Data Structures

Εργασία στις δομές δεδομένων.
Περιλαμβάνει:
- αταξινόμητο πίνακα,
- ταξινομημένο πίνακα,
- απλό δυαδικό δένδρο αναζήτησης,
- δυαδικό δένδρο αναζήτησης τύπου AVL και
- πίνακας κατακερματισμού με ανοικτή διεύθυνση.

Λαχανά Αριστέα
3590
alachana@csd.auth.gr

Κόρκος Αλέξανδρος
3870
alexkork@csd.auth.gr

Περιεχόμενα

1. *Αταξινόμητος πίνακας*

1.1 Εισαγωγή

Για την αποθήκευση των λέξεων μέσα στον πίνακα θα γίνει χρήση ενός strcuct (Element). Η δομή έχει ως εξής: υπάρχει ένα πεδίου τύπου string και όνομα word που αποθηκεύει την λέξη αυτούσια επιπλέον, υπάρχει ένα πεδίο τύπου int και με όνομα appearances που αποθηκεύει τις εμφανίσεις της συγκεκριμένης λέξης μέσα στο αρχείο. Ο πίνακας που αποθηκεύει τις λέξεις θα υλοποιηθεί στον σωρό. Σαφέστερα, για την υλοποίηση του πίνακα θα χρησιμοποιηθούν δυο μεταβλητές τύπου int, η size που είναι το μέγεθος του πίνακα (δηλ. των θέσεων που διαθέτει ο πίνακας) και η μεταβλητή posOfLastElement που κρατάει την θέση του τελευταίου στοιχείου μέσα στον πίνακα. Σε περίπτωση όπου ο πίνακας γεμίσει με στοιχεία, καλείται η μέθοδος resize που μεγαλώνει το μέγεθος του πίνακα (διπλασιάζεται η size) έτσι ώστε να μην χαθούν λέξεις από το αρχείο.

1.2 Λειτουργίες

1.2.1 Εισαγωγή (insert)

Η μέθοδος αρχικά εξετάζει εάν υπάρχει ήδη η λέξη που πρόκειται να εισαχθεί, για αυτό τον έλεγχο καλείται η αναζήτηση που εντοπίζει την θέση της λέξης και την αποθηκεύει σε μια μεταβλητή. Στην περίπτωση που υπάρχει ήδη, η λέξη δεν εισάγεται ξανά στον πίνακα αλλά αυξάνεται ο αριθμός των εμφανίσεων της λέξης αυτής. Αντίθετα όταν δεν υπάρχει εισάγεται στον πίνακα θέτοντας παράλληλα των αριθμό των εμφανίσεων της λέξης σε ένα (1) και η posOfLastElement αυξάνεται κατά ένα (1). Τέλος, ελέγχεται εάν το size ισούται με posOfLastElement, ο έλεγχος αυτός γίνεται για να εξεταστεί εάν ο πίνακας γέμισε. Εάν έχει γεμίσει καλείται η μέθοδος resize.

1.2.2 Αναζήτηση (search)

Για την αναζήτηση θα χρησιμοποιηθεί η search και συγκεκριμένα η public, η μέθοδος αυτή επιστρέφει το πλήθος των εμφανίσεων της λέξης μέσα από τον πίνακα. Ουσιαστικά αυτή η μέθοδος δέχεται από την private search (σε αυτή την μέθοδο υλοποιείται η σειριακή αναζήτηση) τη θέση του στοιχείου μέσα στον πίνακα, όπου την

αποθηκεύει και την χρησιμοποιεί για να επιστρέψει το πλήθος των εμφανίσεων της λέξης.

1.2.3 Διαγραφη (remove)

Αρχικά γίνεται κλήση της συνάρτησης αναζήτησης ώστε να βρεθεί η θέση της λέξης που πρόκειται να διαγραφεί. Σε περίπτωση που η λέξη εμφανίζεται περισσότερες από μια φορές, το πεδίο του struct που αποθηκεύει τον αριθμό των εμφανίσεων της λέξης (appearances). Εάν η τιμή του πεδίου πριν την κλήση της συνάρτησης είναι ένα (1) τότε, αφού κληθεί αφαιρείται η λέξη από τον πίνακα, τα στοιχεία μετακινούνται κατά μια θέση προς τα αριστερά και η μεταβλητή που αποθηκεύει την θέση του τελευταίου στοιχειού πίνακα (posOfLastElement) μειώνεται κατά ένα (1). Τέλος, εάν πραγματοποιηθεί η διαγραφή επιστρέφεται η λογική τιμή true ενώ εάν η λέξη δεν υπάρχει στον πίνακα επιστρέφεται η λογική τιμή false.

2. *Ταξινομημένος πίνακας*

2.1 Εισαγωγή

Όπως και στο αταξινόμητη έτσι και εδώ, για την αποθήκευση των λέξεων μέσα στον πίνακα θα γίνει χρήση ενός strcuct (Element). Η δομή έχει ως εξής: υπάρχει ένα πεδίου τύπου string και όνομα word που αποθηκεύει την λέξη αυτούσια επιπλέον, υπάρχει ένα πεδίο τύπου int και με όνομα appearances που αποθηκεύει τις εμφανίσεις της συγκεκριμένης λέξης μέσα στο αρχείο. Ο πίνακας που αποθηκεύει τις λέξεις θα υλοποιηθεί στον σωρό. Σαφέστερα, για την υλοποίηση του πίνακα θα χρησιμοποιηθούν δυο μεταβλητές τύπου int, η size που είναι το μέγεθος του πίνακα (δηλ. των θέσεων που διαθέτει ο πίνακας) και η μεταβλητή posOfLastElement που κρατάει την θέση του τελευταίου στοιχείου μέσα στον πίνακα. Σε περίπτωση όπου ο πίνακας γεμίσει με στοιχεία, καλείται η μέθοδος resize που μεγαλώνει το μέγεθος του πίνακα (διπλασιάζεται η size) έτσι ώστε να μην χαθούν λέξεις από το αρχείο.

Η μέθοδος αρχικά εξετάζει εάν υπάρχει ήδη η λέξη που πρόκειται να εισαχθεί, για αυτό τον έλεγχο καλείται η αναζήτηση που εντοπίζει την θέση της λέξης και την αποθηκεύει σε μια μεταβλητή. Στην περίπτωση που υπάρχει ήδη, η λέξη δεν εισάγεται ξανά στον πίνακα αλλά αυξάνεται ο αριθμός των εμφανίσεων της λέξης αυτής. Αντίθετα όταν δεν υπάρχει εισάγεται στον πίνακα θέτοντας παράλληλα των αριθμό των εμφανίσεων της λέξης σε ένα (1) και η posOfLastElement αυξάνεται κατά ένα (1). Τέλος, ελέγχεται εάν το size ισούται με posOfLastElement, ο έλεγχος αυτός γίνεται για να εξεταστεί εάν ο πίνακας γέμισε. Εάν έχει γεμίσει καλείται η μέθοδος resize

2.2 Λειτουργίες

2.2.1 Εισαγωγή (insert)

Για την εισαγωγή στο ταξινομημένο πίνακα θα γίνει η χρήση της δυαδικής αναζήτησης και της ταξινόμησης με εισαγωγή. Αρχικά γίνεται η κλήση (η private insert) μια παραλλαγής της δυαδικής αναζήτησης της οποίας το αποτέλεσμα αποθηκεύετε σε μια μεταβλητή. Με παραλλαγή εννοείται πως όταν το στοιχείο υπάρχει ήδη στον πίνακα, η μεταβλητή found που περνιέται με στην συνάρτηση με αναφορά γίνεται true και επιστρέφεται η θέση του στοιχείου μέσα στον πίνακα, αντίθετα όταν το

στοιχείο απουσιάζει η δυαδική αναζήτηση βρίσκει την θέση στην οποία πρέπει να τοποθετηθεί το στοιχείο για να διατηρηθεί η ταξινόμηση και η μεταβλητή found γίνεται false. Έπειτα εξετάζεται από την συνάρτηση εισαγωγής η τιμη της found, εάν είναι true αυξάνεται η τιμή της μεταβλητής appearances του Element (struct) που βρίσκεται μέσα στον πίνακα κατά ένα (1) διαφορετικά, κάθε στοιχείο το οποίο βρίσκεται δεξιά από τη θέση που θα πρέπει να εισαχθεί το νέο στοιχείο, μετακινείται κατά μια (1) θέση προς δεξιά και έπειτα το στοιχείο εισάγεται στην σωστή θέση χωρίς να διαταραχθεί η ταξινόμηση ενώ επιπλέον η posOfLastElement αυξάνεται κατά ένα (1). Τέλος, ελέγχεται εάν το size ισούται με posOfLastElement, ο έλεγχος αυτός γίνεται για να εξεταστεί εάν ο πίνακας γέμισε. Εάν έχει γεμίσει καλείται η μέθοδος resize.

2.2.2 Αναζήτηση (search)

Για την αναζήτηση των στοιχείων μέσα από τον πίνακα θα χρησιμοποιηθεί η δυαδική αναζήτηση (η private search) έτσι ώστε να γίνει εκμετάλλευση της ταξινόμησης και επιστρέφει την θέση του στοιχειού μέσα στον πίνακα εάν υπάρχει διαφορετικά επιστρέφει -1. Αφού βρεθεί η θέση του στοιχείου μέσα στον πίνακα επιστρέφεται ο αριθμός των εμφανίσεων της συγκεκριμένης λέξης, σε κάθε άλλη περίπτωση επιστρέφεται η τιμή μηδέν (0).

2.2.3 Διαγραφή (remove)

Όπως στον αταξινόμητο πίνακα, έτσι και εδώ γίνεται κλήση της συνάρτησης δυαδικής αναζήτησης (η private search) ώστε να βρεθεί η θέση της λέξης που πρόκειται να διαγραφεί. Σε περίπτωση που η λέξη εμφανίζεται περισσότερες από μια φορές, το πεδίο του struct που αποθηκεύει τον αριθμό των εμφανίσεων της λέξης (appearances). Εάν η τιμή του πεδίου πριν την κλήση της συνάρτησης είναι ένα (1) τότε, αφού κληθεί αφαιρείται η λέξη από τον πίνακα, τα στοιχεία μετακινούνται κατά μια θέση προς τα αριστερά και η μεταβλητή που αποθηκεύει την θέση του τελευταίου στοιχειού πίνακα (posOfLastElement) μειώνεται κατά ένα (1). Τέλος, εάν πραγματοποιηθεί η διαγραφή επιστρέφεται η λογική τιμή true ενώ εάν η λέξη δεν υπάρχει στον πίνακα επιστρέφεται η λογική τιμή false.

3. *Απλό δυαδικό δένδρο αναζήτησης*

3.1 Εισαγωγή

Για την αποθήκευση των λέξεων μέσα στο δυαδικό δένδρο χρησιμοποιείται ένα struct (node), η δομή του struct είναι η εξής: χρησιμοποιείται η μεταβλητή word τύπου string για την αποθήκευση της λέξης, μια μεταβλητή appearances τύπου int για την αποθήκευση των εμφανίσεων της συγκεκριμένης λέξης και υπάρχουν τρεις μεταβλητές (parent, left, right) τύπου node (struct) για την αποθήκευση του προγόνου (πατέρα) κόμβου, την αποθήκευση του αριστερού απογόνου (παιδιού) και του δεξιού απογόνου (παιδιού) αντίστοιχα. Για την δημιουργία λοιπόν του δένδρου, θα χρησιμοποιηθεί η root τύπου node που θα αναπαριστά την ρίζα του δένδρου (η τιμή της root κατά την κατασκευή της δομής θέτεται σε nullptr).

3.2 Λειτουργίες

3.2.1 Εισαγωγή (insert)

Για την εισαγωγή των λέξεων στο πίνακα από τον χρήστη καλείται η μέθοδος insert (public), η οποία εξετάζει αρχικά την ρίζα (root) του δένδρου δεν έχει πάρει κάποια λέξη ως τιμή (root == nullptr). Εάν είναι άδειος ο κόμβος τότε η λέξη τοποθετείται στην ρίζα και γίνεται αρχικοποίηση των μεταβλητών του node (appearances = 1, parent = nullptr, right = nullptr, left = nullptr), σε κάθε άλλη περίπτωση καλείται η insert (private), σε αυτή την μέθοδο εξετάζεται εάν η λέξη είναι ίδια με αυτή που έχει ήδη ο κόμβος εάν ναι η μεταβλητή appearances αυξάνεται κατά ένα (1), εάν όχι εξετάζεται εάν είναι λεξικογραφικά μεγαλύτερη από την λέξη που υπάρχει ήδη στον κόμβο δημιουργείται ένας νέος κόμβος δεξιά του κόμβου που εξετάζεται διαφορετικά, (δηλ. είναι λεξικογραφικά μικρότερη) δημιουργείται ένας νέος κόμβος που τοποθετείτε η λέξη στα αριστερά του κόμβου που εξετάζεται.

3.2.2 Διαγραφή (remove)

Την διαγραφή θα την χωρίσουμε σε τρεις περιπτώσεις:

- Διαγραφή κόμβου που δεν έχει παιδιά (δηλ. ενός φύλλου),
- διαγραφή κόμβου που έχει ένα παιδί,

- διαγραφή κόμβου που έχει δύο παιδία.

Σε κάθε περίπτωση γίνεται κλήση της remove (private) όπου πραγματοποιείται η αναζήτηση του κόμβου που πρέπει να διαγραφεί και αποθηκεύετε στην parent (τύπου node) ο πρόγονος του κόμβου που πρέπει να διαγραφεί.

Στη πρώτη περίπτωση εξετάζεται εάν ο πρόγονος είναι κενός σημαίνει ότι βρισκόμαστε στην ρίζα όπου και την διαγράφουμε. Εναλλακτικά εξετάζεται εάν ο κόμβος είναι το αριστερό ή το δεξί παιδί του προγόνου όπου και διαγράφεται.

Στην δεύτερη περίπτωση εάν ο πρόγονος είναι κενός που σημαίνει ότι ήμαστε στην ρίζα, εξετάζουμε ποιο από τα παιδιά του παιδιού είναι κενό και τοποθετούμε το άλλο ως ρίζα το δένδρου ενώ τέλος διαγράφουμε το παιδί (rt). Διαφορετικά εξετάζεται εάν ο κόμβος που πρέπει να διαγραφεί είναι αριστερό ή δεξί παιδί του προγόνου του και εάν ο κόμβος που θα διαγραφεί έχει αριστερό ή δεξί παιδί. Έτσι διακρίνουμε και τέσσερις πιθανές περιπτώσεις:

- Ο κόμβος διαγράφης είναι αριστερό παιδί του προγόνου και ο κόμβος διαγράφης δεν έχει αριστερό παιδί, άρα το αριστερό παιδί του του προγόνου γίνεται το δεξί παιδί του κόμβου διαγράφης, ενώ έπειτα διαγράφεται ο κόμβος αυτός.

- Ο κόμβος διαγράφης είναι αριστερό παιδί του προγόνου και ο κόμβος διαγράφης δεν έχει δεξί παιδί, άρα το αριστερό παιδί του του προγόνου γίνεται το αριστερό παιδί του κόμβου διαγράφης, ενώ έπειτα διαγράφεται ο κόμβος αυτός.

- Ο κόμβος διαγράφης είναι δεξί παιδί του προγόνου και ο κόμβος διαγράφης δεν έχει αριστερό παιδί, άρα το δεξί παιδί του του προγόνου γίνεται το δεξί παιδί του κόμβου διαγράφης, ενώ έπειτα διαγράφεται ο κόμβος αυτός.

- Ο κόμβος διαγράφης είναι δεξί παιδί του προγόνου και ο κόμβος διαγράφης δεν έχει αριστερό παιδί, άρα το δεξί παιδί του του προγόνου γίνεται το δεξί παιδί του κόμβου διαγράφης, ενώ έπειτα διαγράφεται ο κόμβος αυτός.

Στην τρίτη περίπτωση αποθηκεύεται στην successor ο μικρότερος κόμβος κάτω από το δεξί παιδί του κόμβου διαγραφής ώστε να αποθηκευτούν στα πεδία word & appearances του κόμβου διαγραφής οι τιμές word & appearances του successor ενώ έπειτα καλείται ξανά η remove (private) με όρισμα την successor.

3.2.3 Αναζήτηση (search)

Η αναζήτηση επιτυγχάνεται μέσω της κλήσης της search (public) που επιστρέφει την αριθμό εμφανίσεων της λέξης εάν υπάρχει διαφορετικά επιστρέφει 0. Για την εύρεση της λέξης γίνεται κλήση της search (private) η οποία εξετάζει εάν η λεξικογραφική θέση της λέξης που ψάχνουμε είναι μεγαλύτερη ή μικρότερη από έναν δεδομένο κόμβο (η πρώτη εξέταση γίνεται με την ρίζα) και αναλόγως επιλέγετε το σωστό μονοπάτι για την εύρεση της λέξης.

3.3 Διάσχιση

Η λειτουργίες τις διάσχισης του δένδρου υλοποιούνται όλες αναδρομικά.

3.3.1 Ενδοδιατεταγμένη (inorder)

Για κάθε κόμβο, επισκεπτόμαστε πρώτα τους κόμβους του αριστερού του υποδένδρου, έπειτα τον ίδιο τον κόμβο και στη συνέχεια τους κόμβους του δεξιού του υποδένδρου.

3.3.2 Μεταδιατεταγμένη (postorder)

Για κάθε κόμβο, επισκεπτόμαστεπρώτα τους κόμβους του αριστερού του υποδένδρου, έπειτα τους κόμβους του δεξιού του υποδένδρουκαι στη συνέχεια τον ίδιο τον κόμβο.

3.3.3 Προδιατεταγμένη (preorder)

Για κάθε κόμβο, επισκεπτόμαστεπρώτα τον ίδιο τον κόμβο, έπειτα τους κόμβους του αριστερού του υποδένδρουκαι στη συνέχεια τους κόμβους του δεξιού του υποδένδρου.

4. Δυαδικό δένδρο αναζήτησης τύπου AVL

4.1 Εισαγωγή

Κλάση Node: αναπαριστά έναν κόμβο από το AVL δένδρο και περιέχει 5 πεδία:

- string key: η τιμή του κόμβου - λέξη από το αρχείο.
- Node *left: αριστερό παιδί του κόμβου.
- Node *right: δεξί παιδί του κόμβου.
- int height: ύψος του δένδρου.
- int appearances: μετράει πόσες φορές εμφανίζεται η λέξη key στο κείμενο.

Σε ένα δένδρο AVL τα ύψη των δύο παιδιών της ρίζας διαφέρουν το πολύ κατά 1 όπως και δυο παιδιά των παιδιών της ρίζας κ.ο.κ. Για να γίνει η διατήρηση της ιδιότητας αυτής του δένδρου θα χρησιμοποιηθούν στην εισαγωγή και διαγραφή (όταν χρειαστεί) κάποιες μέθοδοι που αναλαμβάνουν την περιστροφή του δένδρου. Σαφέστερα:

- Μέθοδος που υλοποιεί τη δεξιά περιστροφή του δένδρου, ώστε να μη χάσει την ιδιότητά του σαν AVL δένδρο. Δίνεται ο κόμβος y σαν παράμετρος. Δημιουργούνται δύο νέοι κόμβοι: x και T2. Ο x αρχικοποιείται με το αριστερό παιδί του κόμβου y και στη συνέχεια ο T2 αρχικοποιείται με το δεξί παιδί του x. Στη θέση του δεξιού παιδιού του x πάει ο y, ενώ στη θέση του αριστερού παιδιού του y πάει ο T2. Τέλος, το height του κόμβου y αλλάζει καλώντας τη μέθοδο max(int, int) που παίρνει ως ορίσματα:

  1. το height του αριστερού παιδιού του y και
  2. το δεξί height του y αυξημένο κατά 1.

  Ομοίως, αλλάζει το height του x, αλλά με ορίσματα:

  1. το height του αριστερού παιδιού του x και
  2. το δεξί height του x αυξημένο κατά 1.

- Μέθοδος που υλοποιεί την αριστερή περιστροφή του δένδρου, ώστε να μη χάσει την ιδιότητά του σαν AVL δένδρο. Δίνεται ο κόμβος x σαν παράμετρος. Δημιουργούνται δύο νέοι κόμβοι: y και T2. Ο y αρχικοποιείται με το δεξί παιδί του κόμβου x και στη συνέχεια ο T2 αρχικοποιείται με το αριστερό παιδί του y. Στη θέση του αριστερού παιδιού του y πάει ο x, ενώ στη θέση του δεξιού παιδιού του x πάει ο T2. Τέλος, το height του κόμβου x αλλάζει καλώντας τη μέθοδο max(int, int) που παίρνει ως ορίσματα:

  1. το height του αριστερού παιδιού του x και
  2. το δεξί height του x αυξημένο κατά 1.

  Ομοίως, αλλάζει το height του y, αλλά με ορίσματα:

  1. το height του αριστερού παιδιού του y και
  2. το δεξί height του y αυξημένο κατά 1.

4.2 Λειτουργίες

4.2.1 Εισαγωγή (insert)

Μέθοδος που υλοποιεί την εισαγωγή μιας λέξης από το κείμενο στο AVL δένδρο. Αρχικά, γίνεται έλεγχος για το αν ο κόμβος που δίνεται σαν παράμετρος είναι κενός. Σε αυτή τη περίπτωση καλείται η μέθοδος newNode(string) και το αποτέλεσμα αυτής επιστρέφει η insert. Αν ο κόμβος δεν είναι κενός, γίνεται έλεγχος για το αν η λέξη από το κείμενο είναι λεξικογραφικά μεγαλύτερη, μικρότερη ή ίση με τη λέξη που υπάρχει στον κόμβο που δίνεται. Αν είναι μικρότερη, καλείται αναδρομικά η insert δίνοντας ως πρώτη παράμετρο το αριστερό παιδί του κόμβου και αποθηκεύει το αποτέλεσμα στο αριστερό παιδί του κόμβου. Ομοίως, αν είναι η λέξη του κειμένου μεγαλύτερη, αλλά για το δεξί παιδί του κόμβου. Αν η λέξη του κειμένου είναι ίδια με αυτή του κόμβου, αυξάνεται το πεδίο appearances του κόμβου κατά 1 και η μέθοδος επιστρέφει τον κόμβο. Μετά από αυτόν τον έλεγχο, η μέθοδος βρίσκει το height του κόμβου και τη διαφορά των υψών μεταξύ των υποδένδρων καλώντας τις κατάλληλες συναρτήσεις. Τέλος,με βάση τις δύο προηγούμενες τιμές, γίνεται έλεγχος για το αν πρέπει να γίνει δεξιά ή αριστερή περιστροφή στο δένδρο, έτσι ώστε να μη χάσει την ιδιότητά του ως AVL.

4.2.2 Διαγραφή (remove)

Μέθοδος που υλοποιεί την διαγραφή ενός κόμβου του δένδρου AVL. Αρχικά γίνεται έλεγχος για το αν δόθηκε κενός κόμβος. Σε αυτή τη περίπτωση, επιστρέφεται ο κενός κόμβος. Αν η λέξη που είναι προς διαγραφή είναι μικρότερη από τη τιμή του κόμβου, βρίσκεται στο αριστερό υποδένδρο, γι αυτό ξανακαλείται η deleteNode με πρώτο όρισμα το αριστερό παιδί του κόμβου. Αν η λέξη που είναι προς διαγραφή είναι μεγαλύτερη από τη τιμή του κόμβου, βρίσκεται στο αριστερό υποδένδρο, γι αυτό ξανακαλείται η deleteNode με πρώτο όρισμα το δεξί παιδί του κόμβου. Στη περίπτωση που βρεθεί ο κόμβος με τη λέξη που είναι προς διαγραφή, αν ο κόμβος

έχει ένα παιδί, αντικαθίσταται με τον κόμβο-παιδί. Αν δεν έχει παιδί, διαγράφεται με τη χρήση της free. Αν έχει δύο παιδιά, καλεί την minValueNode με όρισμα το δεξί παιδί, για να βρεθεί ο κόμβος με τη μικρότερη τιμή του δεξιού υποδένδρου και αποθηκεύεται στον κόμβο προσωρινό temp. Τότε ο κόμβος αντικαθίστανται με τον temp, ως προς τα πεδία key και appearances. Τότε καλείται ξανά η deleteNode με πρώτο όρισμα το δεξί παιδί του root και με δεύτερο όρισμα το key του temp. Μετά του παραπάνω ελέγχους, αλλάζει κατάλληλα το πεδίο height του root με τη χρήση της height() και γίνεται έλεγχος της διαφοράς των υψών μεταξύ του αριστερού και του δεξιού υποδένδρου, ώστε να γίνουν οι κατάλληλες περιστροφές στο δένδρο.

4.2.3 Αναζήτηση (search)

Μέθοδος που υλοποιεί την αναζήτηση της λέξης που δίνεται από το κείμενο στο AVL. Η μέθοδος επιστρέφει ακέραιο αριθμό, ο οποίος αναπαριστά το πόσες φορές βρίσκεται η λέξη στο κείμενο. Αν ο κόμβος που δίνεται ως όρισμα είναι κενός, επιστρέφει 0. Αν η λέξη βρεθεί σε κάποιον κόμβο του δένδρου, επιστρέφει το πεδίο appearances του κόμβου αυτού. Αν η λέξη του κειμένου είναι μεγαλύτερη από τη τιμή του κόμβου, καλείται ξανά η search αλλά με πρώτο όρισμα το δεξί παιδί του κόμβου. Αν η λέξη του κειμένου είναι μικρότερη από τη τιμή του κόμβου, τότε καλείται πάλι η search, αλλά με πρώτο όρισμα το αριστερό παιδί του κόμβου.

4.3 Διάσχιση

4.3.1 Ενδοδιατεταγμένη (inorder)

Για κάθε κόμβο, επισκεπτόμαστε πρώτα τους κόμβους του αριστερού του υποδένδρου, έπειτα τον ίδιο τον κόμβο και στη συνέχεια τους κόμβους του δεξιού του υποδένδρου.

4.3.2 Μεταδιατεταγμένη (postorder)

Για κάθε κόμβο, επισκεπτόμαστεπρώτα τους κόμβους του αριστερού του υποδένδρου, έπειτα τους κόμβους του δεξιού του υποδένδρουκαι στη συνέχεια τον ίδιο τον κόμβο.

4.3.3 Προδιατεταγμένη (preorder)

Για κάθε κόμβο, επισκεπτόμαστεπρώτα τον ίδιο τον κόμβο, έπειτα τους κόμβους του αριστερού του υποδένδρουκαι στη συνέχεια τους κόμβους του δεξιού του υποδένδρου.

5. *Πίνακας κατακερματισμού με ανοικτή διεύθυνση*

5.1 Εισαγωγή

Για την αποθήκευση των λέξεων στο hashTable, χρησιμοποιείται struct "HashNode", το οποίο έχει τα εξής πεδία:

- string word: η λέξη που εισάγεται στο hashTable από το αρχείο
- int address: η διεύθυνση στο hashTable για τη δοσμένη λέξη
- int appearances: αριθμός που δείχνει πόσες φορές εμφανίζεται η λέξη στο αρχείο

Για τη δημιουργία του hashTable χρησιμοποιείται η root τύπου hashNode που αναπαριστά το αρχικό στοιχείο του table. Επίσης, υπάρχει καθολική μεταβλητή size που αναπαριστά το μέγεθος του table, που είναι το πολύ 100000. Ο κατασκευαστής δημιουργεί τον καινούριο πίνακα δεικτών size θέσεων και θέτει σε κάθε έναν από αυτούς nullptr. Ο καταστροφέας διαγράφει έναν έναν όλους τους δείκτες του πίνακα.

5.2 Λειτουργίες

5.2.1 Εισαγωγή (insert)

Η insert αρχικά καλεί τη findAddress και καταχωρεί τον αριθμό που επιστρέφει στη μεταβλητή step (int). Η step καθορίζει τη θέση του πίνακα, από την οποία θα αρχίσουμε να ψάχνουμε το πού θα τοποθετηθεί η λέξη. Η while επαναλαμβάνεται μέχρι να βρεθεί μια κενή θέση στον πίνακα για τη λέξη ή η λέξη που δίνεται από το αρχείο στον πίνακα. Αν συμβεί το δεύτερο τότε αυξάνουμε το step κατά μια μονάδα (μετακινούμαστε κατά μία θέση δεξιά του πίνακα). Μετά την επανάληψη, εξετάζουμε τις δύο εξής περιπτώσεις:

- Αν έχει βρεθεί κενή θέση στον πίνακα για τη λέξη, την τοποθετούμε.
- Αν η λέξη υπήρχε από πριν στον πίνακα τότε αυξάνουμε το appearances κατά ένα.

5.2.2 Αναζήτηση (search)

Η main καλεί τη search, η οποία έχει μια μεταβλητή τύπου hashNode, την *pos. Η pos αρχικοποιείται καλώντας τη privSearch. Αν η privSearch επιστρέψει nullptr, επιστρέφει η συνάρτηση 0, αλλιώς επιστρέφει τον αριθμό των εμφανίσεων της λέξης για την οποία γίνεται αναζήτηση. Η privSearch εκτελεί μια παρόμοια διαδικασία με την insert. Βρίσκει τη διεύθυνση της λέξης και την καταχωρεί στη μεταβλητή step και ψάχνει τη λέξη αυτή στον πίνακα. Αν τη βρει επιστρέφει ολόκληρο το hashNode της λέξης, αλλιώς επιστρέφει nullptr.