Ανάπτυξη Λογισμικού για Αλγοριθμικά Προβλήματα – Εργασία 2

Φώκος Ορέστης 1115201200190
Απόστολος Μπονωτης 1115201600111

README

Η εργασία δουλεύει για διανύσματα, και για τα κομμάτι Initialization 1,Assignment 1, (Assignment 2 δεν έχει υλοποιηθεί αλλά τόσο το range search όσο και το lsh υπάρχουν), Update 1. Επίσης, έχουμε υλοποιήσει το διάβασμα για αρχείο εισόδου καμπυλών.

Αποφύγαμε όσο το δυνατόν τη χρήση structs και classes για να γλυτώσουμε μεγάλο αριθμό constructors και destructors. Τα σημεία του dataset όπως και των query, αποθηκεύονται σε vector όπου το κάθε σημείο είναι ένα vector, άρα όλα τα queries αποθηκεύονται σε ένα συνολικό vector<vector<int>> queries. Αντίστοιχα, τα σημεία του dataset αποθηκεύονται στο vector<vector<int>> All.

Έχει υλοποιηθεί και ο lsh , από το 1^{ο} κομμάτι του πρότζεκτ. Έχει υλοποιηθεί και η range search για σημεία, η οποία αν αποσχολιαστούν τα κομμάτια της main, δουλεύει σωστά και χωρίς απώλειες μνήμης. Απλά δεν έχει γίνει ο συνδυασμός των 2 για το κομμάτι του Assignment 2.

Οδηγίες μεταγλώττισης/εκτέλεσης στην τελευταία σελίδα.

Δομή dist_id:

Περιέχει μεταβλητή double dist για την απόσταση του nearest neighbor, TEMPLATE id για το id του, και double time για το χρόνο υπολογισμού των παραπάνω.

ΑΡΧΕΙΑ:

- **ann.cpp ann.h :** Υλοποιείται ο αλγόριθμος ann, για ένα query και ένα hashtable, και συνολικότερα για πολλά queries και πολλά (σε εμάς L) hashtables

  Συναρτήσεις:
  ann:
  παράμετροι: δείκτης σε ένα hash table, vector<int> που είναι οι διαστάσεις ενός query, και string που ορίζει το distance type δηλαδή επιλογή μετρικής (πχ. “manhattan”).

  Επιστρέφει αντικείμενο δομής dist_id, το οποίο έχει την απόσταση και το id του nearest neighbor του query στο συγκεκριμένο hash table, δηλαδή που βρίσκεται στο ίδιο bucket του συγκεκριμένου hash table με το query. Επίσης περιέχει το χρόνο υπολογισμού του nearest neighbor.

ann_complete:

παράμετροι: δείκτης σε όλα τα hash table, δείκτης στη δομή που αποθηκεύουμε τις τιμές των μεταβλητών si, δείκτη σε vector<vector<int>> που είναι στην ουσία vector που περιέχει queries.
Επιστρέφει vector <dist_id> , δηλαδή για το ι-οστό query, το i-οστό στοιχείο του vector που επιστρέφει, θα περιέχει το distance, το id, και το χρόνο που αντιστοιχεί στο query αυτό.

- **brute_force.cpp brute_force.h :** Υλοποιούνται οι brute force αναζητήσεις nearest neighbor τόσο στην αρχή για όλα τα σημεία του dataset (για κατάλληλη τιμή της παραμέτρου W), όσο και αργότερα για κάθε σημείο query, για σύγκριση με άλλες μεθόδους, πχ. Ann.

Συναρτήσεις:
brute_min_distance:
παράμετροι : vector<int> με τις συντεταγμένες σημείου query, vector<vector<int>> All που είναι vector όπου κάθε στοιχείο του είναι συντεταγμένες σημείου dataset (σε μορφή vector<int>). Τρίτη παράμετρος string distance_type για επιλογή μετρικής (πχ. “manhattan”)

Βρίσκει με brute force τρόπο το nearest neighbor για σημείο query, από δοσμένο vector<vector<int>> σημείων του dataset, επιστρέφει αντικείμενο τύπου dist_id με απόσταση,id,χρόνο υπολογισμού.

brute_min_distance_all:

Βρίσκει με brute force τρόπο το nearest neighbor κάθε σημείου εντός του δοσμένου dataset All. Χρησιμοποιείται για την εύρεση της μέσης απόστασης nn στο dataset, για τον ορισμό της τιμής της παραμέτρου W.

παράμετροι: vector<vector<int>> All, double & avg_nn_distance, string distance_type

Επιστρέφει vector<dist_id> που περιέχει το nn_distance, id , time, για κάθε σημείο εντός του dataset. Δεύτερη παράμετρος είναι double που θα περνιέται ως δείκτης, στην οποία θα αποθηκεύουμε τη μέσης απόστασης nn στο dataset (για το W). Τρίτη παράμετρος για επιλογή μετρικής (πχ. “manhattan”).

calculations.cpp:

Η modularPow χρησιμοποιεί τη σχέση (a*b)modM = (amodM * bmodM)modM για την αποφυγή του overflow.

Η h_calculation υπολογίζει τις hi χρησιμοποιώντας τη modularPow.

Η decToBinaryConcat δημιουργεί τις gi φτιάχνοντας τη δυαδική ακολουθία από τις h.Στη hashfunction γίνεται mod tableSize για να μπει στο table.

HashTable.cpp:

Χρησιμοποιείται ο πίνακας s δύο διαστάσεων όπου αποθηκεύονται τιμές των random s που χρησιμοποιούνται στην εύρεση της h έτσι ώστε να μπορούμε να βρούμε την τιμή της gi για κάποιο σημείο για το οποίο έχει ήδη υπολογιστεί.Αν η γραμμή του s είναι κενή τότε δεν υπήρχε προηγούμενη τιμή άρα υπολογιζονται τα f s και αποθηκευονται στην κενη γραμμη.

Αποθηκεύεται η τιμή fi(gi) σε καθε bucket gi ωστε να προσδιοριζουμε αν εχει υπολογιστει ηδη η τιμη fi(gi) για να την χρησιμοποιησουμε ή να την δημιουργήσουμε uniformly ανάλογα.Για τις υπολοιπες gi χρησιμοποιείται ο s

Hypercube.cpp:

farray:εξεταζει το bucket του hashtable για να δει αν υπαρχει τιμη για την f.

F_to_int μετατρεπει το binary tag σε δεκαδικη τιμη ωστα να χρησιμοποιηθεί ως index του table του hypercube.

Neighbors:βρισκει ολες τις γειτονικες κορυφες του query και αποθηκευει σε εναν πινακα int τη θεση τους στο table του hypercube.

mind_distances: ελεγχει σε probes γειτονες ποια ειναι η μικροτερη αποσταση με το query και κραταει το id του σημειου και την min distance.Αυτο το κανει για ολα τα queries.

- **File.cpp file .h**

Εδώ περιέχεται η συνάρτηση που είναι υπεύθυνη για το διάβασμα αρχείων και τη μετατροπή τους σε κατάλληλη μορφή για το υπόλοιπο πρόγραμμα.
Η συνάρτηση καλείται τόσο για το διάβασμα του input file, όσο και για το query file. Διαβάζει τα αρχεία, και επιστρέφει vector με σημεία (του dataset ή query), όπου το κάθε σημείο είναι ένα vector<int>.
Το πρώτο στοιχείο κάθε σημείου είναι βέβαια το id του, γι' αυτό και στον υπολογισμό αποστάσεων κλπ στο υπόλοιπο πρόγραμμα, το πρώτο στοιχείο κάθε σημείο αγνοείται.

Συνάρτηση:
read_input_to_vector:

Παράμετροι: string filename για το όνομα του αρχείου που θα διαβάσει.
Διαβάζει το αρχείο και μετατρέπει την είσοδο σε vector από σημεία, δηλ. vector<vector<int>>.

Assignment.cpp:

Lloyds:

Καλεί μία φορά την first assignment για να γίνει το αρχικό insert στο hashtable, και έπειτα, σε λούπα, καλεί την update kai tin lloyds_assignment_point μέχρι να μην υπάρχει αλλαγή cluster για κανένα σημείο.

lloyds_first_assignment_point:

Κανει assign το point στο πιο κοντινο cluster και εισαγει την πληροφορια στο hashtable_points.

Lloyds update:

Δίνεται ως ορισμα το point* centroid ωστε να μπορει να γινει η αλλαγη μεσα στην update.Υπολογιζει το new_centroid συμφωνα με τα points των cluster και το αθροισμα και ανανεωνει το centroid.

lloyds_assignment_point:

Βρισκει το cluster που ανηκει το point,ελεγχει το cluster με το cluster του point στο hashtable.Αν εχει αλλαξει τοτε γινεται update στο hashtable και γινεται διαγραφη του point απο το παλιο cluster και εισαγεται στο καινουργιο.

HashTable.cpp:

Hashtable_points:

Για τον lloyd algorithm χρειαζεται να ξερω αν εγινε καποια αλλαγη ενος point σε καποιο cluster ετσι ωστε να τερματισει ο αλγοριθμος εφοσον ελεγξει πως πλεον δεν γινεται καμια αλλαγη.Το hashTable χρησιμοποιειται για την αποθηκευση στοιχειων της μορφης (point,αριθμος του cluster οπου ανηκει) ωστε η αναζητηση να γινεται αρκετα πιο γρηγορα.Οταν αλλαζει ενα point cluster γινεται το καταλληλο update και στο hashtable

Initialization.cpp:

random_initialization_point:

Παράγει εναν τυχαίο αριθμό από 0 έως Dataset-1 που είναι ο και τον χρησιμοποιεί για να θέσει το κέντρο του cluster τυχαία.

Create_clusters_point:

Δημιουργεί ενα vector μεγέθους k για τα cluster.

Dataset:

Για διαφορετικό τύπο στοιχοίων του dataset χρησιμοποιούμε typedef (type of dataset points) new_type;

ΟΔΗΓΙΕΣ ΜΕΤΑΓΛΩΤΤΙΣΗΣ:

Έχει υλοποιηθεί **makefile** με το συνήθη τρόπο. Compilation με την εντολή make. Συνήθης λειτουργία του make clean.
Για εκτέλεση./lsh (παράμετροι)

Παράδειγμα εκτέλεσης: ./lsh -d DataVectors_5_500x100.csv -o output.txt

Παράμετροι που δεν δίνονται μέσω των αντίστοιχων flags, είτε παίρνουν default τιμές, πχ το L, είτε τις ζητάμε από το χρήστη μέσω κατάλληλου μηνύματος στο console.

ΦΟΙΤΗΤΕΣ:

Ονοματεπώνυμο	Α.Μ.	e-mail	Github:
Φώκος Ορέστης	1115201200190	sdi1200190@di.uoa.gr	https://github.com/OrestisFokos
Απόστολος Μπονωτης	1115201600111	sdi1600111@di.uoa.gr	https://github.com/AkisBon