3η ΕΡΓΑΣΙΑ

ΤΖΕΝΗ ΜΠΟΛΕΝΑ 3170117

ΚΩΝΣΤΑΝΤΙΝΟΣ ΝΙΚΟΛΟΥΤΣΟΣ 3170122

- `void insert(WordFreq item):` **αρχικα για να δουμε αν θα γινει εισαγωγη ως φυλλο το αντικειμενο, τσεκαρουμε αρχικα αν ανηκει στα stop words, αν όχι τοτε καλουμε την insert recursive, και βρισκουμε που θα εισαχθει σαν φυλλο, κάθε φορα πηγαινουμε αριστερα ή δεξια του δεντρου , αναλογως αν το αντικειμενο προς εισαγωγη το key του είναι αντοιστοιχα μικροτερο ή μεγαλυτερο από την κεφαλη του υποδεντρου. Φτανουμε εως οτου να βρουμε την καταλληλη θεση για εισαγωγη του αντικειμενου. Στην εισαγωγη αυτη ξαναγραφουμε όλα τα κλειδια.**

- `void update(String w):` **αρχικα για να δουμε αν θα γινει update το αντικειμενο, τσεκαρουμε αρχικα αν ανηκει στα stop words, αν όχι τοτε καλουμε την update recursive, και βρισκουμε αν θα εισαχθει σαν φυλλο ή όχι(δλδ υπαρχει ηδη και απλα θα ενημερωσουμε τον αριθμο το φορων που υπαρχει). Αν βρουμε head ισο με Null μετα από τις αναζητησεις τοτε σημαινει ότι το αντικειμνο εισαγεται σαν φυλλο, κραταμε μια help variable ετσι ώστε αν γινει εισαγωγη σαν φυλλο, τοτε κάθε φορα όταν επιστρεφεται η recursive να ενημερωνουμε τα head ώστε να αυξανουν το number τους κατά 1, δλδ τον αριθμο τον υποδεντρων που εχουν από κατω. Αν τωρα το αντικειμενο(δλδ το string) εχει ηδη εισαχθει τοτε όταν το βρουμε απλως κανουμε update το ποσες φορες υπαρχει, δλδ τον αριθμο συνχνοτητας αυξανοντας το κατά 1. Επισης αμα το βρουμε κανουμε και έναν μικρο ελεγχο ώστε αν το αντικειμενο αυτό πλεον εχει μαγλυτερη συχνοτητα από το τρεχον αντικειμνο με μαγλυτερη συχνοτητα να ενημερωσουμε το maxFreq με αυτό το αντικειμενο.**

- `WordFreq search(String w):` **καλουμε την search recursive, αν δεν βρεθει το αντικειμενο με συτο το string τοτε απλως επιστρεφουμε null, αλλιως ψαχνουμε αριστερα ή δεξια του υποδεντρου για να βρουμε το αντικειμενο με αυτό το key, αν το βρουμε τοτε ελεγχουμε αν εχει συχνοτητα μεγαλυτερη της μεσης**

συχνοτητας ,αν ναι τοτε με καταλληλες περιστροφες το παμε στην κορυφη. Το σκεπτικο στις περιστροφες είναι ότι αν βρισκομαστε σε αριστερο υποδεντρο τοτε κανουμε δεξια περιστροδη και το δεξι υποδεντρου της νεας κεφαλης το βαζουμε κατω στα αριστερα του πλαιου head, και κανουμε head τη νεα κεφαλη. Αλλιως αν βρισκομαστε σε δεξι υποδεντρο κανουμε αριστερη περοστροφη και το αριστερο υποδεντρο της νεας κεφαλης το βαζουμε κατω δεξια από το παλιο head, και κανουμε και την αλλαγη στο head.

- `void remove(String w):` **αν το αντικειμενο που παμε να κανουμε remove υπαρχει(ελεγχουμε να υπαρχει με τροπο παρομοιο τις εισαγωγης σαν φυλλο, απλα χρησιμοποιοντας μια επαναληψη(while)). Αν υπαρχει το αντικειμενο τοτε καλουε την delete recursive και παμε να το αφαιρεσουμε. Το σκεπτικο με το οποιο υλοποιειεται η αφαιρεση είναι το εξης: πηγενουμε σε αριστρεο ή δεξι υποδεντρο αναλογως το κλειδι, και αφου το βρουμε εχουμε 3 περιπτωσης. Προτων αν το δεντρο είναι φυλλο τοτε απλα γινεται remove κανοντας το Null. ΑΝ τωρα το αντικειμενο αυτό εχει μονο αριστερο ή μονο δεξι παιδι τοτε , κανουμε τον πατερα του να διξει στο παδι του. Αν τωρα το αντικειμενο προς αφαιρεση εχει δυο παιδια τοτε παμε στο δεξι του υποδεντρο και βρισκουμε το μικροτερο στοιχειο. Το μικροτερο αυτό στοιχειο τωρα το βαζουμε να είναι εκει που είναι το αντικειμνο προς αφαιρεση, δλδ το αντικαθιστουμε εμ αυτό. Και επειτα αφου το αντικειμενο αυτό είναι και φυλλο, το διαγραφουμε. Κατά την διαρκεια αυτων τον αλλαγων κανουμε ενημερωνουμε και καταλληλα τα πεδια totalWords και το number των κομβων .**

- `void load(String filename):` **στην load εχουμε διαφορες καταστασεις η οποιες είναι true ή false αναλογως τι punctuations εχουμε βρει προηγουμενως. Οι καταστασεις αυτές μας βοηθανε για να γνωριζουμε τι πρεπει να κανουμε με κάθε string που διαβαζουμε. Ξεκιναμε λοιπον διαβαζουμε το αρχειο γραμμα γραμμα. Εχουμε 3 καταστασεις.**
  1)Αν εχουμε την κατασταση idleState που σημαινει ότι ψαχνουμε να βρουμε μια λεξη ή αριθμο, ετσι ώστε να παμε σε καποια άλλη κατασταη, αν τωρα βρισκουμε μονο κενα ή καποια σημεια στοιξης συνεχιζουμε στην ιδια κατασταση. Αν βρουμε αριθμο σημαινει ότι παμε σε μια κακη κατασταση badIdleState. ΑΝ τωρα βρουμε γραμμα ξεκιναμε την διαδικασια να χτυσουμε ένα string.

2) Η κατασταση badIlde , δλδ κατασταση που το string μας εχει αριθμους και πρεπει να το αγνωισουμε συνεχιζεται εως ωτου βρουμε καποιο σημειο στοιξης ή κενο, δλδ παμε στην καταστη οπου η διαδικασια να βρουμε μια λεξη ξεκιναει από την αρχη, παμε στην πρωτη κατσταση.
3) Αυτή είναι η κατασταση οπου ειμαστε σε διαδικασια και χτιζουμε ένα string(stringBuilderIdle). Αν βρισκουμε συνεχως γραμματα συνεχιζουμε και χτιζουμε το string μας εως οτου να βρουμε space ή punctuation mark. Αμα λοιπον βρουμε ένα απο αυτα τα δυο κανουμε update το string και παμε στην πρωτη κατασταση. Αν όμως βρουμε καποιον αριθμο πριν καν βρουμε space ή punctuation mark τοτε παμε σε μια κακη κατασταση(badIdelState).
Η διαδικασια αυτή συνεχιζεται μεχρι να φτασουμε στο τελος του αρχειου.
Για να κανουμε την διαδικασια της αναγνωσης πιο απλη χρησιμοποιυμε καποιες βοηθητικες μεθοδους(`isEnglishChar, isPunctuationChar, isNumber`) ετσι ώστε να γνωριζουμε τι ειδους χαρακτηρα εχουμε και σε ποια κατασταση θα παμε.

- `int getTotalWords():` **για να παρουμε τα total word απλως εχουμε μια μεταβλητη totalWords και κάθε φορα που κανουμε Insert , update ή remove την ενημερωνουμε καταλληλα.**

- `int getDistinctWords():` **σε κάθε κομβο εχουμε μια μεταβλητη number που κραταει το ποσοι κομβοι υπαρχουν από κατω του, οποτε το συνολο των κομβων(αρα και των διαφορετικων λεξεων) είναι οσοι είναι οι κομβοι κατω από το head (δηλαδη οσο το Number του head) και το head, δλδ είναι Number του head +1.**

- `int getFrequency(String w):` **για να βρουμε την συχνοτητα ενός string αρχικα ψαχνουμε αν υπαρχει το string κανοντας search, αν ναι τοτε τυπονουμε την συχνοτητα του μεσω της μεταβλητης times found που εχουν τα αντικειμενα τυπου wordFreq, αν δεν υπαρχει απλως επιστρεφουμε 0.**

- `WordFreq getMaximumFrequency():`**για να βρουμε το max WordFreq το κραταμε σε μια μεταβλητη , και κανουμε τις καταλληλες ενημερωσεις(αν**

χριαστει) όταν κανουμε insert , update, και remove, σε περιπτωση που remove γινει το αντικειμενο με το max frequency, τοτε ψαχνουμε να βρουμε το νέο αντικειμενο , φτιαχνοντας μια αναδρομικη μεθοδο updateMaxFrequency και το max frequency το ενημερωνουμε καταλληλα. Στην insert και στην Update, απλως ελεγχουμε αν το αντικειμενο που ισαχθηκε εχει μεγαλυτερο frequency(times found) από το τρεχωνν αντικειμενο με max frequency.

- `double getMeanFrequency():` για να βρουμε τη μεση συχνοτητα αρκει να διαιρεσουμε τα συνολικα words με το συνολο των διαφορετικων λεξεων. Το πως βρισκουμε το συνολο των λεξεων και το συνολο των διαφορετικων words εξηγηειται παραπανω, στις αντοιστοιχες μεθοδους.

- `void addStopWord(String w):` για να κανουμε add stop word καλουμε την μεθοδο add της κλασης List, εχουμε φτιαξει μια λιστα από nodes , και επειτα αν η λεξη αυτή υπαρχει ηδη στην λιστα τοτε δεν την εισαγουμε(εχουμε μια μεθοδο isInList), διαφορετικα αν δεν υπαρχει τοτε προσθετουμε στην κεφαλη της λιστας το stop word.

- `void removeStopWord(String w):` για να κανουμε remove stop word καλουμε την μεθοδο deleteString της κλασης List, και αφου τσεκαρουμε ότι η λεξη προς αφαιρεση υπαρχει στη λιστα, τοτε βρισκουμε και κραταμε σε μια βοηθητικη μεταβλητη το προηγουμενο του node προς αφαιρεση , και το κανουμε να δειξει στο επομενο του node προς αφαιρεση. Και καπως ετσι το node προς αφαιρεση πλεον δεν υπαρχει.

- `void printTreeAlphabetically(PrintStream stream):` επειδη εχουμε ένα δυαδικο δεντρο αρκει να κανουμε μια inorder διασχιση στο δεντρο και ετσι τα στοιχεια θα τυποθουν με αλφαβιτικη σειρα. Inorder σημαινει ότι πρωτα τυπονουμε αριστερο παιδι, μετα γονεα και μετα δεξι παιδι.

- `void printTreeByFrequency(PrintStream stream):` για να τυπωσουμε το δεντρο με βαση το frequency , δλδ τον αριθμο ενφανισεων τα

βαζουμε πρωτα όλα σε ένα πινακα wordfrequency με μεγεθος οσο τα distinct words κανοντας μια Inorder διασχηση. Επειτα τα ταξινομουμε χρησιμοποιοντας την μεθοσο meregeSort, την οποια εχουμε χρησιμοποιησει και στην δευτερη εργασια οποτε στο δευτερο pdf δινεται μια καλη εξηγησει για το πως λειτουργει η Mergost. Εχουμε στην κλαση WordFreq , την μεθοδο compareTo η οποια ελεγχει ποιο από δυο wordfreq εχει μεγαλυτερο αριθμο εμφανισης και ετσι επιστρεφονατς τιμες στην Merge sort, 0, 1 ή -1 γνωριζουμε ποια αντικειμενο εχει μεγαλυτερη συχνοτητα. Στην συνεχεια απλα τυπονουμε τα στοιχεια του πινακα που αποτελειται από wordFreq.