Στα πλαίσια του μαθήματος “Δομες Δεδομενων” ζητήθηκε η ανάπτυξη πέντε δομών δεδομένων στην γλώσσα προγραμματισμού “C++” για την αποθήκευση συνεχόμενων ζευγών λέξεων.

1. ΔΥΑΔΙΚΟ ΔΕΝΔΡΟ ΑΝΑΖΗΤΗΣΗΣ
2. ΙΣΟΡΡΟΠΗΜΕΝΟ ΔΕΝΔΡΟ ΑΝΑΖΗΤΗΣΗΣ (AVL)
3. ΑΤΑΞΙΝΟΜΗΤΟΣ ΠΙΝΑΚΑΣ
4. ΤΑΞΙΝΟΜΗΜΕΝΟΣ ΠΙΝΑΚΑΣ
5. ΠΙΝΑΚΑΣ ΚΑΤΑΚΕΡΜΑΤΙΣΜΟΥ ΑΝΟΙΧΤΗΣ ΔΙΕΥΘΥΝΣΗΣ.

Στην συνέχεια να εφαρμοστούν αναζητήσεις σε Q αριθμό λέξεων και να σημειωθούν οι χρόνοι εκτέλεσης και αριθμοί εμφάνισης κάθε λέξης για κάθε δομή ξεχωριστά σε ενα αρχειο κειμενου “output.txt”. Σημειώνεται πως ο αριθμος Q είναι τυχαίος και διαφέρει σε κάθε εκτέλεση. Επιπλέον κατα την έκταση της τεχνικής αναφοράς μια λεξη/συμβολοσειρά θεωρείται ένα ζεύγος συνεχόμενων λέξεων.

Απλό Δυαδικό Δέντρο Αναζήτησης (Αρχειο BinaryTree.h)

Για την αναπαράσταση του δυαδικού δέντρου στο πρόγραμμά κατασκευάστηκε η κλάση node. Κάθε στιγμιότυπο της κλάσης αναπαριστά έναν κόμβο του δένδρου και έχει ως βασικές ιδιότητές τις δύο διευθύνσεις μνήμης των κόμβων-παιδιών, το ένα με μεγαλύτερη και το άλλο με μικρότερη τιμή του κόμβου-γονέα (Η τιμή του κόμβου είναι μία συμβολοσειρά). Η ιδιότητα count απαριθμεί την συχνότητα εμφάνισης μιας συμβολοσειράς. Το αρχικό αντικείμενο το οποίο διαχειρίζεται η main αποτελεί την ρίζα του δένδρου.

Η μέθοδος **node::add(string)** έχει σκοπό την προσθήκη μιας νέας συμβολοσειράς στο δένδρο. Δημιουργεί έναν νέο κόμβο χωρίς φύλλα και τον τοποθετεί στην σωστή θέση σύμφωνα με την τιμή της συμβολοσειράς και τις προβλεπόμενες συγκρίσεις της δομης στο τέλος του δένδρου. Σε περίπτωση που η συμβολοσειρα υπάρχει στο δένδρο τότε ο κόμβος αγνοείται και ο αλγόριθμος αυξάνει την ιδιότητά count κατα μια μονάδα του κόμβου που υπάρχει στο δένδρο με την ίδια συμβολοσειρα.

Τέλος υπάρχει η μέθοδος **node::search(string)** η οποία υλοποιεί αναζήτηση στο δέντρο με στόχο την εύρεση του κόμβου με την συγκεκριμένη συμβολοσειρα και επιστρεφει το πληθος που το συγκεκριμενο ζευγος εισήχθει στην δομη.

Ισορροπημένο Δυαδικό Δέντρο Αναζήτησης (Αρχειο AVL.h)

Για την αναπαράσταση του ισορροπημένου δυαδικού δέντρου στο πρόγραμμα κατασκευάστηκε η κλάση AVL. Κάθε στιγμιότυπο της κλάσης αναπαριστά έναν κόμβο του δέντρου και έχει ως βασικές ιδιότητές τις δύο διευθύνσεις μνήμης των κόμβων-παιδιών, το ένα με μεγαλύτερη και το άλλο με μικρότερη τιμή του κόμβου-γονέα. Η ιδιότητα count αριθμεί την συχνότητά εμφάνισης μιας συμβολοσειράς. Το αρχικό αντικείμενο το οποίο διαχειρίζεται η main αποτελει την ρίζα του δένδρου.

Ο κατασκευαστής της κλάσης δέχεται μια συμβολοσειρα και δημιουργεί έναν κόμβο χωρίς φύλλα.

Η μέθοδος **AVL::add(string)** έχει σκοπό την προσθήκη μιας νέας συμβολοσειρας στο δέντρο. Δημιουργεί έναν νέο κόμβο χωρίς φύλλα και τον τοποθετεί στην σωστή θέση σύμφωνα με την τιμή της συμβολοσειρας και τις προκαθορισμένες συγκρίσεις στο τέλος του δέντρου. Σε περίπτωση που η συμβολοσειρα υπαρχει στο δέντρο τότε ο κόμβος αγνοείται και ο αλγόριθμος αυξάνει την ιδιοτητα count κατα μια μονάδα του κομβου που υπαρχει στο δένδρο με την ίδια συμβολοσειρα.

Στη συνέχεια, με αναδρομικό τρόπο ξεκινώντας από τον τελευταίο κομβο , ελέγχει το βάθος των υποδένδρων κάθε κόμβου και στην περίπτωση που εντοπιστεί μια διάφορα στο βάθος των δυο υποδένδρων μεγαλύτερη της μονάδας, τότε εφαρμόζει τις κατάλληλες ενέργειες ώστε να επέλθει ξανά η ισορροπία, δηλαδή μετα τις διεργασίες η διαφορά των υποδένδρων να είναι μικρότερη ή ιση με το μηδέν κατά απόλυτη τιμή. Τέλος επιστρέφει την διεύθυνσή μνήμης της νέας ρίζας ώστε να αντικαταστήσει το αρχικό κόμβο , ενώ στην περίπτωση που τα υποδενδρα ήταν ισορροπημένα επιστρέφει την διευθυνση μνημης του ίδιου κόμβου .

Τέλος ,η μέθοδος **AVL::search(string)** υλοποιεί αναζήτηση στους κόμβους του δένδρο με στόχο την εύρεση του κόμβου με την ιδια συμβολοσειρά με εκείνη που δέχεται ως όρισμα. Όταν ολοκληρωθεί η εκτέλεση, η μεθοδος επιστρεφει τον αριθμο που το συγκεκριμένο ζεύγος εισήχθει στην δομή.

Μη Ταξινομημένος Πίνακας (Αρχείο UnsortedArray.h)

Για την αναπαράσταση του μη ταξινομημένου πίνακα στο πρόγραμμα κατασκευάστηκε η κλάση UnsortedArray.

Ιδιότητες:

string *pairs: Πίνακας από συμβολοσειρές που περιέχει όλα τα διαφορετικά ζεύγη λέξεων

long long *countsOfPairs: Πίνακας από ακέραιους, που αντιπροσωπεύουν τις συχνότητες εμφάνισης του κάθε ζεύγους στις αντίστοιχες θέσεις τους. Για παράδειγμα, ο ακέραιος που βρίσκεται στο countsOfPairs[0] δείχνει πόσες φορες προστέθηκε, στον πίνακα pairs, το ζεύγος που βρίσκεται στο pairs[0].

long long arraySize: Ακέραιος που αντιπροσωπεύει την ποσότητα μνήμης που έχουμε κατανείμει στους πίνακες.

long long countOfDiffPairs: Ακέραιος που αντιπροσωπεύει το πλήθος διαφορετικών ζεύγων λέξεων που έχουμε προσθέσει.

Μέθοδοι:

UnsortedArray::long long searchPair(string): Ψάχνει γραμμικά, το ζεύγος που δίνεται, μέσα στον πίνακα. Αν το βρει, επιστρέφει την θέση του, αν όχι, επιστρέφει -1.

UnsortedArray::long long countOfPair(string): Επιστρέφει την συχνότητα εμφάνισης του ζεύγους που δίνεται. Αρχικά, ψάχνει το ζεύγος με την UnsortedArray::searchPair(). Αν βρει το ζεύγος, επιστρέφει τον αριθμό που βρίσκεται στην αντίστοιχη θέση στον πίνακα countsOfPairs, αν όχι, επιστρέφει 0.

UnsortedArray::void add(string): Προσθέτει το ζεύγος που δίνεται στον πίνακα. Αρχικά, ψάχνει το ζεύγος με την UnsortedArray::searchPair() κι αν το βρει, υψώνει το "count" του κατά 1 στον πίνακα countsOfPairs. Αν οχι, ελέγχει αν ο πίνακας έχει γεμίσει. Αν είναι γεμάτος, δημιουργεί δύο προσωρινούς πίνακες και αναθέτει τους τρέχων πίνακες σε αυτούς. Ύστερα διπλασιάζει την μνήμη των τρέχων πινάκων και αντιγράφει όλα τα στοιχεία των προσωρινών πινάκων σε αυτούς. Τέλος, προσθέτει το ζεύγος στο τέλος του πίνακα και αναθέτει 1 στο "count" του στον πίνακα countsOfPairs. (Θα μπορούσε να προσθέτει αρκετή μνήμη για ένα μόνο ζεύγος, αντί να την διπλασιάζει κάθε φορά, αλλά αυτό θα ήταν υπερβολικά χρονοβόρο, διότι θα έπρεπε να αντιγράφει όλα τα ζεύγη κάθε φορά και όσα παραπάνω ζεύγη προσθέτει, τόσο πιο αργό γίνεται)

Ταξινομημένος Πίνακας (Αρχείο SortedArray.h)

Για την αναπαράσταση του ταξινομημένου πίνακα στο πρόγραμμα κατασκευάστηκε η κλάση SortedArray. Είναι παράγωγος της UnsortedArray και οι μόνες διαφορές τους είναι οι μέθοδοι searchPair(), countOfPair() και add().

Μέθοδοι:

SortedArray::long long searchPair(string): Ψάχνει δυαδικά, το ζεύγος που δίνεται, μέσα στον πίνακα. Αν το βρει, επιστρέφει την θέση του, αν όχι, επιστρέφει -1.

SortedArray::long long countOfPair(string): Επιστρέφει την συχνότητα εμφάνισης του ζεύγους που δίνεται. Αρχικά, ψάχνει το ζεύγος με την SortedArray::searchPair(). Αν βρει το ζεύγος, επιστρέφει τον αριθμό που βρίσκεται στην αντίστοιχη θέση στον πίνακα countsOfPairs, αν όχι, επιστρέφει 0.

SortedArray::void add(string): Προσθέτει το ζεύγος που δίνεται στον πίνακα. Αρχικά, ψάχνει το ζεύγος με την SortedArray::searchPair() κι αν το βρει, υψώνει το "count" του κατά 1 στον πίνακα countsOfPairs. Αν οχι, ελέγχει αν ο πίνακας έχει γεμίσει. Αν είναι γεμάτος, δημιουργεί δύο προσωρινούς πίνακες και αναθέτει τους τρέχων πίνακες σε αυτούς. Ύστερα διπλασιάζει την μνήμη των τρέχων πινάκων και αντιγράφει όλα τα στοιχεία των προσωρινών πινάκων σε αυτούς. Τέλος, βρίσκει την σωστή θέση του ζεύγους δυαδικά. Ψάχνει δυαδικά τον πίνακα όπως στην SortedArray::searchPair(string) αλλά στο τέλος, όταν γίνει low > high, αντί να επιστρέψει -1, θέτει ως σωστή θέση το low επειδή εκεί θα βρίσκαμε το ζεύγος αν υπήρχε ήδη. Ύστερα, προσθέτει το ζεύγος στη σωστή θέση που βρήκαμε και αναθέτει 1 στο "count" του στον πίνακα countsOfPairs. (Θα μπορούσε να προσθέτει αρκετή μνήμη για ένα μόνο ζεύγος, αντί να την διπλασιάζει κάθε φορά, αλλά αυτό θα ήταν υπερβολικά χρονοβόρο, διότι θα έπρεπε να αντιγράφει όλα τα ζεύγη κάθε φορά και όσα παραπάνω ζεύγη προσθέτει, τόσο πιο αργό γίνεται)

Πίνακας Κατακερματισμού Ανοικτής Διεύθυνσης (Αρχείο HashTable.h)

Για την αναπαράσταση του πίνακα κατακερματισμού στο πρόγραμμα κατασκευάστηκε η κλάση HashTable. Υλοποιήθηκε η τεχνική του διπλού κατακερματισμού, διότι παράγει λιγότερες συγκρούσεις κι επομένως είναι πιο αποτελεσματικός.

Ιδιότητες:

string *table: Πίνακας από συμβολοσειρές που περιέχει όλα τα διαφορετικά ζεύγη λέξεων

long long *countsOfPairs: Πίνακας από ακέραιους, που αντιπροσωπεύουν τις συχνότητες εμφάνισης του κάθε ζεύγους στις αντίστοιχες θέσεις τους. Για παράδειγμα, ο ακέραιος που βρίσκεται στο countsOfPairs[0] δείχνει πόσες φορες προστέθηκε, στον πίνακα table, το ζεύγος που βρίσκεται στο table[0].

long long tableSize: Ακέραιος που αντιπροσωπεύει την ποσότητα μνήμης που έχουμε κατανείμει στους πίνακες.

long long countOfDiffPairs: Ακέραιος που αντιπροσωπεύει το πλήθος διαφορετικών ζεύγων λέξεων που έχουμε προσθέσει.

Μέθοδοι:

HashTable::long long stringToInt(string): Αντιστοιχεί την συμβολοσειρά που δίνεται με έναν ακέραιο. Πολλαπλασιάζει κάθε γράμμα της συμβολοσειράς με έναν θετικό ακέραιο, που διαλέγουμε εμείς, υψωμένο εις τον δείκτη θέσης του γραμματος και τα προσθέτει όλα μαζί. Για τους χαρακτήρες που δεν βρίσκονται στον ascii πίνακα θέτει 255 ως την τιμή τους. Δύο ίδιες συμβολοσειρές επιστρέφουν πάντα τον ίδιο ακεραιο που είναι αναγκαίο για την σωστή καταμέτρηση των ζευγών. Όμως, δύο ίδιοι ακέραιοι δεν σημαίνει ότι προήλθαν από την ίδια συμβολοσειρά. Αυτό δεν είναι μεγάλο πρόβλημα, απλά θα συμβούν παραπάνω συγκρούσεις στις συναρτήσεις κατακερματισμού. Είναι σημαντικό να τις ελαχιστοποιήσουμε και για αυτό διαλέξαμε έναν πολύ μεγάλο πρώτο αριθμό σαν όριο(1e12 + 3) και έναν πρώτο αριθμό κοντά στο πλήθος χαρακτήρων του αλφαβήτου των συμβολοσειρών(47 λόγω των λατινικών γραμμάτων που βρήκαμε). (Σύμφωνα με https://cp-algorithms.com/string/string-hashing.html#calculation-of-the-hash-of-a-string) Αυτή η διαδικασία θα μπορούσε να ενσωματωθεί στις συναρτήσεις κατακερματισμού, αλλά ο κώδικας είναι πιο απλός και κατανοητός με αυτό τον τρόπο.

HashTable::long long hashFunc1(long long): Απλή συνάρτηση κατακερματισμού.

HashTable::long long hashFunc2(long long): Απλή συνάρτηση κατακερματισμού.

HashTable::long long doubleHashFunc(long long, long long): Σύνθετη συνάρτηση κατακερματισμού που χρησιμοποιεί τις δύο απλές παραπάνω.

HashTable::long long nextTableSize(): Επιστρέφει το επόμενο ιδανικό μέγεθος του πίνακα. Είναι σημαντικό το μέγεθος του πίνακα να είναι πρώτος αριθμός για να προσπελαστούν όλες οι θέσεις του πίνακα, καθώς και αρκετά μεγαλύτερος από το πλήθος διαφορετικών ζευγών για να μην συμβούν πολλές συγκρούσεις. Για αυτό επιστρέφει τον επόμενο πρώτο αριθμό που βρίσκει μετά το διπλάσιο του τρέχων μεγέθους.

HashTable::long long searchPair(string): Ψάχνει τον πίνακα, για το ζεύγος που δίνεται, σύμφωνα με την σύνθετη συνάρτηση κατακερματισμού αυξάνοντας το i κατά 1 κάθε φορα που αποτυγχάνει. Αν το βρει, επιστρέφει την θέση του, αν όχι, επιστρέφει -1. Αν βρει την κενή συμβολοσειρά “” σε μια απο τις θέσεις που ψάχνει, αυτό σημαίνει ότι η συμβολοσειρά δεν υπάρχει, εφόσον θα έπρεπε να βρίσκεται εκεί, κι άρα επιστρέφει -1.

HashTable::long long countOfPair(string): Επιστρέφει την συχνότητα εμφάνισης του ζεύγους που δίνεται. Αρχικά, ψάχνει το ζεύγος με την HashTable::searchPair(). Αν βρει το ζεύγος, επιστρέφει τον αριθμό που βρίσκεται στην αντίστοιχη θέση στον πίνακα countsOfPairs, αν όχι, επιστρέφει 0.

HashTable::void add(string): Προσθέτει το ζεύγος που δίνεται στον πίνακα. Αρχικά, ψάχνει το ζεύγος με την HashTable::searchPair() και αν το βρει, υψώνει το “count” του κατά 1 στον πίνακα countsOfPairs. Αν οχι, ελέγχει αν ο μισός πίνακας έχει γεμίσει. Αν ναι, δημιουργεί ένα προσωρινό HashTable αντικείμενο και αναθέτει το τρέχων αντικείμενο σε αυτό. Αναθέτει επιπλέον μνήμη με την βοήθεια της HashTable::nextTableSize() και προσθέτει όλα τα ζεύγη του προσωρινού αντικειμένου στο τρέχων. Τέλος, προσθέτει το ζεύγος σύμφωνα με την σύνθετη συνάρτηση κατακερματισμού αυξάνοντας το i κατά 1 κάθε φορα που αποτυγχάνει και αναθέτει 1 στο “count” του στον πίνακα countsOfPairs. (Θα μπορούσε να προσθέτει αρκετή μνήμη για ένα μόνο ζεύγος, αντί να χρησιμοποιεί

την nextTableSize() κάθε φορά, αλλά αυτό θα καθιστούσε τον κατακερματισμό ανούσιο, καθώς θα συνέβαιναν πολλές συγκρούσεις συνέχεια)