ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ

Συστήματα Διαχείρισης Βάσεων Δεδομένων

Άσκηση 1

Δημήτρης Πλεξουσάκης

Τμήμα Επιστήμης Υπολογιστών

ΗΥ460 – Συστήματα Διαχείρισης Βάσεων Δεδομένων

Διδάσκοντες: Δημήτρης Πλεξουσάκης
1η Σειρά Ασκήσεων

Άσκηση 1 (20 μονάδες) - Αποθήκευση στο δίσκο

Θεωρήστε μαγνητικό δίσκο με τα ακόλουθα χαρακτηριστικά:

- *μέγεθος τομέα (sector) 256 bytes*
- *50 τομείς ανά τροχιά (track)*
- *2000 τροχιές ανά επιφάνεια (surface)*
- *10 διπλής όψης πλατό (platters)*
- *μέσο χρόνο αναζήτησης (seek time) 10 msec,*
- *μέγεθος μπλοκ 1024 bytes*
- *ταχύτητα περιστροφής 7200 rpm*

Υποθέστε ότι σε ένα αρχείο έχουμε 100.000 εγγραφές , 100 bytes η κάθε μία και ότι το αρχείο αυτό αποθηκεύεται στο δίσκο. Επιπλέον δεν επιτρέπεται μια εγγραφή να επεκτείνεται σε δύο μπλοκ.

***(α)* (3 μονάδα)** *Πόση είναι η χωρητικότητα σε bytes μιας τροχιάς, μιας επιφάνειας και ολόκληρου του δίσκου;*

Λύση

Η χωρητικότητα σε bytes μιας τροχιάς είναι: 50 τομείς*256 bytes/τομέα=12.800 bytes.
Η χωρητικότητα μιας επιφάνειας (surface) είναι: (12.800 bytes/τροχιά)*2000 τροχιές=25.600.000 bytes.
Η χωρητικότητα ολόκληρου του δίσκου είναι 20*25.600.000=512.000.000 bytes. (Το 20 προκύπτει από τα 10 πλατό διπλής όψεως, άρα 20 επιφάνειες).

***(β)* (3 μονάδα)** *Πόση είναι η μέγιστη και η μέση καθυστέρηση περιστροφής (rotation delay);*

Λύση

Η μέγιστη καθυστέρηση περιστροφής (maximum rotational delay) είναι: 60/7200=8,33 ms.
Η μέση καθυστέρηση περιστροφής (average rotational delay) είναι το μισό της μέγιστης θεωρώντας ότι η ελάχιστη καθυστέρηση είναι μηδενική, δηλαδή 4,165 ms.

***(γ)* (3 μονάδες)** *Υποθέτοντας ότι μια τροχιά δεδομένων μπορεί να μεταφερθεί σε μία περιστροφή, ποιος είναι ο ρυθμός μεταφοράς (transfer rate);*

Λύση

Για να υπολογίσουμε το transfer rate θα κάνουμε τα εξής:
Έχουμε maximum rotational delay= 8,33 ms, το μέγεθος μιας τροχιάς είναι 12.800 bytes.

Δεδομένου ότι μια τροχιά μπορεί να μεταφερθεί ανά περιστροφή τότε ο ρυθμός μεταφοράς είναι: transfer rate=(12.800 bytes)/8.33 ms= 1.536.614 bytes/sec.

***(δ)* (3 μονάδες)** *Υπολογίστε το blocking factor; Ποιο είναι το μέγεθος του αρχείου σε μπλοκ;*

Λύση

Στην ουσία όταν λέμε blocking factor εννοούμε πόσες εγγραφές χωρούν σε ένα block. Έχουμε
λοιπόν από την εκφώνηση 1024 bytes το μέγεθος του block, και 100 bytes το μέγεθος της κάθε εγγραφής , άρα είναι σαφές πως κάθε block χωράει 1024/100=10 εγγραφές.
Τώρα το μέγεθος ολόκληρου του αρχείου σε blocks είναι 100.000/10=10.000

***(ε)* (4 μονάδες)** *Αν οι σελίδες του αρχείου αποθηκεύονται σειριακά (sequentially) στο δίσκο, με τη σελίδα 1 στο μπλοκ 1 της τροχιάς 1 μιας επιφάνειας, ποια σελίδα αποθηκεύεται στο μπλοκ 1 της τροχιάς 1 της επόμενης επιφάνειας;*

Λύση

Όλη η τροχιά είναι 12.800 bytes με το κάθε block να έχει μέγεθος 1024 bytes, συνεπώς κάθε τροχιά έχει 12.800/1024= 12,5 blocks. Εμείς κρατάμε το ακέραιο μέρος δηλαδή 12 blocks ανά τροχιά. Εφόσον γράφουμε κατά κύλινδρο η επόμενη εγγραφή θα είναι στην 13 σελίδα.

***(στ)* (4 μονάδες)** *Πόσος χρόνος απαιτείται για τη σειριακή ανάγνωση ολόκληρου του αρχείου; Πόσος χρόνος απαιτείται για την ανάγνωση του αρχείου με τυχαία σειρά;*

Λύση

Για την σειριακή ανάγνωση ολόκληρου του αρχείου θα χρειαστεί χρόνος για τις τροχιές και για τους κυλίνδρους. Δηλαδή έχουμε 100.000 εγγραφές από 100 bytes η μια σύνολο 10.000.000 bytes, κάθε τροχιά χωράει 12.800 bytes. Άρα συνολικά και ακριβώς(όχι προσεγγιστικά) , θέλουμε 781, 25 τροχιές. Έχουμε transfer rate= 8,33 ms, συνεπώς ο χρόνος για τις τροχιές είναι: 781,25*8,33=6,5 sec. Τώρα θα πρέπει να προσθέσουμε και τον χρόνο για τους κυλίνδρους(που είναι 40 διότι ο κάθε κύλινδρος έχει 20 τροχιές και συνεπώς για 781 τροχιές είναι 40 κύλινδροι), 40*10 ms=400 ms=0,4 sec.
(10 ms είναι το seek time).
Συνεπώς συνολικά για τη σειριακή ανάγνωση ολόκληρου του αρχείου απαιτείται χρόνος 6,5+0,4= 6,9 sec.

Για τη περίπτωση του χρόνου ανάγνωσης με τυχαία σειρά, τα πράγματα είναι διαφορετικά.
Ο χρόνος για ένα block είναι seek_time + average_rotational_delay + transfer time= 10ms+4,156 ms+0,66 ms=15 ms. Το transfer time είναι ο χρόνος μεταφοράς ενός block και είναι ίσος με το χρόνο πλήρους περιστροφής δια τον αριθμό των block ανα τροχιά δηλ 8,33 / 12,5=0,66 ms
Στο δ ερώτημα είδαμε ότι όλο το αρχείο είναι 10.000 blocks, συνεπώς ο χρόνος που απαιτείται για την ανάγνωση ολόκληρου του αρχείου με τυχαία σειρά είναι: 10.000*15 ms=150 sec.

Άσκηση 2 (40 μονάδες) – Ευρετήρια και B+ Δέντρα

Θεωρείστε μια σχέση Students που έχει 500.000 πλειάδες και είναι αποθηκευμένη σε ένα αρχείο σωρού. Κατασκευάζουμε ένα ευρετήριο για ένα πεδίο StdType της σχέσης Students που δεν είναι κλειδί για τη σχέση. Έστω ότι υπάρχουν 1.000 διαφορετικές τιμές του StdType. Το μέγεθος του πεδίου StdType είναι 8 bytes και των δεικτών 16 bytes (όλων των ειδών δεικτών). Θεωρείστε μέγεθος block 2048 bytes.

Έστω ότι κατασκευάζουμε ένα ευρετήριο με ενδιάμεσο επίπεδο.

***(α)* (10 μονάδες)** *Έστω ότι το ευρετήριο είναι ένα δευτερεύον ευρετήριο ενός επιπέδου*

i. *Δώστε το συνολικό μέγεθος του αρχείου ευρετηρίου (συμπεριλαμβανομένου και του ενδιάμεσου επιπέδου) σε αριθμό blocks.*
ii. *Υπολογίστε το κόστος (σε αριθμό blocks) της αναζήτησης StdType = α, όπου α μια τιμή από το πεδίο ορισμού του StdType. Υποθέστε ότι η αναζήτηση ικανοποιείται από το 0.1% των εγγραφών της σχέσης.*
iii. *Έστω ότι εισάγουμε μια νέα πλειάδα στη σχέση Students. Εξηγείστε πως αλλάζει το ευρετήριο, όταν η εγγραφή έχει μια τιμή στο StdType η οποία*
    a. *δεν εμφανίζεται ήδη σε άλλη πλειάδα και*
    b. *εμφανίζεται ήδη σε άλλη πλειάδα.*

    Δώστε μια εκτίμηση του κόστους αυτών των αλλαγών (σε αριθμό blocks) για κάθε μία από τις δύο περιπτώσεις.

Λύση

Το αρχείο ευρετηρίου θα έχει εγγραφές τύπου <StdType_k, rid > .

Το αρχείο ευρετηρίου θα έχει μια εγγραφή για κάθε διαφορετική τιμή του πεδίου StdType. Από την εκφώνηση ξέρουμε ότι το StdType έχει 1000 διαφορετικές τιμές, άρα συνολικά θα έχουμε 1000 εγγραφές στο ευρετήριο.

Κάθε rid θα δείχνει σε ένα block από δείκτες εγγραφών στο ενδιάμεσο επίπεδο. Αυτό το block από δείκτες εγγραφών θα δείχνει σε μια εγγραφή του αρχείου σωρού με τιμή στο πεδίο ευρετηριοποίησης ίση με StdType_k. Οι δείκτες εγγραφών με ίδιο κλειδί αναζήτησης θα βρίσκονται στο ίδιο block.

i. Ξέρουμε ότι το μέγεθος κάθε εγγραφής του index μας θα είναι ίσο με 8 bytes (μεγεθος StdType) + 16 bytes (μέγεθος rid) = 24 byte.

   Ένα block χωράει 2048 bytes. Άρα σε ένα block χωράνε 2048/24=85,3 δηλαδή 85 εγγραφές

   Έχουμε 1000 εγγραφές στο ευρετήριο, μια για κάθε διαφορετική τιμή του StdType. Άρα, για το ευρετήριο χρειαζόμαστε 1000/85=11,76 δηλαδή 12 block

   Επιπλέον θεωρούμε ότι ο αριθμός των εγγραφών για κάθε διαφορετική τιμή του StdType είναι ίσος. Δηλαδή, για κάθε διαφορετική τιμή θα έχουμε 500.000/1.000 = 500 εγγραφές. Αυτό σημαίνει ότι στο ενδιάμεσο επίπεδο θα πρέπει να έχουμε 500 δείκτες προς τις εγγραφές στο αρχείο σωρού. Άρα, πρέπει να βρούμε πόσους δείκτες των 16 bytes χωράει κάθε block. Αυτό είναι 2048/16=128 δείκτες. Άρα, για

κάθε διαφορετική τιμή του StdType που έχει 500 εγγραφές θα χρειαζόμαστε 500/128=3,9 δηλαδή 4 block. Οπότε συνολικά για το ενδιάμεσο επίπεδο θα θέλουμε 4 block για κάθε διαφορετική τιμή, δηλαδή 4.000 blocks.

Άρα Μέγεθος αρχείου = Μέγεθος ευρετηρίου + Μέγεθος ενδιάμεσου επιπέδου= 12 + 4000=4012 blocks.

ii. Θέλουμε να αναζητήσουμε το StdType = a στο ευρετήριο. Η αναζήτηση ικανοποιείται από το 0.1% των εγγραφών της σχέσης δηλαδή: 500.000 * 0.1% = 500 εγγραφές (από το πρώτο ερώτημα έχουμε ήδη πει ότι κάθε StdType θα έχει 500 εγγραφές αφού ο αριθμός για κάθε διαφορετική τιμή του StdType είναι ίσος, δηλαδή είναι ομοιόμορφα κατανεμημένες οι εγγραφές.)

Άρα έχουμε ότι το StdType θα είναι ίσο με α σε 500 εγγραφές.

Αρχικά πρέπει να αναζητήσουμε το StdType στο ευρετήριο. Ξέρουμε ότι το ευρετήριο είναι ταξινομημένο ως προς το πεδίο του ευρετηριασμού για να αναζητούμε πιο γρήγορα. Κατά μέσο όρο θα πρέπει να ψάξουμε τα μισά μποκ για να βρούμε το α δηλαδή περίπου 6. (Σημείωση: θα μπορούσαμε να χρησιμοποιήσουμε και δυαδική αναζήτηση με κόστος log12=4 blocks)

Για το ενδιάμεσο επίπεδο εφόσον έχουμε φτάσει εκεί με δείκτη, που το StdType του είναι ίσο με α, τότε όλοι οι δείκτες που οδηγούν σε εγγραφές στο heap file θα έχουν τιμή α. Άρα, θα προσπελάσουμε 4 block που περιέχουν τους δείκτες προς τις εγγραφές με StdType = α.

Άρα το κόστος αναζήτησης του ευρετηρίου θα είναι ίσο με 6+ 4 = 10 blocks.

Όμως, βρήκαμε παραπάνω πως για κάθε α, υπάρχουν 500 εγγραφές. Θα πρέπει επομένως να αναζητήσουμε τις 500 αυτές εγγραφές, που δεν ξέρουμε όμως σε ποια block βρίσκονται. Έχουμε αρχείο σωρού, το οποίο είναι unordered. Αν υποθέσουμε οτι βρίσκονται σε διαφορετικά Blocks, θα διαβαστούν 500 blocks.

Σύνολο 510 blocks

iii. α) Εισάγουμε μια νέα πλειάδα στη σχέση Students, της οποία η τιμή του StdType δεν εμφανίζεται ήδη σε άλλη πλειάδα. Αυτό σημαίνει πως δεν υπάρχει ήδη καταχώρηση για αυτό το StdType στο Ευρετήριο. Συνεπώς, θα πρέπει να καταχωρήσουμε μια εγγραφή με το κλειδί αναζήτησης της νέας πλειάδας στο κατάλληλο block, αφού το ευρετήριο είναι ταξινομημένο. Οπότε αρχικά θα πρέπει να αναζητήσουμε το block που θα κάνουμε την καταχώρηση. Κατά μέσο όρο θα προσπελάσουμε 6 block.

Επίσης, πρέπει να τοποθετήσουμε στη σωστή θέση (σύμφωνα με την ταξινόμηση) την εγγραφή μέσα στο ευρετήριο, οπότε ενδεχομένως θα πρέπει να μετακινήσουμε εγγραφές προς τα κάτω για να προσθέσουμε τη νέα, πράγμα που ίσως μας κοστίσει σε I/O. (1 write για την νέα εγγραφή και 1 read & 1write για κάθε εγγραφή κάτω από την νέα, δηλαδή 6+6 I/Os κατά μέσο όρο.

Τέλος, θα πρέπει να προσθέσουμε έναν δείκτη στο ενδιάμεσο επίπεδο που θα δείχνει στην τελευταία καταχώρηση του heap file. Άρα, εφόσον η τιμή δεν εμφανίζεται ήδη σε άλλη πλειάδα θα πρέπει να δεσμεύσουμε ένα νέο block για τον δείκτη.

Συνεπώς, μια εκτίμηση του κόστους αυτών των αλλαγών στο ευρετήριο θα είναι περίπου: 6 blocks (αναζήτηση κατάλληλου μπλοκ στο ευρετήριο) + 12 I/O (ολίσθηση εγγραφών προς τα κάτω) + 1 (εγγραφή στο δεσμευμένο block δεικτών του ενδιάμεσου επιπέδου).

b) Εισάγουμε μια νέα πλειάδα στη σχέση Students, της οποίας η τιμή στο StdType εμφανίζεται ήδη σε άλλη πλειάδα. Αυτό σημαίνει ότι υπάρχει ήδη καταχώρηση για αυτό το StdType στο Ευρετήριο. Άρα θα αναζητήσουμε κατά μέσο όρο 4 blocks για να βρούμε την εγγραφή με δυαδική αναζήτηση. Στη συνέχεια θα πρέπει να διαβάσουμε τα 4 block του ενδιάμεσου επιπέδο, ώστε στο 4ο να εισάγουμε το νέο δείκτη προς το heap file.

Συνεπώς, μια εκτίμηση του κόστους αυτών των αλλαγών στο ευρετήριο θα είναι περίπου: 4 blocks (αναζήτηση του κατάλληλου μπλοκ στο ευρετήριο) + 4 blocks (ανάγνωση των 4 block) +1 (εγγραφή) = 11 block.

***(β)* (10 μονάδες)** *Υποθέστε τώρα ότι κατασκευάζουμε ένα B+-δέντρο (οι δείκτες τιμών δείχνουν στο ενδιάμεσο επίπεδο). Επαναλάβατε το υπο-ερωτήματα (i)-(iii) του ερωτήματος (α) θεωρώντας ότι το δέντρο είναι όσο το δυνατόν πιο γεμάτο.*

Λύση

I. Αν υποθέσουμε ότι σε κάθε κόμβο του B+δέντρου υπάρχουν n τιμές κλειδιών και n+1 δείκτες. Τότε θα πρέπει να ισχύει (n x 8) + ((n + 1) x 16) ≤ 2048. Δηλαδή, n=84.

   Κάθε κόμβος του δέντρου χωράει 84 τιμές και 85 δείκτες.

   Το StdType έχει 1000 διαφορετικές τιμές και θέλουμε το δέντρο να είναι όσο το δυνατόν πιο γεμάτο, συνεπώς θα έχουμε 1000/85 = 12 κόμβους φύλλα (γεμάτα). Εφόσον ένας κόμβος μπορεί να έχει 84 τιμές κλειδιού και 85 δείκτες, ένα root node είναι αρκετό για να δείχνει στα 12 φύλλα.

   Άρα το Ευρετήριο χρειάζεται 13 block.

Για το ενδιάμεσο επίπεδο ομοίως με το α θα θέλουμε 4 block για κάθε διαφορετική τιμή, δηλαδή 4.000 blocks.

Σύνολο 4013 blocks

II. Εφόσον έχουμε B+- δέντρο για την αναζήτηση του StdType θα διαβαστούν στο Ευρετήριο 2 blocks. Η ρίζα και το ζητούμενο φύλλο. Στο ενδιάμεσο επίπεδο θα διαβαστούν τα 4 block που περιέχουν δείκτες προς το heap file με εγγραφές StdType = α. Άρα το κόστος αναζήτησης του ευρετηρίου για StdType = α, θα είναι 2+ 4 = 6 block.

Επειδή στα 4 block έχουμε 500 εγγραφές με StdType = α, τότε θα κάνουμε 500 αναγνώσεις + τις 6 αναγνώσεις για την αναζήτηση. Άρα, θα έχουμε 506 I/O.

III. α)Εισάγουμε μια νέα πλειάδα στη σχέση Students, της οποία η τιμή του StdType δεν εμφανίζεται ήδη σε άλλη πλειάδα. Αυτό σημαίνει πως δεν υπάρχει ήδη καταχώρηση για αυτό το StdType στο B+-δέντρο (σε κάποιο φύλλο) . Άρα αρχικά θα πρέπει να καταχωρήσουμε την τιμή του κλειδιού αναζήτησης σε κάποιο φύλλο. Έχουμε 12 φύλλα με 84 τιμές κλειδιού το καθένα. Άρα, στα φύλλα χωράνε συνολικά 1008 τιμές κλειδιού. Έχουμε καταχωρημένα ήδη τα 1000 άρα θα υπάρχουν κάποια κενά (8).Έχουμε δύο περιπτώσεις:

η νέα τιμή να πέφτει σε φύλλο με κενό

Αρχικά θα κάνουμε αναζήτηση για να βρούμε το κατάλληλο φύλλο για να γίνει η εισαγωγή. Άρα, 2 block. Θα γίνει 1 write για να προσθέσουμε την εγγραφή στο φύλλο. Η τιμή δεν υπάρχει στο ενδιάμεσο επίπεδο άρα θα πρέπει να δεσμευτεί ένα block, όπου θα προσθέσουμε τον δείκτη (1 write) . Άρα το κόστος ενημέρωσης του B+-δέντρου θα είναι 4 I/O. (ή 2 blocks + προσπέλαση και εγγραφή στο νέο block του επιπλέον επιπέδου)

η νέα τιμή να πέφτει σε γεμάτο φύλλο.

Αρχικά θα κάνουμε αναζήτηση για να βρούμε το φύλλο στο οποίο θα έπρεπε να μπει η νέα εισαγωγή. Άρα, 2 block. Αναγκαστικά, αφού δε χωράει το κλειδί αναζήτησης θα πρέπει να διασπαστεί το φύλλο. Συνεπως, θα πρέπει να δημιουργήσουμε ένα νέο φύλλο, να το ενημερώσουμε και να ενημερώσουμε την ρίζα για το νέο φύλλο (3 write). Όπως και πριν, η τιμή δε θα υπάρχει στο ενδιάμεσο επίπεδο άρα θα πρέπει να δεσμευτεί ένα νέο block όπου θα εγγραφεί ο νέος δείκτης προς το heap file. Άρα το κόστος ενημέρωσης του B+-δέντρου σε αυτή τη περίπτωση θα είναι 6 I/O. (ή 2 block + 1 block για το νέο φύλλο + ανάγνωση και διάσπαση του φύλλου που θα έπρεπε να μπει το κλειδί αν δεν ήταν γεμάτο + 1 block στο ενδιάμεσο επίπεδο = 5 block)

β) Εισάγουμε μια νέα πλειάδα στη σχέση Students, της οποίας η τιμή στο StdType εμφανίζεται ήδη σε άλλη πλειάδα. Αυτό σημαίνει ότι υπάρχει ήδη καταχώρηση για αυτό το StdType στο Ευρετήριο. Άρα θα αναζητήσουμε κατά μέσο όρο 2 blocks για να βρούμε την εγγραφή στο B+-δέντρο. Στη συνέχεια θα πρέπει να διαβάσουμε τα 4 block του ενδιάμεσου επιπέδο, ώστε στο $4_{ο}$ να εισάγουμε το νέο δείκτη προς το heap file.

Συνεπώς, μια εκτίμηση του κόστους αυτών των αλλαγών στο ευρετήριο θα είναι περίπου: 2 blocks (αναζήτηση του κατάλληλου μπλοκ στο ευρετήριο) + 4 blocks (ανάγνωση των 4 block) +1 (εγγραφή) = 7 block.

***(γ)* (10 μονάδες)** *Υποθέστε τώρα ότι κατασκευάζουμε ένα πυκνό δευτερεύον ευρετήριο (δηλαδή, δεν υπάρχει ενδιάμεσο επίπεδο). Επαναλάβατε το υπο-ερωτήματα (i) και (ii) του ερωτήματος (α).*

Λύση

I. Σε αυτό το ερώτημα δεν έχουμε ενδιάμεσο επίπεδο για τα ίδια κλειδιά αναζήτησης. Συνεπώς το αρχείο ευρετηρίου θα έχει εγγραφές <StdType, rid> για κάθε μια από τις πλειάδες της σχέσης Student. Το StdType δεν είναι κλειδί, συνεπώς δεν είναι μοναδικό για κάθε εγγραφή. Οπότε θα έχουμε εγγραφές στο ευρετήριο με την ίδια τιμή στο κλειδί αναζήτησης, όσες είναι και οι εγγραφές της σχέσης μας με το συγκεκριμένο StdType.

   Έχουμε index της μορφής <StdType, rid> άρα το μέγεθος της κάθε εγγραφής στο ευρετήριο θα είναι 8bytes (τιμή κλειδιού αναζήτησης) + 16 bytes (pointer προς το heap file) = 24 byte.

   Ένα block χωράει 2048 bytes. Άρα σε ένα block χωράνε 2048/24 = 85 εγγραφές

   Το ευρετήριο θα έχει 500.000 εγγραφές αφού δεν υπάρχει το ενδιάμεσο επίπεδο όποτε θα χρειαστούμε 500.000/85= 5883 blocks

   Άρα το πυκνό ευρετήριο θα έχει μέγεθος 5883 blocks.

II. Θέλουμε να αναζητήσουμε το StdType = a στο πυκνό ευρετήριο. Η αναζήτηση ικανοποιείται από το 0.1% των εγγραφών της σχέσης δηλαδή: 500.000 * 0.1% = 500 εγγραφές. Έχουμε 500 εγγραφές και κάθε μπλοκ χωράει 85 εγγραφές. Άρα, το α θα υπάρχει σε περίπου 6 block. εγγραφές του α.

   Θέλουμε λοιπόν με binary search log(5883) = 13 block για να βρούμε το πρώτο block + 5 για να αναγνώσουμε τα υπόλοιπα block.

   Το κόστος σε I/O θα στη συνέχεια 500 αναγνώσεις

   Σύνολο 13+5+500= 518 IOs

***(δ)* (10 μονάδες)** *Υποθέστε μια αναζήτηση StdType > α, όπου α μια τιμή από το πεδίο ορισμού του StdType, η οποία ικανοποιείται από το 10% των εγγραφών της σχέσης. Εξηγείστε πως μπορείτε να χρησιμοποιείστε το ευρετήρια των ερωτημάτων (α)-(γ) για αυτήν την αναζήτηση και δώστε μια εκτίμηση του κόστους της σε κάθε περίπτωση.*

Λύση

Έχουμε ότι η StdType > α ικανοποιείται από το 10% των εγγραφών της σχέσης. Άρα 500.000 * 10% = 50.000 εγγραφές.
Αν θεωρήσουμε ότι ο αριθμός εγγραφών για κάθε διαφορετική τιμή του StdType είναι ίσος, τότε οι 50.000 εγγραφές θα έχουν 100 διαφορετικές τιμές για StdType. (100*500 = 50.000 εγγραφές)

Ευρετήριο με ενδιάμεσο επίπεδο
Αρχικά με Binary Search θα φτάσουμε στο πεδίο με τιμή α διαβάζοντας log12=4 blocks. Στη συνέχεια πρέπει να διαβάσουμε όλα τα blocks μέχρι το τέλος του ευρετηρίου δηλαδή ακόμα 1 block (ένα μπλόκ χωράει 85 εγγραφές).
Για κάθε εγγραφή μεγαλύτερη του α που υπάρχουν σε αυτά τα 2 block θα πρέπει να διαβάσουμε 4 block του ενδιάμεσου επιπέδου που τους αντιστοιχούν. Άρα συνολικά θα διαβαστούν 4*100 = 400 block στο ενδιάμεσο επίπεδο.

Άρα συνολικά το κόστος αυτής της αναζήτησης θα είναι 5 block στο index + 400 block στο επιπλέον επίπεδο = 405 block.

Έχουμε 50.000 εγγραφές της σχέσης με StdType > a, άρα θα έχουμε 50.000 αναγνώσεις εγγραφών + 405 αναγνώσεις block = 50.405 I/O.
Ευρετήριο με B+-δέντρο
Αρχικά θα αναζητήσουμε το block με την εγγγραφή α. Εφόσον έχουμε B+-δέντρο. Άρα θα διαβάσουμε την ρίζα και το φύλλο του B+-δέντρου που αντιστοιχεί στη τιμή κλειδιού αναζήτησης α.

Είχαμε βρει στο ερώτημα β ότι κάθε κόμβος είναι ένα block που χωράει 84 τιμές κλειδιού και 85 δείκτες. Οπότε, οι 100 επόμενες καταχωρήσεις για StdType > α θα βρίσκεται στο φύλλο που διαβάσαμε κατεβαίνοντας στο δέντρο και στο δεξί γειτονικό του φύλλο (ένα next στη διπλά συνδεδεμένη λίστα). Οι δείκτες των φύλλων θα μας οδηγήσουν στα blocks των ενδιάμεσο επίπεδο.

Άρα συνολικά θα διαβαστούν 4*100 = 400 block στο ενδιάμεσο επίπεδο.

Άρα το κόστος αναζήτησης στο B+-δέντρο θα είναι 2 block (αναζήτηση στο B+-δέντρο) + 1 block (δεξί γειτονικό φύλλο) + 400 block (ενδιάμεσο επίπεδο) = 403 block.

Έχουμε 50.000 εγγραφές της σχέσης με StdType > a, άρα θα έχουμε 50.000 αναγνώσεις εγγραφών + 403 αναγνώσεις block = 50.403 I/O.
Πυκνό Ευρετήριο
Από το γ ερώτημα έχουμε ότι το πυκνό ευρετήριο έχει 5883 block και χωράει 85 εγγραφές κάθε block.

Αρχικά θα ψάξουμε να βρούμε τις τιμές που είναι ίσες με α με δυαδική αναζήτηση. Θα χρειαστούμε log 5883 = 13 blocks.

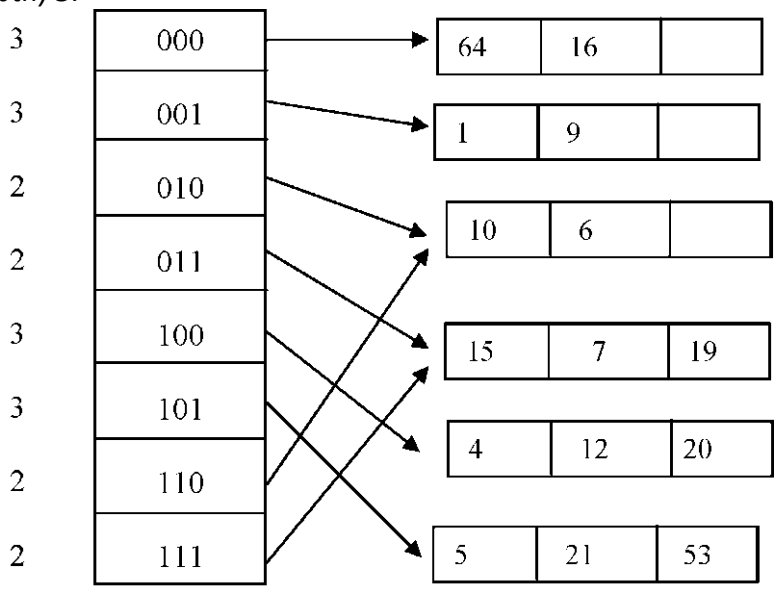
Επίσης, ξέρουμε ότι υπάρχουν 500 εγγραφές με A = α και ότι αυτές αντιστοιχούν σε 6 block(500 / 85 = 6).

Μόλις φτάσουμε εκεί θα διαβάσουμε τις επόμενες τιμές μέχρι το τέλος. Αυτές θα είναι σε 50.000/85=589 blocks

Στη συνέχεια, έχουμε 50.000 εγγραφές της σχέσης με StdType > a, άρα θα έχουμε 50.000 αναγνώσεις εγγραφών + 589 αναγνώσεις block +13 +6 - 1= 50.607 I/O.

Άσκηση 3 (20 μονάδες) – Ευρετήρια Επεκτατού Κατακερματισμού

Θεωρείστε το παρακάτω ευρετήριο επεκτατού κατακερματισμό (extensible hash) με ολικό βάθος (global depth) 3.



***(α)* (4 μονάδες)** *Ποιο είναι το μεγαλύτερο πλήθος τιμών που μπορεί να εισαχθούν σε αυτό χωρίς να χρειαστεί να αυξηθεί το ολικό βάθος του καταλόγου; Εξηγείστε που πρέπει να πάνε αυτές οι τιμές.*

Λύση

Για να βρούμε το μεγαλύτερο πλήθος τιμών που μπορεί να εισαχθούν χωρίς να αυξηθεί το ολικό βάθος του καταλόγου, θα κοιτάξουμε κάθε μια θέση του καταλόγου και θα δούμε πόσα στοιχεία χρειάζεται ο κάδος ώστε να γεμίσει.

Θέση καταλόγου (00)

000 -> χώρος για 1 καταχώριση δεδομένων

100 -> γεμάτος

Θέση Καταλόγου (01)

001 -> χώρος για 1 καταχώριση δεδομένων

101 -> γεμάτος

Θέση Καταλόγου (10)

010 & 110 δείχνουν στον ίδιο κάδο άρα δεν είχε διασπαστεί ο συγκεκριμένος κάδος

010 & 110 -> 4 καταχωρίσεις δεδομένων

Θέση Καταλόγου (11)

011 & 111 δείχνουν στον ίδιο κάδο άρα δεν είχε διασπαστεί ο συγκεκριμένος κάδος

011 & 111 -> 3 καταχωρίσεις δεδομένων

Μέγιστο Πλήθος = 1+1 +4+ 3 = **9** καταχωρίσεις δεδομένων

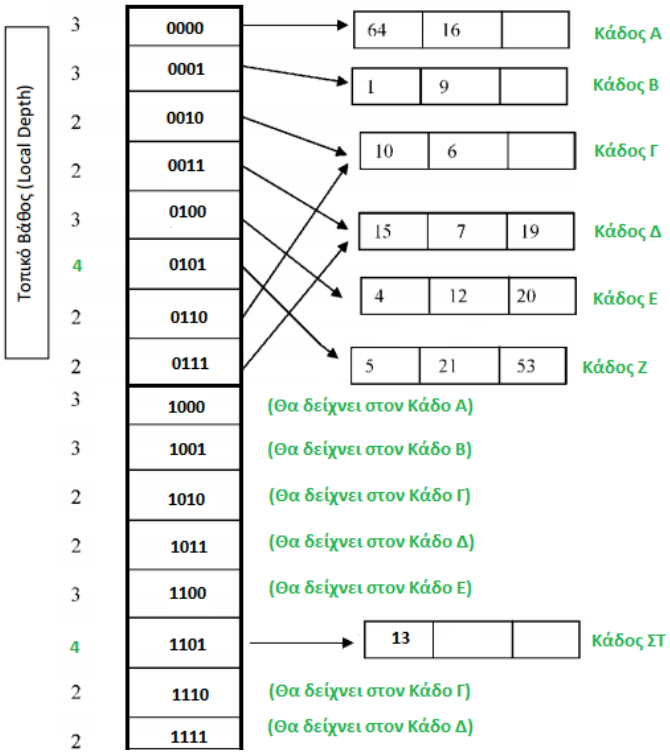
***(β)* (4 μονάδες)** *Ποιο είναι το ελάχιστο πλήθος τιμών που μπορεί να εισαχθούν σε αυτό ώστε να χρειαστεί να αυξηθεί το ολικό βάθος του καταλόγου κατά 1; Δώστε ένα τέτοιο παράδειγμα.*

Λύση

Το μόνο που χρειάζεται για να αυξηθεί το ολικό βάθος από 3 σε 4, είναι να προσπαθήσουμε να κάνουμε εισαγωγή μιας τιμής σε ένα bucket directory το οποίο είναι ήδη γεμάτο και ταυτόχρονα το local depth του είναι ίσο με το global depth, που σημαίνει ότι δεν μπορεί να γίνει split. Άρα η απάντηση για το συγκεκριμένο extensible hash είναι **1**.

Ας δούμε το παράδειγμα.

Εισάγουμε την τιμή 13 (1101) η οποία σύμφωνα με τα διάρθρωση του συστήματος πρέπει να γίνει εισαγωγή στο bucket directory με κλειδί το 101. Το local depth είναι 3 όσο και το global depth, άρα για να γίνει η εισαγωγή το global depth γίνεται 4 αυξάνοντας των αριθμό των ψηφίων σε 4. Το νέο σχήμα που θα προκύψει είναι



(γ) **(4 μονάδες)** *Ποιο είναι το ελάχιστο πλήθος τιμών που μπορεί να εισαχθούν σε αυτό ώστε να χρειαστεί να αυξηθεί το ολικό βάθος του καταλόγου κατά 2; Δώστε ένα τέτοιο παράδειγμα.*

Λύση

Συνεχίζουμε από το παράδειγμα που δόθηκε στο β. Θα εισάγουμε στο Ευρετήριο την τιμή 37. Αυτή έχει δυαδική αναπαράσταση 0101. Παρατηρούμε, ότι στον κατάλογο 0101, ο Κάδος Ζ που πρέπει να μπει αυτή η τιμή είναι πλήρης. Άρα, πρέπει ο κάδος να διασπαστεί με την διαδικασία που περιγράψαμε στο Β ερώτημα. Συνεπώς, το ελάχιστο πλήθος τιμών που πρέπει να εισαχθούν ώστε το ολικό βάθος του καταλόγου να αυξηθεί κατά 2 είναι **2** καταχωρίσεις δεδομένων.

(δ) **(4 μονάδες)** *Ποιο είναι το ελάχιστο πλήθος τιμών που η διαγραφή τους μπορεί να οδηγήσει σε μείωση κατά 1 του ολικού βάθους του καταλόγου της αρχικής κατάστασης του ευρετηρίου. Δώστε ένα τέτοιο παράδειγμα.*

Λύση

Για να μειώσουμε το ολικό βάθος του καταλόγου της αρχικής κατάστασης του ευρετηρίου, αρκεί να διαγράψουμε στοιχεία από τους διασπασμένους κάδους. Αυτό σημαίνει, ότι αν αδειάσουν οι εξής κάδοι θα μειώσουμε τους καταλόγους σε 4. Συνεπώς, το ελάχιστο πλήθος τιμών που η διαγραφή τους μπορεί να οδηγήσει σε μείωση κατά 1 του ολικού βάθους είναι **4** διαγραφές.

***(ε)* (4 μονάδες)** *Θεωρείστε ότι σε κάποια στιγμή το ευρετήριο μεγαλώνει και φτάνει τους 200 κάδους. Ποιο είναι το μικρότερο δυνατό ολικό βάθος σε αυτήν την περίπτωση; Ποιο είναι το μεγαλύτερο δυνατό ολικό βάθος σε αυτήν την περίπτωση;*

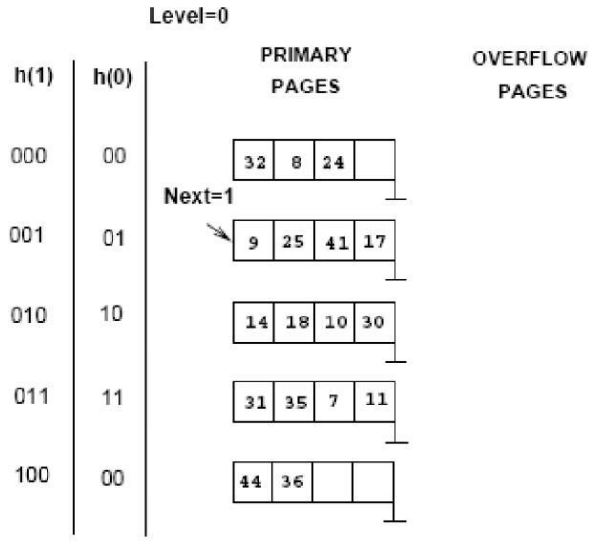
Λύση

Όσο διασπούνται κάδοι το ευρετήριο διπλασιάζεται συνεχώς. Αυτό σημαίνει ότι χρησιμοποιούνται περισσότερα bits για να διακρίνουμε 2 κάδους.

Με 8 bits μπορούμε να περιγράψουμε 256 διαφορετικά αντικείμενα. Συνεπώς, για μια κανονική χρήση του ευρετηρίου 200 κάδοι θα μπορούσαν να αναπαρασταθούν με 8 bit με κάποιους δείκτες να δείχνουν σε κοινούς κάδους. Άρα το ελάχιστο ολικό βάθος θα είναι 8.
Έχουμε ήδη 6 buckets, θα πρέπει κάθε φορά που δημιουργείται και νέο bucket να αυξάνει το ολικό βάθος. Άρα αφού θα προσθέσουμε 194 ακόμα buckets και το τρέχον βάθος είναι 3 το μέγιστο ολικό βάθος που μπορούμε να φτάσουμε είναι 197.

Άσκηση 4 (20 μονάδες) – Ευρετήρια Γραμμικού Κατακερματισμού

Έστω το παρακάτω ευρετήριο γραμμικού κατακερματισμού (linear hash).



Ας υποθέσουμε ότι γίνεται split όποτε δημιουργείτε μια σελίδα υπερχείλισης. Επιπλέον υποθέστε ότι όλες οι μελλοντικές εισαγωγές στο ευρετήριο έχουν τιμές κατακερματισμού που τελειώνουν σε "11". Απαντήστε στις παρακάτω ερωτήσεις για αυτό το ευρετήριο:
***(α)* (5 μονάδες)** *Ποιος είναι ο ελάχιστος αριθμός εισαγωγών που μπορεί να προκαλέσει στον αντίστοιχο κάδο (bucket) τη δημιουργία δύο σελίδων υπερχείλισης. Πόσα splits προκάλεσαν αυτές οι εισαγωγές;*

Λύση

Αρχικά ο δείκτης Next κοιτάζει το δεύτερο bucket. Μόλις έρθει η πρώτη τιμή που θα εισαχθεί στο bucket με τιμή hash 11 τότε θα γίνει overflow και θα δημιουργηθεί ένα pointer bucket που θα κρατάει προσωρινά την καινούργια τιμή. Η εκφώνηση δηλώνει ότι η διαδικασία του split εξαρτάται από την εμφάνιση overflow, άρα θα γίνει split στο bucket που δείχνει η μεταβλητή Next. Μετά το Next θα αυξηθεί και θα γίνει 2 δείχνοντας στο επόμενο bucket. Με μια δεύτερη εισαγωγή στην ίδια θέση με πριν, δεν θα γίνει δεύτερο overflow (δεύτερος δείκτης) γιατί το overflow page του bucket έχει άλλες 3 κενές θέσεις (μέγεθος όσο και το bucket). Με άλλες 3 εισαγωγές (σύνολο 5) θα γίνει νέο overflow 2^{ης} σελίδας στο bucket 11 και split στην θέση που δείχνει η μεταβλητή Next . Το Next θα αυξηθεί και τώρα θα δείχνει στο 11 όπου είναι και τα 2 overflow. Άρα σε αυτό το σημείο έχουμε συνολικά 7 buckets και 2 overflow pages σε αλυσίδα, τα οποία προκλήθηκαν από μόλις **5 εισαγωγές** και είχαν ως αποτέλεσμα **2 splits**.

(β) **(5 μονάδες)** *Βρείτε μια λίστα από επιπλέον καταχωρήσεις που η εισαγωγή τους μπορεί να προκαλέσει ένα split έτσι ώστε να μειωθεί το μήκος της προηγούμενης αλυσίδας υπερχείλισης. Χρησιμοποιήστε όσο το δυνατό λιγότερες καταχωρήσεις ευρετηρίου.*

Λύση

Αρχικά για να γίνει το νέο split τώρα πρέπει η δεύτερη overflow page να γεμίσει. Άρα χρειαζόμαστε τουλάχιστον **3** καταχωρήσεις επιπρόσθετες (σύνολο 8). Λαμβάνοντας υπόψη τις δυαδικές τιμές των αριθμών στο bucket παρατηρούμε ότι το 7 και το 31 θα μετακινηθούν στο νέο split bucket που θα δημιουργηθεί, ανοίγοντας έτσι 2 θέσεις διαθέσιμες. Για να μειωθεί η λίστα overflow αλυσίδα πρέπει να υπάρχουν άλλες 2 τουλάχιστον καταχωρήσεις οι οποίες να έχουν στην δυαδική τους μορφή τα 3 τελευταία bit να ισοδυναμούν με 111. Άρα δεχόμαστε οποιαδήποτε λίστα με καταχωρήσεις που υπακούει σε αυτόν τον κανόνα.

(γ) **(10 μονάδες)** *Ποιος είναι ο μέγιστος αριθμός των καταχωρήσεων που μπορεί να πραγματοποιηθεί, πρίν να πραγματοποιηθεί ένα split, ώστε να μειωθεί το μήκος της αλυσίδας υπερχείλισης*

Λύση

Συνεχίζοντας το προηγούμενο ερώτημα, ο μέγιστος αριθμός καταχωρήσεων θα προκύψει με τιμές που έχουν κλειδιά της μορφής 2^m +3 όπου το m είναι ένας επαρκώς μεγάλος αριθμός. Στην περίπτωση αυτή μπορούμε να προσθέσουμε 3 τιμές χωρίς να γίνει

Σημειώματα

Σημείωμα αναφοράς

Σημείωμα Αδειοδότησης

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Κρήτης» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.