Εργασία Εξαμήνου - Δομές Δεδομένων/C++ - 4619/4735

Η γενική δομή της εργασίας ξεκινάει διαβάζοντας γραμμές από το “commands.txt” και αποθηκεύοντας κάθε λέξη σε έναν πίνακα από λέξεις (5 το πολύ καθώς κάθε εντολή έχει 5 ή λιγότερες λέξεις)
Αρχικά δημιουργούμε ένα αντικείμενο από κάθε δομή έτσι ώστε να υπάρχει στην μνήμη σε κάθε επανάληψη του while loop.
Το filename είναι κάθε φορά στα δικά μας test ένα από τα BUILD_MINHEAP.txt, BUILD_MAXHEAP.txt, BUILD_GRAPH.txt, BUILD_AVLTREE.txt, BUILD_HASHTABLE.txt

MinHeap:

- BUILD MINHEAP BUILD_MINHEAP.txt: κάνουμε insert κάθε φορά ένα στοιχείο του array με την βοήθεια της συνάρτησης HeapifyMin η οποία έχει O(logn) άρα σύνολο η πολυπλοκότητα του build είναι O(nlogn).
- GETSIZE MINHEAP: επιστρέφει τον αριθμό των στοιχείων του σωρού με κόστος O(n) το οποίο πληρώνουμε μία φορά στην αρχή όταν κάνουμε build τον σωρό.
- FINDMIN MINHEAP: επιστρέφει το μικρότερο στοιχείο το οποίο είναι στην κορυφή του σωρού με κόστος O(1)
- INSERT MINHEAP number: προσθέτει τον number στο τέλος του σωρού και κάνει HeapifyMin για να συνεχίσουμε να έχουμε σωρό ελαχίστων με κόστος εισαγωγής O(logn). Η συνάρτηση αυτή είχε πρόβλημα στο replit.com παρόλλο που είναι ίδια με την αντίστοιχη του max heap ενώ στο code blocks έτρεχε κανονικά, οπότε αν δημιουργηθεί κάποιο πρόβλημα μπορείτε να την κάνετε comment out και στο commands.txt και στην main.cpp και να μην χρησιμοποιηθεί.
- DELETEMIN MAXHEAP number: O(1) κόστος για διαγραφή της κορυφής του σωρού και O(logn) για την HeapifyMin για να διορθωθεί ο σωρός.

MaxHeap:

- BUILD MAXHEAP BUILD_MAXHEAP.txt: κάνουμε insert κάθε φορά ένα στοιχείο του array με την βοήθεια της συνάρτησης HeapifyMax η οποία έχει O(logn) άρα σύνολο η πολυπλοκότητα του build είναι O(nlogn).
- GETSIZE MAXHEAP: επιστρέφει τον αριθμό των στοιχείων του σωρού με κόστος O(n) το οποίο πληρώνουμε μία φορά στην αρχή όταν κάνουμε build τον σωρό.
- FINDMIN MAXHEAP: επιστρέφει το μεγαύτερο στοιχείο το οποίο είναι στην κορυφή του σωρού με κόστος O(1)
- INSERT MAXHEAP number: προσθέτει τον number στο τέλος του σωρού και κάνει HeapifyMax για να συνεχίσουμε να έχουμε σωρό ελαχίστων με κόστος εισαγωγής O(logn)
- DELETEMAX MAXHEAP number: O(1) κόστος για διαγραφή της κορυφής του σωρού και O(logn) για την HeapifyMax για να διορθωθεί ο σωρός.

Graph:
Γενικά οι κορυφές και οι ακμές έχουν θετικές ακέραιες τιμές και όχι σύνολο ακμών πάνω από 100000. Επίσης μία κορυφή που δεν ενώνεται με καμία άλλη δεν μπορεί να υπολογιστεί και

αντιμετωπίζεται σαν να μην υπάρχει. Η Bubble Sort με κόστος O(n^2) χρησιμοποιείται όπου χρειάζεται κάποια ταξινόμηση.

- BUILD GRAPH BUILD_GRAPH.txt: χωρίζουμε τα δεδομένα σε nodes1, nodes2, edges με κόστος O(n) και επίσης αρχικοποιούμε σε -1 όλα τα στοιχεία της discovered array η οποία θα είναι βοηθητική αργότερα με κόστος και αυτή η διαδικασία O(n).
- GETSIZE GRAPH: επιστρέφει τα διαφορετικά Nodes και Edges με κόστος O(n^2) και O(n) αντίστοιχα.
- COMPUTESHORTESTPATH GRAPH number1 number2: αν οι δύο κορυφές δεν ενώνονται επιστρέφουμε ότι δεν ενώνονται. Αρχικοποιούμε τα discovered σε -1 όλα (άρα δεν μπορούμε να έχουμε κορυφή με αριθμό -1) και επίσης οι κορυφές έχουν σύνολο βάρη μικρότερα του 100000 το οποίο χρησιμοποιείται ως αρχικό shortest path. Έπειτα η CSP, η οποία δουλεύει αναδρομικά, ελέγχει όλους του συνδυασμούς από το number1 στο number2 και επιστρέφει τον λιγότερο κοστοβόρο με κόστος O(n^2) χρησιμοποιώντας τις discoveredAdd και discoveredRemove οι οποίες έχουν κόστος O(n) καθώς και την notDiscovered η οποία έχει και αυτή κόστος O(n).
- COMPUTESPANNINGTREE GRAPH: μπορεί να χρησιμοποιηθεί μόνο εάν ο γράφος αποτελείται από μία συνιστώσα και όχι παραπάνω. Χρησιμοποιείται ο αλγόριθμος του Prim μέσω της συνάρτησης MST με κόστος O(n^2) και οι συναρτήσεις discoveredAdd, discoveredRemove και η notDiscovered με κόστος O(n) η καθεμία. Τέλος επιστρέφεται το συνολικό βάρος του μικρότερου minimum spanning tree.
- FINDCONNECTEDCOMPONENTS GRAPH: μέσω της συνάρτησης FCC αναδρομικά υπολογίζεται ο αριθμός των συνιστωσών με O(n^2).
- INSERT GRAPH number1 number2 weight: αν δεν υπάρχει το weight το πρόγραμμα δεν μπορεί να τρέξει και θα τερματίσει. Αλλιώς, προσθέτει την ακμή με βάρος weight μεταξύ των 2 nodes number1-number2 με κόστος O(n). Αν η ακμή υπάρχει ήδη επιστρέφεται αντίστοιχο μήνυμα.
- DELETE GRAPH number1 number2: διαγράφει την ακμή των number1-number2 με κόστος O(n) και αν η ακμή δεν υπάρχει επιστρέφεται αντίστοιχο μήνυμα.

AvlTree:

- BUILD AVLTREE BUILD_AVLTREE.txt: κάνουμε insert κάθε φορά ένα στοιχείο του δέντρου με την βοήθεια της συνάρτησης insert η οποία έχει πολυπλοκότητα O(logn) .
- GETSIZE AVLTREE: επιστρέφει τον αριθμό των στοιχείων του δέντρου με κόστος O(n) .
- FINDMIN AVLTREE: επιστρέφει το μικρότερο στοιχείο του δέντρου με κόστος O(logn).
- INSERT AVLTREE number: προσθέτει τον αριθμό στο δέντρο και κάνει τα κατάλληλα rotations ώστε το δέντρο να παραμείνει avl. Το κόστος εισαγωγής είναι O(logn).
- SEARCH AVLTREE number: αναζητά τον αριθμό στο δέντρο με την βοήθεια της συνάρτησης search η οποία έχει κόστος O(logn).

- DELETE AVLTREE number: O(logn) κόστος για την διαγραφή στοιχείου από το δέντρο και O(logn) για τα rotations (αν χρειαστούν).

HashTable:

- BUILD HASHTABLE BUILD_HASHTABLE.txt: κάνουμε insert κάθε φορά ένα στοιχείο στον πίνακα με την βοήθεια της συνάρτησης insert με πολυπλοκότητα O(1) και την hashfunction για να αποφύγουμε τα colisions.
- GETSIZE HASHTABLE: επιστρέφει τον αριθμό των στοιχείων του πίνακα με κόστος O(n)
- SEARCH HASHTABLE number: αναζητά τον αριθμό στον πίνακα με κόστος O(1)
- INSERT HASHTABLE number: προσθέτει τον αριθμό στον πίνακα με πολυπλοκότητα O(1), με την βοήθεια της Hash function.

Αρχεία που χρησιμοποιήθηκαν για testing θα είναι μέσα στο τελικό zip αρχείο.

Σπυριδόπουλος Γεώργιος (4619) - Ανδρέου Παναγιώτης (4735)