Ανάπτυξη Λογισμικού για Αλγοριθμικά Προβλήματα
Άσκηση 1

Δημήτριος Μπούσουλας 1115201500106
Άγγελος Λάλουσης 1115201500081

Κατάλογος αρχείων:

Το παραδοτέο περιλαμβάνει 4 φακέλους, έναν για κάθε υποερώτημα. Συγκεκριμένα:

1. Ο φάκελος LSH_Vec αφορά την εφαρμογή του LSH για διανύσματα και περιλαμβάνει τα εξής αρχεία:

    - average.c: περιέχει υλοποίηση αλγορίθμων για την εύρεση την μέσης απόστασης μεταξύ των διανυσμάτων και για την μέτρηση της πραγματικής απόστασης μεταξύ του query και του κοντινότερου γείτονά του.
    - concat.c: για το concatenation μεσα στο lsh
    - factors.c: για τον υπολογισμών των παραγόντων m
    - hash.c: περιέχει τον κώδικα εισαγωγής σε hash table
    - input.c: για το διάβασμα του dataset και των queries
    - lsh.c: υλοποίηση του του lsh
    - main.c
    - modulo_calc: κώδικας για τον υπολογισμό του m^d % M
    - functions.h
    - structs.h
    - write_output.c: γράφει τα αποτελέσματα σε ένα αρχείο output
    - Makefile

2. Ο φάκελος Cube_Vec αφορά την εφαρμογή του hypercube για διανύσματα και περιλαμβάνει τα εξής αρχεία:

    - average.c: υπολογισμός r και κώδικας εξαντήτικής αναζήτησης
    - cube.c: υλοποίηση του αλγορίθμου του hypercube

- factors.c
- hamming_distance.c
- input.c
- lsh.c
- main.c
- modulo_cal.c
- write_output.c
- functions.h
- structs.h
- Makefile

3. Ο φάκελος LSH_Curves αφορά την εφαρμογή του LSH για καμπύλες που ανάγονται σε διανύσματα μέσω grid και περιλαμβάνει τα εξής αρχεία:

    - average.c:
    - concat_curve.c: concatenation κατά το gridding
    - concat.c: concatenation για το lsh
    - count_d: υπολογισμός του δ
    - dtw.c: υλοποίηση dtw ώστε να μπορεί να υποστηρίξει και backtracking
    - euclidean.c: για τον υπολογισμό της ευκλείδιας απόστασης μεταξύ σημείων
    - factors.c
    - hash.c
    - Input.c
    - lsh.c
    - main.c
    - modulo_calc.c
    - snap.c
    - write_output.c
    - functions.h
    - structs.h
    - Makefile

4. Ο φάκελος Cube_Curves αφορά την εφαρμογή του Cube για καμπύλες που ανάγονται σε διανύσματα μέσω grid και περιλαμβάνει τα εξής αρχεία:
    - concat_curve.c
    - count_d: Περιλαμβάνει τον υπολογισμό του δ
    - cube.c
    - dtw.c

- euclidean.c
- factors.c
- hamming_distance.c
- input.c
- lsh.c
- main.c
- modulo_calc.c
- search.c: Περιλαμβάνει τον κώδικα εξαντλητικής αναζήτησης
- snap.c
- structs.h
- functions.h
- Makefile

Παραδοχές:

- Για να θέσουμε w = 4500 υπολογίζουμε το r (μέση ελάχιστη απόσταση 2 vectors), με την χρήση της average.c τι οποίο επέστρεψε 1026 και θέτουμε w=4500, που είναι περίπου 4*r. Έχουμε σχολιασμένο την κλήση της average για να αποφύγουμε το overhead.
- Για τον υπολογισμό του δ ακολουθούμε ακριβώς την ίδια τακτική, η τιμή η οποία πήραμε για το δ είναι περίπου 0,0022
- Καθώς το δ πρέπει να είναι ακέραιος, το δ αποφασίσαμε να το πολλαπλασιάσουμε με 1000 ώστε να γίνει 2 (έπειτα από στρογγυλοποίηση). Για να γίνει σωστά το snap με τις τιμές του grid και τα δεδομένα κάθε καμπύλης πολλαπλασιάζονται με το 1000 για να βρεθεί με ποιον αριθμό απ' το grid θα αντιστοιχηθεί κάθε συντεταγμένη κάθε σημείου των καμπυλών.
- Στην υλοποίηση του lsh χρειάζεται πολλές φορές να υπολογίζουμε το m^i, όπου το i παίρνει τιμές από 1 εως d-1. Για λόγους πολυπλοκότητας πράξεων αποφασίσαμε να αποθηκεύσουμε αυτά τα d-1 στο πλήθος m. Αυτόν τον υπολογισμό και την αποθήκευση κάνει η factors.c.
- Σε όλα τα εκτελέσιμα έχουμε κάνει την παραδοχή ότι είτε θα δωθούν όλες οι παράμετροι από την γραμμή εντολών είτε θα δωθούν οι default τιμές για τις παραμέτρους και θα διαβαστούν από το πληκτρολόγιο μόνο τα αρχεία input, query και output.

Περιγραφή:

LSH_Vec:

Ξεκινάει διαβάζοντας το input και το query. Έπειτα καλεί τον κώδικα για να πάρει τα αποτελέσματα εξαντλητικής ανζήτησης για το output. Στην συνέχεια εκτελείται ο κώδικας για το lsh, όπου φτιάχνει κ h για κάθε έναν από τους L hashtables. Κάθε h αποτελείται από τα si, τα οποία δημιουργούνται τυχαία με κανονική κατανομή στο διάστημα [0,w). Στην συνέχεια υπολογίζονται οι m_factors και καλείται η lsh_train η οποία εκτελεί τον αλγόριθμο της lsh γεμίζοντας κάθε έναν από τους L hashtables με τα διανύσματα του input. Τέλος καλείται η lsh_serach για κάθε διάνυσμα απ' το query file το οποίο το οδηγεί σε ένα κελί από κάθε hashtable και επιστρέφει το διάνυσμα του input file που ανήκει στο ίδιο κελί και έχει την μικρότερη απόσταση από το query.

Η εντολή εκτέλεσης είναι όπως ζήταγε η εκφώνηση δηλαδή:
./lsh –d<inputfile> –q<queryfile> –k<int> -L<int> -o <outputfile>

Cube_Vec:

Ξεκινάει διαβάζοντας το input και το query. Έπειτα καλεί τον κώδικα για να πάρει τα αποτελέσματα εξαντλητικής αναζήτησης για το output. Στην συνέχεια εκτελείται ο κώδικας για το lsh ελαφρά αλλαγμένος ώστε να μην υπολογίζει τις g. Εφόσον έχουμε τις h από τον lsh καλείται η cube_train όπου εισάγει κάθε διάνυσμα σε μία κορυφή του υπερκύβου. Για λόγους πολυπλοκότητας κάθε κορυφή έχει έναν hashtable ο οποίος περιέχει τα διανύσματα της κορυφής. Στην συνέχεια καλούνται η lsh_search και η cube_search για κάθε διάνυσμα του query ώστε να αντιστοιχηθεί σε μία κορυφή του υπερκύβου και να βρει το κοντινότερο διάνυσμα αυτής της κορυφής αλλά και αλλων 24 (probes-1 δηλαδή) κορυφών ξεκινώντας απ' αυτές που έχουν hamming distance 1 και ανεβαίνοντας αν χρειαστεί.

Η εντολή εκτέλεσης είναι όπως ζήταγε η εκφώνηση δηλαδή:
. /cube –d <input file> –q <query file> –k <int> -M <int> -probes <int> -o<output file>

LSH_Vec:

Αρχικά διαβάζονται οι καμπύλες του dataset. Δημιουργούνται τα L grids και στη συνέχεια με την συνάρτηση snap κάθε καμπύλη αντιστοιχίζεται με L καινούργιες οι οποίες θα αποτελούνται από ακέραιες συντεταγμένες. Στη συνέχεια με την concat_curve μετατρέπονται τα grid_curves σε διανύσματα. ΄Υστερα εφαρμόζεται lsh στα διανύσματα σύμφωνα με τον κώδικα του προηγούμενου ερωτήματος. Εδώ ο κώδικας αυτός παρουσιάζει μικρές παραλλαγές λόγω του ότι τώρα έχουμε L διανύσματα για κάθε καμπύλη και ένα hash table για καθένα από αυτά.

Το τελευταίο κομμάτι που αφορά την εφαρμογή του lsh στα διανύσματα δεν τρέχει σωστά! Σε αυτό το σημείο η εφαρμογή γίνεται killed!

Η εντολή εκτέλεσης είναι όπως ζήταγε η εκφώνηση δηλαδή:
$./curve –d <input file> –q <query file> –k <int> -L <int> -o <output file> -a LSH –h LSH

Cube_Curve:

Ξεκινάει διαβάζοντας το input file. Έπειτα δημιουργεί τα L grids, ώστε να καλέσει την snap η οποία θα αντιστοιχήσει κάθε καμπύλη με L καινούργιες οι οποίες θα αποτελούνται από ακέραιους αριθμούς οι οποίου ανήκουν στα grids. Στην συνέχεια καλείται η concat_curve ώστε να μετατραπούν τα grid_curves σε διανύσματα. Τέλος για τα διανυσματα εκτελείται ο κώδικας που δημιουργήθηκε για την Cube_Vec με μόνη διαφορά ότι για να συγκριθεί η απόσταση μεταξύ 2 διανυσμάτων μετράμε την απόσταση μεταξύ των αρχικών καμπυλών που αντιστοιχούν σε αυτά τα διανύσματα.

Δεν τρέχει σωστά! Με την εκτέλεση του προγράμματος ο υπολογιστής ζορίζεται πολύ και τρέχει για αρκετή ώρα χωρίς να τερματίζει καθώς δεν προλάβαμε να βρούμε το λογικό λάθος στον κώδικα.

Η εντολή εκτέλεσης είναι όπως ζήταγε η εκφώνηση δηλαδή:
$./curve –d <input file> –q <query file> –k <int> -L <int> -o <output file> -a LSH –h Hypercube