ΤΡΙΤΗ ΕΡΓΑΣΙΑ ΣΤΙΣ ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

ΜΠΙΛΙΑΣ ΓΕΩΡΓΙΟΣ 3200278

ΜΠΟΥΡΑΖΑΝΑΣ ΠΑΝΑΓΙΩΤΗΣ 3200268

Αρχικά, για την **void insert(String w)** ουσιαστικά πήραμε μια παράμετρο εάν το **head** είναι κενό να πηγαίνει και να φτιάχνει ένα αντικείμενο **WordFreq,** και να βάζει σαν **head** το νέο **TreeNode** με μέσα του αυτό, αλλιώς θέτει σαν **current** το **head** και μετά μέσα σε μια **while loop** κάνει **traveling** μέχρι να βρει την θέση οπού πρέπει να ενταχθεί το αντικείμενο. Εντωμεταξύ εκεί μέσα κάνει και **update** την **subtreeSize** καθώς και αυξάνει τον **counter** στα κατάλληλα σημεία. Μόλις, λοιπόν, βρίσκει το σημείο που πρέπει να ενταχθεί η λέξη δημιουργεί το νέο αντικείμενο **WordFreq** καθώς και το **treeNode** το οποίο θα ενταχθεί με μέσα του το **WordFreq**. Μέσα σε αυτό το **loop** κοιτάει και εάν υπάρχει αυτή η λέξη στο δέντρο, εάν κάτι τέτοιο ισχύει δεν την ξαναβάζει μέσα απλά αυξάνει την συχνότητα της (**frequency**) ανά 1.

Για την **WordFreq search(String w)** αρχικά θέσαμε το **head** ως **current** και μετά φτιάξαμε μια **while loop** οπού μέσα σε αυτή με την χρήση **if** είτε έβρισκε το αντικείμενο και το επέστρεφε, έκανε και την διαδικασία οπού εάν η συχνότητα του είναι μεγαλύτερη απτή μέση συχνότητα το βάζει σαν **head**, είτε επέστρεφε **null** επειδή δεν το βρήκε. Όσο αφορά την εισαγωγή σαν **head**, ουσιαστικά απλά κάναμε **remove** αυτό που βρήκαμε απτό δέντρο και με την χρήση της insertAsRoot το βάζαμε σαν **head**. Αξίζει να αναφερθεί πως και εδώ γίνονται update τα subtreeSizes μέσα στην **if** αυτή καθώς και αυξάνεται ο **counter** επειδή αρχικά μειώνεται στην **remove**.

Για την **void remove(String w)** βασιστήκαμε στην αντίστοιχη του εργαστήριου. Ουσιαστικά κάναμε **set** το **head** σαν **current** και θέταμε τον **parent** του σαν **null**, κάναμε την κατάλληλη αναδρομική διαδικασία ψάχνοντας να βρούμε αυτό που θέλουμε να κάνουμε **remove**. Άμα ήταν το **head** απλά το θέταμε σαν **head**, αλλιώς το βάζαμε στην κατάλληλη μεριά. Και πάλι κάναμε **update** το **subtreeSize** στα καταλληλά σημεία καθώς και μειώναμε τον **counter**.

Για την **void load(String filename)** αρχικά αρχικοποιούμε ένα **Scanner** ως **null** καθώς και ένα **insert** ως **true** που θα μας χρειαστεί για την εισαγωγή αργότερα. Στην συνέχεια προσπαθούμε να διαβάσουμε το αρχείο και αν για κάποιο λόγο δεν το διαβάζει μας πετάει το κατάλληλο μήνυμα. Μετά δημιουργούμε μια **while loop** όσο υπάρχει ακόμα κάποιο **token** που πρέπει να διαβαστεί και αν τηρεί τις προϋποθέσεις να γίνει **insert**. Μέσα σε αυτό το **loop** αρχικά αποθηκεύουμε αυτήν την λέξη σε μια μεταβλητή και μετά λέμε πως εάν δεν περιέχει νούμερα και δεν είναι ίση με το κενό να συνεχίσει να ψάχνει εάν είναι κατάλληλη για **insert**. Εάν η λέξη είναι μεγαλύτερη του ενός χαρακτήρα πάει για να βγάλει εάν υπάρχουν σημεία στίξης εξωτερικά της. Έτσι κάνουμε δύο **while loops** όπου όσο υπάρχει κάποιο σημείο στίξης τόσο στην αρχή όσο και στο τέλος της το αφαιρεί. Όταν τα αφαιρέσει όλα ή φτάσει να είναι ένας χαρακτήρας σταματάει και πάει να τσεκάρει εάν συνεχίζει να έχει κάποιο σύμβολο μέσα της πέρα από την απόστροφο, εάν κάτι τέτοιο ισχύει θέτει την **insert** ως **false** αφού δεν θέλουμε να ενταχθεί στο δέντρο. Εάν η λέξη είναι από την αρχή 1 χαρακτήρας απλά τσεκάρει εάν δεν είναι αλφάβητο, εάν κάτι τέτοιο ισχύει θέτει την **insert** σε **false**.

Τέλος, λέμε εάν δεν είναι “”, δεν υπάρχει στην λίστα των **stopWords** και η **insert** είναι **true** να την εισάγει μέσα στο δέντρο.

Για την **int getTotalWords()** αυτό που κάναμε είναι ότι λέγαμε εάν το **head** είναι **null** τότε επέστρεψε 0 καθώς δεν υπάρχει καμία λέξη μέσα στο δέντρο αλλιώς καλούσαμε μια μέθοδο που με αναδρομική διαδικασία υπολόγιζε το **sum** όλων των παιδιών του **head** και μας επέστρεφε το **sum** αυτών με την συχνότητα του. Εάν δεν έβρισκε σε κάποια μεριά φύλλο απλά επέστρεφε το 0.

Για την **int getDistinctWords()** απλά επιστρέφαμε το **counter** που προαναφέραμε. Το συγκεκριμένο **counter** αυξανόταν κάθε φορά που κάναμε **insert** ένα νέο αντικείμενο καθώς και όταν κάναμε ένα αντικείμενο **insert** σαν **head**. Επίσης, αυτό μειωνόταν κάθε φορά που κάναμε **remove** ένα αντικείμενο απτό δέντρο.

Για την **int getFrequency(String w)** απλά καλούσαμε την **search** και αναζητούσαμε σ’ αυτή το **w**. Έτσι, μετά βάλαμε μια **if** παράμετρο εάν αυτό που μας επέστρεψε η **search** είναι **null**, δηλαδή δεν υπάρχει μέσα στο δέντρο, να επιστρέφει 0 αλλιώς να μας επιστρέφει την συχνότητα αυτού που επέστρεψε η **search**. (Αυτό το κάναμε με την χρήση μιας **getFreq()** μέσα απτο **WordFreq**).

Για την **WordFreq getMaximumFrequency()** ουσιαστικά παίρνουμε παράμετρο εάν το **head** είναι **null**, δηλαδή δεν υπάρχει κάποιο αντικείμενο στο δέντρο επιστρέφει 0. Αλλιώς καλεί μια μέθοδο **findMax** και επιστρέφει αυτό που θα μας επιστρέψει η **FindMax**. Η **findMax** ουσιαστικά κάνει μια αναδρομική διαδικασία οπού τσεκάρει παντού εάν του κάθε στοιχείου η συχνότητα είναι μεγαλύτερη απτή **max** εκείνη την στιγμή, εάν κάτι τέτοιο ισχύει την θέτει σαν **max**.

Για την **double getMeanFrequency()** απλά καλούμε τις **getTotalWords()** και **getDistinctWords()** και τις θέτουμε σε 2 **double** μεταβλητές την κάθε μια ξεχωριστά ώστε να έχουμε να κάνουμε με **double values**. Στην συνέχεια απλά διαιρούμε τα αποτελέσματα που μας επέστρεψαν και επιστρέφουμε το αποτέλεσμα αυτής της διαίρεσης.

Για την **void addStopWord(String w)** πήραμε μια **if** εάν η **stopWords** είναι κενή να φτιάχνει μια νέα λίστα και μετά απλά με την χρήση της **insertAtBack** κάναμε **insert** την λέξη στην λίστα.

Για την **void removeStopWord(String w)** απλά λειτουργήσαμε με την χρήση **iterator** και διασχίζαμε την λίστα μέχρι να βρει την λέξη ή ο **iterator** να φτάσει να είναι κενός (να έχει διασχισθεί όλη η λίστα). Εάν την βρει μέσα στην λίστα και είναι **head** απλά την βγάζει και βάζει την επόμενη σαν **head** αλλιώς απλά συνδέει την προηγούμενη της με την επόμενη της.

Για την **printTreeAlphabetically(PrintStream stream)** ουσιαστικά σε αυτή καλούσαμε μια άλλη μέθοδο **printInOrder** με όρισμα το **head**. Έτσι αυτή η **printinorder** ουσιαστικά με τον γνωστό αναδρομικό τρόπο της **InOrder** μεθόδου μας εκτύπωνε τα στοιχεία του πίνακα με αλφαβητική σειρά.

Για την **printTreeByFrequency(PrintStream stream)** εάν το **head** είναι **null** απλά κάνει **return**, αλλιώς μέσα σε αυτή δημιουργούμε έναν πίνακα μεγέθους ίσου με τα **TreeNodes** του δέντρου εκείνη την στιγμή και καλούμε μια άλλη μέθοδο **insertToArray** που με **inOrder** αναδρομή βάζει τα στοιχεία του δέντρου μέσα στον πίνακα. Στην συνέχεια μέσα στην **PrintTreeByFrequency** καλούμε μια **QuickSort** μέθοδο που έχουμε φτιάξει για τον πίνακα και μετρά με μια for εκτυπώνουμε αυτόν τον ταξινομημένο πίνακα. Στο τέλος αρχικοποιούμε και την μεταβλητή j σε περίπτωση που θέλουμε να ξανακάνουμε **printByFrequency**.

Οι μέθοδοι **int getDistinctWords()** και **void addStopWord(String w)** έχουν υλοποιηθεί με τέτοιό τρόπο ώστε να μην χρειάζεται καμια **while loop** ή κάτι που να αυξάνει την πολυπλοκότητα τους. Έτσι, η πολυπλοκότητα τους είναι **O(1)**.

Η πολυπλοκότητα της **printTreeAlphabetically(Printstream stream)** είναι **O(N)** κι' αυτό γιατί το κάθε node θα γίνει traverse μόνο μια φορά.

Η πολυπλοκότητα της **printTreeByFrequency(PrintStream stream)** ειναι **O(N)** κι' αυτό γιατί αυτή ειναί χειρότερη απ'την ologn πολυπλοκότητα της **QuickSort**.

Επίσης, και η πολυπλοκότητα των **void insert(String w)**,**WordFreq search(String w)**,**void remove(String w),int getTotalWords(),WordFreq getMaximumFrequency()** θα είναι **O(N)** γιατί και αυτές στην χειρότερη των περιπτώσεων θα κάνουν **traverse** το καθε **TreeNode** μόνο μια φόρα, δηλαδή ουσιαστικά διάσχιση δέντρου.

Η double getMeanFrequency() επειδή καλεί την **int getDistinctWords()** που έχει **complexity O(1)** και την **int getTotalWords()** που στήν χειρότερη των περιπτώσεων έχει **O(N)** θα έχει επίσης **O(N).**

Η **void removeStopWord(String w)** επίσης θα έχει πολυπλοκότητα **O(N)** καθώς στην χειρότερη των περιπτώσεων θα πρέπει να διασχίσει όλη την λίστα.

Η **int getFrequency(String w)** καλεί την **search** η οποία όπως προαναφέραμε έχει **complexity O(N).** Έτσι και η **int getFrequency(String w)** στήν χειρότερη των περιπτώσεων θα έχει **O(N).**

Τέλος ,όσο αφορα την πολυπλοκότητα της **void Load(String filename)** επηρεάζεται από πολλούς παράγοντες όπως το **length** του κάθε **string**, τον αριθμό των **strings** και γραμμών κα.

Εαν υπάρχει κάποιο πρόβλημα παρακαλώ ενημερώστε μας, σε εμας φαίνονται να λειτουργούν όλα ομαλά.

Τα email μας είναι **p3200278@aueb.gr**, **p3200268@aueb.gr**