Κ23γ: Ανάπτυξη Λογισμικού για Αλγοριθμικά Προβλήματα
Χειμερινό εξάμηνο 2016-17
1η Προγραμματιστική Εργασία
Υλοποίηση του Locality Sensitive Hashing (LSH) στη γλώσσα C/C++

Εκπονήθηκε από τους φοιτητές:

- Βασίλειος Δρέττας με ΑΜ:1115201300042
- Κυριακή Ράπτη με ΑΜ:1115201100105

Η άσκηση υλοποιήθηκε στο πλαίσιο του αλγορίθμου LSH (locality sensitive hashing). Σκοπός του προγράμματος για ένα σύνολο δεδομένων P είναι η εύρεση:

1. Των γειτόνων εντός ακτίνας R αντικειμένου q
2. Του κοντινότερου γείτονα αντικειμένου q

Το σύνολο δεδομένων P που αναφέρθηκε παραπάνω περιλαμβάνει αντικείμενα που «ζουν»:

1. Στον d-διάστατο πραγματικό διανυσματικό χώρο βάσει:
    a. Της ευκλείδειας μετρικής
    b. Της μετρικής cosine
2. Στον χώρο Hamming
3. Σε μετρικό χώρο όπου ορίζεται η απόσταση μέσω πίνακα αποστάσεων (metric spaces matrix)

Επιγραμματικά η εκτέλεση του προγράμματος είναι η εξής:

- Το πρόγραμμα διαβάζει δεδομένα από το dataset που θα δοθεί και αποθηκεύει τα δεδομένα σε μια λίστα.
- Ανάλογα με τη μέθοδο που θα ζητηθεί (hamming, cosine, Euclidean, metric spaces matrix) εκτελεί τον αλγόριθμο LSH και τοποθετεί τα δεδομένα σε L Hash Tables (τα δεδομένα τοποθετούνται ανά πίνακα και η διαδικασία επαναλαμβάνεται L φορές) με μέγεθος που καθορίζεται είτε από το k (πχ hamming) είτε από τον αριθμό των στοιχείων N (πχ Euclidean) .
- Διαβάζει τα queries που περιέχουν τα αντικείμενα προς αναζήτηση των γειτόνων τους.
- Βασισμένο στο R που θα δοθεί σαν ακτίνα, το πρόγραμμα υπολογίζει για το εκάστοτε αντικείμενο τους κοντινότερους του γείτονες εντός ακτίνας R
- Υπολογίζει μέσω του Lsh τον προσεγγιστικά κοντινότερο γείτονα και, την απόσταση του από το αντικείμενο καθώς και το χρόνο εύρεσης του γείτονα (ο οποίος υπολογίζεται μέσω timer)

- Υπολογίζει μέσω brute force τον πραγματικά κοντινότερο γείτονα , την απόσταση του από το αντικείμενο και το χρόνο εύρεσης του γ (επίσης υπολογίζεται μέσω timer)
- Εκτυπώνει τα ευρήματα για κάθε αντικείμενο στο σύνολο αναζήτησης.

Τα αρχεία που υλοποιήθηκαν για αυτή την άσκηση είναι τα εξής:

1. dataTypes.h : το αρχείο περιλαμβάνει τους τύπους των δεδομένων που αποθηκεύουμε στη λίστα κάθε φορά ανάλογα με το dataset εισόδου που μας δίνεται. Περιλαμβάνει τρεις κλάσεις: τη κλάση Vector (που χρησιμοποιείται για Euclidean και cosine) , τη κλάση Hamming και τη κλάση MatrixPoint (για το metric spaces matrix)
2. euclideanNode.h: δηλώνει την κλάση EuclideanNode που περιέχει δεδομένα τύπου Euclidean και το id του κάθε αντικειμένου. Αυτό συμβαίνει γιατί ζητείται από τον αλγόριθμο και η αποθήκευση του id (είναι βασικό στοιχείο για την επιτυχή εκτέλεση του LSH: "ID is locality sensitive: depends on *w*-length cells on the *v*-lines.") Επίσης περιλαμβάνει τα πρωτότυπα των συναρτήσεων για τη διαχείριση της κλάσης.
3. euclideanNode.cpp: ορισμός της κλάσης EuclideanNode
4. hashFunction.h : περιλαμβάνει την κλάση της hash function η οποία έχει υλοποιηθεί με templates. Τα templates χρησιμοποιήθηκαν προκειμένου να εξασφαλισθεί η επεκτασιμότητα του κώδικα. Με την αξιοποίηση τους ουσιαστικά χρησιμοποιείται ένας κορμός ο οποίος διαχειρίζεται διαφορετικού τύπου δεδομένα, δλδ, τύπου hamming ή cosine ή Euclidean ή matrix ανάλογα με την περίπτωση. Εδώ κιόλας χρησιμοποιούνται οι κλάσεις που ορίστηκαν στα datatypes.h , euclideanNode.h . Ανάλογα με το όρισμα και το dataset που έχουν δοθεί κατά την είσοδο η hash function χρησιμοποιεί την κατάλληλη κλάση για να διαχειριστεί τα δεδομένα που της δόθηκαν. Σε αυτό το αρχείο εμπεριέχεται ο κώδικας για τη hash function του hamming ενώ οι υπόλοιποι μέθοδοι υλοποιούνται σε διαφορετικά αρχεία.
5. hashFunction.cpp: ορισμός των κλάσεων που δηλώνονται στο hashFunction.h
6. hashFunctionCosine.h : δήλωση της κλάσης της cosine hash function. Όταν έχουμε δεδομένα τύπου cosine η συγκεκριμένη class καλείται από τον κορμό και χρησιμοποιείται για τον κατακερματισμό των δεδομένων.
7. hashFunctionCosine.cpp : το κύριο σώμα των συναρτήσεων που βρίσκονται στην class HashFunctionCosine

8. hashFunctionEuclidean.h : δήλωση της κλάσης της Euclidean hash function. Όταν έχουμε δεδομένα τύπου euclidean η συγκεκριμένη class καλείται από τον κορμό και χρησιμοποιείται για τον κατακερματισμό των δεδομένων.
9. hashFunctionEuclidean.cpp : ορισμός της class HashFunctionEuclidean
10. hashFunctionMatrix.h : δήλωση της κλάσης της Euclidean hash function. Όταν έχουμε δεδομένα που προέρχονται από το metric spaces matrix η συγκεκριμένη class καλείται από τον κορμό και χρησιμοποιείται για τον κατακερματισμό των δεδομένων.
11. hashFunctionMatrix.cpp : το κύριο σώμα των συναρτήσεων που βρίσκονται στην class HashFunctionMatrix.
12. hashtable.h: δήλωση της κλάσης του hash table. Και αυτή η κλάση είναι υλοποιημένη με templates προκειμένου να μπορεί να διαχειρίζεται διαφορετικού τύπου δεδομένα. Ανάλογα με τα δεδομένα του dataset εισόδου καλείται ο κορμός και στη συνέχεια η κατάλληλη hash function και γίνεται ο κατακερματισμός των δεδομένων. Είναι ένας πίνακας από λίστες(buckets) που κάθε φορά δείχνουν σε διαφορετικού τύπου δεδομένα.
13. hashtable.cpp: το κύριο σώμα των συναρτήσεων που βρίσκονται στην class HashTable.
14. List.h: δήλωση της class List. Και αυτή η κλάση υλοποιείται μέσω templates για λόγους επεκτασιμότητας. Η κλάση αυτή χρησιμοποιείται για την αποθήκευση των δεδομένων αλλά και για τα hash tables.
15. List.cpp: ορισμός της class List
16. lsh.h: Η κλάση Lsh εμπεριέχει τους L hashtables που προκύπτουν. Η μέθοδος LSH υλοποιείται με τον κατακερματισμό των δεδομένων L φορές δημιουργώντας L hashtables, όπου ο καθένας διέπεται από τη δική του gfunction (γενικά έχουμε L gfunctions = L*k hfunctions). Σε αυτή την κλάση επίσης γίνεται ανάγνωση των search queries και βρίσκονται οι κοντινότεροι γείτονες εντός ακτίνας R για κάθε ζητούμενο αντικείμενο, ο προσεγγιστικά κοντινότερος γείτονας, ο αληθινά κοντινότερος γείτονας και οι χρόνοι εύρεσης τους.
17. lsh.cpp: το κύριο σώμα των συναρτήσεων που βρίσκονται στην class LSH.
18. main.cpp: αρχείο για την κλήση των κλάσεων και των συναρτήσεων και τη γενικότερη εκτέλεση του προγράμματος. Η main είναι αυτή που καλεί όλα τα απαραίτητα στοιχεία για την επιτυχή υλοποίηση του LSH καθώς και την εμφάνιση των αποτελεσμάτων.
19. Makefile: αρχείο για τη συγκεντρωτική μεταγλώττιση του προγράμματος. Εμπεριέχονται όλες οι εντολές μεταγλώττισης και με την εκτέλεση αυτού του αρχείου γίνεται η επιτυχής μεταγλώττιση του προγράμματος
20. Node.h: δήλωση της κλάσης Node. Η κλάση αυτή χρησιμοποιείται για την υλοποίηση λίστας και υλοποιείται με templates για την προσαρμοστικότητα του προγράμματος ανάλογα με τον τύπο δεδομένων που δίνονται.

21. Node.cpp: ορισμός της class Node.
22. psedoRandomNumbers.h: τα πρωτότυπα των συναρτήσεων για τον υπολογισμό των τυχαίων μεταβλητών είτε βάση την ομοιόμορφη κατανομή είτε τη Gaussian κατανομή.
23. psedoRandomNumbers.cpp: κύριο σώμα των συναρτήσεων που ορίστηκαν στο psedoRandomNumbers.h
24. readFile.h: δήλωση των συναρτήσεων για την ανάγνωση των datasets, των query files για αναζήτηση καθώς και της ακτίνας r
25. readFile.cpp: ορισμός των παραπάνω συναρτήσεων που δηλώθηκαν στο readFile.h

Σημείωση: Στην άσκηση έχει υλοποιηθεί ο περιορισμός του 3L στην αναζήτηση για τον προσεγγιστικά κοντινότερο γείτονα καθώς και ο έλενχος του ID για την εύρεση του κοντινότερου στην ευκλείδια όμως δεν έδιναν τα καλύτερα δυνατά αποτελέσματα και οι έλεγχοι έχουν μπει σε σχόλια.

Εντολές Μεταγλώττισης:

- make (μόνο με αυτή την εντολή το σύστημα θα βρει το makefile – εφόσον έχουμε μόνο ένα- και θα εκτελέσει όλες τις εντολές μεταγλώττισης που εμπεριέχονται στο αρχείο)

Οδηγίες Χρήσης:

Η εντολή εκτέλεσης είναι ./lsh –d <input file> –q <query file> –k <int> -L <int> -o <output file> . Η παράμετρος -k και -L ειναι προαιρετικές. Αν δεν δίνονται έχουν τιμές k=4 και L=5.