ΑΝΑΦΟΡΑ ΣΧΕΔΙΑΣΤΙΚΩΝ ΕΠΙΛΟΓΩΝ

(ΜΙΧΟΠΟΥΛΟΣ ΣΟΦΟΚΛΗΣ - ΙΩΑΝΝΗΣ, ΦΡΑΝΤΖΗΣ ΑΝΔΡΕΑΣ-ΣΥΜΕΩΝ)

Ο αρχικός προβληματισμός μας ήταν η αποδοτική αναπαράσταση και η διαχείριση των δεδομένων μας. Χρειαζόμασταν μια δομή δεδομένων που θα αναπαριστούσε τα δεδομένα με κατάλληλο τρόπο ώστε να μπορούμε να έχουμε άμεση πρόσβαση σε αυτά με την χαμηλότερη δυνατή χρονική πολυπλοκότητα. Έτσι, αποφασίσαμε να χρησιμοποιήσουμε μια δομή γράφου με περιεχόμενο δύο maps ώστε να έχουμε άμεση και γρήγορη πρόβαση στα δεδομένα.

Σχεδιαστικές επιλογές κλάσης γράφου:

- **map<int, vector<int>> adjList:** Το map adjList επιλέχθηκε για την αποθήκευση της λίστας γειτνίασης ώστε να υπάρχει εύκολη εισαγωγή κόμβων-γειτόνων και γρήγορη αναζήτηση. Η δομή αυτή εξασφαλίζει ότι οι κόμβοι και οι γείτονές τους είναι αποθηκευμένοι με βάση το κλειδί (τον κόμβο) και τη λίστα γειτόνων του (το vector<int>).

- **map<int, vector<float>> cordinates:** Το map αυτό χρησιμοποιείται για να κάνουμε την αντιστοίχιση του κάθε κόμβου με των πίνακα συντεταγμένων που πέρνουμε απο τα δεδομένα της εκφώνησης. Επιλέχθηκε map ώστε να έχουμε εύκολη πρόβαση στα δεδομένα του κάθε κόμβου.

Πέρα απο τις βασικές συναρτήσεις του γράφου πχ: (addEdge(), deleteEdge(), addNode()) έχουν υλοποιηθεί και συναρτήσεις που είναι σχεδιασμένες αποκλειστικά για τις ανάγκες τις εργασίας.

- **pair<set<Lnodes, CompareValue>, unordered_set<int>> GreedySearch():**
  Για να υλοποιηθεί η συνάρτηση GreedySearch όπως αναφέρεται και στην εκφώνηση χρειάστηκε να δωθούν στην συνάρτηση τα εξής ορίσματα:
  a) int S: Κόμβος S του γράφου
  b) vector<float>& Xq: Query με συντεταγμένες
  c) int k: k γείτονες που θα βρούμε
  d) int L: Μέγεθος λίστας γειτόνων που ψάχνουμε

  Η συνάρτηση επιστρέφει ένα **set<Lnodes, CompareValue>** και ένα **unordered_set<int>**. Στην συγκεκριμένη συνάρτηση χρησιμοποιήθηκαν sets καθώς μας ευνοούν οι λειτουργίες της std βιβλιοθήκης αυτών και η πολυπλοκότητα που έχουν.

  -**set<Lnodes, Compare Value>** : Το συγκεκριμένο set χρησιμοποιήθηκε σαν το σύνολο τών γειτόνων που επιστρέφει η GreedySearch. Έγινε ordered set με συγκεκριμένη συνάρτηση Compare Value έτσι ώστε οι γείτονες κατα την εισαγωγή τους να ταξινομούνται στην κατάλληλη θέση του set, βάση τις ευκλείδιας απόστασης τους απο το σημείο xq. Το γεγονός οτι οι γείτονες υπάρχουν ταξινομημένοι μέσα στο set, μας κάνει πιο έυκολη την εξαγωγή τους και την ανανέωση τους σε συγκεκριμένες περιπτώσεις που πρέπει να γίνει μεσα στην Greedy.

  -**unordered_set<int>** : Το συγκεκριμένο set χρησιμοποιήθηκε σαν το σύνολο των γειτόνων που

έχει επισκεφθεί η GreedySearch κατα την εκτέλεση της. Το set αυτό μας βοηθάει να κατανοήσουμε πότε η Greedy πρέπει να σταματήσει να αναζητά γείτονες.

- **void RobustPrune():**
  Για να υλοποιηθεί η συνάρτηση RobustPrune όπως αναφέρεται στην εκφώνηση χρειάστηκε να δωθούν στην συνάρτηση τα εξής ορίσματα:
  a) int p: Ο κόμβος ή σημείο του γράφου που θα τροποποιηθεί
  b) unordered_set<int> & Visited: Το σύνολο υποψήφιων γειτόνων
  c) int a: Παράγοντας απόστασης_
  d) int R: Ανώτατο όριο νέων γειτόνων

  Ο ρόλος της συνάρτησης είναι να επιλέγει και να φιλτράρει την έυρεση και την προσθήκη γειτόνων του κόμβου P, βασισμένο σε αποστάσεις. Η συνάρτηση σταματάει την διαδικασία όταν φτάσει το όριο R γειτόνων ή όταν το visited αδειάσει.

Παράλληλα με τον προγραμματισμό των αρχείων Graph.cpp και Graph.h έγινε και ο προγραμματισμός των αρχείων Functions.cpp και Functions.h.
Τα αρχεία Functions περιέχουν βοηθητικές συναρτήσεις που χρειάζονται στην υλοποίηση και επιλέξαμε να τις τοποθετήσουμε σε διαφορετικό αρχείο, γιατι είναι συναρτήσεις που χρησιμοποιούνται και στο αρχείου Vamana και στο αρχείου Graph.

Έτσι με τις υλοποιήσεις των βοηθητικών συναρτήσεων, των βασικών συναρτήσεων του γράφου, την συνάρτηση GreedySearch και την συνάρτηση RobustPrune μπορέσαμε και υλοποιήσαμε τον αλγόριθμο Vamana όπως ακριβώς αναφέρεται και στην εκφώνηση της εργασίας.

Σχεδίαση αρχείου Vamana:

Το αρχείο Vamana.cpp περιέχει όλες τις υλοποιήσεις συναρτήσεων των αρχείων Graph.h και Functions.h και αποτελεί το τελικό αρχείο της εργασίας. Η εκτέλεση του αρχείου παίρνει σαν όρισματατα 3 αρχεία base, query, groundtrouth και εκτελεί πάνω σε αυτά των αλγόριθμο Vamana. Στην αρχή της εκτέλεσης γίνεται το άνοιγμα και η αποθήκευση των ορισμάτων σε vectors. Στην συνέχεια αρχικοποιείται ένας τυχαίος γράφος του οποίου ο κάθε κόμβος θα έχει R ακμές και ένα vector N θέσεων το οποίο θα περιέχει τιμές απο το 0 μέχρι το n-1 αντίστοιχα με τους κόμβους. Έπειτα υπολογίζεται το medoid του base αρχείου που έχει δωθεί σαν όρισμα και ξεκινάει η διαδικασία του αλγορίθμου Vamana όπως ακριβώς φαίνεται και στην εκφώνηση της εργασίας. Μετά την ολοκλήρωση της επαναληπτικής διαδικασίας που σημαίνει το τέλος του αλγορίθμου Vamana καλείται η GreedySearch για κάθε query του αρχείου query και ελέγχεται το αποτέλεσμα με βάση το αρχείο ground trouth υπολογίζοντας ένα ποσοστό επιτυχίας.

Περιεχόμενα Εργασίας:
Includes/Graph -> Υλοποίηση γράφου.

Includes/Functions -> Βοηθητικές συναρτήσεις.
Tests -> Test για τις συναρτήσεις της εργασίας.
Vamana.cpp -> Υλοποίηση Vamana αλγόριθμου με την βοήθεια των Includes
Makefile
ReadMe.md