Ανάπτυξη Λογισμικού για Αλγοριθμικά Προβλήματα

Υλοποίηση Δομής για την εύρεση των κοντινότερων γειτόνων στη γλώσσα C/C++

Απόστολος Φλωράκης-1115201400217

Γενική Περιγραφή: Το πρόγραμμα έχει σχεδιαστεί βάσει της εκφώνησης της εργασίας, ακολουθώντας την θεωρία και τεχνικές LSH των διαφανειών και όσων συζητήθηκαν στην τάξη. Δεχόμενο ως είσοδο ένα αρχείο που περιέχει διανύσματα, τα αποθηκεύει στην αντίστοιχη δομή (πίνακα LSH η υπερκύβο) βάσει της μετρικής που έχουμε επιλέξει και στη συνέχεια πραγματοποιεί τα ερωτήματα εύρεσης των πλησιέστερων γειτόνων. Για την υλοποίηση έχει χρησιμοποιηθεί η γλώσσα C++ χωρίς τη χρήση STL όσον αφορά τις δομές δεδομένων.

Αρχεία εισόδου-εξόδου: Η μορφή των αρχείων εισόδου όσο για το input όσο και για τα queries έχει διατηρηθεί ίδια με αυτή των αρχείων που μας δόθηκε αρχικά. Κάτι τέτοιο σημαίνει πως δεδομένου πως δεν ακολουθούσαν την μορφή της εκφώνησης έκρινα καλύτερο να μην τα πειράξω και να διαχειρίζομαι τα ids εσωτερικά στο πρόγραμμα αλλά και να δηλώνω την μετρική και την ακτίνα με τη βοήθεια επιπλέον ορίσματος στη γραμμή εντολών. Όσον αφορά το αρχείο εξόδου δημιουργείται στον τρέχοντα κατάλογο και σαν μορφή συμφωνεί με αυτή που υποδεικνύεται στην εκφώνηση.

Μεταγλώττιση και Εκτέλεση: Τόσο το lsh όσο και το cube πρόγραμμα μοιράζονται πολλές κοινές συναρτήσεις και δομές δεδομένων. Για το λόγο αυτό έχω τοποθετήσει στο φάκελο που περιέχει τα παραδοτέα αρχεία δυο Makefile. Το Makefile.lsh και το Makefile.cube. Με τη χρήση της εντολής **make -f Makefile.lsh** μπορεί να παραχθεί το lsh εκτελέσιμο ενώ με τη χρήση της **make -f Makefile.cube** το cube εκτελέσιμο. Με την make clean -f ακολουθούμενη από το εκάστοτε Makefile μπορεί να γίνει διαγραφή των επιμέρους .o αρχείων.
Όσον αφορά την εκτέλεση του προγράμματος έχω προσθέσει δυο ορίσματα γραμμής εντολών ένα που αφορά τη μετρική και ένα για την ακτίνα. Έτσι η εκτέλεση μπορεί να γίνει:

./lsh -d <inputfile> -q <queryfile> -k <int> -L <int> -o <outputfile> -m <metric> -r <radius>

./cube -d <inputfile> -q <queryfile> -k <int> -M <int> -probes <int> -o <outputfile> -m <metric> -r <radius>

Αρχεία:

- **lsh.cc:** Περιέχει τη main του lsh προγράμματος που είναι υπεύθυνη για το διάβασμα των αρχείων, τη δημιουργία των δομών δεδομένων την υποβολή των ερωτημάτων και την καταγραφή των αποτελεσμάτων.
- **cube.cc:** Αντίστοιχο για το cube πρόγραμμα.
- **HashTable.cc:** Περιέχει τους ορισμούς των συναρτήσεων του αντίστοιχου .h αρχείου.
- **HashTable.h:** Περιέχει τις δηλώσεις και ορισμούς που αφορούν την κλάση HashTable καθώς και Tables. Η κλάσεις αυτές μοντελοποιούν κάθε ένα πίνακα αλλά και ένα πίνακα από πίνακες κατακερματισμού αντίστοιχα.
- **HyperCube.cc:** Περιέχει τους ορισμούς των συναρτήσεων του αντίστοιχου .h αρχείου.
- **HyperCube.h:** Περιέχει τις δηλώσεις και ορισμούς που αφορούν την κλάση Cube. Η κλάση αυτή μοντελοποιεί τον υπερκύβο.
- **VectorList.h:** Περιέχει τις δηλώσεις και ορισμούς που αφορούν την κλάση ListNode και VectorList. Η κλάσεις αυτές μοντελοποιούν μια λίστα από vector.
- **VectorList.cc:** Περιέχει τους ορισμούς των συναρτήσεων του αντίστοιχου .h αρχείου.
- **Vector.h:** Περιέχει τις δηλώσεις και ορισμούς που αφορούν την κλάση Vector. Η κλάση αυτή μοντελοποιεί το διάνυσμα.
- **Vector.cc:** Περιέχει τους ορισμούς των συναρτήσεων του αντίστοιχου .h αρχείου.
- **utils.h:** Περιέχει δηλώσεις χρήσιμων αυτόνομων συναρτήσεων που χρησιμοποιούνται σε διάφορα σημεία του προγράμματος.
- **utils.cc:** Περιέχει τους ορισμούς των συναρτήσεων του αντίστοιχου .h αρχείου.
- **Makefile.lsh:**
- **Makefile.cube:**

Σχόλια:

- Τα αντικείμενα Vectors που αναπαριστούν τα διανύσματα εισόδου δημιουργούνται μια φορά και τοποθετούνται σε ένα πίνακα. Στον πίνακα αυτόν και με και με τη βοήθεια pointers θα δείχνουν οι pointers των λιστών των hashtable. Αυτό γίνεται ώστε τα διανύσματα να μπορούν να δημιουργηθούν και να διαγραφούν μια φορά ενώ παράλληλα να μπορούν να “υπάρχουν” ταυτόχρονα σε όλα τα hashtable.
- Οι υπόλοιπες υπερπαράμετροι του προγράμματος εκτός αυτών που δίνονται ως όρισμα στη γραμμή εντολών έχουν οριστεί μέσω define στο utils.h αρχείο.
- Για την παραγωγή τυχαίων αριθμών στην κανονική κατανομή έχω χρησιμοποιήσει την τεχνική Marsaglia υλοποιώντας αντίστοιχη συνάρτηση.

- Ως συνάρτηση f στην ευκλείδεια μετρική στο cube έχω χρησιμοποιήσει το mod.
- Στον υπερκύβο η συνθήκη τερματισμού της αναζήτησης είναι η εξής. Η αναζήτηση σταματάει όταν ξεπεράσουμε τον μέγιστο αριθμό κορυφών που μπορούμε να επισκεφτούμε **και** όταν ξεπεράσουμε τον μέγιστο αριθμό των σημείων που μπορούμε να εξετάσουμε. Για την ακρίβεια όταν έχει δοθεί αριθμός κορυφών που πρέπει να εξεταστούν τις εξετάζουμε όλες ανεξαρτήτων των σημείων που επισκεπτόμαστε. Άμα σαν τιμή στο probes έχει δοθεί το 0, τότε εξετάζουμε σημεία σε γειτονικές κορυφές (απόσταση hamming 1,2 κλπ) έως ότου συμπληρώσουμε τον μέγιστο αριθμό σημείων M.
- Για την αποφυγή του overflow έχω χρησιμοποιήσει long long μεταβλητή και επιλογή r μικρού μεγέθους.
- Για την αποφυγή αρνητικού αποτελέσματος στην πράξη % έχω υλοποιήσει mod που ξεπερνάει το πρόβλημα δίνοντας πάντα θετικές τιμές.

Αποτελέσματα και Συγκρίσεις

Μερικές ενδεικτικές εκτελέσεις του προγράμματος (όχι κατ' ανάγκη οι βέλτιστες) για τα αρχεία siftsmall/input_small(10000 εγγραφές) και siftsmall/query_small(1000 ερωτήματα)

k=4,l=4,w=350,r=[0,800] για το LSH
probes = 2 για το cube

Eucledian	LSH	CUBE
Mean tLSH	0.00160546	0.00322921
Mean tReal	0.0172348	0.0169664
Max Fraction	1.68201	3.22741
Mean Fraction	1.04636	1.08006

Παρατηρούμε πως το cube παρότι είναι σχετικά γρήγορο έχει μεγαλύτερο μέγιστο κλάσμα από το lsh γιατί ουσιαστικά ψάχνουμε 2 κορυφές του υπερκύβου.

L = 5

Cosine	LSH	CUBE
Mean tLSH	0.00678771	0.00193623
Mean tReal	0.00734517	0.00709059
Max Fraction	1.7123	6.96827
Mean Fraction	1.00627	1.06648

Παρατηρούμε πως στο cosine LSH ο χρόνος δεν διαφέρει και πολύ από την εξαντλητική μέθοδο. Αυτό δικαιολογείται από το γεγονός πως για k=4 δημιουργούνται ουσιαστικά 16 bucket στα οποία διασκορπίζονται οι εγγραφές χωρίς να αξιοποιείται όλος ο πίνακας. Έτσι οι αναζητήσεις σε όλους τους πίνακες θα τείνουν να ισοδυναμούν με μια μεγάλη αναζήτηση. Στο cube ο χρόνος μειώνεται αισθητά όμως έχουμε και πάλι μεγαλύτερο μέγιστο κλάσμα παρότι το μέσο κλάσμα είναι ικανοποιητικό καθώς ψάχνουμε 2 κορυφές του υπερκύβου.

K = 8, L=5 για το LSH
k=11, probes = 10 για το cube

cosine	LSH	CUBE
Mean tLSH	0.00122513	0.00138682
Mean tReal	0.00721835	0.00724153
Max Fraction	1.99258	4.08422
Mean Fraction	1.04343	1.07694

Για το LSH παίρνοντας k = 8 καταφέρνουμε να “σκορπίσουμε” τις εγγραφές στο hashtable μειώνοντας έτσι το χρόνο εκτέλεσης σε σχέση με το k=4 διατηρώντας ταυτόχρονα ένα χαμηλό μέσο και μέγιστο κλάσμα. Για το cube επιλέγοντας k=8 και probes = 10 επιτυγχάνουμε να μειώσουμε το μέγιστο κλάσμα διατηρώντας σταθερό μέσο κλάσμα και χρόνο εκτέλεσης.

M=500, k=6

cube	eucledian	cosine
Mean tLSH	0.00495969	0.00273642
Mean tReal	0.0175871	0.00720539
Max Fraction	2.53467	2.78088
Mean Fraction	1.05343	1.03037

Μια άλλη εναλλακτική που έχουμε προς την εκτέλεση του cube είναι να επισκεπτόμαστε γειτονικές κορυφές μέχρι να εξετάσουμε ένα μέγιστο αριθμό σημείων M. Παρατηρούμε λοιπόν πως και για τις δυο μετρικές εξετάζοντας ουσιαστικά 500 σημεία δηλαδή το 1/20 του dataset πετυχαίνουμε ικανοποιητικά αποτελέσματα.

Παραμετροποίηση των μεθόδων για επίτευξη ίσου μέγιστου κλάσματος

K=8, probes = 20 για το cube

K = 4, L = 5 για το LSH

Eucledian	LSH	CUBE
Mean tLSH	0.00171808	0.00337553
Mean tReal	0.0171984	0.0174772
Max Fraction	2.16269	2.18987
Mean Fraction	1.04384	1.06959

Παρατηρούμε πως με την παραμετροποίηση αυτή πετυχαίνουμε σχεδόν ίδιο κλάσμα με την ίδια μετρική και για τις δυο τεχνικές.

Χωρική – Χρονική Σύγκριση Μεθόδων

Για τη μέτρηση του χώρου που καταλαμβάνουν οι δυο δομές έχει κατασκευαστεί συνάρτηση που μετράει το χώρο μέσα από το πρόγραμμα με τη χρήση του sizeof().

Για κ=4,L=3 ο χώρος που καταναλώνει το lsh είναι 802.032 bytes ενώ για το cube είναι 244.300 bytes. Το αποτέλεσμα είναι λογικό αφού για το lsh δημιουργούνται οι τριπλάσιοι πίνακες κατακερματισμού άρα και το μέγεθος θα είναι περίπου τρεις φορές πιο μεγάλο.

Όσον αφορά τους χρόνους εκτέλεσης οι δυο δομές επιτυγχάνουν πολύ παρόμοιους χρόνους οι οποίοι εξαρτώνται κατά πολύ από την εκάστοτε παραμετροποίηση.