Ανάπτυξη Λογισμικού για Αλγοριθμικά Προβλήματα

Εργασία 3

Συμμετέχοντες: Πρωτόπαπας Ευάγγελος, 1115201400169
Χρήστου Κωνσταντίνος, 1115201400229

Comments

Για compile:

run ‘make’
Δημιουργούνται 2 εκτελέσιμα, το proteins και το segments

Για run:

Για το 1ο μέρος της εργασίας τρέχουμε το εκτελέσιμο proteins ως εξής:
$ ./proteins -i <bio_input_file> -c <clustering_config_file>

Για το 2ο μέρος της εργασίας τρέχουμε το εκτελέσιμο segments ως εξής:
$ ./segments -i <input_csv_unsegmented> -o <output_csv_segmented> -c <clustering_config_file> -s <size_of_dataset>

Σχόλιο: Στο segments η είσοδος είναι μόνο στο -i το athens.csv. Στο output θα βγει το segment.csv και επίσης στα αρχεία ανάλογα με τι clustering τρέχουμε.
π.χ. το lsh_ways_frechet.dat, kmeans_ways_frechet.dat
Αν θέλουμε να βγάλουμε τα αρχεία με την μετρική DTW θα πρέπει να αλλάξουμε την τιμή της μεταβλητής metric στην main από “classic_frechet” σε “classic_DTW” και από “DFD” σε “DTW” αντίστοιχα

Implementation Notes

Στο πρώτο μέρος απλά διαβάζουμε το αρχείο της εισόδου και κάνουμε τα 2 clustering με βάση την μετρική crmsd και την μετρική frechet αφού γίνει πρώτα η μετατόπιση και η περιστροφή των καμπυλών. Ο αλγόριθμος του clustering τρέχει αρκετές φορές μέχρι να βρούμε ένα τοπικά optimal k.

Στο δεύτερο μέρος, για το 1ο υποερώτημα χρησιμοποιήσαμε μια βιβλιοθήκη για την C++ που λέγεται libosmium με στόχο να κάνουμε το αρχικό parsing του osm αρχείου. Σε γενικές γραμμές με την βιβλιοθήκη αυτή φτιάξαμε ένα cache αρχείο που από το αρχικό osm αποθηκεύσαμε στο cache τα locations των nodes και με ένα 2ο πέρασμα ταιριάξαμε από το cache αρχείο τα locations στα αντίστοιχα ways τους και έτσι πήραμε το τελικό athens.csv, το οποίο το έχουμε ανεβάσει στο link https://www.sendspace.com/file/xmle04

Για το 2ο υποερώτημα κάνουμε το parsing του athens.csv ως εξής: Κάνουμε ένα πέρασμα του αρχείου για να δούμε ποια ζευγάρια (lat,lon) περιέχονται σε παραπάνω από 1 δρόμους (δηλαδή σε διασταυρώσεις) και τα αποθηκεύουμε και κάνουμε ένα 2ο πέρασμα στο οποίο διαβάζουμε από το athens.csv, υπολογίζουμε τις καμπυλότητες των τμημάτων καθώς διαβάζουμε το αρχείο και ανάλογα αν η καμπυλότητα κάποιου τμήματος ξεπερνάει ένα όριο το οποίο το υπολογίσαμε με διάφορες δοκιμές και το έχουμε ορίσει με ένα define και αν δεν ανήκει σε διασταύρωση αυτό το τμήμα, τότε σπάμε τον δρόμο σε εκείνο το σημείο. Με τον τρόπο αυτό, αν υπάρχουν συνεχόμενες διασταυρώσεις θα σπάσει κάποιος δρόμος στο τέλος των διασταυρώσεων, εκτός αν ξεπεράσει κάποιο όριο μεγέθους το οποίο το έχουμε ορίσει επίσης με define.

Στο 3ο υποερώτημα κάνουμε το clustering που ζητείται με τον kmedoids αλγόριθμο.

Στο 4ο υποερώτημα κάνουμε τα clustering χρησιμοποιώντας τα buckets του lsh πίνακα. Αρχικά ορίζουμε το μέγεθος του πίνακα να είναι ίσο με το μέγεθος του dataset και ανάλογα με το πως θα μπουν οι καμπύλες μέσα στον πίνακα παίρνουμε τα buckets που περιέχουν στοιχεία και τα ορίζουμε σαν clusters. Έπειτα, παίρνουμε αυτά τα clusters και χρησιμοποιώντας τον mean frechet update αλγόριθμο, βρίσκουμε ένα κέντρο για αυτά τα clusters και με βάση αυτό υπολογίζουμε και την σιλουέτα. Με διάφορες εκτελέσεις που κάναμε και δοκιμές, βρήκαμε ότι το k των grids όταν είναι ίσο με 2 δουλεύει καλά γενικά αλλά για κάποια συγκεκριμένα μεγέθη datasets (συνήθως μεγάλα) μπορεί κάποιο k μεγάλο, μεγαλύτερο του 10 μπορεί και του 20, μπορεί να δώσει καλύτερα αποτελέσματα γιατί θα δημιουργηθούν περισσότερα clusters.