Δομές Δεδομένων – 3η Εργασία

ΟΜΑΔΑ

Χρύσα Μαυράκη : 3130128

Ελευθέριος Χατζηαράπης : 3130225

Υλοποίηση Βασικών Μεθόδων:

WordFreq search (String w) : Δημιουργούμε μια private μέθοδο findRecursive η οποία επιστρέφει τον κόμβο αν η αναζήτηση είναι επιτυχείς ή null σε περίπτωση που η αναζήτηση ήταν ανεπιτυχείς. Τη μέθοδο αυτή τη χρησιμοποιεί η search για να μπορεί να έχει κόμβο τον οποίο ψάχνουμε.Όταν λοιπον το βρούμε συγκρίνουμε το πόσες φορές εμφανίζεται αυτή η λέξη με το μέσο όρο επαναλήψεων όλων των λέξεων.Αν είναι μεγαλύτερο απο το μέσο όρο τότε αφαιρεί το κόμβο χρησιμοποιώντας τη μέθοδο remove και εισάγει το στοιχείο χρησιμοποιώντας τη μέθοδο insert που εκτελεί εισαγωγή στη ρίζα.Αν είναι μικρότερο απλά επιστρέφει τον κόμβο τον οποίο βρήκε.

Void insert(WordFreq item)

Καλεί την βοηθητικη μέθοδο insertR για να κάνει εισαγωγή στη ρίζα.

void update(String w):Πρώτα αναζητάμε το κόμβο με το string w για να μας επιστρέψει τα στοιχεία του(το αντικείμενο του WordFreq).Αν η αναζήτηση μας επιστρέψει null(δε βρέθηκε κόμβος με στοιχείο w) τότε δημιουργεί ένα καινούργιο αντικείμενο τύπου WordFreq με ορίσματα (w,1),δηλαδή ένα string w με συχνότητα 1.Μετά το εισάγουμε στο δέντρο.Αν η αναζήτηση δε μας επιστρέψει null(αυτο σημαίνει πως βρήκε ενα κομβο με w) τότε αυξάνουμε τη συχνότητα του κόμβου για τη συγκεκριμένη λεξη κατα 1.

void remove(WordFreq item) : Δημιουργούμε μια private μέθοδο rem(String w,TreeNode temp) την οποία χρησιμοποιεί η remove για να αφαιρέσει το στοιχείο μέσα από το δέντρο.Άν το temp==null,τότε επιστρέφει.Άν το κλειδί του temp είναι μικρότερο από το w τότε temp = temp.r;(το temp ισουτε με το δεξι του κομβο) και μετά καλούμε πάλι τη rem(w,temp).Αυτό συμβαίνει γιατί έχουμε δυαδικό δέντρο οπότε τα στοιχεία του δεξιού υποδέντρου είναι μεγαλύτερα ενώ τα στοιχεία του αριστερού υποδέντρου ειναι μικρότερα.Αντίστοιχα αν το κλειδί του temp είναι μεγαλύτερο απο το w τοτε temp = temp.l; και κάλεσμα

της rem(w,temp).Όταν λοιπόν το το κλειδί είναι ίσο με το w ενωνει το αριστερό και το δεξί παιδί χρησιμοποιώντας τη μέθοδο joinLR(temp.l,temp.r) που θα δούμε πιο κάτω.

void load(String filename) : Φορτώνουμε το αρχείο filename για ανάγνωση.Χωρίζουμε το κείμενο όπου υπάρχει κενό, αγνοούμε τη λέξη αν περίεχει γράμματα και ύστερά καλούμε μια βοηθητική μέθοδο removepunctiation για να βγάλει τα σημεία στήξης. Ύστερά καλούμε την Update για να βάλει τις λέξεις στα σωστά σημεία του δέντρου.

int getTotalWords():Καλεί την μέθοδο countT(head) η οποία με αναδρομικό τρόπο κάνει μια διάσχιση στο δυαδικό δέντρο αναζήτησης επιστρέφοντας το συνολικό άθροισμα των συχνοτήτων.

int getDistinctWords():Επιστρέφει το σύνολο των ενεργών στοιχείων της κεφαλής(δηλαδή του πίνακα).Αυτο υποδεικνύει το συνολο των διαφορετικών λέξεων.

int getFrequency(String w):Αναζητάμε το το κόμβο με το στοιχείο του w και αποθηκεύουμε το αντικείμενο του. Αν το αντικείμενο που επέστρεψε είναι null τότε η μέθοδος

επιστρέφει 0.Σε άλλα περίπτωση επιστρέφει τη συχνότητα του.

Wordfreq getMaximumFrequency()

Καλούμε την traverseR με όρισμα την ρίζα.

double getmeanFrequency()

Υπολογίζει το άθροισμα όλων των εμφανίσεων(με τη χρήση της calcFrequency) απο τη ρίζα και το διαιρεί με το πλήθος των λέξεων.

Υλοποίηση Βοηθητικών Μεθόδων :

Private int calcFrequency(TreeNode h)

Αναδρομική συνάρτηση που υπολογίζει το άθροισμα όλων των frequency απ τον δοσμένο κόμβο και παρακάτω.

private TreeNode joinLR(TreeNode left,TreeNode right)

Αν δεν υπάρχει δεξί παιδί τότε επιστρέφει το αριστερό. Αν υπάρχει τότε θέτει το δεξί ίσο με την βοηθ. μέθοδο PartR με όρισμα 0. Έπειτα θέτει το αριστερό παιδί του δεξιού ίσο με το αρχικό αριστερό. Επιστρέφει το καινούριο right, το οποίο θα είναι η ρίζα.

Private TreeNode partR(TreeNode h, int k)

Αντικαθιστά τη ρίζα h με τον k-οστό (με αφετηρία το 0) μικρότερο κόμβο του υποδέντρου. Καλείται αναδρομικά από τον εαυτό της επιλέγοντας αν θα αναζητήσει τον κόμβο στο εκάστοτε αριστερό ή δεξί υποδέντρο. Έπειτα, εκτελεί τα απαιτούμενα rotations (αντίθετα με το υποδέντρο όπου έγινε η αναζήτηση, δηλαδή αριστερό υποδένρο -> δεξί rotation και αντίστροφρα). Εντέλει επιστρέφεται το k-οστό μικρότερο στοιχείο, το οποίο θα είναι και η συνολική ρίζα.

Private TreeNode RotR(TreeNode h)

Ορίζουμε ενα TreeNode x= h.l

Θέτουμε το h.l=x.r

Θέτουμε το x.r=h

Επιστρέφουμε το x;

Αντίστοιχα γίνεται και η αριστερή περιστροφή.

private boolean containsNumber(String word):Αναζητά στο word να βρει αν υπάρχει καποιος αριθμός.Σε περίπτωση που υπάρχει επιστρέφει true αλλιως false.

private String removePunctuation(String word):Ορίζει ένα string nWord και μετά μέσα σε ενα βρόχο εξετάζει ένα ένα τους χαρακτήρες του word.Σε περίπτωση που δεν είναι γραμμα ή δεν είναι απόστροφη τότε επαναλαμβάνει το βρόχο.Σε άλλη περίπτωση προσθέτει τους χαρακτήρες στο nWord.

private WordFreq findRecursive(TreeNode p,String element)

Αναδρομική μέθοδος αναζήτησης.

private int calcFrequency(TreeNode h)

Υπολογίζει αναδρομικά το άθροισμα της συχνότητας εμφάνισης για όλους τους παρακάτω κόμβους.

private TreeNode insertR(TreeNode h,Wodfreq x)

αναδρομική μέθοδος εισαγωγής όπου ελεγχει το κλειδί του x με το κλειδί του h αν το πρώτο είναι μικρότερο τότε καλεί αναδρομικά τον εαυτό της με ορίσμα h.l αλλιώς το h.r. Τελειώνει όταν το h==null

private WordFreq TraverseR(TreeNode h)

Αν ο κόμβος είναι Null επιστρέφουμε null.

Θέτουμε ως max το αντικείμενο του h.

Και αποθηκεύουμε τα MaxLeft,MaxRight καλώντας αναδρομικά την traverseR με κόμβους το h.l,h.r αντίστοιχα.Ύστερα συγρίνουμε το Maxleft και το MaxRight με το max και επιστρέφουμε το συνολικά μεγαλύτερο.