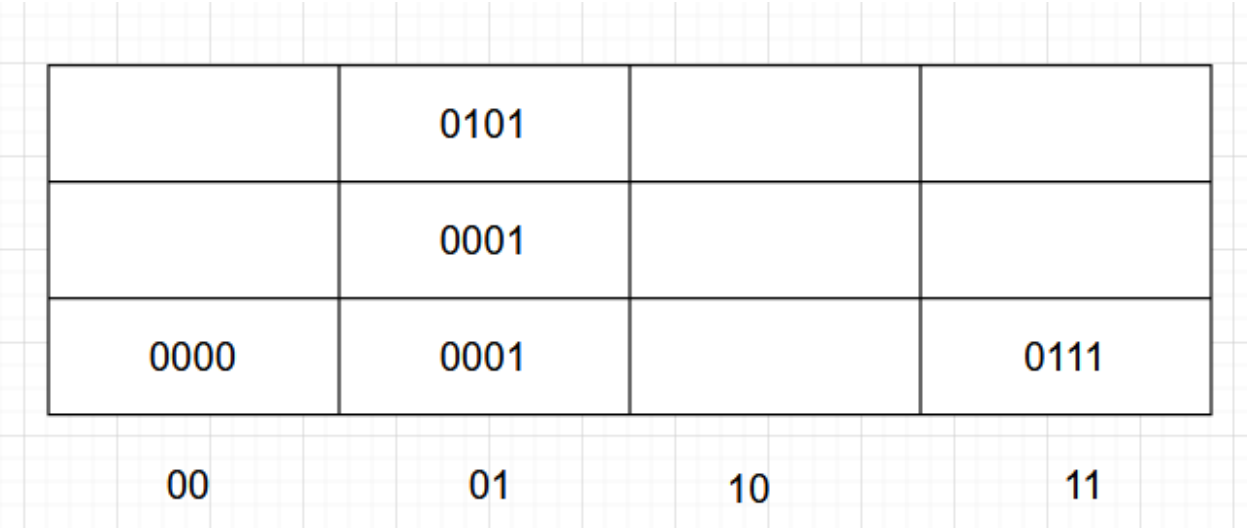


Utilization = 4/6 = 60% περίπου

<u>Εισαγωγή 0111</u>

Εδώ αυξάνεται το i σε 2 και αυξάνονται και τα buckets σε 4 (διότι έχουμε utilization 5/6 = 80%).

Ο πίνακας θα γίνει έτσι :



Utilization = 5 / 12 = 41%

<u>Εισαγωγή 0010 , 0111 , 0010 , 0011</u>

Μέχρι εδώ χωράνε τα στοιχεία και ο πίνακας γίνεται ως εξής :

	0101		0111
	0001	0010	0111
0000	0001	0010	0011
00	01	10	11

Utilization = 9/12 = 70%

<u>Εισαγωγή 0100</u>

Λόγω του παραπάνω utilization θα αλλάξει ο αριθμός των κάδων από 4 σε 5 και θα αλλάξει το i από 2 σε 3.

Ο τελικός πίνακας θα έχει την εξής μορφή :

	0101		0111	
	0001	0010	0111	
0000	0001	0010	0011	0100
000	*01	*10	*11	100

Άσκηση 6

a)

Εφόσον έχουμε d = 10 θα έχουμε 2^10 = 1024 κάδους

Το μέγεθος ενός κάδου είναι 2048 bytes και το μέγεθος μιας εγγραφής του ευρετηρίου είναι 4 bytes. Άρα κάθε κάδος περιέχει 2048/4 = 512 εγγραφές. Τέλος , αφού έχουμε 1024 κάδους θα έχουμε 512*1024 = **524288 εγγραφές** στο ευρετήριο.

b)

Εφόσον η μία εγγραφή στο ευρετήριο είναι 4 bytes το μέγεθος του ευρετηρίου θα είναι :

524288 * 4 = 2,097,152 bytes

c)

Το μέγεθος ενός κάδου είναι 2400 bytes , το μέγεθος μιας εγγραφής δεδομένων είναι 400 bytes , άρα ο μέγιστος αριθμός εγγραφών σε έναν κάδο είναι:

2400/400 = 6 εγγραφές.

Εφόσον έχουμε 1024 κάδους ο μέγιστος αριθμός εγγραφών στο αρχείο δεδομένων θα είναι :

6 * 1024 = 6,144 εγγραφές.

Άσκηση 7

Δεν συμφωνώ με την επιλογή του σχεδιαστή για τους εξής λόγους :

Αρχικά , αφού έχει βάλει τους μισθούς σε ένα Χ εύρος δεν θα μπορεί να εκτελέσει range queries (να ψάξει πχ όλους τους εργαζόμενους με μισθούς από 1500 έως 2000 ευρώ).

Επίσης, γνωρίζοντας την κατάσταση της χώρας, οι περισσότεροι Έλληνες θα μαζεύονταν στον κάδο 1 καθώς εκεί κυμαίνονται οι περισσότεροι μισθοί σήμερα (1000 με 1500 ευρώ) , το οποίο θα μας οδηγούσε στην εξής κατάσταση κακής χρήσης hashing :

Διότι θα μαζεύονταν όλες οι εγγραφές σε έναν κάδο και θα είχαμε πολλές φορές υπερχείλιση.

Για αυτό θεωρώ ότι θα ήταν καλύτερη η χρήση ενός ευρετηρίου B+ Tree.