Ανάπτυξη λογισμικού για Αλγοριθμικά Προβλήματα

Μάρκος Βαρβαγιάννης

sdi1400017@di.uoa.gr

1η Εργασία

Υλοποίηση δομής για την εύρεση κοντινών γειτόνων στη γλώσσα C/C++

Η άσκηση αφορά την υλοποίηση δομών δεδομένων και αλγορίθμων για την γρήγορη και αποτελεσματική εύρεση των κοντινών γειτόνων. Σε μεγάλα και πραγματικά δεδομένα, η εξαντλητική αναζήτηση είναι ιδιαίτερα χρονοβόρα και δαπανηρή σε χρόνο, επομένως η ανάγκη κατασκευής τέτοιων δομών κρίνεται επιτακτική.
Έχουν υλοποιηθεί σε C++ όλα τα ζητούμενα της εργασίας, όπως περιγράφεται στην εκφώνηση. Οι αλγόριθμοι που υλοποιήθηκαν δηλαδή είναι:

- Ο αλγόριθμος **LSH** για διανύσματα στον d-διάστατο πραγματικό χώρο βάσει της *ευκλείδειας μετρικής* και της *ομοιότητας συνημιτόνου* (cosine similarity).
- Η μέθοδος τυχαίας προβολής στον **υπερκύβο**, που θα στηρίζεται στην προηγούμενη υλοποίηση του LSH.

Μεταγλώτιση

Για να μεταγλωτιστεί το πρόγραμμα τρέξτε **make**

Εκτέλεση

- Για το LSH:
  ./lsh –d <input file> –q <query file> –k <int> -L <int> -o <output file>
- Για τον υπερκύβο:
  ./cube –d <input file> –q <query file> –k <int> -M <int> -p <int> -o <output file>

Μορφή αρχείων:

Το πρόγραμμα τρέχει στη μορφή των αρχείων input και query που δόθηκαν στο eclass, με την προσθήκη απλά των:

- **@metric cosine** στην περίπτωση που θέλουμε την μετρική ομοιότητας συνημιτόνου και του
- **Radius: <double>** στο query set.

Αρχεία Κώδικα/Επικεφαλίδες:

- **lsh.cpp** , **cube.cpp**
  Τα αρχεία που περιέχουν τις main των lsh και cube αντίστοιχα
- **lsh_helping.cpp**, **cube_helping.cpp**
  lsh_helping.h, **cube_helping.h**

Βοηθητικές συναρτήσεις για το lsh (και για το cube αντίστοιχα) όπως οι συναρτήσεις που φτιάχνουν και επεξεργάζονται τις αντίστοιχες δομές.

- **mutual_helping.cpp**
  mutual_helping.h
  Αρχεία που περιέχουν κοινές συναρτήσεις και για τα δύο προγράμματα, όπως οι συναρτήσεις ειδόσου, ευκλείδιας και cosine απόστασης
- **dVector.cpp**
  dVector.h
  Περιέχει τον ορισμό της κλάσης του σημείου - vector
- **hFunction.cpp**
  hFunction.h
  Περιέχει τον ορισμό της κλάσης συνάρτησης h
- **parameter_values.h**
  Περιέχει ως defines κάποιες σταθερές και για τα δύο προγράμματα

Περιγραφή Δομών

- Για τον **LSH** χρησιμοποιήθηκε ένας hashtable (unordered_map) που έχει ως key_type string και mapped_type λίστες με δείκτες σε αντικείμενα κλάσης σημείου. Το διάνυσμα της g μετατρέπεται σε string και γίνεται hashing με κλειδί αυτό. Η προσπέλαση στα δεδομένα με αυτόν τον τρόπο είναι ιδιαίτερα γρήγορη. Ταυτόχρονα, με αυτόν τον τρόπο αποτρέπονται τα collisions διαφορετικών g για ίδιες τιμές της συνάρτησης

κατακερματισμού.
- Για τον **υπερκύβο** χρησιμοποιήθηκε ένας απλός πίνακας 2^k θέσεων με λίστες από δείκτες σε αντικείμενα κλάσης σημείου. Η ακολουθία από bits που προκύπτει μετά την εφαρμογή της f γίνεται ένας ακέραιος αριθμός που χρησιμοποιείται ως index για την προσπέλαση στον πίνακα.

Μετρήσεις

Ενδεικτικές εκτελέσεις:

- ./lsh -d data/input_small -q data/query_small -o data/output.txt
- ./cube -d data/input_small -q data/query_small -o data/output.txt -p 16 -M 2000 -k 5

Εν γένει παρατηρείται ότι τα αποτελέσματα του lsh είναι αρκετά καλύτερα από αυτά του υπερκύβου και όσον αφορά το κλάσμα προσέγγισης του κοντινότερου γείτονα, αλλά και όσον αφορά το χρόνο. Φυσικά μπορούμε να παραμετροποιήσουμε τον υπερκύβο με τέτοιο τρόπο ώστε ο χρόνος να γίνεται πολύ μικρός, αλλά τότε χάνουμε πολύ σε ακρίβεια. Αντιθέτως, σχετικά με τη χωρική πολυπλοκότητα, ο υπερκύβος υπερτερεί, καθώς έχουμε εναν απλό πίνακα από λίστες, σε αντίθεση με τον LSH που έχουμε L HashTables. Ενδεικτικά στο παράδειγμα των παραπάνω εκτελέσεων, οι Hashtables καταλαμβάνουν 61464 byttes, ενώ ο υπερκύβος μόλις 512!
Συμπερασματικά, το ποια θα είναι κατάλληλη μέθοδος για να χρησιμοποιήσουμε εξαρτάται αν θέλουμε να δώσουμε μεγαλύτερη βαρύτητα στο χρόνο (που επιλέγουμε τον LSH) ή στο χώρο (επιλέγουμε

τον υπερκύβο), δηλαδή εξαρτάται από τη φύση του εκάστοτε προβλήματος που έχουμε να αντιμετωπίσουμε.

Σημειώσεις:

- Χρησιμοποίησα το εργαλείο git για versioning.
- Προσπάθησα όσο μπορούσα να κάνω refactoring στον κώδικα, ωστόσο λόγω έλλειψης χρόνου, η διαδικασία θα ολοκληρωθεί στα επόμενα παραδοτέα.