2η Εργασία – Κατακερματισμός και αναζήτηση για χρονοσειρές

Δουκάκης-Χιώτης Στέφανος, ΑΜ: 1115201800276
Κωστής Σειράς, ΑΜ: 1115201800174

Directory Structure

/bin → objective files for each .cpp file
/fred-frechet → header and source files for continuous frechet
/hdr → header files
/src → stores the .cpp source files
/unit-tesing → files for unit testing

Η δομή των φακέλων ακολουθεί την 1η εργασία, επομένως για το ερώτημα Ai χρησιμοποιούνται τα αρχεία LSH.cpp και Cube.cpp στον φάκελο /src/algorithms. Για τα ερωτήματα Aii και Aiii χρησιμοποείται το Curves.cpp στον ίδιο φάκελο. Η main συνάρτηση για όλο το ερώτημα Α βρίσκεται στο αρχείο Curvesmain.cpp στον /src/mains.

Περιεχόμενα 1ης εργασίας:

Utils.hpp:
Περιέχει τις βασικές κλάσεις και μεθόδους που χρησιμοποιούν οι τρεις αλγόριθμοι LSH, Hypercube και Clustering. Για την αναπαράσταση των διανυσμάτων χρησιμοποιούνται οι κλάσεις Data_point, Data_query, όπου όλες κληρονομούν την Data_item στην οποία αποθηκεύονται ως string το id του διανύσματος και οι συντεταγμένες του. Τα σημεία που αποτελούν το input αποθηκεύονται ως Data_point, ενώ το κάθε query ως Data_query. Αντίστοιχα οι κλάσεις Solver και HashTable λειτουργούν ως οι base κλάσεις των LSH_Solver, Cube_Solver και LSH_HashTable, Cube_Hashtable. Οι base κλάσεις έχουν χαρακτηριστικά που μοιράζονται όλες οι κλάσεις παιδιά. Η κλάση hFunction αποτελεί την συνάρτηση h με την οποία γίνεται το hashing . Η μέθοδος readItems διαβάζει τα data διανύσματα ή τα queries και η getNumbersWithHammingDistance() υπολογίζει όλες τις κορυφές με hamming distance ‘x’ για τον αλγόριθμο τυχαίας προβολής στον υπερκύβο. Η avgDistance() χρησιμεύει στην δημιουργία του w. Οι Solvers αναθέτουν στον global pointer distanceFunction, που βρίσκεται στο utils.cpp, την ανάλογη μετρική που θα χρησιμοποιηθεί.

LSH.hpp:
Η κλάση LSH_Solver είναι υπεύθυνη για την διεκπεραίωση του αλγορίθμου LSH. Αποθηκεύει τα διανύσματα data και query που τα κατακερματίζει σε L LSH_Hashtables. Η κλάση LSH_HashTable έχει μια συνάρτηση G (κλάση gFunction) που μαζί με ένα σύνολο απο h συναρτήσεις κατακερματίζει τα data στα buckets του hashtable. Η gFunction, μεταξύ άλλων περιέχει τον vector linearCombinationElements, ο οποίος κρατά τα ζεύγη <r1, h1>. Η μέθοδος solve() του LSH_Solver καλεί για κάθε query την NNandRS() η οποία επιστρέφει ταξινομημένα τα n Approximate Neighbors του query, την BruteForceSearch() που επιστρέφει τα n Exact Neighbors και τέλος την WriteResult() που γράφει τα αποτελέσματα στο ouput αρχείο.

Cube.hpp:
Όπως και στο LSH.hpp αρχείο υπάρχουν οι κλάσεις Cube_Solver και Cube_Hashtable με αντίστοιχους ρόλους. Η Cube_Hashtable έχει έναν vector απο h συναρτήσεις που χρησιμοποιούνται

για τον κατακερματισμό όπως και k unordered_map, όσες και οι h, όπου κάθε map για κάθε τιμή της h που αντιπροσωπεύει, αναθέτει τυχαία το 0 ή 1. Επίσης διαθέτει ένα vector απο την κλάση Vertex_point, η οποία κρατάει την κάθε τιμή κατακερματισμού του κάθε σημείου, το σύνολο των οποίων με την ίδια τιμή πρακτικά ορίζουν την κάθε κορυφή του υπερκύβου. Η solve() είναι παρόμοια με αυτη του LSH_Solver.

Clustering.hpp:
Για την συσταδοποίηση το κάθε διάνυσμα αναπαρίσταται απο την κλάση Clustering_data_item η οποία κληρονομεί την Data_point. Η κλάση Clustering_Solver καλεί την μέθοδο lloyd(), την reverseAssignmentLSH() ή την reverseAssignmentCube() αναλόγα με το input του χρήστη.
Η μέθοδος initpp() δημιουργεί τα αρχικά κεντροειδή για τον εκάστοτε αλγόριθμο.
Η reverseAssignmentLSH() ορίζει μια ειδική κλάση Solver, την LSH_Solver_Clustering, η οποία διαθέτει ένα ειδικό LSH_Hashtable, LSH_HashTable_Clustering που εκτελεί την Reverse Approach μέσω της μεθόδου clusteringRangeSearch(). Με αντίστοιχο τρόπο λειτουργεί η reverseAssignmentCube().

Περιεχόμενα 2ης εργασίας:

Curves.hpp:
Περιέχει τις κλάσεις Frechet_query και Frechet_point, που κληρονομούν τις Data_query και Data_point αντίστοιχα, που η μεταβλητή altered_coordinates αποθηκεύει τις μετατροπές των συντεταγμένων. Η κλάση snapping έχει δύο κλάσεις παιδιά Discrete_snapping και Continuous_Snapping.Η Discrete_snapping εκτελεί το snapping, duplicate removal και concatenation. Αντίθετα, η Continuous_snapping εκτελεί snapping και την MinimaMaxima συνάρτηση.
Η κλάση Frechet_Solver είναι υπεύθυνη για την διεκπεραίωση των αλγορίθμων Frechet. Στον constructor για τον Discrete Frechet το w είναι ορισμένο με την τιμή 200, ενώ στον Continuous Frechet είναι ένα πολλαπλάσιο του 500, ανάλογα με τις διαστάσεις των χρονοσειρών που θα δοθούν. Επίσης και στους δύο Frechet αλγορίθμους η τιμή delta είναι ορισμένη με 10. Όλες αυτές οι τιμές ορίστηκαν μετά απο πειραματισμό, για τα καλύτερα αποτελέσματα στο εύρος των αρχείων που δόθηκαν. Η solveDiscreteFrechet() καλεί για κάθε query την NNandRSDiscrete(), η οποία με την σειρά της καλεί για κάθε hashtable την NearestNeighboursDiscrete(), για να βρεθεί η προσεγγιστικά κοντινότερη χρονοσειρά. Έπειτα, καλείται η bruteForceSearchDiscrete() και μετά η writeResult() που γράφει στο αρχείο εξόδου τα αποτελέσματα. Η solveContinuousFrechet() λειτουργεί με παρόμοιο τρόπο.
Η κλάση Frechet_Hashtable έχει την συνάρτηση padding() που χρησιμοποείται για την εισαγωγή των σημείων (insertDiscrete() και insertContinuous()) στα ή στο hashtable.
Επίσης υπάρχουν οι γενικές συναρτήσεις discreteFrechet(), discreteFrechetRecursion() και calculateDistance() που υπολογίζουν την Discrete Frechet απόσταση, οι συναρτήσεις bruteForce, η συνάρτηση για το filtering για τον Continuous Frechet και η constructFredFrechetCurve για την μετατροπή ενός Frechet_query ή Frechet_point σε κλάση τύπου Curve της βιβλιοθήκης για τον Continuous Frechet.

Curves_clustering.hpp

Περιέχει το declaration των κλάσεων και συναρτήσεων που χρησιμοποιούνται για την υλοποίηση του clustering των curves. Στην περίπτωση που ο χρήστης θέλει να κάνει το assignment στο clustering me lloydAssignment του δίνεται η δυνατότητα να επιλέξει αν το update step των centroids θα γίνεται με mean frechet curve ή με την κλασσική προσέγγιση που χρησιμοποιήθηκε στην προγηούμενη εργασία. Στο initialization των centroids χρησιμοποιείται η κατάλληλη συνάρτηση απόστασης ανάλογα με το αν ο χρήστης διάλεξε mean Frechet curve ή meanVector.

Για το unit-tesing χρησιμοποείται το framework acutest απο το repository https://github.com/mity/acutest.
Τα δύο test παίρνουν την πρώτη καμπύλη απο το nasd_input.csv και ελέγχουν το snapping για τους Discrete και Continuous Frechet. Επιστρέφουν pass αν οι συντεταγμένες μετά το snapping είναι λιγότερες απο τις αρχικές, αλλιώς fail.

Η μεταγλώττιση γίνεται με το Makefile που διατίθεται και η εκτέλεση των προγραμμάτων γίνεται με τις εντολές που έχουν δοθεί στην εκφώνηση της εργασίας, με την διαφορά ότι στο ./search οι παράμετροι -algorithm και -metric χρειάζονται δύο παύλες (–), όπως επίσης και στο ./cclustering οι παράμετροι -assignment και -update.
Τα προγράμματα δεν τρέχουν επαναληπτικά και κατα την εκτέλεση εκτυπώνουν στην οθόνη τις παραμέτρους που χρησιμοποιούν και τον χρόνο εκτέλεσής τους.