Δομές Δεδομένων

Γραμμένος Θεόδωρος, Μπούας Χρήστος

A.E.M.: 3294 - 3315

Περιεχόμενα

Εισαγωγή

Στο κειμενό αυτό περιγράφεται η υλοποίηση της εργασίας μας στο μάθημα των δομών δεδομένων, η οποία αφορούσε την υλοποίηση κάποιων δομών σε C++. Αρχικά, αναλύονται οι δομές και οι μέθοδοί τους (ο τρόπος λειτουργίας τους και ο σκοπός τους). Έπειτα περιγράφεται ο τρόπος λειτουργίας της συνάρτησης main.

Υλοποίηση δομών

Συνολικά υλοποιήθηκαν 7 δομές, οι 5 που περιέχονταν στην εργασία και 2 βοηθητικές(στοίβα και λίστα). Όλες οι δομές υπάγονται σε μία superclass DataStructure η οποία περιέχει 2 virtual μεθόδους τις οποίες υιοθετούν όλες οι δομές: insert και getSize.

1 Δέντρο AVL

Για το δέντρο AVL δημιουργήθηκε ένα struct AVLNode το οποίο παριστάνει έναν κόμβο του δένδρου και αποτελείται από:

- Δύο pointers στο αριστερό και δεξί παιδί του κόμβου.
- Μία μεταβλητή height με το ύψος του υποδένδρου που ορίζεται από αυτόν τον κόμβο.
- Μία μεταβλητή value με το περιεχόμενο του κόμβου.

numOfElements	Ο αριθμός των στοιχείων στο δένδρο
root	Δείκτης στη ρίζα του δένδρου

Πίνακας 1: Μεταβλητές της κλάσης

Ο παράγοντας ισορροπίας(bf) ενός κόμβου είναι:
$bf =$ ύψος αριστερού υποδένδρου $-$ ύψος δεξιού υποδένδρου

Μέθοδοι

1.1 newnode

Η συνάρτηση newnode δημιουργεί ένα νέο κόμβο και του αρχικοποιεί τις τιμές του.

1.2 search

Η συνάρτηση search εντοπίζει εάν ένα στοιχείο υπάρχει στο δένδρο και επιστρέφει true ή false.

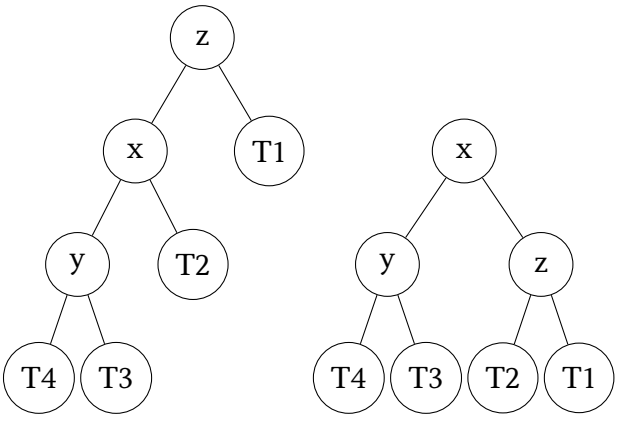
1.3 getHeight & getBalanceFactor

Οι βοηθητικές αυτές συναρτήσεις υπολογίζουν: το ύψος του υποδένδρου που έχει ρίζα τον κόμβο που δίνεται ως όρισμα και τον παράγοντα ισορροπίας του δοσμένου κόμβου. Χρησιμοποιούμε τις συναρτήσεις αυτές γιατί σε κάποιες περιπτώσεις χρειάζεται να υπολογίσουμε αυτές τις τιμές για την τιμή NULL και δεν μπορούμε να εξετάσουμε απευθείας την τιμή height ενός κόμβου.

1.4 recursiveInsert

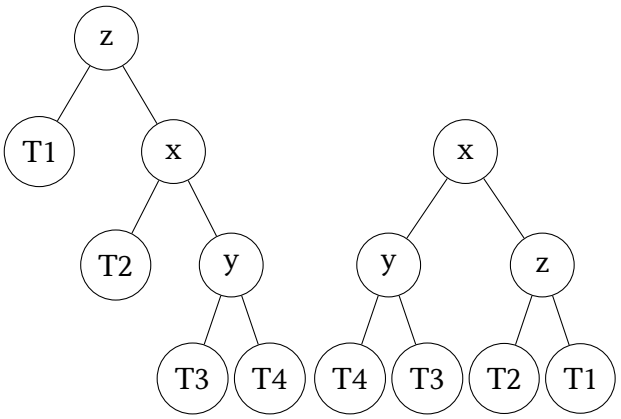
Η συνάρτηση αυτή εισάγει αναδρομικά ένα στοιχείο στο δένδρο AVL. Πρώτα ελέγχεται εάν δοθεί ως όρισμα NULL pointer. Τότε, βρισκόμαστε στη σωστή θέση του δένδρου και απλώς δημιουγείται ένας νέος κόμβος μέσω της newnode. Έπειτα εξετάζεται εάν χρειάζεται να μετακινηθούμε πιο κάτω στο δένδρο. Σε αυτή την περίπτωση ξανακαλείται η συνάρτηση με όρισμα το δεξί ή το αριστερό παιδί του κόμβου που βρισκόμαστε και αναθέτουμε τον κόμβο που επιστρέφει στο αντίστοιχο παιδί. Αφού τοποθετηθεί το στοιχείο στη σωστή θέση, αρχίζει να "μαζεύεται" η αναδρομή. Για κάθε κόμβο στη διαδρομή έως το στοιχείο ενημερώνουμε το ύψος του και βρίσκουμε τον παράγοντα ισορροπίας του. Στη συνέχεια εξετάζεται εάν υπάρχει ανισορροπία στον κόμβο και εκτελούνται οι απαραίτητες περιστροφές:

- Αν ο παράγοντας ισορροπίας του κόμβου είναι > 1 και το στοιχείο τοποθετήθηκε στα αριστερά του αριστερού παιδιού εκτελείται μια απλή δεξιά περιστροφή.



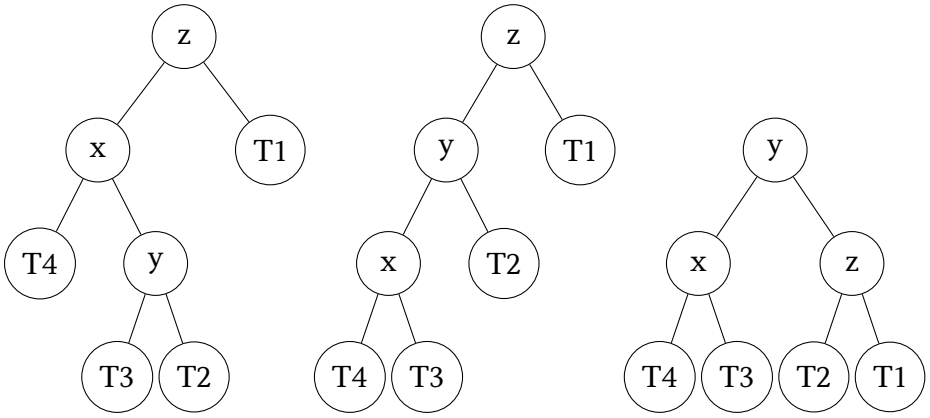
Σχήμα 1: Δεξιά περιστροφή

- Αν ο παράγοντας ισορροπίας του κόμβου είναι -1< και το στοιχείο τοποθετήθηκε στα δεξιά του δεξιού παιδιού εκτελείται μια απλή αριστερή περιστροφή.



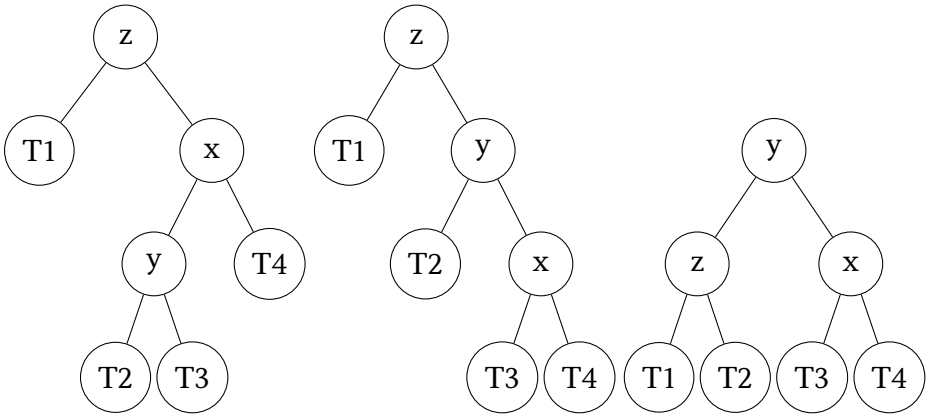
Σχήμα 2: Αριστερή περιστροφή

- Αν ο παράγοντας ισορροπίας του κόμβου είναι > 1 και το στοιχείο τοποθετήθηκε στα δεξιά του αριστερού παιδιού εκτελείται μια αριστερή περιστροφή στο αριστερό παιδί και μία δεξιά περιστροφή στον κόμβο.



Σχήμα 3: Αριστερή-δεξιά περιστροφή

- Αν ο παράγοντας ισορροπίας του κόμβου είναι -1< και το στοιχείο τοποθετήθηκε στα αριστερά του δεξιού παιδιού εκτελείται μια δεξιά περιστροφή στο δεξί παιδί και μια αριστερή περιστροφή στον κόμβο.



Σχήμα 4: Αριστερή-δεξιά περιστροφή

Σε κάθε περίπτωση επιστρέφεται η νέα ρίζα του υποδένδρου για να τοποθετηθεί στη σωστή θέση από το επόμενο instance της συνάρτησης που θα εκτελεστεί.

1.5 insert

Η συνάρτηση insert καλεί την recursiveInsert δίνοντάς της ως όρισμα το στοιχείο που θα εισαχθεί και έναν δείκτη στη ρίζα του δέντρου.

1.6 leftrotate & rightrotate

Οι συναρτήσεις rightrotate και leftrotate υλοποιούν την δεξιά και αριστερή περιστροφή, όπως φαίνονται στα σχήματα 1 και 2 αντίστοιχα.

1.7 remove

Η συνάρτηση remove αφαιρεί ένα στοιχείο από το δένδρο. Σε αυτή τη συνάρτηση χρησιμοποιείται η στοίβα που υλοποιήθηκε προκειμένου να αποθηκευτούν οι κόμβοι που διασχίζονται μέχρι να βρεθεί το στοιχείο(το μονοπάτι έως το στοιχείο). Αφού εντοπιστεί το στοιχείο διακρίνονται 3 περιπτώσεις, ανάλογα με τον αριθμό των παιδιών του κόμβου που περιέχει το στοιχείο:

- Εάν ο κόμβος δεν έχει παιδιά τότε απλώς διαγράφεται ο κόμβος και ενημερώνεται ο γονέας του
- Εάν ο κόμβος έχει 1 παιδί τότε το παιδί τοποθετείται στη θέση του κόμβου

- Εάν ο κόμβος έχει 2 παιδιά τότε τοποθετούμε στη θέση του διαγραμμένου στοιχείου, το μεγαλύτερο στοιχείο του αριστερού υποδένδρου. Σε αυτήν την περίπτωση χρησιμοποιείται η συνάρτηση deleteLargestSubtreeElement.

Στη συνέχεια ελέγχονται όλα τα στοιχεία στη στοίβα για τυχόν ανισορροπίες που προέκυψαν. Εάν εντοπιστούν, πραγματοποιούνται οι απαραίτητες περιστροφές.

1.8 deleteLargestSubtreeElement

Η συνάρτηση δέχεται ως όρισμα έναν κόμβο και τον γονέα του. Εντοπίζει το μεγαλύτερο στοιχείο στο υποδένδρο που ορίζεται από αυτόν τον κόμβο, το διαγράφει και επιστρέφει την τιμή του.

1.9 findmin

Επιστρέφει το ελάχιστο στοιχείο στο δέντρο, το οποίο βρίσκεται κάτω αριστερά σε αυτό.

2 Maxheap & Minheap

Οι υλοποιήσεις διαφέρουν στο ότι στο maxheap χρησιμοποιείται το vector ενώ στο minheap γίνεται δυναμική διαχείριση “με το χέρι”.

Μέθοδοι

2.1 insert

Η μέθοδος insert εισάγει ένα νέο στοιχείο στο σωρό. Το στοιχείο αρχικά εισάγεται στο τέλος του πίνακα/vector και “ανεβαίνει” στο δένδρο όσο είναι μικρότερο/ μεγαλύτερο από το γονέα του, ώστε να τοποθετηθεί στη σωστή θέση.

2.2 removeMin/removeMax

Οι μέθοδοι αυτοί αφαιρούν το ελάχιστο/μέγιστο στοιχείο από το δένδρο. Για να γίνει αυτό, αρχικά μετακινείται το τελευταίο στοιχείο του πίνακα στη ρίζα και στη συνέχεια το στοιχείο “κατεβαίνει” στο δέντρο με συνεχείς αντιμεταθέσεις, φέρνοντας το μέγιστο/ελάχιστο στη ρίζα και διατηρώντας τις ιδιότητες του σωρού.

3 Hashtable

Η υλοποίηση του έγινε με διπλό κατακερματισμό.

Μεταβλητές

Η κλάση που υλοποιεί τη δομή περιέχει τις μεταβλητές:

- Τον πίνακα, table, στον οποίο αποθηκεύονται τα στοιχεία. Ο πίνακας δεσμεύεται δυναμικά.
- Έναν πίνακα bool, hash_keys, στον οποίο δηλώνονται οι κατειλημμένες θέσεις του παραπάνω πίνακα. Το μέγεθος του πίνακα ορίζεται δυναμικά.
- Μία μεταβλητή που αποθηκεύει το μέγεθος του πίνακα, size.
- Μία μεταβλητή που μετράει τον αριθμό των στοιχείων στον πίνακα, numOfElements.
- Μία μεταβλητή τύπου float που αποθηκεύει το τρέχον ποσοστό στοιχείων στον πίνακα, spaceUsed. Η spaceUsed ορίζεται ως: numOfElements/size.

Μέθοδοι

Η δομή υποστηρίζει τις εξής μεθόδους:

- Έναν κενό κατασκευαστή, ο οποίος: δεσμεύει χώρο για τους δύο πίνακες (8 θέσεις στον καθένα), ενημερώνει το size, αρχικοποιεί όλες τις θέσεις του hash_keys σε false, ορίζει τα numOfElements και spaceUsed σε 0.
- Εναν getter: getsize που επιστρέφει τον αριθμό των στοιχείων που υπάρχουν στον πίνακα.
- Δύο Hash συναρτήσεις, για τον διπλό κατακερματισμό. Οι συναρτήσεις παίρνουν ως όρισμα μία τιμή, και επιστρέφουν το Hash Key της.
- Μία συνάρτηση που υλοποιεί διπλό κατακερματισμό, doubleHashing(value, i). Όπου i ο πολλαπλασιαστής της δεύτερης Hash συνάρτησης.
- Μία συνάρτηση που διπλασιάζει το μέγεθος του πίνακα, Hash_Table_X2(). Η συνάρτηση: 1) δημιουργεί προσωρινό πίνακα-αντίγραφο του table, 2) δεσμεύει διπλάσιο χώρο για τους πίνακες table και hash_keys, 3) αντιγράφει τα στοιχεία πίσω στον αρχικό πίνακα table, χρησιμοποιώντας διπλό κατακερματισμό.

- Μία συνάρτηση αναζήτησης της θέσης ζητούμενου στοιχείου, getElementPos(value), η οποία αναζητά με την αντίστροφη διαδικασία από αυτή του διπλού κατακερματισμού, τη θέση του value, στον πίνακα.
- Συνάρτηση insert, η οποία εισάγει ένα στοιχείο στον πίνακα. Αρχικά ελέγχει αν ο χώρος που καταλαμβάνεται, spaceUsed, είναι μεγαλύτερος του 50%. Αν είναι, τον διπλασιάζει μέσω της Hash_Table_X2(). Αν δεν είναι, πραγματοποιεί εισαγωγή, χρησιμοποιώντας την doubleHashing().
- Συνάρτηση search, η οποία αναζητά ένα στοιχείο στον πίνακα, και επιστρέφει μια bool τιμή ανάλογα. Πραγματοποιεί την αντίστροφη διαδικασία από αυτή του διπλού κατακερματισμού.

4 Γράφημα

Η υλοποίησή του έγινε με λίστες γειτονικότητας.

Αρχικά, δημιουργήθηκε ένα struct Vertex, που αναπαριστά τον κόμβο του γραφήματος. Στο struct υπάρχουν: α) ένας δείκτης σε λίστα (η λίστα με τους γείτονες του κόμβου), και β) το id του κόμβου (ακέραιος αριθμός).

Μεταβλητές

Το γράφημα περιλαμβάνει τις εξής μεταβλητές:

- Έναν πίνακα, κάθε θέση του οποίου είναι δείκτης σε Vertex. Το μέγεθος του πίνακα δεσμεύεται δυναμικά.
- Δύο μεταβλητές, vertices και edges, οι οποίες αποθηκεύουν τον αριθμό κόμβων και ακμών αντίστοιχα.
- Μία μεταβλητή, capacity, που αποθηκεύει το μέγεθος του πίνακα.
- Πίνακες S και distance, βοηθητικοί για την Dijkstra(). Ο S αποθηκεύει ποιες κορυφές έχουμε ελέγξει, ενώ ο distance αποθηκεύει το μικρότερο κόστος για να φτάσουμε από την αρχική κορυφή σε όλες τις άλλες.

Μέθοδοι

- Έναν κενό κατασκευαστή.
- Getters για τον αριθμό των κόμβων και των ακμών.

- Μία συνάρτηση, findpos(id), που παίρνει ως όρισμα ένα id κόμβου και επιστρέφει τη θέση του κόμβου στον πίνακα (εάν υπάρχει).
- Μία συνάρτηση, deg(id), που επιστρέφει τον βαθμό της κορυφής. Χρησιμοποιεί την findpos για να βρει τη θέση της κορυφής στον πίνακα και έπειτα απαριθμεί τους κόμβους της λίστας που εκτείνεται από τη συγκεκριμένη θέση του πίνακα.
- Δύο συναρτήσεις, vertexExists(id) και edgeExists(v1,v2), που επιστρέφουν αν υπάρχει αυτή κορυφή ή, αντίστοιχα, η ακμή.
- Μία συνάρτηση, insertVertex(id), που εισάγει κόμβο με το συγκεκριμένο id. Ελέγχει αν υπάρχει αρκετός χώρος στον πίνακα. Έπειτα δημιουργεί νέο Vertex.
- Μία συνάρτηση, insertEdge(v1,v2), που εισάγει ακμή ανάμεσα σε δύο κόμβους. Ελέγχει αν υπάρχουν και οι δύο κόμβοι. Αν δεν υπάρχει κάποιος από τους δύο, τον δημιουργεί καλώντας την insertVertex(id).
- Μία συνάρτηση, removeEdge(v1,v2), που διαγράφει μια συγκεκριμένη ακμή.
- Συναρτήσεις DFS και RDFS, που υλοποιούν τους αντίστοιχους αλγορίθμους.
- Συνάρτηση Dijkstra(id,id), που υλοποιεί τον αλγόριθμο του Dijkstra.
- Συνάρτηση pickNode(), βοηθητική για την Dijkstra().
- Συνάρτηση, connectedComponents(), που βρίσκει τα συνεκτικά κομμάτια του γραφήματος.
- Συνάρτηση MST(), για την εύρεση του κόστους του ελάχιστου εκτεινόμενου δέντρου.
- Συνάρτηση, increaseSIze(), που αυξάνει το μέγεθος του πίνακα κατά 10 θέσεις.

Η συνάρτηση main

Η συνάρτηση main διαβάζει τις εντολές του αρχείου commands.txt, τις εκτελεί και γράφει το αποτέλεσμα στο αρχείο output.txt. Για κάθε γραμμή του

αρχείου, εξετάζει το περιεχόμενό της και αντιστοιχίζει στη δομή που αναφέρεται έναν αριθμό χρησιμοποιώντας τη συνάρτηση
structuretonumber. Έπειτα, καλεί τις αντίστοιχες μεθόδους. Αξίζει να αναφερθούμε στον τρόπο με τον οποίο αντιμετωπίζεται η ενολή BUILD: Για όλες τις δομές εκτός του γραφήματος, εκμεταλευόμαστε το γεγονός ότι υπάγονται στην abstract κλάση DataStructure η οποία υποστηρίζει τη μέθοδο insert. Έτσι, έχουμε την συνάρτηση buildFromFile η οποία δέχεται ως ορίσματα το αρχείο που περιέχει τις τιμές και ένα γενικό αντικείμενο τύπου DataStructure. Το γράφημα χρειάζεται ειδική μεταχείριση καθώς το insert μπορεί να αναφέρεται είτε σε κορυφή, είτε σε ακμή και η διάταξη του input file είναι διαφορετική. Για αυτόν το λόγο έχουμε κάνει overload τη συνάρτηση buildFromFile ώστε εάν δέχεται ως δεύτερο όρισμα Graph, να λειτουργεί διαφορετικά. Για τη χρονομέτρηση χρησιμοποιείται η βιβλιοθήκη chrono της C++.

Πως συνεργαστήκαμε

Αρχικά, ο καθένας μας ανέλαβε να υλοποιήσει συγκεκριμένες δομές και τη συνάρτηση main. Δημιουργήσαμε ένα repository στο GitHub όπου ο καθένας μας τοποθέτησε τον κώδικά του, επιτρέποντας στον άλλον να τον ελέγξει και να τον δοκιμάσει. Για το χτίσιμο της εργασίας χρησιμοποιείται το εργαλείο CMake.