Αναφορά 1 Άσκησης στις Δομές Δεδομένων και Αρχείων

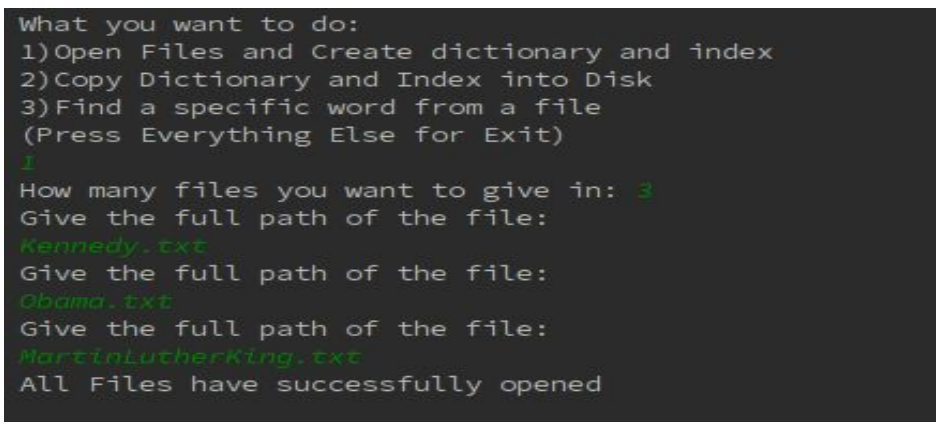
Η πρώτη εργασία των δομών δεδομένων και αρχείων είχε σκοπό να μας εξοικειώσει με την επεξεργασία Αρχείων. Συγκεκριμένα σημεία τα οποία έπρεπε να δώσουμε έμφαση για την διεκπεραίωση της είναι, η ικανότητα προσπέλασης ήδη υπαρχόντων αρχείων και η μετάβαση στην κεντρική μνήμη πληροφοριών από αυτά. Η σωστή διαχείριση πληροφοριών με την χρήση σωστών δομών και μεθόδων επεξεργασίας δομών παίρνοντας υπόψη την πολυπλοκότητα του κάθε ενός και επιλέγοντας τον καλύτερο σε απόδοση. Και τέλος την έγγραφή και ανάγνωση σε αρχεία, και πιο συγκεκριμένα σε σελίδες αυτών, ώστε να μην χρειάζεται να χάνουμε χρόνο καθώς ο δίσκος είναι σημαντικά πιο αργός από την κεντρική μνήμη.

Κατασκευή της δομής δεδομένων στην Κεντρική Μνήμη

Σε αυτό το κομμάτι χρειάστηκε να διαβάσουμε ένα πλήθος αρχείων το οποίο εξαρτάται από τον χρήστη (καθώς ο ίδιος επιλέγει πόσα αρχεία πρόκειται να δώσει), να τα ανοίξει και να δημιουργήσει ένα Array το οποίο θα περιέχει το λεξικό και παρόμοια μία List η οποία θα περιέχει το ευρετήριο. Για να γίνει το συγκεκριμένο κομμάτι δημιουργήσαμε μία κλάση, την FilesOpen, η οποία υλοποιεί τις εξής βασικές μεθόδους FilesToStructs() και DeleteUseless(). Πιο συγκεκριμένα μέσα από την μέθοδο FilesToStruct() ανοίγουν τα επιλεγμένα αρχεία, τα οποία ο χρήστης έχει δώσει, και διαβάζονται μία μία λέξη διαδοχικά σαν String. Αφού καθαριστεί ανάλογα το String με την χρήση της DeleteUseless() (αν πρόκειται να χρησιμοποιηθεί για το λεξικό όλοι οι ειδικοί χαρακτήρες γίνονται “” ενώ για το ευρετήριο “ ” ώστε να κρατηθούν οι αποστάσεις σωστά) αποθηκεύεται σε 2 λίστες (μία για την δημιουργία του λεξικού και μία για του ευρετηρίου).

Για την δημιουργία του Array του λεξικού χρησιμοποιήθηκε η λίστα η οποία προοριζόταν για το λεξικό, και αφού διαγράφηκαν όλοι οι χαρακτήρες οι οποίοι υπάρχουν επανειλημμένα με την χρήση της κλάσης HashSet ταξινομήθηκε χωρίς να υπάρχει case sensitivity με την χρήση της κλάσης Collection τέλος δημιουργήσαμε από την ταξινομημένη λίστα που περιλαμβανόταν μία μόνο φορά το κάθε στοιχείο της, τον Array τον οποίο χρειαζόμασταν τελικά.

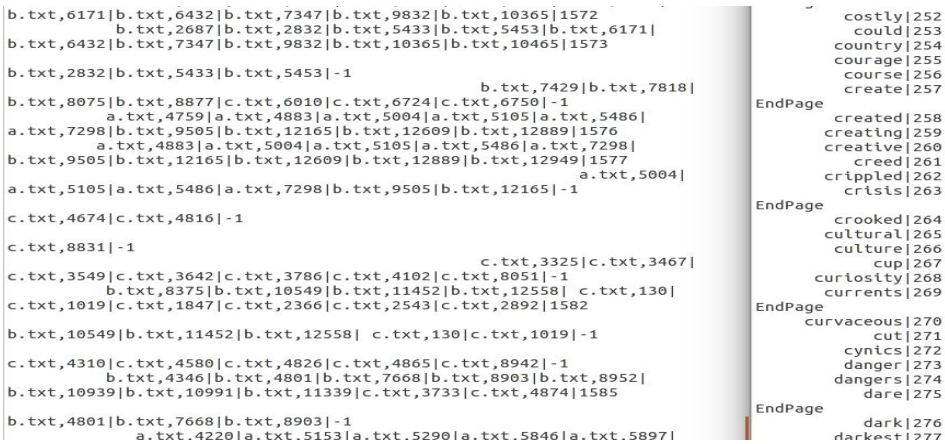
Για την δημιουργία της λίστας χρησιμοποιήσαμε την λίστα από Strings την οποία είχε δημιουργηθεί για αυτή την δουλειά και προσπελάσαμε κάθε στοιχείο της ξεχωριστά ελέγχοντας το στο λεξικό που ήδη είχε δημιουργηθεί ώστε να βρούμε σε ποια λέξη του λεξικού αναφέρεται το String της λίστας μας, ώστε σε ένα παράλληλο πίνακα από Strings να αποθηκεύουμε την ακριβή πληροφορία για το αρχείο το οποίο αναφερόμαστε αρχικά και για το μέγεθος σε byte από την αρχή αυτού του αρχείου όπου βρίσκεται η λέξη, ή οποίαλόγω του τρόπου που έχει δομηθεί το πρόβλημα είναι ίση με το μέγεθος των προηγούμενων κόμβων στη λίστα μας σε byte. Τέλος το μόνο που χρειάστηκε ήταν ο παράλληλος Array να μετατραπεί στην απαιτούμενη λίστα.



Κατασκευή Δομής Αρχείου

Για το συγκεκριμένο κομμάτι χρειάστηκε να δημιουργηθεί μία κλάση Dictionary και μία Index, όπου και οι δύο περιλαμβάνουν μέσα τις μεθόδους WriteIntoFile και ReadFromFile. Σε αυτό το κομμάτι θα μας απασχολήσει η μέθοδος WriteIntoFIle. Για να καταφέρουμε να γράψουμε το Dictionary χρειάστηκε να πάρουμε διαδοχικά ένα-ένα τα στοιχεία του, να τα μετατρέψουμε σε String, να μετατρέψουμε το String σε συγκεκριμένο format των 20 byte (16byte για το max word και 4 για int) των κάθε ένα και παίρνοντας 6 κάθε φορά από αυτά να τα περάσουμε σε ένα buffer το οποίο μετατρέπεται έπειτα σε byte array των 128 byte σύνολο ο οποίος με την χρήση της RanfomAccessFile έπειτα αποθηκεύεται σε συγκεκριμένο page του Dictionary. Αυτή η διαδικασία επαναλαμβάνεται μέχρι να φτάσουμε και στο τελευταίο page.

Έπειτα για την δημιουργία του Index αρχείου ακολουθήθηκε παρόμοια διαδικασία με μερικές εξαιρέσεις λόγο των πολλών εγγραφών όπου μπορεί να υπάρχουν σε κάποιες σελίδες. Για τον συγκεκριμένο λόγο, αρχικά ακολουθήθηκε η ίδια διαδικασία με το Dictionary για το μέγεθος των στοιχείων τα οποία αυτό έχει. Αυτό σημαίνει ότι περνούσε ένα ένα τα στοιχεία του Index μέχρι να φτάσει σε 10 από αυτά ανά σελίδα(Λόγω του ότι το μέγεθος του κάθε ενός είναι 12byte αρα στον buffer θα χωρούσαν συνολικά μόνο 10 και όσα περισσέψουν αν περισσέψουν τα περνάει σε μία λίστα όπου κάθε κόμβος της θα έχει όλες τις θέσεις που εμφανίζεται η λέξη για την οποία αναφερόμαστε, η οποία θα χρησιμοποιηθεί σε δεύτερη φάση, ενώ αν είναι λιγότερο από 10 απλά το κάνει formart ώστε να έχει σταθερά συγκεκριμένο αριθμό byte). Το αρχείο θα περιέχει σε πρώτη φάση σελίδες ίσες με τον αριθμό των λέξεων των οποίων υπάρχουν στο λεξικό και θα αποθηκεύει όλες τις απαραίτητες πληροφορίες σε κάθε σελίδα με χρήση buffer (128 bytes σύνολο) και έπειτα αν αυτές φτάνουν μόνο σε μία σελίδα στο τέλος της σελίδας την τιμή -1, σε αντίθετη περίπτωση θα αποθηκευτεί στο τέλος ο αριθμός της σελίδα στην οποία υπάρχει η συνέχεια αυτής κάθε φορά, παίρνοντας υπόψιν πόσες σελίδες θα χρειαστούν για να αποθηκευτούν τα αντικείμενα που προηγούνται από αυτά. Παράδειγμα, αν έχουμε εισάγει μόνο ένα αρχείο με 20 λέξεις στο λεξικό και οι λέξεις που βρίσκονται στις θέσεις 3, 7 και 9 θα χρειαστούν πάνω από μία σελίδα(συγκεκριμένα για την 3 θα χρειαστούν άλλες 3 σελίδες, για την 7 άλλη 1 και για την 9 άλλες 2), σε πρώτη φάση θα αποθηκεύσει σε 20 σελίδες τις βασικές πληροφορίες μας με τις τιμές -1 σε όλες εκτός από τις σελίδες 3,7 και 9 και στο τέλος της σελίδα 3 θα αποθηκεύσει ότι υπάρχει η συνέχεια της πληροφορίας στην σελίδα 21 ενώ παίρνοντας υπόψιν πόσες σελίδες θα χρειαστούν ακόμα για αυτή την λέξη(δηλαδή 3) αποθηκεύει στο τέλος της 7 σύμφωνα με αυτό σε ποια σελίδα θα είναι η συνέχεια της(δηλαδή την τιμή 24 αφού η προηγούμενη θα ξεκινήσει από την 21 και θα χρειαστεί 3 σελίδες). Όμοια θα αποθηκευτεί στην 9 η τιμή 25(αφού η προηγούμενη ξεκινάει από την 24 και θα χρειαστεί ακόμα μία σελίδα). Έπειτα αφού πλέον θα είναι διαδοχικά από την 21 και μετά οι πληροφορίες μας επαναλαμβάνει το ίδιο βήμα μέχρι να δει ότι είμαστε στην τελευταία σελίδα για κάθε λέξη και να βάλει στο τέλος της την τιμή -1. Αντίθετα αποθηκεύετε απλά η επόμενη σελίδα η οποία έχει ήδη κρατηθεί ώστε να συνεχίσει σε αυτήν η πληροφορία.

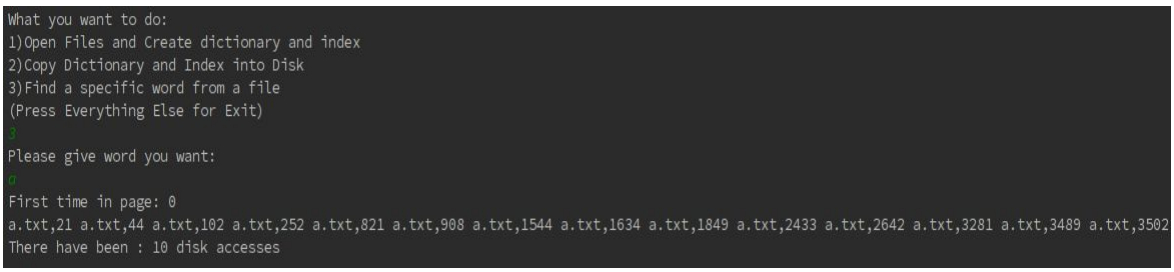


Αριστερά το Index και δεξιά το Dictionary αν έχουν αποθηκευτεί όμως σαν String στο δίσκο

Αναζήτηση στη δομή δεδομένων

Για αυτό το κομμάτι στο πρώτο σκέλος χρειάστηκε να υλοποιηθεί ένας διάδικος αλγόριθμος αναζήτησης ο οποίος θα έπαιρνε την μεσαία σελίδα αφού πρώτα την διάβαζε από το αρχείο και ανάλογα αν υπήρχε σε αυτή η λέξη η οποία έδωσε ο χρήστης, σταματούσε παρουσιάζοντας όλα τα σημεία στα οποία βρισκόταν η λέξη, ενώ σε αντίθετη περίπτωση και ανάλογα με το που βρισκόταν στο ταξινομημένο Dictionary έπαιρνε είτε από τα αριστερά είτε από τα δεξιά την απαιτούμενη σελίδα μέχρι να βρεθεί η δοσμένη λέξη. Σε κάθε περίπτωση ελέγχετε αν η λέξη που έδωσε ο χρήστης υπάρχει ή όχι στο λεξικό. Και θα μας επιστρέψει θετικό αριθμό, την σελίδα που βρέθηκε, αν αυτή βρεθεί ενώ σε αντίθετη περίπτωση αρνητικό. Από την στιγμή που χρησιμοποιήσαμε δυαδική αναζήτηση σημαίνει ότι η πολυπλοκότητας μας θα πήγαινε σε log(n) πράγμα το οποίο μας είναι πολύ χρήσιμο ειδικά όταν έχουμε ανάγνωση από δίσκο που η ταχύτατα του μπορεί να μας επηρεάσει.

Για το δεύτερο κομμάτι στην κλάση Index μέσω της μεθόδου ReadFromFile θα παίρνει σαν όρισμα το page στο οποίο εμφανίζεται η απαιτούμενη λέξη, θα το διαβάζει μέχρι να φτάσει στο τελευταίο όρισμα του το οποίο έχουμε βάλει εμείς, αν είναι -1 σταματάει ενώ σε αντίθετη περίπτωση ξανά διαβάζει την σελίδα αυτή την φορά που υπάρχει σαν τελευταίο όρισμα μέχρι να εμφανιστεί ο αριθμός -1 σαν τελευταίο όρισμα μας στην επόμενη σελίδα όπου θα διαβαστεί. Και επιστρέφει ένα ακέραιο για το πόσες σελίδες επισκέφτηκε, τον οποίο τον προσθέτουμε με αυτόν που έχουμε από την μεταβλητή που αναφέρεται στο ίδιο κομμάτι από το προηγούμενο ερώτημα για να υπολογίσουμε τον αριθμό των προσβάσεων που είχαμε συνολικά στον δίσκο.



Παράδειγμα αν έχουμε ανοίξει μόνο ένα αρχείο από όπου ζητάμε την λέξη ‘a’
όπου υπάρχουν πολλές εγγραφές(πάνω από 10 σύνολο)

Για να εκτελεστεί ο κώδικας χρειάζεται να εκτελεστούν με την σειρά τα Steps στον κώδικα από το Menu ώστε να κάνει create τα Αναγκαία Files, αφού δημιουργηθούν μπορεί να εκτελεστεί και μόνο το 3

Βιβλιογραφία-Παραπομπές

Για την διεκπεραίωση της άσκησης χρησιμοποιήθηκαν οι ακόλουθες παραπομπές

- Σημειώσεις του μαθήματος στο Courses
- “Απόλυτη java”,Savitch, Κεφάλαιο 10, “Είσοδος-Έξοδος Αρχείων”
- https://docs.oracle.com/javase/7/docs/api/java/util/HashSet.html Σχετικά με την HashSet
- http://beginnersbook.com/2013/12/hashset-class-in-java-with-example/ Σχετικά με την HashSet
- https://docs.oracle.com/javase/7/docs/api/java/io/RandomAccessFile.html Σχετικά με την RandomAccessFile
- http://tutorials.jenkov.com/java-io/randomaccessfile.html Σχετικά με την RandomAccessFile