

Μέρος 1ο: LSH & Hypercube

1 Μεταγλώττιση & Εκτέλεση

Τα προγράμματα μεταγλωττίζονται με τη χρήση του αρχείου Makefile και συγκεκριμένα τις εντολές:

- `make`
 - Για τη δημιουργία και των δύο εκτελέσιμων
- `make lsh`
 - Για τη δημιουργία μόνο του εκτελέσιμου `lsh`
- `make cube`
 - Για τη δημιουργία μόνο του εκτελέσιμου `cube`

Για την εκτέλεση του LSH:

```
./lsh -d <input file> -q <query file> -k <int> -L <int> o- <output file>
```

Όπου L είναι το πλήθος των hash tables που θα δημιουργηθούν και k είναι το πλήθος των συναρτήσεων κατακερματισμού h_i ανά hash table.

Για την εκτέλεση της τυχαίας προβολής στον υπερκύβο:

```
./cube -d <input file> -q <query file> -k <int> -M <int> -p <int> o- <output file>
```

Όπου k είναι η διάσταση του υπερκύβου, M το μέγιστο πλήθος σημείων που θα ελέγχονται σε κάθε αναζήτηση και p το μέγιστο πλήθος κορυφών που θα ελέγχονται ανά αναζήτηση.

2 Οργάνωση Κώδικα

Ο κώδικας οργανώνεται σε διαφορετικά αρχεία ανάλογα με το σκοπό και τη λειτουργικότητά του. Συγκεκριμένα στα εξής αρχεία:

- `lsh.cpp`
 - Περιέχει την υλοποίηση του LSH καθώς και την εκτέλεση του με εκτύπωση αποτελεσμάτων και στατιστικών
- `cube.cpp`
 - Περιέχει την υλοποίηση της προβολής στον υπερκύβο καθώς και την εκτέλεση του με εκτύπωση αποτελεσμάτων και στατιστικών
- `hash_table.cpp/.hpp`

- Περιέχει την υλοποίηση των πινάκων κατακερματισμού που χρησιμοποιεί ο αλγόριθμος LSH
- `hasher.cpp/.hpp`
 - Περιέχει την υλοποίηση των μετρικών που χρησιμοποιούνται και από τις δύο μεθόδους (ευκλείδεια απόσταση και συνημιτόνου)
- `point.cpp/.hpp`
 - Περιέχει την υλοποίηση του ΑΤΔ point το οποίο αναπαριστά τα σημεία των datasets και τα διανύσματα των αλγορίθμων
- `util.cpp/.hpp`
 - Περιέχει γενικές συναρτήσεις που χρησιμοποιούνται και από τις δύο μεθόδους

3 Σύγκριση Μεθόδων

Για την σύγκριση των δύο μεθόδων θα χρησιμοποιήσουμε την **μετρική συνημιτόνου**.

3.1 Ενδεικτικές Εκτελέσεις

Στη συνέχεια βλέπουμε τις επιδόσεις των μεθόδων για διαφορετικές παραμέτρους. Είναι πολύ σημαντικό να σημειωθεί ως σημείο αναφοράς ότι ο ντετερμινιστικός αλγόριθμος πλησιέστερου γείτονα τρέχει κατά μέσο όρο σε χρόνο 0.05s. Χρησιμοποιούμε ένα dataset 10000 σημείων ως είσοδο και ένα dataset 100 σημείων για queries.

3.1.1 LSH

k	L	Max Approx.	Avg. Approx	Avg. Time
3	5	1	1	0.2292
4	5	1.0559	1.0011	0.2089
6	5	1.1177	1.0050	0.1469
8	5	1.1070	1.0058	0.0900
10	5	1.5502	1.0367	0.0456
12	5	1.8626	1.0400	0.0416
14	5	2.1293	1.0489	0.0298
15	5	2.1771	1.0786	0.0202
16	5	2.3813	1.0956	0.0274
16	6	2.1134	1.0783	0.0421
16	7	2.0756	1.0808	0.0357
16	8	1.8307	1.0690	0.0235
16	9	1.7638	1.0438	0.0341
16	10	1.6221	1.0505	0.0464

3.1.2 Hypercube

k	M	p	Max Approx.	Avg. Approx	Avg. Time
4	500	4	5.0334	1.3621	0.0027
5	500	4	4.9590	1.4644	0.0025
6	500	4	3.3547	1.2262	0.0024
6	1000	4	3.1239	1.1584	0.0050
6	1000	6	1.7797	1.0844	0.0054
6	1000	10	2.4349	1.0989	0.0052
6	2000	10	2.6356	1.0793	0.0069
7	3000	15	2.2929	1.0542	0.0119
7	3000	32	1.7262	1.0447	0.0131
7	3000	64	1.6721	1.0509	0.0138
5	3000	15	2.2484	1.0857	0.0149
5	3000	32	1.8530	1.0498	0.0150

3.1.3 Παρατηρήσεις & Σχόλια

Από τις παραπάνω εκτελέσεις παρατηρούμε ότι και οι δύο μέθοδοι καταφέρνουν αρκετά καλά αποτελέσματα με σχετικά παρόμοια ακρίβεια (συγκεκριμένα για $k = 16, L = 9$ στο LSH και $k = 7, M = 3000, p = 64$ στον υπερκύβο). Παρατηρούμε όμως ότι η μέθοδος του υπερκύβου είναι περίπου 2 φορές πιο γρήγορη από το LSH και περίπου 3 φορές πιο γρήγορη από τον ντετερμινιστικό αλγόριθμο.

Επίσης, για το LSH παρατηρούμε ότι δεν έχει νόημα η παράμετρος k να παίρνει τιμές κάτω από 14 για τη συγκεκριμένη μετρική, διότι ο χρόνος που πετυχαίνει είναι χειρότερος από τον ντετερμινιστικό αλγόριθμο. Αυτό συμβαίνει διότι η συνάρτηση κατακερματισμού έχει πολλές συγκρούσεις για μικρά k .

Ακόμη, στον υπερκύβο παρατηρούμε ότι είναι σημαντικό όταν αυξάνεται το k να αυξάνεται και το p , το οποίο είναι λογικό αφού όταν έχουμε πολλές κορυφές θα πρέπει να επιτρέπουμε στον αλγόριθμο να επισκέπτεται περισσότερες.

3.2 Κατανάλωση Χώρου

3.2.1 LSH

Για την υλοποίηση του LSH χρησιμοποιούνται L hash tables, καθένα από τα οποία χρησιμοποιεί τη δική του συνάρτηση κατακερματισμού και κάθε μια απ' αυτές έχει k "υπο-συναρτήσεις" με ξεχωριστά διανύσματα ανάλογα τη μετρική.

Για κάθε hash table δημιουργούνται k buckets, τα οποία αναπαρίστανται με C++ vectors και σε αυτά αποθηκεύονται οι δείκτες των σημείων που είναι αποθηκευμένα σε μια εξωτερική δομή για την αποφυγή της πολλαπλής (για την ακρίβεια k -πλής) αποθήκευσης των σημείων.

3.2.2 Υπερκύβος

Στην περίπτωση του υπερκύβου, έχουμε 2^k κορυφές οι οποίες έχουν ίδια μορφή με τα buckets των hash tables (C++ vectors με pointers σε σημεία), μόνο μια συνάρτηση κατακερματισμού με 2^k υποσυναρτήσεις και k συναρτήσεις $f : x \rightarrow [f_1(h_1(x), \dots, f_{2^k}(h_{2^k}(x)))] \in \{0, 1\}^{2^k}$. Για την υλοποίηση των συναρτήσεων f χρησιμοποιείται η δομή unordered_map της C++ όπου κάθε τιμή που δεν περιέχεται ήδη στο map προβάλεται τυχαία στο $\{0, 1\}$.