Δομές Δεδομένων και Αρχείων

ΑΝΑΦΟΡΑ ΔΕΥΤΕΡΗΣ ΑΣΚΗΣΗΣ-ΑΝΑΖΗΤΗΣΗ ΣΕ ΑΡΧΕΙΑ ΜΕ ΔΕΙΚΤΗ ΔΕΙΚΤΟΤΗΣΗΣ(INDEXING)

Καλογεράκης Στέφανος|2ο Έτος|14/5/17

Σκοπός

Σκοπός της πρώτης άσκησης ήταν η ανάπτυξη τον δομών ενός B-tree και μιας Posting List που αποθήκευε και διάβαζε δεδομένα από εξωτερικά αρχεία. Η υλοποίηση του προγράμματος έγινε στην γλώσσα προγραμματισμού Java με την χρήση του εργαλείου Eclipse.

Υλοποίηση

Με την εκκίνηση εκτέλεσης του κώδικα πραγματοποιείται το διάβασμα των δεδομένων από τα εξωτερικά αρχεία που δίνονται και η εισαγωγή τους σε αρχεία για το btree και την posting list. Το διάβασμα των αρχείων γίνεται δυναμικά **από τον φάκελο του πρότζεκτ** επιτρέποντας και την εισαγωγή και άλλων αρχείων προς επεξεργασία με το επίθεμα txt(“.txt”) ενώ κάθε φορά που γίνεται επιτυχής επεξεργασία του κάθε αρχείου εμφανίζεται το αντίστοιχο μήνυμα.

Η κάθε λέξη επεξεργάζεται/εισάγεται και στις δύο δομές ξεχωριστά. Οι λέξεις έχουν αυστηρά μέγεθος μέχρι 12 bytes όπως ορίζεται από την εκφώνηση ενώ όποιες το υπερβαίνουν λαμβάνονται υπόψη μόνο τα πρώτα 12 byte. Επίσης πριν την εισαγωγή σε οποιαδήποτε δομή οι λέξεις μετατρέπονται όλες σε μικρά γράμματα.

Αξίζει να σημειωθεί συναντήθηκε πρόβλημα στην μετατροπή των κλάσεων σε Object γεγονός οδήγηση στην χρήση Treemaps παράγοντας όμως το ζητούμενο αποτέλεσμα αποθηκεύοντας σε εξωτερικά αρχεία τις δομές που δημιουργήσαμε.

Ακόμη ο τελικός κώδικας που παραδόθηκε δεν πραγματοποιεί ακριβώς το ζητούμενο της εκφώνησης για λόγους βελτιστοποίησης. Στα σχόλια του κώδικα υπάρχουν γραμμές που επιτρέπουν την αποθήκευση κάθε φορά στην μνήμη των δομών που επιθυμούμε οι οποίες όμως δεν χρησιμοποιούνται καθώς μειώνεται αισθητά η απόδοση. Το τρέχον πρόγραμμα αποθηκεύει σε εξωτερικό αρχείο τις δύο δομές με την ολοκλήρωση της επεξεργασίας τους και όχι κάθε φορά. Να επαναλάβουμε όμως ότι και ***ο ζητούμενος τρόπος υλοποιείται*** *με την χρησιμοποίηση ορισμένων γραμμών κώδικα που είναι στα σχόλια.*

Μετά το τέλος εισαγωγής των δεδομένων εμφανίζονται μια σειρά από δεδομένα χρήσιμα για την υλοποίηση της άσκησης. Αυτά είναι οι συνολικές λέξεις που υπήρχαν στα αρχεία, οι συνολικές διαφορετικές, οι συνολικές προσβάσεις στον δίσκο για την δημιουργία του btree, και οι μέσες προσβάσεις για την δημιουργία του δέντρου.

Έπειτα, φαίνεται ένα μενού επιλογών στο οποίο υλοποιούνται τα ζητούμενα ερωτήματα. Το μενού επιλογών απαιτεί έγκυρη επιλογή ακεραίου για να λειτουργήσει ειδάλλως, τερματίζεται το πρόγραμμα. Οι επιλογές του μενού περιγράφονται παρακάτω

- Στην πρώτη επιλογή, ο χρήστης δίνει μια λέξη της επιλογής του και εμφανίζεται αναλυτικά σε ποια κείμενα βρίσκεται η συγκεκριμένη λέξη και πόσες θέσεις bytes απέχει από την αρχή του. Σε περίπτωση που δεν υπάρχει εμφανίζεται σε κάθε

περίπτωση ενώ κάθε φορά θα εμφανιστεί ο αριθμός των προσβάσεων που έγιναν στον δίσκο.

- Η δεύτερη επιλογή, πραγματοποιεί αναζήτηση λέξεων που υπήρχαν στα αρχεία που δίνονταν στην posting list. Τα αρχεία και πάλι διαβάζονται από εξωτερικό αρχείο δυναμικά ,από τον φάκελο File_read δίνοντας και πάλι δυνατότητα προσθήκη .txt αρχείων στον συγκεκριμένο φάκελο προς επεξεργασία. Στο τέλος εμφανίζονται αναλυτικά όπως και την πρώτη επιλογή τα κείμενα που συναντάται η λέξη και τα bytes που απέχει από την αρχή του αρχείου. Η τέταρτη επιλογή είναι παραπλήσια καθώς πραγματοποιεί αναζήτηση στα αρχεία του φακέλου File_read_random με τις λέξεις να είναι τυχαίες.
- Στην τρίτη επιλογή, γίνεται αναζήτηση λέξεων που υπήρχαν στα αρχεία που δίνονταν στο btree. Τα αρχεία διαβάζονται με αντίστοιχο τρόπο όπως το προηγούμενο ερώτημα. Στο τέλος εμφανίζεται και πάλι ο μέσος αριθμός προσβάσεων στον δίσκο. Να σημειωθεί ότι και η πέμπτη επιλογή πραγματοποιεί τις ίδιες λειτουργίες με τις παραπάνω με την μόνη διαφορά ότι τα αρχεία που έχει πρόσβαση είναι σε άλλο φάκελο(File_read_random) με τις λέξεις που περιέχουν να είναι τυχαίες και δεν βρίσκονται απαραίτητα σε κάποιο από τα αρχεία που δόθηκαν
- Στην τελευταία επιλογή, εκτυπώνεται το λεξικό που ζητείται στο πρώτο ερώτημα. Η εκτύπωση αυτή δεν ήταν απαραίτητη αλλά τα αρχεία που δημιουργήθηκαν τα δεδομένα αποθηκεύονται και επεξεργάζονται σαν Object γεγονός που δεν τα καθιστά προς ανάγνωση από άνθρωπο. Εμφανίζεται λοιπόν για λόγους επαλήθευσης της ορθότητας της υλοποίησης.

Αποτελέσματα

Μετά την υλοποίηση του κώδικα είμαστε σε θέση να συμπληρώσουμε τον παρακάτω πίνακα:

	Α. Εισαγωγή	**Β. Επιτυχής Αναζήτηση**		**Γ. Τυχαία αναζήτηση**	
Μέθοδος	Μέσος αριθμός προσβάσεων στο δίσκο κατά την δημιουργία του B-tree	Μέσος αριθμός προσβάσεων δίσκου στο B-tree για 100 αναζητήσεις λέξεων που υπάρχουν	Μέσος αριθμός προσβάσεων σε σελίδες δίσκου στο Ευρετήριο	Μέσος αριθμός προσβάσεων δίσκου στο B-tree για 100 αναζητήσεις τυχαίων λέξεων	Μέσος αριθμός προσβάσεων σε σελίδες δίσκου στο Ευρετήριο
Μέτρηση	3,2	3,5	330,2	3,8	635,27

Επιπροσθέτως, ο υπολογισμός του βαθμού n του btree γίνεται από τον τύπο:
N = (n-1)*4bytes[info]+(n-1)*12 bytes [String]+4 bytes [number of records]+4 bytes [father]+4 bytes *n[child]. Ισχύει ότι συναρτήσει του $n = \frac{N-8}{20}$.

Το N είναι ο αριθμός των bytes ανά σελίδα. Έτσι με N=128bytes που μας δίνεται προκύπτει ότι n = 6,8 άρα η τάξη είναι 6.

Τα αποτελέσματα του παραπάνω πίνακα είναι απόλυτα λογικά. Αρχικά γνωρίζουμε από θεωρία ότι ο μέσος αριθμός προσβάσεων για ένα δυαδικό δέντρο υπολογίζεται από τον τύπο $c = logn\left(\frac{N}{b}\right)$ όπου N ο συνολικός αριθμός των στοιχείων και b το μέγεθος της σελίδας. Στο πρόβλημά μας n=6, N=1496 και b=128 και προκύπτει c=3,07 τιμή που πλησιάσει σε όλες τις μετρήσεις για το b-tree ενώ υπάρχει μια απόκλιση καθώς δεν υπάρχει πλήρης βελτιστοποίηση . Αναμενόμενα επίσης σε αναζήτηση τυχαίων λέξεων οι προσβάσεις είναι περισσότερες καθώς υπάρχουν λέξεις που δεν υπάρχουν στο btree και γίνονται έτσι παραπάνω αναζητήσεις. Αντίστοιχα ισχύει και στην περίπτωση του posting list. Το posting list και το btree είναι επίσης λογικό να απέχουν κατά πολύ στον μέσο αριθμό προσβάσεων καθώς το posting list για να βρει την πρώτη σελίδα που υπάρχει η λέξη πραγματοποιεί ουσιαστικά σειριακή αναζήτηση.

Πηγές

http://stackoverflow.com/questions/5924237/java-read-all-txt-files-in-folder

https://www.youtube.com/watch?v=VwV1VKvtVtI

http://www.di.ufpb.br/lucidio/Btrees.pdf

http://turing.plymouth.edu/~zshen/Webfiles/notes/CS322/CodeSample/com/mhhe/clrs2e/BTree.java

https://github.com/jankotek/JDBM3/blob/master/src/main/java/org/apache/jdbm/BTree.java

http://www.cs.columbia.edu/~allen/S14/NOTES/DisplaySimpleTree.java

https://www.youtube.com/watch?v=ymvFVkJ_SDI

https://www.youtube.com/watch?v=hQut8sCGcDw&t=21s

http://code.google.com/p/himmele/source/browse/trunk/Algorithms%20and%20Data%20Structures/BTree/src/BTree.java

https://www.tutorialspoint.com/java/util/treemap_get.htm

http://beginnersbook.com/2013/12/treemap-in-java-with-example/

https://gist.github.com/belun/2214237

http://www.sourcecodesworld.com/articles/java/java-data-structures/Reading_and_Writing_Trees.asp

http://math.hws.edu/eck/cs124/javanotes3/c11/ex-11-2-answer.html

https://books.google.gr/books?id=wyM-DQAAQBAJ&pg=PA436&lpg=PA436&dq=java+how+to+save+a+binary+tree+in+a+file&source=bl&ots=IOrFpizNfv&sig=p5ncnAl0aU25bcfOXWii5gRPM60&hl=el&sa=X&ved=0ahUKEwiBh6LZoujTAhUDNhoKHa7eDIQ4ChDoAQhdMAg#v=onepage&q=java%20how%20to%20save%20a%20binary%20tree%20in%20a%20file&f=false

http://stackoverflow.com/questions/4909615/what-pattern-should-be-used-in-a-java-util-scanner-to-fetch-the-next-string-iden