2017

Κατακερματισμός και αναζήτηση για πολυγωνικές καμπύλες

Ανάπτυξη Λογισμικού για Αλγοριθμικά Προβλήματα
Μέρος 1ο

Κανελλόπουλος Στέφανος
(Α.Μ. 1115201200050)

Χανιωτάκης-Ψύχος Χαρίδημος
(Α.Μ. 1115201200194)

Περιγραφή προγράμματος

Αρχικά, το πρόγραμμα μας δέχεται ένα αρχείο εισόδου με καμπύλες δύο, τριών ή τεσσάρων διαστάσεων. Από κάθε καμπύλη δημιουργούμε K καμπύλες πλέγματος, οι οποίες στην συνέχεια συγκολλούνται για να δημιουργήσουν το key για την ζητούμενη hash function (classic ή probabilistic). Έπειτα γίνεται η εισαγωγή της καμπύλης στο hashtable με βάση το index που μας δώσει η hash function. Η παραπάνω διαδικασία επαναλαμβάνεται L φορές, το οποίο έχει σαν αποτέλεσμα την δημιουργία της τελικής δομής αναζήτησης.

Στην συνέχεια, το πρόγραμμα μας δέχεται ένα αρχείο εισόδου με καμπύλες ιδίας διάστασης με τις παραπάνω καμπύλες, για την εκτέλεση της αναζήτησης. Θα πρέπει αυτές να μετατραπούν σε καμπύλες πλέγματος, όμοια με πριν, και με βάση τα indexes που θα δώσει η επιλεγμένη hash function για την κάθε μια, θα γίνει η αναζήτηση στα αντίστοιχα buckets του hashtable. Από αυτά τα buckets θα ελεγχθούν μόνο οι καμπύλες που έχουν ίδια καμπύλη πλέγματος και θα υπολογιστεί η απόσταση με βάση την μετρική που επέλεξε ο χρήστης. Σε περίπτωση που δεν βρεθεί τέτοια καμπύλη θα υπολογιστεί η απόσταση από κάθε καμπύλη των συγκεκριμένων buckets και θα επιλεχθεί η πιο κοντινή. Το ίδιο ισχύει και για τους R-nearest γείτονες. Στο τελευταίο βήμα της αναζήτησης, εκτελείται μια εξαντλητική αναζήτηση στο σύνολο ενός hashtable για να βρεθεί η αληθινά πιο κοντινή καμπύλη.

Τέλος, τα αποτέλεσμα εξάγονται σε ένα αρχείο που έχει επιλέξει ο χρήστης.

Τα παραπάνω αποτελούν την υλοποίηση ενός προγράμματος που σχετίζεται με την δημοσίευση με τίτλο "Locality-Sensitive Hashing of Curves" των A. Driemel and F. Silvestri.

Link: http://drops.dagstuhl.de/opus/volltexte/2017/7203/pdf/LIPIcs-SoCG-2017-37.pdf

Αρχεία κώδικα και σύντομη περιγραφή τους

- makefile: Μεταγλώττιση προγράμματος και παραγωγή εκτελέσιμου αρχείου
- main.c: Ελέγχει την ορθή λήψη των παραμέτρων από τον χρήστη και καλεί τις δυο κύριες συναρτήσεις, preprocessing() και search()
- structs.c: Περιλαμβάνει όλες τις δομές που ήταν απαραίτητες για την υλοποίηση του προγράμματος (hash table, linked list κτλ.)
- functions.c: Περιλαμβάνει τις συναρτήσεις που είναι κοινές τόσο για την δημιουργία της δομής αναζήτησης (pre-processing) όσο και για την εύρεση των ζητημένων καμπυλών (search)
- functions.h: Περιλαμβάνει όλες τις standard libraries, τα δικά μας αρχεία επικεφαλίδων όπως και τις συμβολικές σταθερές
- hashtable.c: Περιλαμβάνει τις συναρτήσεις για το hash table (init, insert, destroy) όπως και τις συναρτήσεις για τις δύο μεθόδους hashing (classic, probabilistic)
- hashtable.h: Αρχείο επικεφαλίδας του hashtable.c
- metric_functions.c: Περιλαμβάνει τις συναρτήσεις για τον υπολογισμό απόστασης μεταξύ δύο καμπυλών.
- metric_functions.h: Αρχείο επικεφαλίδας του metric_functions.c
- output_functions.c: Περιλαμβάνει τις συναρτήσεις για την εξαγωγή των αποτελεσμάτων στο output file
- output_functions.h: Αρχείο επικεφαλίδας του output_functions.c
- preprocessing.c: Περιλαμβάνει τις συναρτήσεις για την δημιουργία της δομής αναζήτησης
- preprocessing.h: Αρχείο επικεφαλίδας του preprocessing.c
- quicksort.c: Περιλαμβάνει τις συναρτήσεις για την υλοποίηση της quicksort για ταξινόμηση μονοδιάστατου πίνακα από integers
- quicksort.h: Αρχείο επικεφαλίδας του quicksort.c
- search.c: Περιλαμβάνει τις συναρτήσεις για την αναζήτηση για κάθε query curve, είτε με την χρήση της παραμέτρου –stats είτε χωρίς αυτήν
- search.h: Αρχείο επικεφαλίδας του search.c

Μεταγλώττιση και χρήση του προγράμματος

Για την μεταγλώττιση του προγράμματος περιλαμβάνεται αρχείο makefile και πραγματοποιείται με την εντολή "./make" στον τρέχοντα κατάλογο. Με την εντολή "./make clean" γίνεται ο καθαρισμός των αρχείων αντικειμένων.

Το πρόγραμμα εκτελείται με την παρακάτω εντολή:

```
./lsh -d <input file> -q <query file> -k <int> -L <int> -o
<output file> -stats [optional] -function {DFT, DTW} -hash
{classic, probabilistic}
```

Σε περίπτωση που κάποιο από τα παραπάνω ορίσματα δεν δοθεί στο command line, ο χρήστης έχει την δυνατότητα σε runtime να τα δώσει (θα ζητηθούν όλα όσα λείπουν).

Επιπλέον παρατηρήσεις/παραδοχές

- Για την εύρεση του true nearest neighbor πραγματοποιείται εξαντλητική αναζήτηση, δηλαδή σε όλα τα buckets, μόνο σε ένα hashtable και όχι στα L, καθώς περιλαμβάνουν τις ιδίες καμπύλες που έχουν κατακερματιστεί διαφορετικά.
- Τόσο στο input file όσο και στο query file η πρώτη γραμμή πάντα θα περιέχει την πληροφορία για το dimension και την διάμετρο (R) αντίστοιχα.
- Για το μέγεθος του table κάθε hashtable έχουμε επιλέξει τον αριθμό των καμπυλών στο input file διά τέσσερα.
- Στον παραδοτέο φάκελο *src* συμπεριλαμβάνεται ένα mini_dataset και ένα mini_query με βάση τα οποία ελέγξαμε τον κώδικα μας, για την ορθή λειτουργία του.

Έγινε έλεγχος με valgrind για memory leaks και τα αποτελέσματα φαίνονται παρακάτω.

```
==14487==
==14487== HEAP SUMMARY:
==14487==     in use at exit: 40,532 bytes in 2,055 blocks
==14487==   total heap usage: 5,191 allocs, 3,136 frees, 574,052 bytes allocated
==14487==
==14487== Searching for pointers to 2,055 not-freed blocks
==14487== Checked 54,668 bytes
==14487==
==14487== LEAK SUMMARY:
==14487==    definitely lost: 19,780 bytes in 840 blocks
==14487==    indirectly lost: 20,752 bytes in 1,215 blocks
==14487==      possibly lost: 0 bytes in 0 blocks
==14487==    still reachable: 0 bytes in 0 blocks
==14487==         suppressed: 0 bytes in 0 blocks
==14487==
```

Παρατηρούμε ότι το συγκεντρωτικό memory leak του προγράμματος ανέρχεται στα 40.532 bytes από τα συνολικά 574.052 bytes, που αντιστοιχούν στο 7% της συνολικής μνήμης που χρησιμοποιήθηκε.