Κ22: Λειτουργικά Συστήματα – Χειμερινό Εξάμηνο ΄18

Ονοματεπώνυμο: Κωνσταντίνος Πασχόπουλος

Αριθμός Μητρώου: 1115201500127

Το source code μου αποτελείτε από τα εξής αρχεία: Το mytypes.h περιέχει όλους τους τύπους δεδομένων που χρησιμοποιώ. Το mygraph.c περιέχει την main αυτής της εργασίας. Τα graph_interface.c και graph_interface.h περιέχουν όλες τις συναρτήσεις που χρησιμοποιώ και τους ορισμούς τους αντίστοιχα.

Για την υλοποίηση του γράφου χρησιμοποιώ τους τύπους δεδομένων που βρίσκονται στο αρχείο mytypes.h. Ουσιαστικά έχω υλοποιήσει μια συνδεδεμένη λίστα από συνδεδεμένες λίστες. Όλοι οι κόμβοι είναι συνδεδεμένοι στην ίδια λίστα και από τον κάθε κόμβο ξεκινάει μια λίστα που συνδέει όλες τις ακμές που έχουν αφετηρία αυτόν τον κόμβο. Έτσι στο struct Graph είναι αποθηκευμένη η αρχή της λίστας που συνδέει όλους τους κόμβους και στο κάθε struct Node είναι αποθηκευμένη η αρχή της λίστας που συνδέει όλες τις ακμές που ξεκινάνε από το συγκεκριμένο struct Node.

mygraph.c

Αρχικά ελέγχω τα ορίσματα που έδωσε ο χρήστης κατά την κλήση του προγράμματος. Στη συνέχεια αν ο χρήστης έδωσε κάποιο αρχείο ως είσοδο το ανοίγω και με την συνάρτηση readCSVFile δημιουργώ τον γράφο που αντιστοιχεί σε αυτό το αρχείο. Τέλος με τη χρήση μιας while loop και μιας switch δέχομαι συνεχώς εντολές από τον χρήστη, μέχρι να δώσει την εντολή e, οπότε και τερματίζω το πρόγραμμα. Πρώτα αποθηκεύω την κάθε εντολή σε έναν ενδιάμεσο buffer (στην περίπτωση που πρέπει να εισαχθεί καινούργιος κόμβος δεσμεύω το όνομα του δυναμικά στη συνέχεια), το user_input, μεγέθους MAX_INPUT και μετά με τη χρήση

της sscanf χωρίζω την κάθε εντολή στα συστατικά της κομμάτια για να καλέσω σωστά την συνάρτηση που της αναλογεί.

graph_interface.c

Για την διαγραφή κάποιου κόμβου χρησιμοποιώ την συνάρτηση deleteNode. Η διαδικασία της διαγραφής χωρίζεται σε τρία μέρη. Πρώτα διαγράφω όλες τις ακμές που καταλήγουν στον κόμβο που θέλω να διαγράψω, μετά διαγράφω όλες τις ακμές που ξεκινάνε από τον κόμβο και τέλος διαγράφω τον ίδιο τον κόμβο. Για το πρώτο και το τρίτο μέρος χρησιμοποίησα διπλούς δείκτες (edges_ptr και nodes_ptr). Αν δεν το έκανα αυτό θα έπρεπε να ελέγξω αν αυτό που θέλω να διαγράψω είναι το κεφάλι της λίστας ή όχι και να δράσω αντίστοιχα. Με αυτόν τον τρόπο όμως μπορώ να τα διαγράψω με ένα πέρασμα. Με τον ίδιο τρόπο διαγράφω τις ακμές στην συνάρτηση deleteEdge. Επίσης σε αυτή τη συνάρτηση με την τιμή του w ελέγχω αν θα διαγράψω μια ακμή ή όσες υπάρχουν. Αν το w είναι -1 σημαίνει ότι ο χρήστης δεν έδωσε κάποιο συγκεκριμένο βάρος, άρα τις διαγράφω όλες.

Για τα ερωτήματα 7, 8 και 9 καλώ τις συναρτήσεις simpleCycles, findCircles και traceflow αντίστοιχα. Κάθε μία από αυτές τις συναρτήσεις έχει και μια βοηθητική αναδρομική συνάρτηση. Η βασική δομή είναι ίδια σε αυτές τις συναρτήσεις. Κάθε φορά κάνω Depth First Search (DFS) στον γράφο ξεκινώντας από τον κόμβο που δίνεται και σταματάω όταν ικανοποιηθεί η συνθήκη του κάθε ερωτήματος. Όταν γίνει αυτό εκτυπώνω το μονοπάτι που αποθηκεύω στην μεταβλητή path. Στο 7 επειδή ψάχνω simple cycles ο DFS σταματάει όταν πάει να επισκεφτεί τον ίδιο κόμβο δεύτερη φορά. Επειδή είναι αναδρομική συνάρτηση όταν γίνει αυτό επιστρέφει στην προηγούμενη κλήση της συνάρτησης και συνεχίζει από εκεί. Αντίστοιχα στο 8 που ψάχνω κύκλους ο DFS σταματάει όταν πάει να επισκεφτεί την ίδια ακμή δεύτερη φορά. Τέλος στο 9 πάλι δεν μπορεί να επισκεφτεί την ίδια ακμή και εκτυπώνει το μονοπάτι κάθε φορά που βρίσκει τον Nj αρκεί να μην έχει ξεπεραστεί το μέγιστο μήκος διαδρομής.

Χρησιμοποίησα DFS επειδή με ενδιέφερε να ακολουθήσω μια διαδρομή μέχρι τέλους πριν πάω στην επόμενη για αυτό και τον θεώρησα τον κατάλληλο αλγόριθμο.