Δομές δεδομένων και αρχείων

Αναφορά 3ης εργασίας

Θωμάς Λάγκαλης

Μάιος 2023

Περιεχόμενα

1 Εισαγωγή

Στη παρούσα εργαστηριακή άσκηση έγινε υλοποίηση ενός B+ Tree ως κύρια δομή σε συνδυασμό με συνδεδεμένη λίστα ως δευτερεύον δομή δεδομένων. Συγκεκριμένα, τα δεδομένα που επεξεργάζονται στο πρόγραμμα είναι λέξεις οι οποίες περιέχονται σε αρχεία. Το δέντρο (B+Tree) αποτελεί το λεξικό με τις λέξεις τω αρχείων και η συνδεδεμένη λίστα το ευρετήριο. Τα κλειδιά του δέντρου είναι οι λέξεις και τα values, τα οποία βρίσκονται μόνο στα φύλλα, περιέχουν τις απαραίτητες πληροφορίες για το εκάστοτε κλειδί (λέξη), δηλαδή σε ποια αρχεία και σε ποιες θέσεις των αρχείων εμφανίζεται η λέξη. Οι πληροφορίες αυτές περιέχονται σε μία συνδεδεμένη λίστα κάθε κόμβος της οποίας περιέχει δύο κλειδιά, το ένα είναι το αρχείο στο οποίο περιέχεται η λέξη και το δεύτερο είναι η θέση της λέξης στο αρχείο αυτό.

Σκοπός της εργασίας είναι η εξοικείωση με τα B(+) Tries και η εφαρμογή τους για τη διαχείριση δεδομένων από αρχεία. Επίσης, η κατανόηση της απόδοσης των B+ Tries και η καταλληλότητά τους για εφαρμογές που χειρίζονται δεδομένα στον δίσκο. Τέλος, έγινε σύγκριση της απόδοσης της αναζήτησης για δέντρα διαφορετικών τάξεων (M).

2 Οδηγίες εκτέλεσης κώδικα

Η εκτέλεση του προγράμματος βασίζεται σε μία διεπαφή (Command Line Interface - CLI). Μέσω του CLI εκτελούνται τρεις εντολές:

- **test**: εκτελεί 100 αναζητήσεις σε ένα δέντρο τάξης 10 και 100 αναζητήσεις σε δέντρο τάξης 20. Τα δέντρα αυτό έχουν αρχικοποιηθεί ειδικά για τον σκοπό του test με τα αρχεία 1.txt και 2.txt που δίνονται για τους σκοπούς της άσκησης. Έπειτα, εκτυπώνει αποτελέσματα που ζητούνται στην εκφώνηση, δηλαδή μέσος αριθμός συγκρίσεαων και μέσος αριθμός επισκέψεων σε κόμβους.
- **insert** <όνομα αρχείου>: εισάγονται σε ένα άλλο δέντρο που δεν έχει αρχικοποιηθεί και είναι τάξης 10 (M=10) τα κλειδιά (λέξεις) που περιέχονται στο αρχείο το όνομα του οποίου δίνεται σαν όρισμα στην εντολή.
- **search** <κλειδί>: Γίνεται αναζήτηση του κλειδιού (της λέξης) στο ίδιο δέντρο που εισάγονται τα στοιχεία με την εντολή insert. Έπειτα, τυπώνονται σειριακά οι πληροφορίες για το κλειδί ή δεν τυπώνεται τίποτα αν το κλειδί δεν βρέθηκε.
- **quit**: Το πρόγραμμα τερματίζεται.

Όλες οι παραπάνω οδηγίες μπορούν να βρεθούν και κατά την εκτέλεση του προγράμματος με την εντολή **help**

```
>>>> help
Commands:
    test - Performs 100 searches -with a tree of order 10 and a tree of order 20- with random words from 1.txt and 2.txt and prints results.

    insert <file name> - Inserts all the words of <file name> in the B+Tree.

    search <word> - Searches for <word> in the B+Tree and prints the results if exist.

    quit
>>>>
```

Εικόνα 1: Διεπαφή (CLI) του προγράμματος.

3 Γενική δομή του κώδικα

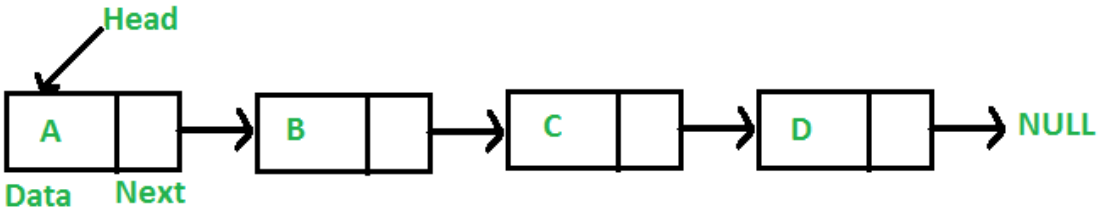
Ο κώδικας ακολουθεί μία αντικειμενοστραφή δομή. Συγκεκριμένα, υπάρχουν 6 κλάσεις από τις οποίες η **BpTree** (B plus Tree) και η Linked List αναφέρονται στις υπό μελέτη δομές δεδομένων της εργαστηριακής άσκησης. Η κλάση **Menu** υλοποιεί ουσιαστικά τη διεπαφή χρήστη-προγράμματος (CLI), η **MyUtils** περιέχει χρήσιμες μεθόδους για την υλοποίηση του προγράμματος ενώ η **MultiCounter** είναι η κλάση που περιέχει τους μετρητές για τον προσδιορισμό των μέσο αριθμών προσβάσεων σε κόμβους και τον μέσο αριθμό συγκρίσεων.

3.1 BpTree Class

Η υλοποίηση της δομής του B+ Tree πάρθηκε έτοιμη από το gitHub, ο σύνδεσμος της δίνεται στο παράρτημα **πηγές**. Από τη συγκεκριμένη κλάση χρειάστηκαν κυρίως οι συναρτήσεις **put** (βάζει κλειδιά στο δέντρο) και **get** (κάνει αναζήτηση και επιστρέφει το value του κλειδιού, αν υπάρχει). Επίσης, έγιναν ορισμένες αλλαγές στον κώδικα που αφορούν κυρίως την τάξη του δέντρου. Συγκεκριμένα, με τη συνάρτηση getOrder() και με τη προσθήκη του μη στατικού στοιχείου order είναι δυνατόν να λαμβάνεται η τάξη του δέντρου (το M) εκτός της κλάσης BpTree (πχ από τη Main). Ακόμη, προστέθηκε ένας ακόμα contructor μέσω του οποίου γίνεται να οριστεί η τάξη του δέντρου. Οι αλλαγές αυτές έγιναν αποκλείστηκα για τον σκοπός της παρούσας άσκησης και συγκεκριμένα για την λειτουργία του **test** και για την εκτύπωση των αποτελεσμάτων/μετρήσεων.

3.2 LinkedListed Class

Η κλάση LinkedList υλοποιήθηκε για της ανάγκες της άσκησης και περιέχει τις λειτουργίες/μεθόδους: **toString()**, οι οποία επιστρέφει ένα String με τις πληροφορίες της συνδεδεμένης λίστας και η **insert(key)** η οποία εισάγει στη συγκεκριμένη δομή ένα ζεύγος πληροφοριών στη συγκεκριμένη περίπτωση της άσκησης το ζεύγος είναι String-int(eger). Να σημειωθεί πως η συνδεδεμένη λίστα αποτελεί το ευρετήριο μίας λέξης, δηλαδή το περιέχει το όνομα του αρχείου (String) που περιέχει την λέξη και τον αριθμό της θέσης της μέσα στο αρχείο (int).



Εικόνα 2: Linked List

4 Αξιολόγηση αποτελεσμάτων

Τα αποτελέσματα που παρουσιάζονται στην εικόνα 3 αναφέρονται σε 100 αναζητήσεις που πραγματοποιήθηκαν σε δύο δέντρα (B+Tries) που περιέχουν τα δεδομένα των 1.txt και 2.txt. Το M αναφέρεται στη τάξη του δέντρου. Εξ ορισμού ο μέγιστος αριθμός παιδιών κάθε κόμβου είναι M ενώ ο μέγιστος αριθμός κλειδιών είναι για κόμβος που περιέχει k παιδιά είναι k-1. Όταν γίνεται εισαγωγή κλειδιού σε δέντρο που περιέχει M-1 κλειδιά τότε ο κόμβος κάνει split (”σπάει”) δηλαδή, το μέσο κλειδί εισάγεται στον κόμβο-πατέρα ενώ τα κλειδιά που είναι μικρότερα πάνε στον αριστερό κόμβο και τα υπόλοιπα στον δεξιό.

```
>>>> test
M=10
Average Nodes Accesses = 5.0
Average Comparisons = 17.02

M=20
Average Nodes Accesses = 4.0
Average Comparisons = 21.81
>>>>
```

Εικόνα 3: Αποτελέσματα 100 αναζητήσεων για M=10 και M=20

4.1 Σχολιασμός μέσου αριθμού επισκέψεων σε κόμβους

Όπως φαίνεται και στην εικόνα 3 ο μέσος αριθμός επισκέψεων σε κόμβους για M=10 είναι μεγαλύτερος από την περίπτωση όπου M=20 (5 και 4 επισκέψεις αντίστοιχα). Να σημειωθεί πως το B+Tree είναι πλήρως ισορροπημένο και όλη η ζητούμενη πληροφορία υπάρχει πάντα στα φύλλα (το δεύτερο χαρακτηριστικό αποτελεί την ειδοποιό διαφορά του B+Tree με το B Tree). Συνεπώς, ο αριθμός των επισκέψεων σε κόμβους ανά αναζήτηση δίνει και το βάθος του δέντρου, αφού ο αλγόριθμος της αναζήτησης δεν χρησιμοποιεί κάποιο backtracing

(οπισθοδρόμηση στη διαδρομή που ακολουθεί). Τέλος, να επισημαίνεται πως το βάθος του δέντρου είναι περίπου $log_M(n)$.

Λόγω των προαναφερθέντων χαρακτηριστικών του B+Tree προκύπτει πως το για M=10 το βάθος του δέντρου είναι μεγαλύτερο απ' ότι για M=20. Αυτό είναι λογικό δεδομένου ότι για μικρότερο M κάθε κόμβος "χωράει" λιγότερα κλειδιά οπότε γίνεται περισσότερες φορές split (όταν γίνεται split στην ρίζα το βάθος του δέντρου αυξάνεται). Επομένως, όσο η τάξη του δέντρο ελαττώνεται, το βάθος του δέντρου αυξάνεται.

4.2 Σχολιασμός μέσου αριθμού συγκρίσεων

Από την εικόνα 3 γίνεται αντιληπτό πως για M=10 ο μέσος αριθμός συγκρίσεων είναι μικρότερος απ' ότι για M=20 (17.02 και 21.81 συγκρίσεις αντίστοιχα). Όπως έχει προαναφερθεί τα B+Tree με μεγαλύτερο M περιέχουν κόμβους με μεγαλύτερο αριθμό κλειδιών. Δεδομένου ότι οι συγκρίσεις σε κάθε κόμβο συμβαίνουν σειριακά είναι εύλογο να ειπωθεί πως για έναν κόμβο ο αριθμός των συγκρίσεων είναι μεγαλύτερος για μεγαλύτερα M (περισσότερα κλειδιά στον κόμβο). Το ερώτημα που προκύπτει είναι: κατά πόσο οι λιγότερες επισκέψεις σε κόμβους επηρεάζουν το αποτέλεσμα των συγκρίσεων;

Αρχικά, θα εξετασθεί η χειρότερη περίπτωση συγκρίσεων (μέγιστες δυνατές συγκρίσεις). Ας υποθέσουμε πως ένα δέντρο είναι M τάξης δηλαδή κάθε κόμβος του περιέχει το πολύ M-1 παιδιά οπότε, στη χειρότερη περίπτωση θα γίνουν M-2 συγκρίσεις (σε γεμάτους κόμβους). Δεδομένου βάθους h ο μέγιστος αριθμός συγκρίσεων που μπορούν να γίνου (οι συγκρίσεις γίνονται σειριακά μέσα στου κόμβους) είναι $(M-2) \times h$, η σχέση αυτή αποτελεί ένα άνω φράγμα για τον **συνολικό** αριθμό των συγκρίσεων. Οπότε, για τις περιπτώσεις δέντρων της άσκησης προκύπτουν τα εξής:

- M=10: Μέγιστος δυνατός αριθμός συγκρίσεων = 40
- M=20: Μέγιστος δυνατός αριθμός συγκρίσεων = 72

Οπότε, τα άνω φράγματα συμφωνούν με τα αποτελέσματα, δηλαδή για μεγαλύτερο M ο μέγιστος δυνατός αριθμός συγκρίσεων είναι μεγαλύτερος. Αυτό όμως δεν δικαιολογεί πλήρως τα αποτελέσματα του προγράμματος. Οι παραπάνω εκτιμήσεις των αριθμών συγκρίσεων υποθέτουν πως όλοι οι κόμβοι είναι πλήρως γεμάτοι (100% πληρότητα κλειδιών). Για να γίνει μία καλύτερη θεωρητική εκτίμηση του μέσου αριθμού συγκρίσεων για κάθε M, θα γίνει η υπόθεση ότι οι κόμβοι είναι 50% γεμάτοι. Επίσης, για λόγους απλούστευσης των υπολογισμών θα γίνει πάλι η υπόθεση πως σε κάθε κόμβο με k κλειδιά θα γίνουν k-1 συγκρίσεις, δηλαδή για M=10 θα γίνουν 3.5 (μεταξύ 3 και 4 λαμβάνεται μέσος όρος) συγκρίσεις σε κάθε κόμβο ενώ για M=20 θα γίνουν 8.5 συγκρίσεις. Πολλαπλασιάζοντας με το βάθος του δέντρου για τον συνολικό αριθμό συγκρίσεων προκύπτει:

- M=10: Μέσος αριθμός συγκρίσεων = 22.5
- M=20: Μέσος αριθμός συγκρίσεων = 34

Τα θεωρητικά αποτελέσματα είναι αρκετά κοντά στα αποτελέσματα που λήφθηκαν από το πρόγραμμα (πειραματικά αποτελέσματα) όμως είναι λίγο μεγαλύτερα. Αυτό οφείλεται στην υπόθεση πως σε κάθε κόμβο συγκρίνονται όλα τα κλειδιά σειριακά γεγονός που δεν ισχύει στη πραγματικότητα, αφού η θέση του κλειδιού είναι δυνατόν να βρεθεί πριν γίνουν όλες οι συγκρίσεις. Άρα, ο αριθμός μέσος αριθμός συγκρίσεων σε **κάθε**

κόμβο είναι μικρότερος από την υπόθεση που έγινε άρα και ο συνολικό αριθμός των συγκρίσεων είναι μικρότερος από τα παραπάνω θεωρητικά αποτελέσματα.

Συμπερασματικά, ο αριθμός των συγκρίσεων είναι ανάλογος της τάξης του δέντρου καθώς και του ύψους του. Οπότε, στη περίπτωση της άσκησης που η τάξη του δέντρου αυξήθηκε κατά μία δεκάδα (από M=10 σε M=20) και ως εκ τούτου το βάθος ελαττώθηκε κατά μία μονάδα (από h=5 σε h=4) ο αριθμός των συγκρίσεων αυξήθηκε.

4.3 Πώς θα αξιολογούσατε τα αποτελέσματα εάν το B+-Tree και το Ευρετήριο ήταν στο δίσκο;

Η δομή του B+Tree είναι ιδανική για εφαρμογές που επεξεργάζονται δεδομένα στον δίσκο. Ο λόγος είναι ο μικρός αριθμός επισκέψεων σε κάθε κόμβο. Η πρόσβαση σε μία σελίδα του δίσκου είναι μία διαδικασία η οποία κοστίζει αρκετά σε χρόνο. Για τον αυτό τον λόγο, ο αριθμός προσβάσεων στον δίσκο οφείλει να ελαχιστοποιηθεί. Οργανώνοντας την τάξη του δέντρου ώστε κάθε κόμβος να αντιστοιχεί σε μία σελίδα του δίσκου, η δομή B+Tree προσφέρει έναν αρκετά μικρό αριθμό προσβάσεων σε κόμβους (άρα και σε σελίδες δίσκου). Επίσης, στο B+Tree τα δεδομένα βρίσκονται όλα στα φύλλα του δέντρου τα οποία είναι συνδεδεμένα μεταξύ τους (συνδεδεμένη λίστα), γεγονός που επιταχύνει την αναζήτηση εύρους κλειδιών (range query).

Αν το λεξικό ήταν στο δίδσκο το B+Tree θα αποτελούσε μία αρκετά καλή επιλογή δομής. Από την άλλη το ευρετήριο με συνδεδεμένη λίστα ίσως να μην ήταν τόσο αποτελεσματική δομή αν κάθε κόμβος θεωρηθεί μία σελίδα στον δίσκο. Αν όμως, η κάθε συνδεδεμένη λίστα μπει σε μία σελίδα δίσκου (ή ακόμα αν κάθε σελίδα περιέχει παραπάνω από μία συνδεδεμένη λίστα) τότε ο αριθμός προσβάσεων στον δίσκο για το ευρετήριο μειώνεται σημαντικά. Επομένως, το B+Tree αποτελεί μία αποτελεσματική δομή αν το λεξικό ήταν στο δίσκο λόγω τον μικρό αριθμό επισκέψεων σε κόμβους (στον δίσκο), ενώ η συνδεδεμένη λίστα του ευρετηρίου χρίζει προσοχή στην υλοποίησή της ώστε να απαιτεί λίγες προσβάσεις στον δίσκο.

5 Πηγές

- Linked List εικόνα 2: https://www.geeksforgeeks.org/what-is-linked-list/
- B+ Tree υλοποίηση: https://github.com/jojozhuang/dsa-java/blob/master/ds-btree/src/main/java/johnny/dsa/ds/BTree.ja