Δομές Δεδομένων και Αρχείων

ΑΝΑΦΟΡΑ ΠΡΩΤΗΣ ΑΣΚΗΣΗΣ-ΕΠΕΞΕΡΓΑΣΊΑ ΑΡΧΕΙΩΝ

Καλογεράκης Στέφανος|2ο Έτος|22/3/17

Σκοπός

Σκοπός της πρώτης άσκησης ήταν η εξοικείωση και η αξιολόγηση της απόδοσης των μεθόδων αναζήτησης στον δίσκο με την άσκηση να απαρτίζεται από 4 διαφορετικές αναζητήσεις που εφαρμόζονται σε δυαδικό αρχείο.

Οι μέθοδοι που μας ζητήθηκαν να υλοποιήσουμε ήταν κατά σειρά:

- Σειριακή αναζήτηση στο αρχείο για τυχαίο κλειδί
- Δυαδική αναζήτηση στο αρχείο για τυχαίο κλειδί
- Δυαδική αναζήτηση με ομαδοποίηση των ερωτήσεων
- Δυαδική αναζήτηση με χρήση προσωρινής μνήμης

Υλοποίηση

Με την εκκίνηση εκτέλεσης του κώδικα εμφανίζεται στον χρήστη μενού επιλογής των μεθόδων που αναφέρθηκαν παραπάνω με τον χρήστη να πρέπει να εισάγει τον ακέραιο που αντιστοιχεί στην επιλογή που επιθυμεί. Σε περίπτωση που ο χρήστης δεν δώσει ακέραιο αριθμό ή ο ακέραιος που δοθεί δεν ανήκει στις επιλογές του μενού εμφανίζεται μήνυμα λάθους και το πρόγραμμα τερματίζεται.

Σε όλες τις περιπτώσεις δημιουργείται στην αρχή ταξινομημένο δυαδικό αρχείο(.bin extension) το οποίο περιείχε ακεραίους από 1 μέχρι 10^7 όπως ζητούσε η εκφώνηση. Κατά την εισαγωγή κάθε σελίδα φορτώναμε 128 αριθμούς σε έναν buffer μέχρι να εξαντληθούν όλοι οι αριθμοί.

Αξίζει να σημειωθεί ότι το αποτέλεσμα της διαίρεσης του αριθμού των στοιχείων δια το μέγεθος της σελίδας ήταν ακέραιος αριθμός γεγονός ιδιαίτερα σημαντικό αφού δεν έπρεπε να λάβουμε υπόψιν περιπτώσεις για το buffer της τελευταίας σελίδας που δεν θα γέμιζε

.

<u>*Στο ερώτημα της σειριακής αναζήτησης*</u>, φορτωνόταν κάθε φορά στην κύρια μνήμη 128 ακέραιους σειριακά και εφαρμόζαμε σε αυτούς δυαδική αναζήτηση μέχρι να βρεθεί το αποτέλεσμα. Παρατηρήθηκε όμως ότι αυτή η μέθοδος αναζήτησης δεν ήταν ιδιαίτερα αποτελεσματική και αρκετά αργή ειδικά όσο το κλειδί αναζήτησης μεγάλωνε. Ήταν δηλαδή γρήγορη, μόνο σε περιπτώσεις με πολύ μικρά κλειδιά. Αυτό φαίνεται και από τον μέσο

αριθμό προσβάσεων για 10.000 αναζητήσεις τυχαίων κλειδιών

```
Key is 9576603
Found on page:74818
Accesses demanded:74818

Key is 8651802
Found on page:67593
Accesses demanded:67593

Average accesses demanded: 40821.61
Programm terminated.Thanks for using!!!
```

Εικόνα 1: Ερώτημα Α

Ενδεικτικά λοιπόν για 10.000 στοιχεία ο μέσος αριθμός προσβάσεων ήταν 40.821,61

Στο ερώτημα της δυαδικής αναζήτησης στο αρχείο για τυχαίο κλειδί, χρησιμοποιήσαμε και πάλι δυαδική αναζήτηση αλλά αυτή την φορά την χρησιμοποιήσαμε για τις σελίδες δίσκου. Βρίσκουμε αρχικά την μεσαία σελίδα. Σε περίπτωση που βρίσκεται εκεί η αναζήτηση τελειώνει. Σε αντίθετη περίπτωση ελέγχουμε τον πρώτο και τον τελευταίο αριθμό της μεσαίας σελίδας. Αν το κλειδί είναι μικρότερο του πρώτου μεταβαίνουμε στο αριστερό μισό του αρχείου ενώ αν είναι μεγαλύτερο του τελευταίου μεταβαίνουμε στο δεξί μισό και αυτό μέχρι να βρεθεί το κλειδί. Η υλοποίηση της κύριας πραγματοποιήθηκε αναδρομικά.

Σε αντίθεση με την σειριακή η συγκεκριμένη αναζήτηση ήταν αρκετά πιο αποτελεσματική αφού η μετάβαση σε μεγαλύτερες σελίδες απαιτεί πολύ λίγες προσβάσεις. Η μόνη πιθανότητα η σειριακή να είναι πιο γρήγορη είναι μόνο για πολύ μικρά κλειδιά στις πρώτες σελίδες του αρχείου. Μετά από την εκτέλεση του κώδικα η διαφορά αυτή αποτυπώθηκε πολύ πιο καθαρά αφού αυτή η μέθοδος είχε μέσο όρο προσβάσεων 15,28

```
Key: 7860984
Key found on page: 61414
Accesses demanded:1514

Key: 5596073
Key found on page: 43720
Accesses demanded:1528

Average accesses demanded: 15.28
Programm terminated.Thanks for using!!!
```

Εικόνα 2: Ερώτημα Β

Στο επόμενο ερώτημα, της δυαδικής αναζήτησης για ομαδοποιημένα κλειδιά, η υλοποίηση έμοιαζε αρκετά με την υλοποίηση του προηγούμενου ερωτήματος αφού η αναζήτηση πραγματοποιούνταν με τον ίδιο τρόπο. Η διαφορά, ήταν ο σε αυτό πρώτα ζητούσαμε τα κλειδιά τα οποία και ταξινομήσαμε με αύξουσα σειρά. Το στοιχείο αυτό απλοποίησε σε

μεγάλο βαθμό την αναζήτηση, αφού μετά την ταξινόμηση όποιες σελίδες είχαν ελεγχθεί δεν χρειάζονταν επανέλεγχο.

```
Page 77739
Key 9956945
Key found on page: 77789
Accesses demanded 9
Page 77789
Key 9977761
Key found on page: 77952
Accesses demanded 6
Page 77952

Average accesses demanded: 13.901
Programm terminated.Thanks for using!!!
```

Εικόνα 3: Ερώτημα Γ

Θα αναμέναμε επομένως, καλύτερη απόδοση της συγκεκριμένης μεθόδου σε σχέση με την προηγούμενη αφού όταν εισέρχονται καινούργια κλειδιά η αναζήτηση δεν ξεκινάει από την αρχή. Το στοιχείο αυτό επιβεβαιώνεται μετά την εκτέλεση του κώδικα με τον μέσο αριθμό προσβάσεων να είναι 13,901 για 10.000 κλειδιά. Αξίζει να σημειωθεί ότι σε περίπτωση που ένα κλειδί βρίσκεται σε σελίδα που έχει ήδη φορτωθεί δεν προσμετρείται ως πρόσβαση.

Στο τελευταίο ερώτημα με την αναζήτηση στην cache, η αναζήτηση βασίστηκε και πάλι στην δυαδική αναζήτηση όπως πραγματοποιήθηκε στο δεύτερο ερώτημα. Αυτή την φορά όμως έπρεπε να δημιουργήσουμε ουρά Κ στοιχείων, με τις σελίδες που βρισκόταν τα κλειδιά που αναζητούσαμε. Πιο αναλυτικά, κατά την είσοδο ενός κλειδιού, γινόταν πρώτα αναζήτηση στην ουρά αν ήταν σε κάποια σελίδα της. Σε αυτή την περίπτωση δεν μετρούσε ως πρόσβαση. Σε αντίθετη περίπτωση, εντοπιζόταν η σελίδα του κλειδιού και εφόσον η ουρά δεν ήταν γεμάτη (περιείχε λιγότερα από Κ στοιχεία) η σελίδα εισαγόταν στην ουρά. Σε περίπτωση που ήταν γεμάτη η ουρά διαγραφόταν η πιο παλιά σελίδα που βρισκόταν σε αυτή.

Για Κ=1 και για μια ενδεικτική αναζήτηση 10000 αποτελεσμάτων προέκυψαν τα παρακάτω αποτελέσματα:

```
Adding the new buffer to cache...
Key 693083
The key doesn't exist in cache
Key found on page: 5415

Searching using cache
**********************************************
Removing the oldest buffer...
Adding the new buffer to cache...

Average accesses demanded: 12.46
Programm terminated.Thanks for using!!!
```

Εικόνα 4: Ερώτημα Δ, για Κ=1

Για Κ=50 και πάλι μετά από υπολογισμούς προκύπτει ότι:

```
Searching using cache
**************************************************
Removing the oldest buffer...
Adding the new buffer to cache...

Average accesses demanded: 9.936
Programm terminated.Thanks for using!!!
```

Εικόνα 5: Ερώτημα Δ, για Κ=50

Τέλος για Κ=100 προκύπτουν τα παρακάτω δεδομένα

```
Key 766016
The key doesn't exist in cache
Key found on page: 5985

Searching using cache
**************************************************
Removing the oldest buffer...
Adding the new buffer to cache...

Average accesses demanded: 7.481
Programm terminated.Thanks for using!!!
```

Εικόνα 6: Ερώτημα Δ, για Κ=100

Τα παραπάνω αποτελέσματα ήταν λίγο πολύ αναμενόμενα, καθώς θα περιμέναμε όταν Κ=1 ο μέσος αριθμός προσβάσεων να είναι πολύ κοντά με την δυαδική αναζήτηση του δεύτερου ερωτήματος(15,28 μέσος αριθμός) αφού πραγματοποιεί ουσιαστικά την ίδια λειτουργία

έχοντας ικανότητα να αποθηκεύει μονάχα μια σελίδα. Επίσης όπως ήταν λογικό παρατηρήθηκε ότι όσο αυξανόταν το Κ τόσο μίκραινε ο αριθμός των προσβάσεων με όλο και παραπάνω σελίδες να είναι αποθηκευμένες στην cache μνήμη.

Να σημειωθεί ότι για την διευκόλυνση επίλυσης του προβλήματος χρησιμοποιήθηκε η Deque, κλάση της java που θυμίζει την κλάση Queue καθώς η πρώτη περιλαμβάνει τις μεθόδους της τελευταίας εμπλουτισμένες.

Ακολουθεί συγκεντρωτικός πίνακας όλων των ερωτημάτων

Μέθοδος	Α	Β	Γ	Δ(Κ=1)	Δ(Κ=50)	Δ(Κ=100)
Απόδοση	40.821,61	15,28	13,901	12,46	9,936	7,481

Συνοψίζοντας λοιπόν την άσκησης ο λιγότερος αποτελεσματικός τρόπος αναζήτησης ήταν του ερωτήματος Α (σειριακή) και ο πιο αποτελεσματικός του Δ για μεγαλύτερο Κ για λόγους που επισημάνθηκαν παραπάνω. Γενικώς όσο πιο δομημένη η αναζήτηση τόσο πιο αποτελεσματική.

Για τους σκοπούς της άσκησης, και όπως ζητήθηκε από για την δημιουργία-επεξεργασία των αρχείων χρησιμοποιήθηκε η RandomAccessFile με ορισμένες μεθόδους όπως seek(), καθώς και οι κλάσεις ByteArrayOutputStream και DataOutputStream. Αυτό βέβαια μας οδήγησε να δημιουργήσουμε πίνακες buffers τύπου byte για την αποθήκευση απευθείας των δεδομένων από την ByteArrayOutputStream και την αποφυγή περιττών loop(για την μετατροπή και το γράψιμο των στοιχείων σε buffer τύπου int). Συνεπώς και οι δυαδικές αναζητήσεις τροποποιήθηκαν κατάλληλα για να χειρίζονται bytes με εξαίρεση το τελευταίο ερώτημα που στην περίπτωση αναζήτησης στην ουρά θα περιπλέκονταν αρκετά τα πράγματα.

Βιβλιογραφία

Όλες οι πηγές που αξιοποιήθηκαν για την πραγματοποίηση του πρότζεκτ αναγράφονται αναλυτικά παρακάτω:

- Σημειώσεις και κώδικες από τα φροντιστήρια του μαθήματος
- https://cs.wmich.edu/gupta/teaching/cs1120/1120Fall12web/codeJava/advanceFileIO.txt
- http://www.javapractices.com/topic/TopicAction.do?Id=245
- https://docs.oracle.com/javase/7/docs/api/java/io/DataOutputStream.html
- http://stackoverflow.com/questions/11743267/get-random-numbers-in-a-specific-range-in-java
- http://www.java2novice.com/java-search-algorithms/binary-search/
- http://knowledgebuddy.weebly.com/queuecode.html
- http://introcs.cs.princeton.edu/java/43stack/
- http://stackoverflow.com/questions/2709128/how-to-create-an-array-arraylist-stack-and-queue-in-java

- http://cs.lmu.edu/~ray/notes/queues/
- https://docs.oracle.com/javase/7/docs/api/java/util/Deque.html