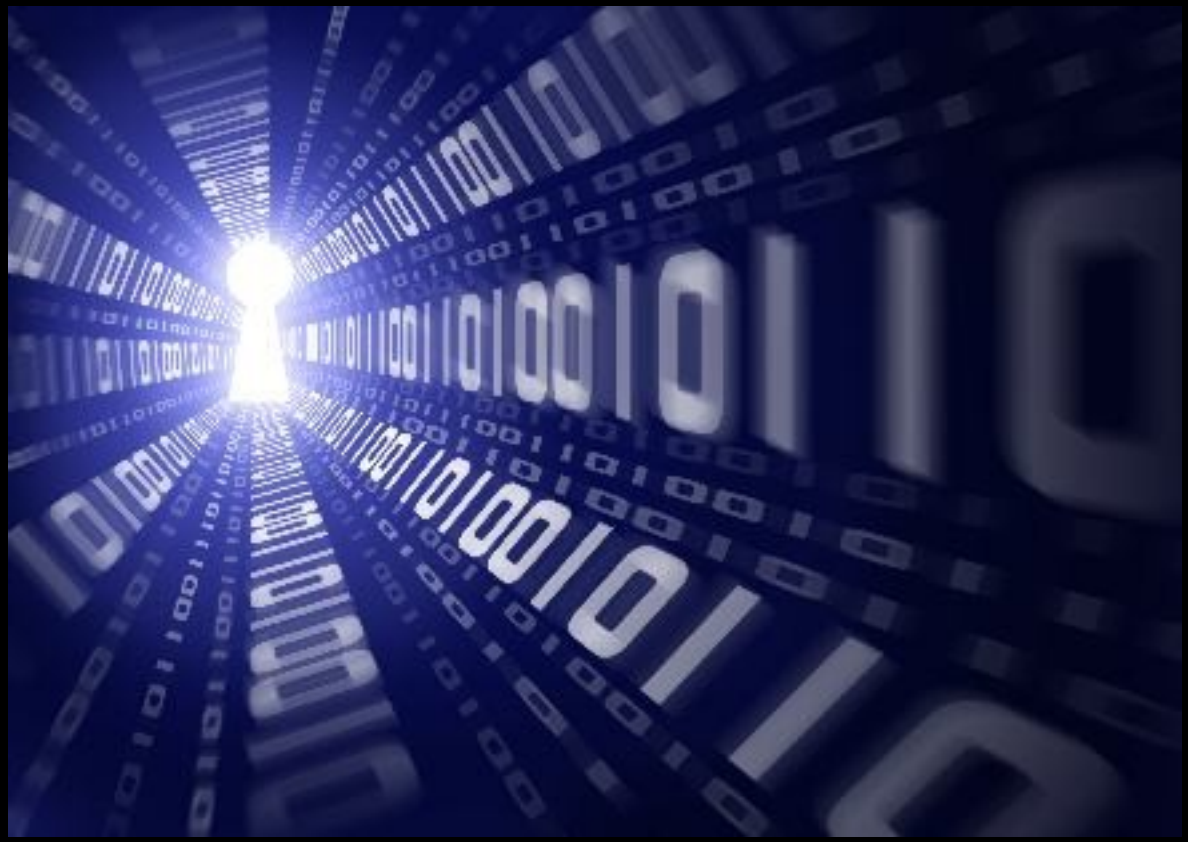
ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ
Εργασία 3



ΜΑΓΚΟΣ ΡΑΦΑΗΛ-ΓΕΩΡΓΙΟΣ 3100098

ΜΠΟΓΔΑΝΟΣ ΜΙΧΑΗΛ 3100123

Τάξεις που χρησιμοποιούνται και στις 2 υλοποιήσεις:

1)Τάξη WordFreq

Μεταβλητές:

-Private int freq: Συχνότητα εμφάνισης της λέξης του συγκεκριμένου αντικειμένου.

-Private String word: Η λέξη που περιέχει το αντικείμενο.

Κατασκευαστές:

- WordFreq(String word,int freq) : Δημιουργεί ένα αντικείμενο WordFreq αναθέτοντας στις αντίστοιχες μεταβλητές τα αντίστοιχα ορίσματα.
- WordFreq(String word) : Δημιουργεί ένα αντικείμενο WordFreq κάνοντας χρήση του παραπάνω κατασκευαστή με ορίσματα word και 1 (τοποθετούμε την τιμή 1 διότι κάθε αντικείμενο που θα κατασκευαστεί σημαίνει ότι εμφανίζεται τουλάχιστον 1 φορές).

Μέθοδοι:

-public String getWord() : Επιστρέφει την μεταβλητή word του αντικειμένου.

-public int getFreq() : Επιστρέφει την μεταβλητή freq του αντικειμένου.

-public void increase() : Αυξάνει την μεταβλητή freq κατά 1.

-public String toString() : Υπερφορτώνει την default μέθοδο toString() ώστε να τυπώνει κατάλληλα τις πληροφορίες που περιέχονται στο αντικείμενο.

2)Τάξη WordComparator: Συγκρίνει δύο αντικείμενα WordFreq ως προς την λέξη που περιέχουν.Η σύγκριση γίνεται με την μέθοδο compareTo(String arg0) που επιστρέφει αρνητικό αριθμό εάν το this αντικείμενο προηγείται λεξικογραφικά του arg0,θετικό αριθμό εάν προηγείται το arg0 και μηδέν εάν είναι ίσα.Εάν τα προς σύγκριση αντικείμενα δεν είναι τύπου WordFreq,το πρόγραμμα τερματίζεται αφού πρώτα εμφανίσει μήνυμα λάθους.

3)Τάξη FreqComparator: Συγκρίνει δύο αντικείμενα WordFreq ως προς την συχνότητα εμφάνισης που περιέχουν.Η μέθοδος επιστρέφει -1 εάν η συχνότητα του πρώτου ορίσματος είναι μικρότερη ή ίση του δεύτερου,ενώ αλλιώς επιστρέφει 1.Εάν τα προς σύγκριση αντικείμενα δεν είναι τύπου WordFreq,το πρόγραμμα τερματίζεται αφού πρώτα εμφανίσει μήνυμα λάθους.

4)Τάξη LinkedList: Η λίστα του εργαστηρίου με τροποποιήσεις.

Προσθέθηκαν οι μέθοδοι :

-public Node<T> MergeSort(Node<T> c,Comparator<T> cmp) : Μέθοδος ταξινόμησης που υλοποιεί την mergesort σύμφωνα με το πρόγραμμα του βιβλίου(σελ.392-393) με μόνη τροποποίηση ότι δέχεται όρισμα έναν Comparator.

-public boolean constains(T val) : Επιστρέφει true εάν υπάρχει το όρισμα στην λίστα,αλλιώς επιστρέφει false.

-T removeSelected(T val) : αφαιρεί και επιστρέφει το όρισμα εάν υπάρχει στην λίστα.

-public void print(PrintStream ps) : Τροποποίση της μεθόδου ώστε να δέχεται ως όρισμα ένα PrintStream για να τυπώνει τα περιεχόμενα της λίστας.

ΜΕΤΡΗΤΗΣ ΛΕΞΕΩΝ ΜΕ Δ.Δ.Α.

1)Τάξη BinarySearchTree : Έγινε χρήση του δέντρου του εργαστηρίου με μερικές τροποποιήσεις για τις ανάγκες των insert και search οι οποίες και επεξηγούνται ακολούθως:

-Private μεταβλητές int sum,WordFreq max,WordFreq min,boolean insertion που περιέχουν τα πεδία άθροισμα συχνοτήτων ,μέγιστο (κατα συχνότητα) αντικείμενο WordFreq, ελάχιστο (κατα συχνότητα) αντικείμενο WordFreq,εάν true εισαγωγή στην ρίζα αλλιώς εισαγωγή ως φύλλο .Τα παραπάνω πεδία αρχικοποιούνται με 0,null,null,false αντίστοιχα στον κατασκευαστή.

(Σημείωση: Η μεταβλητή sum συμπίπτει και με τον συνολικό αριθμό λέξεων του αρχείου.)

-Μέθοδοι WordFreq getMin() και getMax(): Επιστρέφει το αντικείμενο με την ελάχιστη/μέγιστη τιμή συχνότητας.

-Μέθοδος double calculateMean(): διαιρεί την μεταβλητή sum(άθροισμα συχνοτήτων εμφάνισης) με τον συνολικό αριθμό των μοναδικών λέξεων στο δέντρο(μεταβλητή size του δέντρου).

-Μέθοδος void changeInsertion(boolean val): αλλάζει την boolean μεταβλητή insertion ώστε να αλλάξει ο τρόπος εισαγωγής(ως ρίζα ή ως δέντρο).

-Μέθοδος WordFreq search(WordFreq element): Αναζητά στο ΔΔΑ το όρισμα.Εάν δεν βρεθεί επιστρέφει null.Εάν βρεθεί,έχει προστεθεί έλεγχος ώστε εάν η συχνότητα του αντικειμένου που βρέθηκε είναι μεγαλύτερη της μέσης συχνότητας να γίνεται αφαίρεση του συγκεκριμένου κόμβου και εκ νέου εισαγωγή του στην ρίζα(υπάρχει πρόβλεψη ώστε να αλλάζει ο τρόπος εισαγωγής κατά την κλήση της search).Εάν βρεθεί αλλά δεν ικανοποιεί την συνθήκη με την μέση συχνότητα τότε απλά επιστρέφουμε το αντικείμενο που βρέθηκε.

-Μέθοδος void insert(WordFreq element): Εισάγει στο ΔΔΑ το όρισμα είτε στην ρίζα είτε ως φύλλο(με βάση την boolean insertion).Εάν η insertion είναι true τότε καλείται η εισαγωγή στην ρίζα.Αλλιώς, το στοιχείο εισάγεται ως φύλλο.Οι αλλαγές στη μέθοδο βρίσκονται στον υπολογισμό του μεγίστου/ελαχίστου αλλά και του sum.Για τον υπολογισμό του μεγίστου και του ελαχίστου προσθέτουμε κατάλληλες συγκρίσεις στα σημεία που η μέθοδος προσθέτει το στοιχείο.Ομοίως και για την ανανέωση του sum τοποθετούμε κατάλληλα αθροίσματα.

-Μέθοδος void clearStats(): Απομακρύνει τις τιμές των στατιστικών για μελλοντική χρήση της δομής.

2)Τάξη BinaryTreeBasedWordCounter:

Μεταβλητές:

-Private LinkedList<String> stopWords: Συνδεδεμένη λίστα που κρατά τις stopWords.

-Private BinarySearchTree bst: Το ΔΔΑ που θα κρατά τις λέξεις.

Κατασκευαστής:

Υπάρχει ένας κατασκευαστής που αρχικοποιεί την λίστα και το ΔΔΑ.Στο τελευταίο χρησιμοποιούμε για την κατασκευή του ως όρισμα την κλάση WordComparator(λεξικογραφική σύγκριση).

Μέθοδοι:

-public void addStopWord(String word) : Προσθέτει στην λίστα το όρισμα.

-public void removeStopWord(String word) : Αφαιρεί από την λίστα το όρισμα,κάνοντας χρήση της μεθόδου removeSelected της λίστας.Εάν η λέξη δεν υπάρχει τυπώνεται μήνυμα λάθους.

-public void load(String path) : Διαβάζει το αρχείο που δίνεται ως όρισμα.Βασική προυπόθεση είναι το αρχείο να είναι στο ίδιο directory με το πρόγραμμα.Διαβάζει κάθε γραμμή,αφαιρεί τους " χαρακτήρες,τα σημεία στίξης καθώς και τα πρόσθετα κενά(τα δύο τελευταία αφαιρούνται με χρήση regex).Έπειτα «κόβει» σε λέξεις την γραμμή με βάση τον χαρακτήρα κενού και ελέγχει εάν η λέξη δεν είναι αριθμός και δεν είναι stop word τότε προστίθεται στο ΔΔΑ.

-public int getTotalWords() : Επιστρέφει την μεταβλητή sum του ΔΔΑ.

-public int getDistinctWords() : Επιστρέφει το μέγεθος του ΔΔΑ(είναι οι μοναδικές λέξεις του κειμένου,αφού το κλειδί είναι η λέξη και τα duplicates δεν εισάγονται σε νέο κόμβο).

-public int getFrequency(String word) : Αρχικά αλλάζει ο τύπο εισαγωγής σε εισαγωγή στην ρίζα(από την search του ΔΔΑ).Εάν η αναζήτηση αποτύχει επιστρέφουμε 0,αλλιώς επιστρέφουμε την συχνότητα του αντικειμένου που βρέθηκε.

-public double getMeanFrequency() : Επιστρέφει την μέση συχνότητα εμφάνισης καλώντας την calculateMean του ΔΔΑ.

-public String getMaximumFrequency() : Επιστρέφει την μεταβλητή max του ΔΔΑ και καλει την toString σ'αυτό.

-public String getMinimumFrequency() : Επιστρέφει την μεταβλητή min του ΔΔΑ και καλει την toString σ'αυτό.

-public void printByFrequency(PrintStream ps) : Δημιουργεί ένα προσωρινό ΔΔΑ με όρισμα στον κατασκευαστή του ένα αντικείμενο FreqComparator(σύγκριση με βάση την συχνότητα) και τοποθετούμε τα στοιχεία του αρχικού ΔΔΑ στο προσωρινό ΔΔΑ.Καλούμε την clearStats στο βασικό ΔΔΑ,τοποθετούμε τα περιεχόμενα του προσωρινού ΔΔΑ με χρήση toString() (αναδρομική μέθοδος που συνεχώς πηγαίνει στον αριστερότερο κόμβο και ύστερα στους δεξιότερους) στο PrintStream και ξαναγεμίζουμε το αρχικό ΔΔΑ.

-public void printAlphabetically(PrintStream ps) : Τυπώνει το ΔΔΑ με χρήση της toString(),αφού στο ΔΔΑ έχουμε χρησιμοποιήσει WordComparator που όταν τοποθετεί τα δεδομένα τα συγκρίνει λεξικογραφικά και τοποθετεί είτε αριστερά(λεξικογραφικά μικρότερο του πατέρα) είτε δεξιά(λεξικογραφικά μεγαλύτερο του πατέρα).

ΥΛΟΠΟΙΗΣΗ ΜΕΤΡΗΤΗ ΜΕ HASHTABLE

Τάξη MyHashTable<K,V>:

-Εσωτερική ιδιωτική τάξη Node<V>που αναπαριστά τους κόμβους στον πίνακα κατακερματισμού.

-Private V val : μεταβλητή που θα περιέχει την τιμή του κόμβου.

-Private K key : μεταβλητή που θα περιέχει το κλειδί του κόμβου.

-Private Node<V> next : αναφορά στον επόμενο κόμβο(για υλοποίηση της χωριστής αλυσίδωσης,σε περίπτωση συγκρούσεων).

-Public Node(K k, V v) : κατασκευαστής που τοποθετεί τα κατάλληλα πεδία στις αντίστοιχες private μεταβλητές και θέτει την αναφορά του επόμενου κόμβου σε null.

-Public V getVal() : επιστρέφει την μεταβλητή val.

-Public void setVal(V v) : αναθέτει το όρισμα στην μεταβλητή val.

-private Node<V> []data : Πίνακας τύπου Node που κρατά τα στοιχεία.

-private int size : Μέγεθος του πίνακα χαρακτήρων.

-private int elements : Πλήθος των στοιχείων που βρίσκονται αποθηκευμένα στον πίνακα.

- public MyHashTable(int size) : Κατασκευαστής που παίρνει ως όρισμα το μέγεθος του πίνακα.Το όρισμα διαιρείται με το 5 και ακολούθως το τοποθετούμε στο πεδίο size της τάξης.Επίσης αρχικοποιούμε το πεδίο elements στο 0 και δημιουργούμε τον πίνακα data.

-Μέθοδος void clear(): Αδιάζει την δομή,θέτοντας null σε κάθε θέση του πίνακα data.

-Μέθοδος LinkedList<V> values() : αλλάζειΔιατρέχει τον πίνακα data και για κάθε κόμβο που συναντά,προσθέτει αυτόν αλλά και τους επόμενους του(μέχρι να συναντήσει null) σε μια συνδεδεμένη λίστα.

-Μέθοδος int hash(K key) : Επιστρέφει την απόλυτη τιμή της παράστασης key.hashCode()%size (η hashCode σε περίπτωση υπερχείλισης επιστρέφει αρνητικό αριθμό και για αυτό το λόγο τοποθετήθηκε απόλυτο).Προφανώς χρησιμοποιήθηκε αυτούσια η hashCode της java.

-Μέθοδος void put(K key, V value) : Τοποθετεί στον πίνακα το όρισμα value στην θέση που σχετίζεται με το κλειδί key.Εάν κάποιο από τα ορίσματα είναι null τότε παράγεται IllegalArgumentException.Αλλιώς,υπολογίζουμε την θέση του πίνακα με βάση το κλειδί και την συνάρτηση κατακερματισμού.Εάν η θέση που βρέθηκε περιέχει δεδομένα,τότε διατρέχουμε την λίστα στο συγκεκριμένο σημείο μέχρι να βρούμε τον κόμβο με το ζητούμενο κλειδί.Εάν βρήκαμε το κλειδί,τοποθετούμε την νέα τιμή value.Αλλιώς,σημαίνει ότι δεν υπάρχει το κλειδί οπότε τοποθετούμε το νέο κόμβο με το αντίστοιχο κλειδί και τιμή και επίσης θέτουμε τον επόμενο κόμβο τον πρώτο κόμβο στον πίνακα data(δηλαδή πλέον πρώτος κόμβος είναι ο νεος).Επίσης αυξάνουμε το elements κατα 1.

-Μέθοδος remove(K key) : Αφαιρεί το στοιχείο που σχετίζεται με το κλειδί key.Βρίσκουμε την θέση του πίνακα data από τον κατακερματισμό του κλειδιού.Εάν στη θέση αυτή ο κόμβος περιέχει το κλειδί τότε το αφαιρούμε,και στη θέση του βάζουμε τον κόμβο που δείχνει το πεδίο next του προς αφαίρεση κόμβου.Ακολούθως αφαιρούμε 1 από το elements.Εάν το στοιχείο δεν είναι στην αρχή της λίστας τότε κρατάμε 2 αναφορές στον προηγούμενο και στον τρέχον κόμβο.Διασχίζουμε την λίστα τον κόμβων στη συγκεκριμένη θέση του πίνακα μέχρι είτε να φτάσουμε στο τέλος είτε να βρούμε το κλειδί.Στην πρώτη περίπτωση σημαίνει ανεπιτυχής αναζήτηση.Στην δεύτερη περίπτωση σημαίνει ότι βρήκαμε στο κλειδί οπότε αφαιρούμε τον ζητούμενο κόμβο που περιέχει το κλειδί(το πεδίο next του προηγούμενου κόμβου δείχνει στο next του τρέχον κόμβου) και μειώνουμε κατα 1 το elements.

-Μέθοδος boolean containsKey(K key) : Αρχικά βρίσκουμε με κατακερματισμό την θέση του πίνακα data. Εάν βρεθεί η θέση του πίνακα και δεν είναι κενή τότε ψάχνουμε τους κόμβους(ή τον κόμβο αν δεν έχουμε σύγκρουση στην συγκεκριμένη θέση) μέχρι να βρούμε το κλειδί.Σε περίπτωση που δεν βρεθεί το κλειδί ή η θέση του πίνακα είναι κενή τότε επιστρέφει false.Αλλιώς επιστρέφει true.

-Μέθοδος V getValue(K key) : Αρχικά βρίσκουμε με κατακερματισμό την θέση του πίνακα data. Εάν βρεθεί η θέση του πίνακα και δεν είναι κενή τότε ψάχνουμε τους κόμβους(ή τον κόμβο αν δεν έχουμε σύγκρουση στην συγκεκριμένη θέση) μέχρι να βρούμε το κλειδί.Σε περίπτωση που δεν βρεθεί το κλειδί ή η θέση του πίνακα είναι κενή τότε επιστρέφει null.Αλλιώς επιστρέφει την τιμή του κόμβου.

-Μέθοδος boolean isEmpty() : Εάν η μεταβλητή elements είναι 0 τότε επιστρέφουμε true,αλλιώς false.

-Μέθοδος int size() : Επιστρέφει την μεταβλητή elements.

Τάξη HashTableBasedWordCounter: Η λίστα των stopWords καθώς και οι μέθοδοι add/remove που την συνοδεύουν ορίζονται εντελώς όμοια με την τάξη BinarySearchTreeBasedWordCounter.

Μεταβλητές:

-Private int totalWords: Συνολικός αριθμός λέξεων του αρχείου.

-Private WordFreq min,max: Μεταβλητές που θα κρατάνε το ελάχιστον και το μέγιστο(με βάση τη συχνότητα εμφάνισης) αντικείμενο.

-Private MyHashTable<String,WordFreq> : Πίνακας κατακερματισμού στον οποίο θα τοποθετούνται τα αντικείμενα WordFreq.

Κατασκευαστής:

Υπάρχει ένας κατασκευαστής που αρχικοποιεί την λίστα και τον πίνακα κατακερματισμού με το ανάλογο μέγεθος.Επίσης αρχικοποιούμε το totalWords σε 0 και τα min/max σε null.

Μέθοδοι:

-public void load(String path) : Η μέθοδος είναι πανομοιότυπη με την αντίστοιχη load στην υλοποίηση με ΔΔΑ.Το σημείο που διαφέρει,είναι το σημείο εισαγωγής της λέξης όπου προστίθεται στον πίνακα κατακερματισμού εάν δεν υπάρχει ήδη και επίσης κάνουμε τις ανάλογες συγκρίσεις για εύρεση min και max καθώς και αυξάνουμε κατα 1 το totalWords.Εάν υπάρχει ήδη,τότε το εξάγουμε , χρησιμοποιούμε την μέθοδο increase του αντικειμένου ,αυξάνουμε το totalWords κατα 1 και τέλος κάνουμε τις απαραίτητες συγκρίσεις για τα min και max.

-public int getTotalWords() : Επιστρέφει την μεταβλητή totalWords.

-public int getDistinctWords() : Επιστρέφει το μέγεθος του πίνακα κατακερματισμού(είναι οι μοναδικές λέξεις του κειμένου,αφού το κλειδί είναι η λέξη και τα duplicates δεν εισάγονται σε νέο κόμβο).

-public int getFrequency(String word) : Γίνεται χρήση της μεθόδου get του MyHashTable και ακολούθως εάν βρέθηκε το αντικείμενο επιστρέφουμε την συχνότητα του,αλλιώς επιστρέφομε 0.

-public String getMaximumFrequency() : Επιστρέφει την μεταβλητή max καλούμε την toString σ’αυτό.

-public String getMinimumFrequency() : Επιστρέφει την μεταβλητή min και καλούμε την toString σ’αυτό.

-public double getMeanFrequency() : Επιστρέφει την μέση συχνότητα εμφάνισης διαιρώντας το totalWords με το size του MyHashTable.

-public void printByFrequency(PrintStream ps) : Καλούμε την μέθοδο values του MyHashTable και επιστρέφεται μια λίστα με όλα τα περιεχόμενα του πίνακα.Ακολούθως κάνουμε χρήση της μεθόδου mergesort της λίστας και με όρισμα την κεφαλή της λίστας και έναν FreqComparator προκύπτει μια ταξινομημένη λίστα με αύξουσα σειρά συχνότητας την οποία και τυπώνουμε.

-public void printAlphabetically(PrintStream ps)) : Ομοίως με την printByFrequency μόνο που δίνουμε ως όρισμα έναν WordComparator ώστε να προκύψει λεξικογραφικά ταξινομημένη λίστα.