Τεχνική Έκθεση Εργασίας Δομών Δεδομένων 2020

Μπαρακλιλής Ιωάννης

ΑΕΜ: 3685

Email: imparakl@csd.auth.gr

Κατσάκη Σοφία

ΑΕΜ: 3656

Email: skatsaks@csd.auth.gr

Εισαγωγή: Η συνάρτηση main

Η συνάρτηση main ανοίγει το αρχείο με όνομα “input.txt”, διαβάζει μία προς μία τις γραμμές του αρχείου αυτού και τις αναλύει σε μεμονωμένες λέξεις τις οποίες εισάγει στην δομή HashTable. Παράλληλα, διαλέγει Q_SIZE (ορίζεται στη αρχή με #define) λέξεις που αποτελούν ένα σύνολο Q και τις αναζητά στις παραπάνω δομές. Για κάθε μία δομή δημιουργεί αντίστοιχο αρχείο κειμένου με όνομα “HashTable_output.txt” για την δομή BinarySearchTree, AVL_Tree και HashTable αντίστοιχα όπου τυπώνει τα αποτελέσματα της αναζήτησης κάθε στοιχείου του Q για κάθε δομή με μορφή:

“Word : [λέξη του Q], appeared: [αριθμός εμφανίσεων] times”

Τέλος, δημιουργεί ένα αρχείο με όνομα “Time_Count_output.txt” όπου τυπώνονται οι χρόνοι αναζήτησης (συμπεριλαμβανομένου το χρόνου της εκτύπωσης των αποτελεσμάτων στα αρχεία) σε millisecond όλων των λέξεων του Q για κάθε δομή.

Επίσης, πρέπει να σημειωθεί ότι σε αυτή την υλοποίηση, λέξη θεωρείται ακολουθία αλφαβητικών χαρακτήρων (λατινικών).

Επιπλέον, πρέπει να σημειωθεί ότι αν υπάρξει η αδυναμία να ανοίξει οποιοδήποτε από τα παραπάνω αρχεία, το πρόγραμμα τερματίζει με κωδικό εξόδου 1.

Αναλυτικά η λειτουργία της main επιτελεί τα εξής:

Αρχικά, ανοίγει το αρχείο με όνομα input.txt (αν δεν ανοίξει, το πρόγραμμα τερματίζει με κωδικό εξόδου 1), γίνεται δέσμευση χώρου για το σύνολο Q (θέσεων Q_SIZE που δηλώνεται στην αρχή με #define) και δηλώνεται το HashTable.

Στη συνέχεια, διαβάζονται μία προς μία οι γραμμές του input.txt και εισάγονται στις δομές με τρόπο: Περνώνται ένας-ένας οι χαρακτήρες της γραμμής και αν είναι αλφαβητικοί (λατινικοί) μετατρέπονται (ο έλεγχος για το αν είνας χαρακτήρας είναι αλφαβητικός καθώς η μετατροπή σε πεζό γίνονται με την συνάρτηση processChar,η περιγραφή της οποίας βρίσκεται παρακάτω) σε πεζούς (αν δεν είναι ήδη) και επικολλούνται στο τέλος μίας προσωρινής συμβολοσειράς temp που αρχικά είναι κενή. Αν βρεθεί χαρακτήρας που δεν είναι αλφαβητικός, τότε αν η temp δεν είναι κενή δηλαδή αν περιέχει λέξη προς εισαγωγή (δηλαδή αν υπάρχουν πολλοί μη-αλφαβητικοί χαρακτήρες στην σειρά ώστε δεν εισάγεται το κενό) εισάγεται στις δομές και στη συνέχεια αρχικοποιείται σε κενή συμβολοσειρά για να χρησιμοποιηθεί στην συνέχεια για να εισάγει νέα λέξη. Όταν τελειώσουν οι χαρακτήρες μίας γραμμής γίνεται μία (τελευταία για την γραμμή) προσπάθεια εισαγωγής της temp (εφόσον είναι μη κενή) στις δομές εφόσον μπορεί να μην υπάρχει μη-αλφαβητικός χαρακτήρας στο τέλος της γραμμής που μπορεί να προκαλέσει εισαγωγή της temp στις δομές. Παράλληλα, σε κάθε εισαγωγή λέξης στις δομές καλείται η randomInsert (η περιγραφή της βρίσκεται παρακάτω) που δίνεται η temp ως όρισμα για να προστεθεί πιθανώς η temp στο σύνολο Q.

Μετά, ανοίγει το αρχείο “HashTable_output.txt” και το “Time_Count_output.txt” (αν δεν ανοίξουν, το πρόγραμμα τερματίζει με κωδικό εξόδου 1).

Μετά, γίνεται αναζήτηση των λέξεων του συνόλου Q στην δομή, τυπώνονται τα αποτελέσματα των αιτημάτων αναζήτησης στα αντίστοιχα αρχεία και αποθηκεύονται οι χρόνοι των ερωτημάτων και εκτύπωσης τους στο αρχείο (με την χρήση της βιβλιοθήκης chrono σε nanosecond που μετατρέπονται στην συνέχεια σε millisecond) οι οποίοι τυπώνονται στο τέλος στο αρχείο “Time_Count_output.txt”.

Τέλος κλείνουν όλα τα αρχεία που χρησιμοποίησε το πρόγραμμα.

Στο αρχείο main.cpp περιλαμβάνονται επίσης (όπως αναφέρθηκε και παραπάνω) και οι συναρτήσεις processChar και randomInsert:

- Η συνάρτηση bool processChar(char &c) δέχεται τον χαρακτήρα c ως όρισμα με αναφορά και ελέγχει αν είναι αλφαβητικός. Αν είναι, μετατρέπει τον c σε πεζό (αν δεν είναι ήδη) και επιστρέφει true. Διαφορετικά, αν δηλαδή δεν είναι πεζός, επιστρέφει false και δεν αλλάζει τον c.
- Η συνάρτηση void randomInsert(const std::string &word, std::string *Q, long &currQ) η οποία αποφασίζει αν θα εισαχθεί η word στο σύνολο Q και σε ποια θέση με τον τρόπο: Αν το currQ είναι μικρότερο από Q_SIZE (το Q έχει λιγότερα από Q_SIZE στοιχεία) η word εισάγεται στο Q στην θέση currQ (επόμενη της προηγούμενης που εισήχθηκε προηγουμένως ή στην πρώτη θέση αν είναι η πρώτη λέξη του Q) και αυξάνεται η τιμή του currQ. Ο σκοπός αυτού είναι να «γεμίσει» πρώτα ο πίνακας «σίγουρα» πριν εισάγουμε τυχαία τις λέξεις. Διαφορετικά, με πιθανότητα 15% εισάγεται η word σε μία τυχαία θέση του Q.

Πίνακας Κατακερματισμού με ανοικτή διεύθυνση

Μία σύντομη παρουσίαση του πίνακα κατακερματισμού ανοικτής διεύθυνσης

Αρχικά, εντός του public μέρους της κλάσης HashTable ορίζεται η κλάση Node τα αντικείμενα της οποίας αποτελούν στοιχεία του πίνακα κατακερματισμού με public μέλη:

- word τύπου std::string που αποθηκεύει την συμβολοσειρά
- t_a τύπου int που αποθηκεύει τον αριθμό εμφανίσεων της λέξης που αποθηκεύει το word.
- Έναν κατασκευαστή χωρίς ορίσματα που θέτει ως word την κενή συμβολοσειρά και t_a το 0.
- Έναν κατασκευαστή με δύο ορίσματα, τα t (τύπου int) και w (τύπου std::string) που θέτει ως τιμή του word το w και ως τιμή του t_a το t.
- Έναν (friend) ostream operator << που τυπώνει τα δεδομένα του αντικειμένου στο ρεύμα εξόδου με μορφή: “Word : [συμβολοσειρά αποθηκευμένη], appeared: [αριθμός εμφανίσεων] times”.

Η κλάση HashTable έχει ως μέλη δεδομένων (private) : τον δείκτη ht σε αντικείμενα Node.

Ως μεθόδους public που μπορεί να καλέσει ο χρήστης έχει τις:

- const HashTable::Node* Search(const std::string &k), που δέχεται ως όρισμα ένα std::string k και εκτελεί αναζήτηση στον πίνακα και επιστρέφει const δείκτη σε Node αν βρεθεί η λέξη στον πίνακα. Διαφορετικά, επιστρέφει nullptr.
- bool Insert(const std::string & k), που δέχεται ως όρισμα ένα std::string k και εισάγει την λέξη k στην δομή (προσθήκη νέας λέξης ή αύξηση του μετρητή εμφανίσεων αν ήδη υπάρχει) και επιστρέφει bool τιμή με την κατάσταση της επιτυχίας.

Ως private μεθόδους που δεν έχει πρόσβαση ο χρήστης έχει τις:

- bool p_search(const std::string & k, long &loc), που δέχεται ως όρισμα ένα std::string k και εκτελεί αναζήτηση στον πίνακα με τετραγωνική μέθοδο (τετραγωνική αναζήτηση – quadratic probing) και αν βρεθεί η λέξη k σε κόμβο του πίνακα μεταβάλει την τιμή του loc στην θέση του πίνακα και επιστρέφει true. Διαφορετικά, αν βρεθεί κενό στοιχείο του πίνακα κατά την αναζήτηση (δεν υπάρχει η λέξη σε κανένα κόμβο) μεταβάλει την τιμή του loc στην θέση του κενού αυτού στοιχείου στον πίνακα και επιστρέφει false. Τέλος, αν δεν βρεθεί το στοιχείο στον πίνακα και δεν βρεθεί κενή θέση, τότε μεταβάλλεται η τιμή του loc σε -1 και επιστρέφεται false
- long Hash(const std::string &k), που δέχεται ως όρισμα ένα std::string k και είναι η συνάρτηση κατακερματισμού της δομής όπου επιστρέφει την θέση του πίνακα που αντιστοιχεί το k.

Επιπλέον η κλάση HashTable έχει και

- Constructor χωρίς ορίσματα, δεσμεύει χώρο για τον δείκτη ht.
- Destructor που διαγράφει την δεσμευμένη μνήμη του ht.

Αναλυτική παρουσίαση των μεθόδων της κλάσης HashTable

Η μέθοδος Search

Η μέθοδος Search δέχεται ως όρισμα ένα std::string k (const με αναφορά) και επιστέφει const δείκτη στο στοιχείο του πίνακα (τύπου HashTable::Node) όπου αποθηκεύεται η λέξη k, αν βρεθεί. Αν δεν βρεθεί επιστρέφεται nullptr.

Αναλυτικά,

Αρχικά, καλείται η μέθοδος p_search (η παρουσίαση της βρίσκεται παρακάτω).

Αν έχει επιστρέψει true, σημαίνει ότι η λέξη βρίσκεται ήδη στην δομή στην θέση l (μεταβλητή που δόθηκε σαν όρισμα στην p_search). Σε αυτή την περίπτωση αρκεί να επιστρέψω την διεύθυνση του στοιχείου στην θέση l του πίνακα.

Αν έχει επιστρέψει false, επιστρέφω nullptr (το στοιχείο δεν υπάρχει).

Η μέθοδος p_search

Η μέθοδος p_search δέχεται ως όρισμα ένα std::string k (const με αναφορά) και μία μεταβλητή long loc με αναφορά και μεταβάλει την τιμή του loc να είναι η θέση του στοιχείου που βρέθηκε και επιστρέφει bool μεταβλητή με την κατάσταση της επιτυχίας. Στην πράξη εκτελεί τετραγωνική αναζήτηση.

Αναλυτικά,

Αρχικά, καλεί την συνάρτηση Hash (συνάρτηση κατακερματισμού, η παρουσίαση της βρίσκεται παρακάτω) με όρισμα το k και αποθηκεύει την θέση που επιστρέφει ως μεταβλητή θέσης στοιχείου r. Επίσης, χρησιμοποιεί έναν μετρητή επαναλήψεων c (με αρχική τιμή 0) και έναν μετρητή «επόμενου βήματος» inc (με αρχική τιμή 1).

Στη συνέχεια εκτελείται τετραγωνική αναζήτηση με τον τρόπο:

Όσο ο μετρητής c δεν έχει την τιμή N/2 (αναζητήσεις με μεγαλύτερη τιμή είναι ανώφελες στην τετραγωνική αναζήτηση, συνεπώς όταν φτάσει αυτή την τιμή θεωρούμε την αναζήτηση να έχει τελειώσει) εκτελεί τον βρόχο:

Αν στην μεταβλητή θέσης στοιχείου (r) έχουμε ίδια λέξη με αυτή που ψάχνουμε, τότε τελείωσε η αναζήτηση και μεταβάλω την τιμή του loc στην θέση r και επιστρέφω true.

Αν στην μεταβλητή θέσης στοιχείου (r) έχουμε κενή συμβολοσειρά, τότε η λέξη που ψάχνουμε σίγουρα δεν υπάρχει (διαφορετικά θα την βρίσκαμε σε προηγούμενη επανάληψη) και μεταβάλω την τιμή του loc στην θέση r και επιστρέφω false (το στοιχείο δεν βρέθηκε).

Αν δεν ισχύει τίποτα από τα παραπάνω, τότε αυξάνω το βήμα επανάληψης (c) κατά 1 (επόμενη επανάληψη), μεταβάλω την θέση του r σε (r+inc)%N δηλαδή στην επόμενη επανάληψη θα γίνει έλεγχος στο στοιχείο που δείχνει η νέα τιμή του r. Σημειώνεται εδώ ότι στην τετραγωνική αναζήτηση το inc+=2 με αρχική τιμή 1 και αύξηση της μεταβλητής θέσης στοιχείου κατά inc ισοδυναμεί με την μέθοδο αναζήτησης που δίνεται συνήθως ως ορισμός ((αρχική θέση που επιστρέφει το Hash + βήμα^2) % N) αλλά είναι σαφώς πιο γρήγορη (αφού δεν χρειάζεται να υπολογίσει το τετράγωνο που είναι χρονοβόρα πράξη.

Τέλος, αν η επανάληψη τερματίσει λόγω της συνθήκης επανάληψης, σημαίνει ότι δεν υπάρχει το στοιχείο στον πίνακα ούτε υπάρχει κενή θέση, επομένως τίθεται ως τιμή του loc το -1 και επιστρέφεται false (το στοιχείο δεν βρέθηκε).

Η μέθοδος Hash

Η μέθοδος Hash δέχεται ως όρισμα ένα std::string k (const με αναφορά) και είναι η συνάρτηση κατακερματισμού του πίνακα που επιστρέφει την θέση του πίνακα (long) όπου αντιστοιχεί το k.

Αναλυτικά,

Δεν κάνει τίποτα παραπάνω από το να αθροίζει την τιμή (ascii) κάθε χαρακτήρα του k πολλαπλασιασμένο με το 5 (σταθερή τιμή) υψωμένο στην θέση του αντίστοιχου χαρακτήρα και επιστρέφει αυτό το άθροισμα modulo (τελεστής %) το μέγεθος του πίνακα.

Δηλαδή αν έχω k = $c_0c_1...\ c_{n-1}c_n$ τότε η Hash επιστρέφει ($c_0 + c_1 * 5 + ... + c_{n-1} * 5^{n-1} + c_n * 5^n$) % (μέγεθος πίνακα).

Πρέπει να σημειωθεί ότι αν το μήκος της συμβολοσειράς υπερβαίνει το 25, θεωρείται αυθαίρετα ως μήκος της συμβολοσειράς το 25 (τελευταίος χαρακτήρας συμβολοσειράς στην θέση 24) εξαιτίας περιορισμών μνήμης.

Η μέθοδος Insert

Η μέθοδος Insert δέχεται ως όρισμα ένα std::string k (const με αναφορά) και το εισάγει στην δομή αν δεν υπάρχει, διαφορετικά αυξάνει τον αριθμό εμφανίσεων του στοιχείου που το αποθηκεύει. Τέλος, επιστρέφει bool μεταβλητή με την κατάσταση της επιτυχίας.

Αναλυτικά,

Αρχικά, καλείται η μέθοδος p_search (η παρουσίαση της βρίσκεται παραπάνω).

Αν έχει επιστρέψει true, σημαίνει ότι η λέξη βρίσκεται ήδη στην δομή στην θέση loc (μεταβλητή που δόθηκε σαν όρισμα στην p_search). Σε αυτή την περίπτωση αρκεί να αυξήσω τον μετρητή εμφανίσεων στην θέση loc του πίνακα και επιστρέφω true (επιτυχής εισαγωγή).

Αν έχει επιστρέψει false, τότε αν δεν έχει τιμή loc -1 (αν υπάρχει κενή θέση), τότε αρκεί να δημιουργήσω και να εισάγω στην θέση loc ένα στοιχείο Node με τιμή εμφανίσεων 1 και συμβολοσειρά το k και τελικά επιστρέφω true (επιτυχής εισαγωγή).

Διαφορετικά, δηλαδή σε περίπτωση p_search==false και loc==-1 απλά επιστρέφω false εφόσον δεν υπάρχει κενή θέση για εισαγωγή.

Ο κατασκευαστής της HashTable

Ο κατασκευαστής της HashTable απλά δεσμεύει N θέσεις (ορίζεται στο πάνω μέρος του αρχείου του HashTable.cpp με #define) και τις αναθέτει στον δείκτη ht.

Σημειώνεται ότι έχει οριστεί ως N το 1000003 που κρίνεται επαρκής για την δεδομένη υλοποίηση εφόσον είναι πρώτος αριθμός (έτσι μειώνονται οι πιθανότητες συγκρούσεων) και έχει τουλάχιστον διπλάσιο χώρο από αυτόν που χρειάζεται για αποθήκευση όλων των στοιχείων που θέλουμε, κάτι απαραίτητο για την τετραγωνική αναζήτηση (το λεξικό Webster περιέχει 470000 λέξεις όπου 1000003 >470000*2 οπότε θεωρούμε ότι καλύπτει την συγκεκριμένη υλοποίηση).

Ο καταστροφέας της HashTable

Ο καταστροφέας της HashTable απλά αποδεσμεύει την μνήμη που δείχνει ο δείκτης ht.