Αναφορά 1ης Προγραμματιστικής Άσκησης Στο Μάθημα Τεχνολογίες Υλοποίησης Αλγορίθμων 2022-2023

ΑΜ = 1070263

Ον/μο = ΣΠΥΡΟ ΣΟΥΛΙ

Έτος = 6ο

Υλοποίηση

Αρχικά, χρησιμοποιήθηκε η βιβλιοθήκη LEDA με τα παρακάτω headers :

graph_misc.h: παρέχει βοηθητικά προγράμματα σχεδίασης γραφημάτων και οπτικοποίησης.

ugraph.h: παρέχει τύπους δεδομένων για ένα μη κατευθυνόμενο γράφημα, μαζί με διάφορες συναρτήσεις και αλγόριθμους για μη κατευθυνόμενα γραφήματα.

graph_alg.h: παρέχει αλγόριθμους συντομότερης διαδρομής, αλγόριθμους ελάχιστης έκτασης δέντρων και αλγόριθμους ροής δικτύου.

basic.h : παρέχει ορισμένες βασικές λειτουργίες βοηθητικού προγράμματος και μακροεντολές που χρησιμοποιούνται σε όλη τη βιβλιοθήκη LEDA.

dynamic_trees.h: παρέχει τύπους δεδομένων και αλγόριθμους για δυναμικά δέντρα

Για την υλοποίηση των συνεκτικών συνιστώσεων χρησιμοποιήθηκε η κλάση της LEDA TreeCohesiveComponent , στην οποία χρησιμοποιήθηκαν :

Ως public attributes :

node firstnodepointer = δείκτης πρώτου στοιχείου των κόμβων

node lastnodepointer = δείκτης τελευταίου στοιχείου των κόμβων

int Size = μήκος κόμβων

Ως private attributes :

list<node> NodeList = λίστα με κόμβους

Μετά χρησιμοποιήθηκαν οι κατάλληλοι constructors ώστε να μπορούμε να δώσουμε κατάλληλες τιμές στα attributes μας και να μπορούμε να τα προσπελάσουμε αργότερα.

TreeCohesiveComponent(node Node)

int GetSize()

node GetFirstNode()

node GetLastNode()

…

Ύστερα με τις κατάλληλες συναρτήσεις επιτύχαμε την υλοποίηση που μας ζητήθηκε από την εργασία.

friend bool operator == (TreeCohesiveComponent const &LeftOperand, TreeCohesiveComponent const &RightOperand)

int EdgeCostCompare(const edge &a, const edge &b)

TreeCohesiveComponent SearchNode(list<TreeCohesiveComponent> &ComponentList, node Node)

…

Για την υλοποίηση των γραφημάτων χρησιμοποιήθηκε η κλάση UGRAPH<int, int> με ακέραιο κόστος για τις ακμές.

Ύστερα χρησιμοποιήθηκε ο αλγόριθμος Kruscal για να δημιουργήσω μια λίστα με στιγμιότυπα της TreeCohesiveComponent. Με την συνάρτηση αυτή υλοποιούμε την ένωση των 2 δενδρικών συνεκτικών συνιστώσεων με την δημιουργία ενός στιγμιοτύπου TreeCohesiveComponent , ύστερα διαγράφονται τα στιγμιότυπα που ενώθηκαν και το στιγμιότυπο που δημιουργήθηκε εισάγεται στην αρχική λίστα.

TreeCohesiveComponent MergeTreeCohesiveComponents(TreeCohesiveComponent FirstTreeCohesiveComponent, TreeCohesiveComponent SecondTreeCohesiveComponent)

bool TreeCohesiveComponentAddEdge(list<TreeCohesiveComponent>& TreeCohesiveComponentList, node EdgeSourceNode, node EdgeTargetNode)

UGRAPH<int, int> Kruscal(UGRAPH<int, int>& UndirectedGraph, list<edge>& CircleEdgeList)

…

Για το τυχαίο συνεκτικό γράφημα δημιούργησα ένα κενό στιγμιότυπο της ugraph για να μπορεί να δωθεί ως είσοδος. Χρησιμοποιήθηκαν οι κατάλληλες συναρτήσεις για αυτόν τον λόγο :

rand_simple_undirected_graph(InitGraph, NumNodes, NumEdges);

Make_Biconnected(InitGraph);

CopyGraph(UndirectedGraphNew, InitGraph);

…

Για να υλοποιήσουμε το κόστος των πλευρών από 10 έως 10000 χρησιμοποιήσα :

UndirectedGraphNew.assign(tempEdge, (rand() % 990) + 10);

Για την υλοποίηση χειρότερης περίπτωσης του Kruscal τα κόστη των ακμών ορίσθηκαν ως P-1

UndirectedGraphNew.assign(tempEdge, UndirectedGraphNew.index(UndirectedGraphNew.target(tempEdge)) - 1);

Για να φτιάξω τα διαφορετικά γραφήματα βασίστηκα σε συναρτήσεις που υλοποιούν το γράφημα ανάλογα με την επιλογή που κάνουμε ως χρήστες :

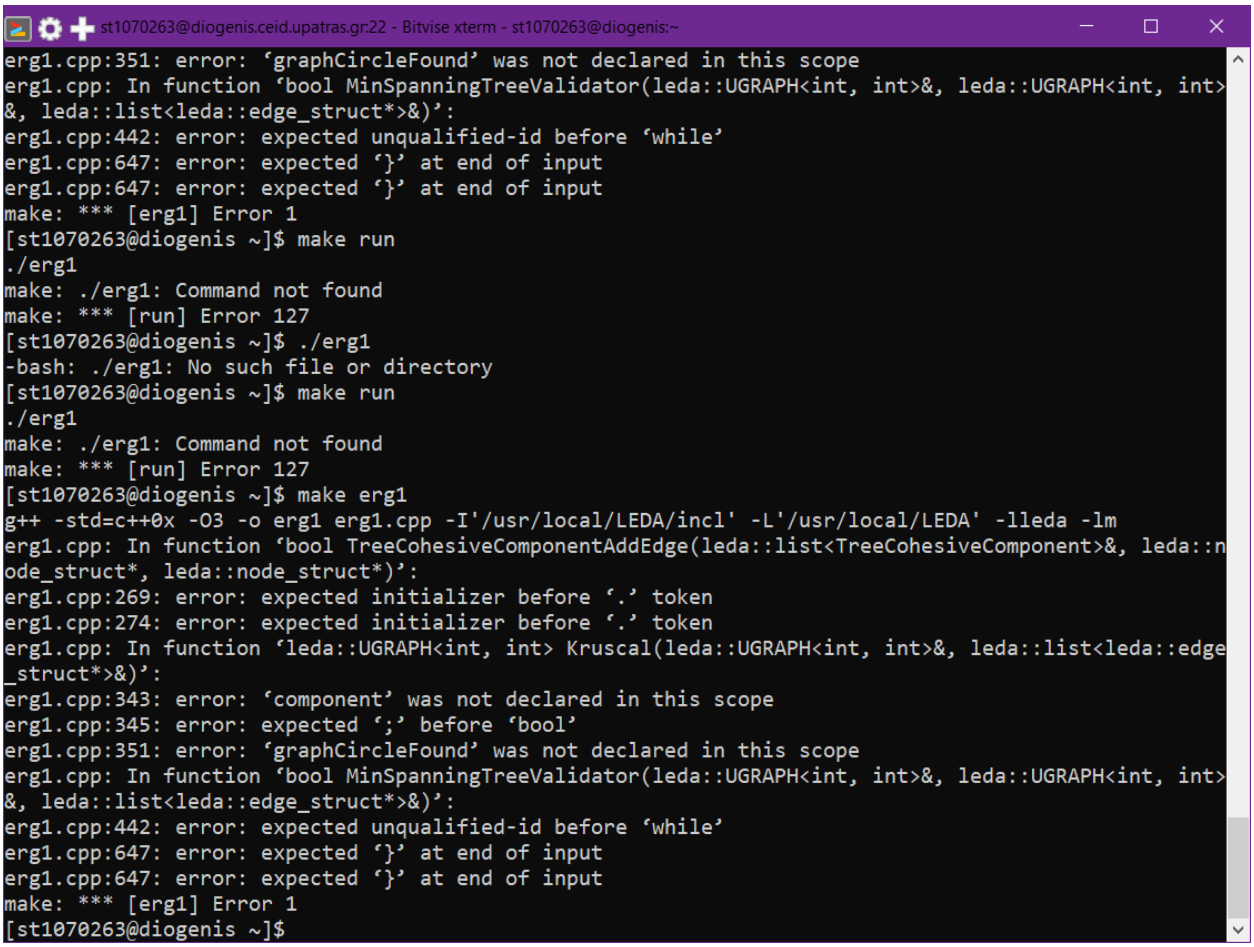
Grid_graph = grid_graph(UndirectedGraphNew, NumNodes);

```
Random_graph = rand_simple_undirected_graph(InitGraph, NumNodes,
NumEdges);

Make_Biconnected(InitGraph);

CopyGraph(UndirectedGraphNew, InitGraph);

srand(time(NULL));

UndirectedGraphNew.assign(tempEdge, (rand() % 990) + 10);


Synthetic_graph = complete_ugraph(UndirectedGraphNew, NumNodes);

UndirectedGraphNew.assign(tempEdge,
UndirectedGraphNew.index(UndirectedGraphNew.target(tempEdge)) - 1);
```

Πειραματική αξιολόγηση

Υπάρχουν κάποια συντακτικά λάθη στον κώδικα που δεν με αφήνουν να κάνω σωστά τα πειράματα μου. Αν και πιστεύω ότι άμα δείτε τον κώδικα θα διαπιστώσετε ότι η λογική μου είναι σωστή το μόνο που μπορώ να κάνω είναι μια θεωρητική πειραματική αξιολόγηση για να μπορέσω να δώσω στο περίπου τα αποτελέσματα τα οποία θα είχα.

Αμα τρέξετε τον κώδικα θα διαπιστώσετε τα παρακάτω errors τα οποία δεν μπόρεσα να καταλάβω πως να τα φτιαξω :



Για rand γραφήματα :

Είσοδος	Kruscal	Min_spanning_tree
4000		0 sec
8000		0 sec
16000		0 sec

Για grid γραφήματα :

Είσοδος	Kruscal	Min_spanning_tree
200		0.8 sec
300		0.20 sec
400		0.22 sec

Για synth γραφήματα :

Είσοδος	Kruscal	Min_spanning_tree
2000	500 sec	1 sec
3000	1500 sec	2 sec
4000	4000 sec	3 sec