Δομές Δεδομένων Και Αρχείων

Άσκηση 1η

Έκθεση Άσκησης:

Ζητούμενο 1: Κατασκευή αρχείου Ν αριθμών

Για την υλοποίηση του συγκεκριμένου ερωτήματος χρησιμοποιήθηκε ο κώδικας που μας εξηγήθηκε στο πρώτο φροντιστήριο του μαθήματος. Με άλλα λόγια, αρχικά δημιουργήθηκε ένα ByteArrayOutputStream συνάμα με ένα DataArrayOutputStream στα οποία γράφονταν τυχαίοι αριθμοί που παράγονται από την Random (με την ιδιότητα next int, η οποία υπάρχει στην java). Μέσω μιας for loop γράφουμε κάθε φορά χίλιους αριθμούς και τους εναποθέτουμε στον buffer μας. Από εκεί και πέρα γράφουμε στο αρχείο που δημιουργείται αυτόματα μέσω της εντολής “rw” της RandomAccessFile(γράφει στο υπάρχων αν το όνομα του αρχείου παραπέμπει σε ήδη υπάρχων αρχείο). Αφού κάθε φορά που πάμε να γράψουμε κάνουμε πρόσβαση στον δίσκο, για 100.000 θα χρειαστούμε 100 προσβάσεις, κάτι που επιβεβαιώνουμε αφού μέσω μιας μεταβλητής που αυξάνει από το 0 όσες φορές γράφουμε στο αρχείο εκτυπώνεται ότι ο αριθμός προσβάσεων είναι όντως, όπως αναμενόταν, 100.

Ζητούμενο 2: Ταξινόμηση αρχείου στο δίσκο

Για να υλοποιηθεί το παραπάνω ζητούμενο έγινε αρχική δήλωση του συνολικού πίνακα που θα υποστεί ταξινόμηση στο τέλος, και του buffer που θα γράφει αρχικά στον πίνακα και στην συνέχεια σε κάθε αρχείο από τα δέκα που μας ζητούνται. Συνεχίζοντας την υλοποίηση, κρατείται σε μία μεταβλητή το offset του αρχείου που θα χωρίσουμε στην συνέχεια σε 10. Έτσι, με την βοήθεια μιας for loop 1-10(buffer χιλίων αριθμών όποτε για 10.000 αρχεία θέλω 10 φόρες), παίρνουμε 10.000 στοιχεία κάθε φορά και τα τοποθετούμε ανά χίλια μέσω του buffer στον πίνακα που φτιάξαμε στην αρχή. Έπειτα μέσω της mergesort (βρέθηκε στο web) ταξινομούμε τον πίνακα. Τέλος, με τον buffer που δημιουργήσαμε στην αρχή, γίνεται μεταφορά αρχείων (ανά χίλια) με μία for loop (πάλι 0-10) .Προσοχή το offset του αρχείου πρέπει να κρατηθεί ώστε να μην πανωγράφονται τα στοιχεία(στην ουσία είναι το x->(0-10)*buffer.length). Έτσι, μετά το τέλος της διαδικασίας έχουμε 10 ταξινομημένα αρχεία, χάρις την αρχική for loop(Πάλι από 0 εώς 10 καθώς θέλουμε 10 αρχεία). Τέλος εφόσον έχουμε μία μεταβλητή που να μετράει τις προσβάσεις στο δίσκο επιβεβαιώνουμε ότι οι προσβάσεις που έγιναν στο δίσκο για να δημιουργηθούν τα δέκα αρχεία είναι 200 (για κάθε αρχείο είναι 20 αφού κάνουμε μία πρόσβαση κάθε φορά που διαβάζουμε από το αρχείο που δίνουμε και μία κάθε φορά που γράφουμε στο καινούργιο που δημιουργούμε, και καθώς για ένα καινούργιο αρχείο, αφού ο buffer είναι χιλίων αριθμών, η διαδικασία διαβάσματος και γραψίματος θα γίνει 10 φορές). Στην συνέχεια, στο δεύτερο σκέλος του ζητουμένου μας ζητείται να ενώσουμε όλα αυτά τα αρχεία σε ένα ταξινομημένο. Για αυτό χρησιμοποιούμε έναν πίνακα 2 διαστάσεων (μια διάσταση όσο οι buffers και μια 1000 στοιχεια που χωράει ο κάθε ένας). Ύστερα, βρίσκουμε το min με μια if συνάρτηση μέσα σε μια φορ για 10 φορές (αφού έχουμε 10 buffer). Τέλος παίρνουμε περιπτώσεις, άμα πρέπει να ξαναφορτώσουμε τον buffer και σε περίπτωση που πρέπει να ξαναφορτώσουμε αν υπάρχουν άλλα στοιχεία στο αρχείο για να διαβάσουμε. Σε περίπτωση που

το ένα αρχείο έχει τελειώσει, μέσω μίας αυτοσχέδιας συνάρτησης διαγράφουμε το αρχείο. Έτσι στο τέλος έχουμε ένα ταξινομημένο αρχείο, με τόσους αριθμούς όσους και το αρχείο που χωρίσαμε στην αρχή του ερωτήματος. Για να γίνουν όλα αυτά είναι σημαντικό να κρατάμε σε πίνακα την θέση του κάθε buffer όπως επίσης και τα offset των αρχείων σε πίνακες. Όσον αναφορά τις προσβάσεις, βλέπουμε πάλι ότι έχουμε τις αναμενόμενες (200) καθώς πάλι διαβάζουμε 100 φορές (10 αρχεία από δέκα φορες) και γράφουμε 100 φορές χίλιους αριθμούς (buffer).

Ζητούμενο 3: Αναζήτηση στο ταξινομημένο αρχείο

Το συγκεκριμένο ερωτημα αποτελείται από δύο σκέλη. Το πρώτο είναι το κομμάτι της σειριακής αναζήτησης. Για την υλοποίηση του συγκεκριμένου κομματιού, αρχικά δηλώνουμε έναν buffer 1000 αριθμών. Στην συνέχεια, με ένα while loop όσο δεν έχουμε φτάσει να φορτώνουμε για 100στη φορά και το boolean που έχουμε θέσει με σκοπό να σταματήσουμε την επαναληπτική διαδικασία την πρώτη φορά που θα βρούμε τον αριθμό που θέλουμε (αν τον βρούμε). Παράλληλα εκτυπώνουμε τις προσβάσεις στο δίσκο. Όπώς περιμένουμε, στην σειριακή αναζήτηση σε περίπτωση που δεν βρούμε το αρχείο έχουμε 100 προσβάσεις δίσκου(Πρέπει να διαβαστεί όλο το αρχείο για να επιβεβαιωθεί ότι δεν υπάρχει ο αριθμός). Για 20 επιτυχημένες προσπάθειες παρατηρώ ότι το αποτέλεσμα του μέσου όρου των προσβάσεων στο δίσκο είναι περίπου 46,2 αν έχω επιτυχία.

Για την δυαδική αναζήτηση ακολουθήθηκαν τα βήματα που περιγράφονται στην εκφώνηση της άσκησης. Με την βοήθεια ενός αλγόρυθμου από το Web θέτοντας 2 μεταβλητές start =0 και end =100 (0 η αρχή και 100 η max τιμή (100*1000 =100.000)). Από αυτές τις δύο τιμές παίρνουμε το middle και ανάλογα με τις περιπτώσεις που αναγράφονται στην άσκηση τροποποιούμε το start η το end αντίστοιχα. Μέσω πάλι μιας boolean μεταβλητής σταματάμε την διαδικασία στην πρώτη φορά που θα βρεί αριθμό. Αποτελέσματα για 20 επιτυχημένες προσπάθειες (τυχαία νούμερα) ο μέσος όρος των προσβάσεων για επιτυχημένες προσπάθειες είναι περίπου 5,3 ενώ ο μέσος όρος των αποτυχημένων είναι περίπου 6. Το αποτέλεσμα μας φαίνεται απολύτως λογικό καθώς η αναζήτηση του αριθμού είναι δυαδική

Συνολικά καταλήγω στο επιθυμιτό αποτέλεσμα καθώς βλέπω ότι για μεγάλο σχετικά όγκο δεδομένων στην δυαδική αναζήτηση κερδίζω πολύ λιγότερες προσβάσεις στον δίσκο που είναι κάτι που επιδιώκω. (Περίπου σταθερό ποσό προσβάσεων ανεξάρτητα επιτυχίας η αποτυχίας εύρεσης κλειδιού με την δυαδική αναζήτηση).

Πηγές:

https://www.baeldung.com/java-binary-search (αλγόρυθμος της δυαδικής αναζήτησης)
https://www.mkyong.com/java/how-to-delete-file-in-java/ (συνάρτηση που επηρεάστηκα για να φτιάξω αυτήν που διαγράφει τα αρχεία).
https://www.geeksforgeeks.org/merge-sort/ (ο κώδικας της mergesort που βρήκα)