Ανάπτυξη Λογισμικού για Αλγοριθμικά Προβλήματα
1η Προγραμματιστική Εργασία - README

Αντωνίου Στέφανος, sdi1800008
Κοκόλιας Κωνσταντίνος, sdi1800077

Αρχεία

types.h

Οι δηλώσεις των τύπων για τα σημεία (pointType), για τη δομή που περιέχει τις παραμέτρους για τις συναρτήσεις h( ) των LSH και Hypercube (hashParametersType), και τις δομές για τα αποτελέσματα (nearest-neighbors, range-search και cluster).
Για το pointType, το id είναι ένα string και τα coords ένας δείκτης σε double που δεσμεύει δυναμικά έναν πίνακα όσο και οι διαστάσεις που δίνονται στο input file. Το hash_id το αποτέλεσμα της id( ) = $r_1 * h_1 + ... + r_k * h_k$ στην μέθοδο LSH ενώ το hash_id_hypercube είναι το concat όλων των h του σημείου, τα οποία χρησιμοποιούνται μόνο από την μέθοδο που έχει επιλέξει ο χρήστης.
Το hashParametersType περιλαμβάνει τις τυχαίες τιμές των r, v και t που αντιστοιχεί σε κάθε μία συνάρτηση h( ).

data_handling.h
data_handling.cpp

Η συνάρτηση get_data δέχεται το όνομα του αρχείου (input ή query) και επιστρέφει έναν vector με τα σημεία.
Η getNumOfDimensionsInFile επιστρέφει το μέγεθος του σημείου (δηλαδή το πλήθος των συντεταγμένων τους).
Η getNumOfLines επιστρέφει το πλήθος των σημείων σε κάθε αρχείο.

functions.h
functions.cpp

Η συνάρτηση mod(D, d) επιστρέφει το πραγματικό modulo του D με το d.
Η L2 είναι η μετρική L2, δέχεται δύο σημεία και τις διαστάσεις τους και επιστρέφει την απόσταση μεταξύ τους.
Η hammingDistance επιστρέφει το hamming distance μεταξύ δύο αριθμών.

GenericHashTable.h

Μία template abstract κλάση για τα hash tables που θα χρησιμοποιηθούν από τα πιο ειδικά hash tables των LSH και Hypercube.
*Επειδή η κλάση χρησιμοποιεί templates, είναι ολόκληρη υλοποιημένη σε ένα .h αρχείο για να μην υπάρχουν errors.

HashTable_LSH.h
HashTable_LSH.cpp

Η υλοποίηση της κλάσης των hash tables για τον αλγόριθμο LSH.
Αρχικοποιείται με τις τυχαίες τιμές των r, v και t για κάθε μία h( ) που περιέχει (η παράμετρος k δίνεται από τον χρήστη)
Για κάθε ένα σημείο που εισάγεται στο hash table, υπολογίζεται και αποθηκεύεται το id του (αλγόριθμος από τις διαφάνειες) έτσι ώστε να βρούμε το g( ) (*hash_function), που

χρησιμοποιεί την h( ) (αλγόριθμος από τις διαφάνειες).

HashTable_Hypercube.h
HashTable_Hypercube.cpp

Η υλοποίηση της κλάσης των hash tables για τον αλγόριθμο Hypercube.
Ουσιαστικά το κάθε bucket του table αντιστοιχεί σε μία κορυφή της δομής του υπερκύβου που έχει k διαστάσεις.
Στην αρχικοποίηση, όπως και στο LSH, αποθηκεύονται οι τυχαίες τιμές για τις h( ), και επίσης αρχικοποιείται ένα map f, στο οποίο αντιστοιχίζονται οι τιμές 0-1 για κάθε h( ), έτσι ώστε αν ξανασυναντηθεί να μην πάρει τυχαία τιμή αλλά αυτή που πήρε πριν.

LSH_class.h
LSH_class.cpp

Η υλοποίηση του αλγορίθμου LSH.
Περιέχει ενα reference στα σημεία του dataset, έναν πίνακα L hash tables, και τις παραμέτρους που έδωσε ο χρήστης (συμπεριλαμβανομένης και της μετρικής).
Στην αρχικοποίηση, δεσμεύεται ο κατάλληλος χώρος για τα hash tables και τα αρχικοποιεί (με μέγεθος n/32 που παρατηρήσαμε ότι είναι το ιδανικό).
Nearest neighbors:
Αρχικά υπολογίζουμε με brute force τις πραγματικές καλύτερες αποστάσεις από ένα query και τους αντίστοιχους χρόνους.
Μετά, για τον υπολογισμό της nearest neighbors, βρίσκει το bucket που ανήκει το query, και ψάχνει και συγκρίνει τα σημεία του bucket με ίδιο id (*αν δεν υπάρχει κάποιο τέτοιο σημείο, ελέγχει ολόκληρο το bucket). Αυτό γίνεται για όλα τα hash tables που έχουμε.
Range search:
Ψάχνει για κάθε bucket τα σημεία με μικρότερη απόσταση από R και τα επιστρέφει.

Hypercube_class.h
Hypercube_class.cpp

Η υλοποίηση του αλγορίθμου Hypercube.
Αντίστοιχες αρχικοποιήσεις/παράμετροι όπως ο LSH.
Στις nearest neighbors και range search, ελέγχει τις |probes| πιο κοντινές κορυφές του κύβου με βάση το hamming distance και max σημεία που θα συγκρίνει μέχρι και |M| (συνολικά, ανεξάρτητα από κορυφές).

clustering.h
clustering.cpp

Η υλοποίηση των αλγορίθμων για το clustering.
Η μέθοδος του clustering που θα χρησιμοποιηθεί (Lloyd’s, range search LSH ή Hypercube) θα καθοριστεί από το πλήθος/τύπο των παραμέτρων που θα δοθούν στην δημιουργία του αντικειμένου τύπου clustering, όπου θα γίνουν και οι κατάλληλες αρχικοποιήσεις.
Για κάθε έναν αλγόριθμο, αρχικοποιούνται τα κεντροειδή μέσω της k_means και καλείται ο αντίστοιχος αλγόριθμος:
lloyds_algorithm:
Αναθέτει όλα τα σημεία στο κοντινότερο κεντροειδές, μετα υπολογίζει τα καινούργια κεντροειδή (το μέσο διάνυσμα), μέχρι τα καινούργια κεντροειδή να είναι πολύ κοντινά με τα προηγούμενα.
rangeassignment_LSH/Hypercube:
Αναθέτει τα R-πιο κοντινά σημεία στο κάθε κεντροειδές (μέχρι να μην εισέρχονται

καινούργια για κάθε διπλασιασμό της R, όπου θα γίνει σύγκριση των υπολοίπων όπως στον Lloyd’s) και ενημερώνει τα κεντροειδή όπως και στον Lloyd’s.
*Η τιμή που επιλέξαμε για να σταματήσει η ενημέρωση των κεντροειδών είναι 0.1
Έπειτα υπολογίζει το μέσο silhouette για κάθε συστάδα και συνολικά και το επιστρέφει.

main_Clustering.cpp
main_Hypercube.cpp
main_LSH.cpp

Παραδείγματα εκτέλεσης των προγραμμάτων και αποθήκευση σε output file των αποτελεσμάτων.

Οδηγίες Μεταγλώττισης:
make lsh/cube/cluster για την παραγωγή του αντίστοιχου εκτελέσιμου ανάλογα με τον αλγόριθμο
make clean για εκκαθάριση των .o και των εκτελέσιμων

Οδηγίες Χρήσης του προγράμματος

$./lsh –i <input file> –q <query file> –k <int> -L <int> -o <output file> -N <number of nearest> -R <radius>

$./cube –i <input file> –q <query file> –k <int> -M <int> -probes <int> -o <output file> -N <number of nearest> -R <radius>

$./cluster –i <input file> –c <configuration file> -o <output file> -complete <optional> -m <method: Classic OR LSH or Hypercube>

*Αν παραληφθεί κάποια παράμετρος θα δοθούν default τιμές, και αν κάποιο αρχείο δεν υπάρχει θα ξαναζητηθεί.

https://github.com/kostiskok/Project_Emiris_Erg1