

ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάθημα: Δομές Δεδομένων

Εργασία Εξαμήνου

Αρτεμάκης Εμμανουήλ, ΑΕΜ: 3859

Έμιλη Μιχαήλ , ΑΕΜ: 2988

# Κλάση ζευγαριού

Για τις υλοποιήσεις των δομών του αταξινόμητου και ταξινομημένου πίνακα καθώς και για τον πίνακα κατακερματισμού υλοποιήθηκε η βοηθητική κλάση myPair η οποία έχει δύο δημόσια πεδία που αναπαριστούν την λέξη (πεδίο word) και την συχνότητα εμφάνισης της (πεδίο occurrences).

# Αταξινόμητος Πίνακας

Για το συγκεκριμένο πρόβλημα αυτή η δομή είχε ως πεδία ένα πίνακα δεικτών της κλάσης myPair (πεδίο A) και τρία πεδία που αφορούν χαρακτηριστικά αυτού του πίνακα (πεδία: size, elements, searchPos). Επίσης ο πίνακας Α υλοποιήθηκε ως πίνακας δεικτών προκειμένου να μην δεσμεύεται μνήμη από αντικείμενα τύπου myPair μέχρι να χρειαστεί.

## Κατασκευαστές

Οι κατασκευαστές που υλοποιήθηκαν είναι 2. Ο κενός που αρχικοποιεί τον πίνακα Α με 10 θέσεις και δίνει αντίστοιχες τιμές στα υπόλοιπα πεδία. Επίσης υλοποιήθηκε και ένας κατασκευαστής που δέχεται ως όρισμα του μέγεθος του αρχικού πίνακα, αλλά δεν χρησιμοποιήθηκε κάπου αλλού στο πρόβλημα.

## Καταστροφέας

Υλοποιήθηκε και καταστροφέας προκειμένου να αποδεσμεύεται η μνήμη που μπορεί να δεσμεύουν τα αντικείμενα στα οποία δείχνουν οι δείκτες του πίνακα Α.

## Μέθοδος bool insert(string & word)

Η λέξη προς καταχώρηση περνάει σαν αναφορική προκειμένου να περνάει ο δείκτης της στις διάφορες μεθόδους της κλάσης και να μην δημιουργούνται διαρκώς νέα αντικείμενα.

Στην μέθοδο αυτή αρχικά γίνεται αναζήτηση της λέξης προκειμένου να ελεγχθεί αν υπάρχει ήδη στον πίνακα και αν υπάρχει απλά προστίθεται ένα στο πεδίο occurrences (+1) του αντικειμένου myPair που έχει την λέξη word. Αν η λέξη δεν υπάρχει τότε δημιουργείται ένα νέο αντικείμενο τύπου myPair και καταχωρείται στην θέση του πίνακα μετά από το τελευταίο στοιχείο του.

Επίσης γίνεται έλεγχος αν ο χώρος των αντικειμένων myPair στον πίνακα υπερβαίνουν το 50% του πίνακα. Αν συμβαίνει αυτό τότε δημιουργείται πίνακας με νέο μέγεθος διπλάσιο από το προηγούμενο και καταχωρούνται τα στοιχεία σε αυτόν.

## Μέθοδος bool search(const string & word, int & occurrences)

Αυτή η μέθοδος χρησιμοποιείται από τον χρήστη της κλάσης προκειμένου να διαπιστώσει αν υπάρχει αντικείμενο που έχει καταχωρημένο την λέξη word και αν ναι τότε να μάθει και τον αριθμό των εμφανίσεων αυτής της λέξης. Αν το αντικείμενο υπάρχει τότε επιστρέφεται True και η μεταβλητή occurrences που έχει περάσει ως αναφορική ενημερώνεται, διαφορετικά επιστρέφεται False και η μεταβλητή occurrences παίρνει τιμή 0.

## Μέθοδος myPair\* search (const string & word)

Αυτή η μέθοδος χρησιμοποιείται από την ίδια την κλάση, ενώ είναι στο μέρος των ιδιωτικών στοιχείων, καθώς θα αντικατασταθεί από άλλη στην κλάση παιδί SortedArray, που όμως θα παίρνει διαφορετικά ορίσματα. Εδώ γίνεται και ενημέρωση του πεδίου searchPos.

Πραγματοποιεί σειριακή αναζήτηση της λέξης που δίνεται, αφού ο πίνακας είναι αταξινόμητος.

## Μέθοδος bool erase(const string & word)

Αυτή η μέθοδος χρησιμοποιείται από τον χρήστη προκειμένου να διαγράψει ένα στοιχείο από τον πίνακα. Αν η διαγραφή ήταν επιτυχής τότε επιστρέφεται true, αλλιώς επιστρέφεται false.

Αν πραγματοποιηθεί διαγραφή, τότε γίνεται μετάθεση των στοιχείων του πίνακα ώστε να συμπληρωθεί η κενή θέση. Στο τέλος της μεθόδου γίνεται και έλεγχος προκειμένου να διαπιστωθεί αν το μέγεθος του πίνακα είναι μικρότερο από το ¼ του μεγέθους. Αν συμβαίνει αυτό τότε γίνεται αλλαγή του μεγέθους του πίνακα στο ½ του αρχικού.

## Υπόλοιπες μέθοδοι

Οι υπόλοιπες μέθοδοι υλοποιήθηκαν για τον έλεγχο της κλάσης, καθώς και για την τελική ενημέρωση του προβλήματος.

* unsigned int getSize(): επιστρέφει το μέγεθος του πίνακα
* unsigned int getElements(): επιστρέφει τον αριθμό των λέξεων στον πίνακα
* unsigned int getOccurrences(): επιστρέφει τις συνολικές εμφανίσεις λέξεων στον πίνακα
* void print(): εκτυπώνει τον πίνακα Α
* string get(int i): επιστρέφει την λέξη στην θέση i του πίνακα Α. Υλοποιήθηκε για να γίνει το σύνολο Q

# Ταξινομημένος Πίνακας

Αυτή η δομή υλοποιήθηκε ως παιδί της δομής Array. Έτσι τα πεδία και οι κατασκευαστές της ήταν τα ίδια με αυτά που περιεγράφηκαν παραπάνω. Παρακάτω περιγράφονται οι μέθοδοι που άλλαξαν προκειμένου να υλοποιηθεί αυτή η δομή.

## Μέθοδος bool insert(string & word)

Αυτή η μέθοδος καλείται από τον χρήση για την καταχώρηση μιας λέξης. Αν η λέξη υπάρχει ήδη μέσα επιστρέφεται false και απλά αυξάνεται η τιμή του πεδίου occurrences του αντικειμένου που περιέχει την λέξη word. Αν δεν υπάρχει, τότε καταχωρείται και επιστρέφεται true.

Στην αρχή γίνονται έλεγχοι προκειμένου να διαπιστωθεί αν η λέξη δεν υπάρχει στον πίνακα. Συγκεκριμένα ελέγχονται με τη σειρά:

* Αν ο αριθμός των στοιχείων είναι 0. Αν ισχύει καταχωρείται η λέξη και σταματάει η μέθοδος
* Αν η λέξη είναι μεγαλύτερη από την τελευταία λέξη
* Αν η λέξη είναι μικρότερη από την 1η λέξη
* Αν υπάρχει η λέξη στον πίνακα

Στην συνέχεια καταχωρείται η λέξη αν δεν υπάρχει στην σωστή θέση που είναι γνωστή από το πεδίο searchPos και γίνεται μετάθεση των στοιχείων του πίνακα. Επίσης και σε αυτή την κλάση υπάρχει αύξηση του μεγέθους του πίνακα αν το πλήθος των στοιχείων του ξεπεράσει το 50% του μεγέθους του πίνακα.

## Μέθοδος bool search(const string & word, int & occurrences)

Αυτή η μέθοδος είναι υλοποιημένη όπως την μέθοδο στον γονέα, με την διαφορά ότι η μέθοδος search στο ιδιωτικό της μέλος καλείται με διαφορετικά ορίσματα.

## Μέθοδος myPair\* search (const string & word, int begin, int end)

Αυτή η μέθοδος πραγματοποιεί δυαδική αναζήτηση αναδρομικά της λέξης word στον πίνακα, αφού είναι ταξινομημένος. Εδώ γίνεται και ενημέρωση του πεδίου searchPos.

## Μέθοδος bool erase(const string & word)

Αυτή η μέθοδος είναι υλοποιημένη όπως την μέθοδο στον γονέα, με την διαφορά ότι η μέθοδος search στο ιδιωτικό της μέλος καλείται με διαφορετικά ορίσματα.

# Πίνακας Κατακερματισμού με ανοιχτή διεύθυνση

Παρόμοια με τις παραπάνω κλάσεις υλοποιήθηκε και αυτή. Αντίστοιχα υπήρχαν τα πεδία size, elements και A, ενώ οι κατασκευαστές και ο καταστροφέας της κλάσης υλοποιήθηκαν με παρόμοια λογική.

## Μέθοδος unsigned int hash(const string & word)

Σε αυτή τη μέθοδο αντιστοιχίζεται η λέξη word ένα μοναδικό κλειδί. Αυτό επιτυγχάνεται διατρέχοντας κάθε γράμμα της λέξης και πολλαπλασιάζοντας σε κάποια δύναμη του 26 (αριθμός των γραμμάτων του αγγλικού αλφαβήτου). Στην συνέχεια επιστρέφεται το υπόλοιπο της διαίρεσης του αριθμού που προκύπτει με το μέγεθος του πίνακα.

## Μέθοδος unsigned int hash2(unsigned int)

Σε αυτή τη μέθοδο επιστρέφεται η θέση τοποθέτησης ή αναζήτησης στοιχείου στον πίνακα, ενώ καλείται σε περίπτωση που η θέση που επιστράφηκε είναι δεσμευμένη (κατά την καταχώριση) ή έχει λέξη και δεν είναι κενός δείκτης (περίπτωση αναζήτησης).

## Μέθοδος bool insert(const string & word)

Αντίστοιχα καταχώρηση της λέξης word και επιστροφή true ή αύξηση του πεδίου occurrences κατά 1 και επιστροφή false.

Στην περίπτωση της καταχώρισης καλείται η συνάρτηση hash για την αντιστοίχιση της λέξης word με ένα μοναδικό κλειδί και με μία θέση στον πίνακα Α και η μέθοδος hash2 (αν χρειαστεί), προκειμένου η λέξη να καταχωρηθεί σωστά.

## Μέθοδος bool search (const string &, int & occurrences)

Αυτή η μέθοδος σε περίπτωση επιτυχούς αναζήτησης επιστρέφει true και καταχωρεί τον αριθμό των εμφανίσεων της λέξης στην μεταβλητή occurrences. Διαφορετικά επιστρέφει false και καταχωρεί 0 στην μεταβλητή occurrences.

## Μέθοδος myPair\* search(const string & word)

Αυτή η μέθοδος καλείται από την μέθοδο search που καλεί ο χρήστης και κάνει αναζήτηση της λέξης στον πίνακα Α αξιοποιώντας τις μεθόδους hash και hash2. Αν βρει την λέξη επιστρέφει το αντίστοιχο αντικείμενο, αλλιώς επιστρέφει ένα κενό δείκτη.

## Υπόλοιπες συναρτήσεις

Οι υπόλοιπες μέθοδοι υλοποιήθηκαν για τον έλεγχο της κλάσης, καθώς και για την τελική ενημέρωση του προβλήματος. Η μέθοδος getSize() και getElements() απλά επιστρέφουν τις τιμές των αντίστοιχων πεδίων. Η getOccurrences() διατρέχει όλο τον πίνακα Α και επιστρέφει τις συνολικές εμφανίσεις των λέξεων. Η μέθοδος print() διατρέχει όλο τον πίνακα Α και εκτυπώνει την κάθε λέξη και τις αντίστοιχες εμφανίσεις της.

# Κλάση Κόμβου

Η κλάση Node (κόμβου) υλοποιήθηκε ώς εσωτερική κλάση της BST και χρησιμοποιήτε για την αναπαράσταση ενος κόμβου. Αποτελείται από τα πεδία occurrences , το οποίο αποθηκεύει τις εμφανίσεις τον λέξεων στο δέντρο, word, left , right (τα παιδιά του κόμβου) και height.

## Κατασκευαστής

Ο κατασκευαστής της Node δέχεται ως όρισμα μια λέξη και στην συνέχεια αρχικοποιεί το occurrences ως 1, το height ως 1 και τα παιδιά left-right ώς nullptr.

## Μέθοδος print()

Η μέθοδος αυτή χρησιμοποιείται για την εκτύπωση του κάθε κόμβου , δηλαδή την λέξη και τις εμφανίσεις που του αντιστοιχεί.

# Δυαδικό Δέντρο Αναζήτησης

Όπως έχει αναφερθεί πιο πάνω, το BST υλοποιήθηκε με την εσωτερική βοηθητική κλάση Node. Έχει ώς πεδίο την Node\* root η οποία αναπαριστά το πρώτο στοιχείο του δέντρου.

## Μέθοδος Node\* Insert (string const&w, Node\* n)

Αυτή η μέθοδος υλοποιήθηκε για την εισαγωγή λέξης στο δέντρο. Καλείται από την Insert (string Const& w) {root = Insert(w,root)} η οποία περνάει σαν όρισμα την root. Η μέθοδος αναδρομικά ελέγχει κάθε φορά αν η εισαγόμενη λέξη είναι μεγαλύτερη ή μικρότερη από τον κόμβο στον οποίο βρίσκεται. Σε περίπτωση που βρεθεί ότι η λέξη υπάρχει ήδη τότε απλώς αυξάνεται το occurrences.

## Μέθοδος bool Search (std::string & w, int & o)

Αυτή η μέθοδος σε περίπτωση επιτυχούς αναζήτησης επιστρέφει true και καταχωρεί τον αριθμό των εμφανίσεων της λέξης στην μεταβλητή o. Διαφορετικά επιστρέφει false.

## Μέθοδος Node\* deleteNode(std::string const& w, Node\* n)

Σε αυτή την μέθοδο αναζητείται αναδρομικά η λέξη προς αφαίρεση w. Μόλις βρεθεί ελέγχεται εάν ο κόμβος είναι leaf node( no children) ή αν έχει 1 ή 2 παιδία . Σε περίπτωση που έχει 1 παιδί τότε απλώς δηλώνουμε εναν temp κόμβο ο οποίος ισούται με αυτόν που θέλουμε να αφαιρέσουμε ,αποθηκεύουμε στον κόμβο το δεξιό ή αριστερό υπόδεντρο και εν τέλη κανουμε delete την temp. Στα 2 παιδιά ψάχνει την μικρότερη τιμή στον δεξιό πίνακα και τον αντικαθηστά.

## Μέθοδος void deleteTree(Node\* n)

Η συγκεκριμένη μέθοδο κάνει delete αναδρομικά το δέντρο αρχίζοντας απο τα subtrees.

## Υπόλοιπες συναρτήσεις

Οι υπόλοιπες μεθόδοι είναι:

* Inorder, Postorder, Preorder: Τυπώνουν το δέντρο με διαφορετικούς μεθόδους.

Ο inorder για κάθε κόμβο ξεκινά από το αριστερό υποδέντρο, συνεχίζει στον ίδιο κόμβο και μετά στο δεξιό υποδέντρο. Ο preorder για κάθε κόμβο ξεκινά από ίδιο κόμβο και στην συνέχεια πάει πρώτα στο αριστερό υποδέντρο και μετά στο δεξί. Ο Postorder για κάθε κόμβο πρώτα αρχίζει από το αριστερό υποδέντρο, στην συνέχεια στο δεξί και τέλος στον ίδιο κόμβο.

* GetOccurences(), getSize(): Επιστρέφει το σύνολο των εμφανισεων όλων των λέξεων και το σύνολο των λέξεων στο δέντρο.
* minValue(Node\*n): Ψάχνει την μικρότερη τιμή στο δέντρο αρχίζοντας απο τον κόμβο n.

# Δυαδικό Δέντρο Αναζήτησης AVL

Η δομή υλοποιήθηκε ως παιδί της BST. Μαζί κληρονομεί και την εσωτερική κλάση Node. Η διαφορά της AVL με την BST είναι ότι όποτε γίνει Insert ή Delete ενος κόμβου τότε γίνονται rotations-balancing έτσι ώστε η διαφορά σε ύψος του δεξιού και αριστερού κόμβου να μην είναι μεγαλύτερος απο 1.

## Μέθοδος Node\* Insert(std::string const& w, Node\* n)

Η διαφορά αυτής της Insert με της BST είναι ότι γίνεται Balancing κάθε φορά.

Η είσοδος γίνετε ώς εξης:

1. Γίνετε κανονικά η Insert όπως την BST
2. Update το height του node και μετά υπολογίζουμε το balance.
3. Αν το δέντρο είναι unbalanced, τοτε υπαρχουν 4 περιπτώσεις που μπορουν να υπάρχουν:
   1. Left Left Case (single right rotation)
   2. Left Right
   3. Right Right
   4. Right left

## Μέθοδος Node\* deleteNode(std::string const& w, Node\* n)

Ίδια με της BST με την διαφορά ότι γίνετε Balancing όπως την Insert το οποίο εξηγείτε παραπάνω.

## Μεθόδοι Rotations

* rightRotate(Node\* &y) : Υλοποιέι την μονή δεξιά περιστροφή. Δηλαδή το δέντρο μετακινείται προς τα δεξιά μια φορά. Σε node x αποθηκεύεται το αριστερό παιδί του y, στην συνέχεια στο αριστερό παιδί αποθηκεύεται το δεξί παιδί του x (το οποίο είναι τώρα το προηγούμενο y->left), και στην συνέχεια αποθηκεύεται το y στον x->right. Έπειτα ανανεώνονται τα ύψη των nodes και επιστρέφεται τον x.
* LeftRotate(Node\*&y): Αντίστοιχα με την rightRotate.
* leftRightRotate(Node\* &n): Καλείται πρώτα η leftRotate στον n->left και μετά η rightRotate.
* rightLeftRotate(Node\* &n): Αντίστοιχα.

## Μέθοδος getBalance(Node\* n)

Επιστρέφει το balance του κόμβου.

## Μέθοδος height(Node\* n) const

Επιστρέφει το ύψος του κόμβου.

# Άλλες συναρτήσεις

Αυτές οι συναρτήσεις χρησιμοποιήθηκαν από την κύρια συνάρτηση, ενώ υλοποιήθηκαν σε ξεχωριστό αρχείο προκειμένου να είναι πιο ευανάγνωστος ο κώδικας.

## Μέθοδος bool isWord(const string &word)

Εξετάζει αν η δοσμένη συμβολοσειρά word είναι λέξη και επιστρέφει ανάλογα true ή false. Τα κριτήρια για να θεωρείται ότι μία συμβολοσειρά είναι λέξη είναι να έχει σώμα χαρακτήρων. Για την διαπίστωση αυτή χρησιμοποιήθηκαν κανονικές εκφράσεις.

## stringstream splitWord(string word, string delimeter)

Έγινε η παρατήρηση ότι οι χαρακτήρες ‘- -‘ μπορεί να χωρίζουν δύο λέξεις. Γι’ αυτό υλοποιήθηκε αυτή η συνάρτηση προκειμένου να διαχωρίζει αυτές τις λέξεις και να επιστρέφει ένα ρεύμα συμβολοσειρών που περιέχει τελικά τις χωρισμένες λέξεις.

## Μέθοδος clearWord(string &word)

Αυτή η συνάρτηση καθαρίζει την λέξη μέσα από κανονικές εκφράσεις από χαρακτήρες που δεν είναι γράμματα και επίσης μετατρέπει όλους τους χαρακτήρες της συμβολοσειράς σε πεζούς.

# Κύρια συνάρτηση

Αυτή η συνάρτηση δέχεται σαν 1ο όρισμα (main.exe “path” ή ./main “path”) το μονοπάτι του αρχείου. Κατά την κύρια συνάρτηση γίνονται οι παρακάτω λειτουργίες:

* Αρχικοποίηση δομών και καταχώριση λέξεων σε αυτές
* Δημιουργία του συνόλου Q από τυχαίες λέξεις παρμένες από την δομή του Αταξινόμητου Πίνακα
* Αναζήτηση και εμφάνιση των λέξεων που αναζητούνται σε κάθε δομή και καταγραφή του χρόνου
* Τελική ενημέρωση του χρήστη για τις αποδόσεις των δομών

# Αποτελέσματα στο small-file.txt

Παρακάτω παρατίθεται ένας πίνακας με τα αποτελέσματα μίας εκτέλεσης του προγράμματος με το small-file.txt.

|  |  |  |  |
| --- | --- | --- | --- |
| Δομή | Χρόνος αναζήτησης (s) | Συνολικές Λέξεις | Συνολικές Εμφανίσεις λέξεων |
| Αταξινόμητος Πίνακας | 0.702086 | 15939 | 244593 |
| Ταξινομημένος Πίνακας | 0.425177 | 15939 | 244593 |
| Πίνακας κατακερματισμού | 0.495997 | 15939 | 244593 |
| Δυαδικό Δέντρο Αναζήτησης | 0.449993 | 15939 | 244593 |
| AVL δέντρο | 0.456438 | 15939 | 244593 |