ΑΜ:2017030021 ΟΝΟΜΑΤΕΠΩΝΥΜΟ:ΙΩΑΝΝΗΣ-ΙΑΣΩΝ ΓΕΩΡΓΑΚΑΣ

ΑΝΑΦΟΡΑ 1ης ΕΡΓΑΣΤΗΡΙΑΚΗΣ ΑΣΚΗΣΗΣ : Επεξεργασία αρχείων και Εξωτερική ταξινόμηση

ΥΛΟΠΟΙΗΣΗ ΑΣΚΗΣΗΣ

Για την υλοποίηση της 1ης εργαστηριακής δημιούργησα τρία εκτελέσιμα αρχεία (main) στο eclipse στην έκδοση του jdk 1.8.0_161 , ένα για κάθε φάση του project.Γενικά χρησιμοποιήσα τις έτοιμες υλοποιήσεις των συναρτήσεων *readIntArrayFromDisk και* write*IntArrayFromDisk από το πακέτο Utils*

1ο Μέρος : Κατασκευή αρχείου Ν αριθμών

Στη πρώτη φάση του project η κλάση (**WriteNumbersBinaryFile**) δημιουργεί ένα αρχείο(jason_2017030021.dat) και εκχωρεί 100.000 τυχαίους ακέραιους αριθμούς οι οποίοι παράγονται από την συνάρτηση randInt σε αυτό χρησιμοποιώντας ένα πίνακα (buffer) των 1000 αριθμών προσομοιώνοντας έτσι τη λογική των data Pages του λειτουργικού συστήματος με την οποία διαβάζει αρχεία. Ακόμη μετράει τις προσβάσεις δίσκου που χρειάστηκαν να γίνουν για την επιτυχή εγγραφή των αριθμών στο αρχείο, ως πρόσβαση δίσκου θεωρώ κάθε ανάγνωση και εγγραφή στο δίσκο.

2ο Μέρος : Ταξινόμηση του αρχείου στον δίσκο

Στη δεύτερη φάση του project η κλάση (SortingFile) χωρίζει το μη ταξινομημένο αρχείο σε αρχεία με το ίδιο μέγεθος δηλαδή των 10.000 αριθμών το καθένα(40.000 bytes) και ταξινομεί το καθένα ξεχωριστά. Ειδικότερα διαβάζει 10000 αριθμούς απο το αρχείο διαβάζοντας 1000 κάθε φορά(μέγεθος data page), τους τοποθετει σε ενα πίνακα(Buffer) και χρησιμοποιώντας τη μέθοδο quickSort (https://www.baeldung.com/java-quicksort) τους ταξινομεί στη συνέχεια γράφει τον πίνακα στο δίσκο ανα 1000 αριθμούς(μέγεθος data page).Ακόμη για κάθε εγγραφή και ανάγνωση μιας data page(1000 αριθμούς) μετράει μία προσβάση δίσκου. Στη συνέχεια κάνουμε συγχώνευση και ταξινόμηση των αρχείων σε ένα τελικό αρχείο. Συγκεκριμένα δημιουργώ έναν 2D array 11x1000 (ListOfBuffers) και διαβάζω τους πρώτους 1000 αριθμούς απο τα αντίστοιχα αρχεία. Συγκρίνω τα πρώτα στοιχεία απο τους 10 πρώτους Buffers(ListOfBuffers[i][] i =0,..,9) με την **searchMinElement** και βαζω το μικρότερο στη πρώτη θέση του 11ου Buffer(ListOfBuffers[10]),επαναλαμβάνω την ίδια διαδικασία με την διαφορά οτι κάθε φορα που παίρνω ενα στοιχείο ως το μικρότερο απο την σύγκριση τοτε αυξάνω τον arrayHelpBuffer[i] κατά 1 . Ο τελευταίος είναι ο δείκτης που χρησιμοποιω για να συγκρίνω το αντίστοιχο στοιχείο των 10 Buffers. Τον πίνακα arrayHelpBuffer τον εχω ορίσει ως εναν integer πίνακα 10 θέσεων(όσα δηλαδη και τα αρχεια), τον εχω αρχικοποιήσει ίσο με 0 και τον χρησιμοποιώ για να γνωρίζω ποιο στοιχείο πρεπει να συγκρίνω απο καθε Buffer (ListOfBuffers) και πότε απο εναν Buffer (ListOfBuffers) έχω συγκρίνει ολα του τα στοιχεία.Ακόμη όταν ο εκάστοτε arrayHelpBuffer ισούται με 1000 τότε σημαίνει οτι πρέπει να ξαναδιαβάσω 1000 αριθμούς απο το αντίστοιχο αρχείο.Επιπλέον οταν διαβάζω 1000 νέους αριθμούς απο το αρχείο μετράω μια πρόσβση δίσκου και αυξάνω τον αντίστοιχο Buffer(secondHelpBuffer) κατα 1 . Ο τελευταίος είναι ένας πίνακας 10 θέσεων(όσα δηλαδή και τα αρχεία) και μετράει πόσες φορές εχω διαβάσει νέους αριθμούς από το αρχείο όταν φτάνει ο εκάστοτε το 10 τότε διαγράφω το αρχείο γιατί δε μου χρειάζεται πια.Τέλος χρησιμοποίω έναν integer fullBuffer για να μετράω ποτε ο Buffer

(listOfBuffers[11][]) έχει γεμίσει ώστε να τον γράψω στο τελικο ταξινομημένο αρχείο όταν δηλαδή το fullBuffer είναι ίσο με 1000 . Ακόμη μετράει τις προσβάσεις δίσκου που χρειάστηκαν για να ταξινομήσω το αρχείο(FinalSorting_Jason_2017030021.dat), ως πρόσβαση δίσκου θεωρώ κάθε ανάγνωση και εγγραφή στο δίσκο.

Μέρος 3ο: Αναζήτηση στο ταξινομημένο αρχείο

Στη τρίτη φάση του project στην κλάση (SearchFile) έχω δημιουργήσει δύο μεθόδους μια για σειριακή αναζήτηση στο αρχείο(sequentialSearchFile) και μια για δυαδική αναζήτηση στο αρχείο(binarySearchFile).Γενικά στην κλάση κάνω 20 αναζητήσεις για τιμές κλειδιών που υπάρχουν στο αρχείο και 20 για τίμες κλειδιών που δεν υπάρχουν.Ειδικότερα για να διασφαλισω οτι όντως τα 20 στοίχεια υπάρχουν παίρνω τους πρώτους 20 αριθμούς απο το μη ταξινομημένο αρχείο, ενω για τις τιμές κλειδιών που δεν υπάρχουν παράγω 20 τυχαίους αριθμούς με την μέθοδο randInt για εύρος τιμών εκτος του ορίου που έχω θέσει για τους αριθμούς που εκχωρούσα όταν δημιουργούσα το αρχείο.Οσον αφορά για τις μεθόδους για σειριακήκαι δυαδική αναζήτηση στο αρχειο, στην σειριακή διαβάζω κάθε φορά 1000 αριθμους απο το αρχείο και καλώ τη μέθοδο linearSearch(http://www.codenuclear.com/linear-or-sequential-search-algorithm/) η οποια κάνει σειριακή αναζήτηση στο πίνακα(Buffer) και επαναλαμβάνω την ίδια διαδικασία 100 φορές(όσα και τα data pages).Για την δυαδική αναζήτηση στο αρχείο διαβάζω αρχικά το μεσαίο buffer του αρχειου(θέση 50.000) κάνοντας binarySearch στο πίνακα με την συνάρτηση binarySearch(https://www.geeksforgeeks.org/binary-search/) και αν δεν το βρω τότε αν ο αριθμός είναι μικρότερος από τον μικρότερο αριθμό του buffer, τότε διαβάζω το αριστερό μισού του αρχείου ενώ αν ο αριθμός είναι μεγαλύτερος από τον μεγαλύτερο αριθμό του Buffer διαβάζω το δεξί μέρος του αρχείου.Επαναλαμβάνω τη διαδικασία εώς του το βρώ αν δεν το βρώ τότε έχω αποτυχημένη αναζήτηση. Ακόμη μετράει τις προσβάσεις δίσκου που χρειάστηκαν για να βρω το στοιχείο ή για να δω ότι δεν υπήρχε, ως πρόσβαση δίσκου θεωρώ κάθε ανάγνωση και εγγραφή στο δίσκο.

Μέθοδος	Κατασκεύη Αρχείου	Ταξινόμηση Αρχείου	Σειριακή Αναζήτηση(Επιτυχημένη)	Σειριακή Αναζήτηση(Αποτυχημένη)	Δυαδική Αναζήτηση(Επιτυχημένη)	Δυαδική Αναζήτηση(Αποτυχημένη)
Απόδοση	100 προσβάσεις δίσκου	400 προσβάσεις δίσκου	47 προσβάσεις δίσκου	100 προσβάσεις δίσκου	5 προσβάσεις δίσκου	7 προσβάσεις δίσκου

Για την κατασκευή του αρχείου χρειαστήκαμε 100 προσβάσεις γιατί γράφώ 100000 αριθμούς ανα buffer των 1000 αριθμών άρα ήταν αναμενόμενο.Για την ταξινόμηση χρειαστήκαν 400 προσβάσεις συνολικα διότι χρειάζονται 200 προσβάσεις για το “σπάσιμο” του αρχικού αρχειου σε δέκα ταξινομημένα αρχεία και άλλες 200 για τη συγχώνευση των αρχείων και τη ταξινόμηση τους στο τελικό.Στη σειριακή αναζήτηση όταν είναι επιτυχημένη τότε χρειάζεται 47 προσβάσεις διότι άθροισα τις προσβάσεις που χρειάστηκαν για την έυρεση και των 20 στοιχείων και μετα το διαίρεσα με το 20, ενώ για την αποτυχημένη χρειάζεται 100 προσβάσεις επειδή η πολυπλοκότητα ειναι O(N) και αφού

το N = 100.000/1.000 = 100 στη χειρότερη θα χρειαστούν 100 προσβάσεις.Στη δυαδικη αναζήτηση όταν είναι επιτυχημένη τότε χρειάζεται 5 προσβάσεις διότι άθροισα τις προσβάσεις που χρειάστηκαν για την έυρεση και των 20 στοιχείων και μετα το διαίρεσα με το 20,ενώ για την αποτυχημένη χρειάζεται 7 αφου η πολυπλοκότητα της δυαδικης ειναι logN με βάση το 2 που είναι ίσο με 6.8 περίπου για N = 100.000/1.000 = 100.