ΕΡΓΑΣΙΑ 3η

Συντελεστές

Καραγιάννης Γεώργιος (3190074)

Τάτας Αλέξανδρος (3190196)

Ανάλυση μέρους Β

Για την 3η εργασία του μαθήματος έχουμε αναλάβει να υλοποιήσουμε την δομή δεδομένων του ΔΤΤ (Δυαδικό Δέντρο Αναζήτησης) η οποία βασίζεται πάνω στο interface WordCounter που μας δόθηκε. Η υλοποίηση του BST λοιπόν, περιλαμβάνει τις μεθόδους: insert, search, remove, load, getTotalWords, getDistinctWords, getFrequency, getMaximumFrequency, getMeanFrequency, addStopWord, removeStopWord, printTreeAlphabetically, printTreeByfrequency καθώς επίσης και κάποιες πρόσθετες υποστηρικτικές μεθόδους όπως οι compare, inOrder και InFreqOrder. Επίσης η κλάση BST περιλαμβάνει και μια άλλη ιδιωτική κλάση στο εσωτερικό της (την TreeNode) η οποία απαρτίζεται από τις μεθόδους nodeInSubTree, getSubTreeSize, setLeft, setRight, getItem, getLeft και getRight. Αναλυτικά τώρα για κάθε μέθοδο του μέρους Β ξεχωριστά:

- **nodesInSubTree(TreeNode curNode)**: Μέσω αναδρομής ελέγχει ποιό είναι το μέγεθος του υποδέντρου που ορίζει ο κόμβος curNode συμπεριλαμβανομένου του ίδιου του κόμβου (δηλαδή από πόσους κόμβους αποτελείται το υποδέντρο).

- **getSubTreeSize():** Επιστρέφει το αποτέλεσμα της nodesInSubTree (ταυτόχρονα ενημερώνει και το πεδίο subTreeSize του κόμβου).

- **setLeft(TreeNode left), setRight(TreeNode right):** Η κάθε μια αναλαμβάνει να θέσει νέο κόμβο αριστερά ή δεξιά του τρέχοντος κόμβου αντίστοιχα.

- **getItem():** Επιστρέφει το αντικείμενο WordFreq που περιέχεται στον τρέχοντα κόμβο του BST.

- **getLeft(), getRight():** Επιστρέφει τον δεξιό ή τον αριστερό κόμβο του τρέχοντος κόμβου.
- **compare(String word1, String word2):** Υποστηρικτική μέθοδος που συγκρίνει δυο strings. Η μέθοδος συγκρίνει έναν-έναν του χαρακτήρες των words και εν τέλει επιστέφει 0 αν τα strings είναι ίδια, αρνητικό ακέραιο αν το words1 είναι μικρότερο του word2 ή θετικό ακέραιο στην αντίστροφή περίπτωση.

- **insert(String w):** Μέθοδος η οποία αναλαμβάνει να εισάγει την λέξη w στο Δέντρο Δυαδικής Αναζήτησης. Αναλυτικότερα, όσον αφορά την υλοποίηση της, ξεκινώντας από την ρίζα του δέντρου ελέγχουμε αν το w ταυτίζεται με το κλειδί του τρέχοντος κόμβου (με την βοήθεια της compare) και αν αυτό ισχύει τότε αυξάνουμε το frequency της λέξης και φυσικά το πεδίο countOfWords. Αν όχι, περιορίζουμε την αναζήτηση στα αριστερά ή τα δεξιά του τρέχοντος κόμβου ανάλογα με το αν το w είναι μικρότερο ή μεγαλύτερο και επαναλαμβάνουμε την ιδιά διαδικασία. Αν καταλήξουμε να εξερευνούμε ολόκληρο το μονοπάτι του δέντρου χωρίς αποτέλεσμα τότε εισάγουμε ως "παιδί" του τελευταίου κόμβου που ελέγξαμε ένα αντικείμενο WordFreq του οποίου κλειδί θα είναι το w και συχνότητα θα είναι το 1(κάνουμε ουσιαστικά μια εισαγωγή ως φύλλο και αυξάνουμε το πεδίο countOfNodes).

- **search(String w):** Λαμβάνει ως όρισμα μια λέξη w και αν την βρει τότε επιστρέφει το αντικείμενο στο οποίο αυτή αντιστοιχούσε μέσα στο ΔΔΑ. Η ιδέα πίσω από την search είναι κυρίως ο αλγόριθμος "binary search". Ξεκινώντας από την ρίζα και με την βοήθεια της μεθόδου compare συγκρίνουμε αν το κλειδί του τρέχοντος κόμβου ταιριάζει με την w οπότε και επιστρέφουμε το περιεχόμενο του κόμβου. Διαφορετικά, περιοριζόμαστε στο αριστερό η το δεξί υποδέντρο του κόμβου και επαναλαμβάνουμε την ίδια διαδικασία έως ότου έχουμε σίγουρο αποτέλεσμα για να επιστρέψουμε.

- **remove(String w):** Μέθοδος η οποία αναλαμβάνει να αφαιρέσει την λέξη w από το Δέντρο Δυαδικής Αναζήτησης. Με την βοήθεια της υποστηρικτικής μεθόδου compare, η remove διασχίζει σταδιακά το ΔΔΑ ώσπου να βρει τον κόμβο που έχει για κλειδί του το w και μέσω κατάλληλων διεργασιών και επεξεργασίας δεικτών αφαιρεί το εν λόγω κόμβο από το δέντρο.

- **load(String filename):** Μέθοδος η οποία δέχεται ως όρισμα ένα αρχείο txt και μέσω διάφορων φιλτραρισμάτων λέξεων κατασκευάζει ένα ΔΔΑ. Αρχικά,

το txt μετατρέπεται σε string και με την σειρά του αυτό δέχεται επεξεργασία ώστε όλοι οι χαρακτήρες του κειμένου να είναι πεζοί. Έπειτα, ανιχνεύονται όλοι οι χαρακτήρες αλλαγής γραμμής και αντικαθίστανται από τον χαρακτήρα του κενού ‘ ‘ ούτως ώστε να μπορούμε να διαχωρίζουμε αργότερα όλες τις λέξεις μεταξύ τους. Στην συνέχεια, απομονώνουμε το κείμενο που υπάρχει εντός αλλά και εκτός των παρενθέσεων (ώστε να απελευθερωθούμε πλήρως από τους χαρακτήρες της παρένθεσης) και στο τέλος αφότου όλες οι εν δυνάμει λέξεις είναι σαφώς διαχωρισμένες με τον χαρακτήρα του κενού ‘ ’ τις εισάγουμε σε μια πρόχειρη λίστα allWordsList με την βοήθεια της μεθόδου split και της put. Δημιουργούμε επίσης μια τελική λίστα fixedWordsList στην οποία εισάγουμε όσες από τις λέξεις της allWordsList πέρασαν τον έλεγχο εγκυρότητας (ειδικοί χαρακτήρες κλπ.) τον οποίο διενεργήσαμε με την χρήση αρκετών εντολών if. Στο τέλος και αφότου έχουμε βρει όλες τις έγκυρες λέξεις του κειμένου και έχουμε κάνει insert στο ΔΔΑ, ζητάμε από τον χρήστη να εισάγει ποιες λέξεις “stop words” θα ήθελε να αποκλειστούν από τον υπολογισμό και με την βοήθεια της μεθόδου remove του BST αφαιρούμε από το δέντρο αυτές τις εξαιρούμενες λέξεις.

- **getTotalWords():** Επιστρέφει τον συνολικό αριθμό λέξεων που βρέθηκαν στο BST συνυπολογιζόμενου και του frequency της λέξης σε κάθε κόμβο. Το μόνο που γίνεται είναι η επιστροφή του πεδίου countOfWords το οποίο ενημερώνεται μέσα στις μεθόδους insert και remove, οπότε προφανώς και η πολυπλοκότητα της μεθόδου είναι O(1).

- **getDistinctWords():** Επιστρέφει τον συνολικό αριθμό διαφορετικών λέξεων στο BST , δηλαδή δεν λαμβάνει υπόψιν του τυχόν συχνότητα>1 για κάποια λέξη (επιστρέφει ουσιαστικά τον αριθμό κόμβων του δέντρου). Το μόνο που γίνεται είναι η επιστροφή του πεδίου countOfNodes το οποίο ενημερώνεται μέσα στις μεθόδους insert και remove, οπότε προφανώς και η πολυπλοκότητα της μεθόδου είναι O(1).

- **getFrequency(String w):** Μέθοδος που επιστρέφει την συχνότητα της λέξης w στο ΔΔΑ. Η βασική ιδέα είναι ότι καλείται η μέθοδος search για αυτή την λέξη και αν εκείνη αποφανθεί ότι αυτή η λέξη βρέθηκε τότε επιστρέφεται η συχνότητα της, διαφορετικά επιστρέφεται 0.

- **getMaximumFrequency():** Μέθοδος που επιστρέφει το αντικείμενο/λέξη με την μεγαλύτερη συχνότητα μέσα στο δέντρο. Ουσιαστικά αυτό που γίνεται είναι μια κλήση της inOrderFreq από την οποία προκύπτει μια λίστα με αντικείμενα WordFreq στην οποία και αναζητούμε το αντικείμενο με το

μεγαλύτερο frequency (βάσει του πολύ γνωστού αλγορίθμου εύρεσης μεγίστου σε ακολουθίες).

- **getMeanFrequency():** Μέθοδος που επιστρέφει τον μέσο όρο των συχνοτήτων μέσα στο δέντρο. Καλείται η μέθοδος inFreqOrder η οποία κατασκευάζει μια λίστα με αντικείμενα WordFreq. Από αυτή την λίστα αθροίζονται οι συχνότητες των λέξεων και το αποτέλεσμα διαιρείται με το μέγεθος της λίστας δημιουργώντας έτσι τον μέσο ορό.

- **addStopWord(String w):** Προσθέτει την "stop word" w στην λίστα stopWordsList που είναι πεδίο του BST μέσω της μεθόδου put. Εφόσον η μέθοδος κάνει απλώς μια κλήση της μεθόδου put (η put ανήκει στο QueueInterface) η οποία ως γνωστόν είναι O(1), έτσι και η εν λόγω μέθοδος θα είναι της τάξης του O(1).

- **removeStopWord(String w):** Αφαιρεί την "stop word" w από την λίστα stopWordsList που είναι πεδίο του BST μέσω της μεθόδου put.

- **printTreeAlphabetically(PrintStream stream):** Εκτυπώνει μέσω PrintStream το δέντρο σε αλφαβητική σειρά κόμβων καλώντας την μέθοδο inOrder με όρισμα τον δείκτη ρίζας head (από όπου και θέλουμε και να ξεκινήσει την διάσχιση).

- **printTreeByFrequency(PrintStream stream):** Εκτυπώνει μέσω PrintStream το δέντρο σε αύξουσα σειρά βάσει της συχνότητας κάθε λέξης στους κόμβους του δέντρου καλώντας την μέθοδο inFreqOrder. Αναλυτικότερα, καλείται η μέθοδος inFreqOrder (με όρισμα τον δείκτη head) από την οποία δημιουργείται μια λίστα με αντικείμενα τύπου WordFreq η οποία με την σειρά της ταξινομείται σε αύξουσα σειρά βάσει συχνότητας και υστέρα με την χρήση PrintStream εκτυπώνεται το κατάλληλο αποτέλεσμα.

- **inOrder(TreeNode node):** Υποστηρικτική μέθοδος που αναλαμβάνει να διασχίσει αναδρομικά το υποδέντρο που ορίζει ο κόμβος node από τα αριστερά προς τα δεξιά δημιουργώντας σταδιακά μια πρόταση (στο string πεδίο BufferForInOrder της BST) με τις λέξεις που βρίσκει με την σειρά (βάσει των κλειδιών τύπου string, πάνω στα οποία στηρίχτηκε προηγουμένως η κατασκευή του ΔΔΑ). Η πρόταση που επιστρέφεται είναι

τύπου string και περιέχει τις λέξεις ταξινομημένες και χωρισμένες από τον χαρακτήρα του κενού ‘ ’ ούτως ώστε να εκτυπωθεί από την printTreeAlphabetically.

- **inFreqOrder(TreeNode node):** Όπως περίπου η inOrder έτσι και η inFreqOrder διασχίζει με αναδρομικό αλγόριθμο το δέντρο. Όμως, σε αυτή την περίπτωση κατασκευάζεται σταδιακά μια λίστα με αντικείμενα τύπου WordFreq (συγκεκριμένα η λίστα inOrderWords που είναι αρχικά δηλωμένη ως πεδίο της κλάσης BST) η οποία και επιστρέφεται για εκμετάλλευση κυρίως από τις μεθόδους printTreeByFrequency, getMaximumFrequency και getMeanFrequency.

***Να σημειωθεί ότι έχει γίνει χρήση της ουράς μονής σύνδεσης που υλοποιήσαμε στην 1η εργασία μέσω του interface “QueueInterface” και εμπλουτισμένη αυτή την φορά με την υποστήριξη generics και με μερικές επιπλέον μεθόδους που βοηθούν στην λύση της 3ης εργασίας.**