Κ23γ: Ανάπτυξη Λογισμικού για Αλγοριθμικά Προβλήματα

1η Προγραμματιστική Εργασία - Αναζήτηση και συσταδοποίηση διανυσμάτων στη C++

Μαρία Σιακαβέλλα - 1115201400181

Τσάμης Αντώνιος - 1115201300183

Περιγραφή του προγράμματος

Το πρόγραμμα περιέχει κώδικα που υλοποιεί τους αλγορίθμους LSH, Hypercube και Clustering με τις διάφορες προδιαγραφές και παραλλαγές που περιγράφονται στην εκφώνηση της εργασίας. Ο κώδικας βρίσκεται χωρισμένος σε δύο φακέλους. Στον φάκελο mains περιέχονται τρεις mains, μια για κάθε αλγόριθμο. Στον φάκελο source περιέχονται headers και υλοποιήσεις κλάσεων, δομών και συναρτήσεων για τους τρεις βασικούς αλγορίθμους (LSH, Hypercube και Clustering), και για βοηθητικές αλλά θεμελιώδεις για το πρόγραμμα διαδικασίες όπως το διάβασμα των δεδομένων και οι πράξεις μεταξύ διανυσμάτων. Όλοι οι αλγόριθμοι υλοποιήθηκαν με βάση τις διαφάνειες του μαθήματος, τις διαλέξεις και την εκφώνηση της εργασίας. Θα θέλαμε να σημειώσουμε μια, ενδεχομένως, ιδιαιτερότητα στην υλοποίηση μας. Όλοι οι αλγόριθμοι λειτουργούν στατικά (static). Το διάβασμα και η αποθήκευση των δεδομένων γίνεται σε μια στατική δομή και στη συνέχεια ο αλγόριθμος που πρόκειται να εκτελεστεί δημιουργεί και αυτός με τη σειρά του μια στατική δομή (για την αποθήκευση π.χ. των συναρτήσεων g και h και για τα σημεία καθε bucket από κάθε hash table για τον lsh). Προτιμήσαμε αυτήν την υλοποίηση γιατί σύμφωνα με την περιγραφή της εκφώνησης δεν πρόκειται ποτέ να χρειαστούν δύο στιγμιότυπα της δομής ενός αλγορίθμου, οπότε το μοναδικό στιγμιότυπο για το οποίο υπάρχει ανάγκη αποφασίσαμε να είναι στατικό, κυρίως για την διευκόλυνση μας κατά την ανάπτυξη του κώδικα, την αποφυγή πολλαπλών αναθέσεων, αναφορών και στιγμιοτύπων και την απλοποίηση του προγράμματος.

Github link: mariasiak/project_2023 (github.com)

Κατάλογος των αρχείων κώδικα / επικεφαλίδων και περιγραφή τους

Κώδικας υπάρχει σε δύο φακέλους. Στον φάκελο source βρίσκονται κεφαλίδες και υλοποιήσεις διαφόρων modules. Στον φάκελο mains υπάρχουν τρεις main συναρτήσεις μια για κάθε αλγόριθμο (lhs, cube, clustering). Στις τρεις main γίνεται το διάβασμα της εισόδου, η κλήση του κατάλληλου αλγορίθμου με τα σωστά

ορίσματα και η δημιουργία του αρχείου εξόδου. Παρακάτω παρουσιάζονται περισσότερα σχόλια για κάθε module του φακέλου source:

cluster.hpp
Περιέχει τον ορισμό και την υλοποίηση ενός struct συστάδας, που με τη σειρα του περιέχει το κεντροειδές της συστάδας και έναν πίνακα με τα αναγνωριστικά των σημείων της συστάδας. Περιέχει επίσης μια συνάρτηση για τον υπολογισμό του κεντροειδούς ως τον μέσο όρο των σημείων της συστάδας.

clustering.cpp & clustering.h
Περιέχει την κλάση του clustering με μεθόδους για την αρχικοποίηση των κεντροειδών με την μέθοδο K-means++, την εκτέλεση του αλγορίθμου Lloyd's, την εκτέλεση της αντίστροφης μεθόδου ακτίνας με lsh και την εκτέλεση της αντίστροφης μεθόδου ακτίνας με hypercube. Επίσης περιέχει μια συνάρτηση για τον υπολογισμό του score Σιλουέτας και κάποιες μεθόδους εκτύπωσης και προσπέλασης.

Παρατήρηση: ο υπολογισμός του score Σιλουέτας παίρνει πολύ χρόνο.

common.h
Περιέχει defines, includes και typedefs καθοριστικά για όλο το πρόγραμμα και γίνεται Include στα headers όλων των κλάσεων.

data_handler.cpp & data_handler.h
Περιέχει συναρτήσεις και μεταβλητές για την φόρτωση, αποθήκευση και προσπέλαση του dataset. Όλα το dataset σώζεται στατικά εδώ και όλοι οι άλλοι αλγόριθμοι χρησιμοποιούν getters αυτού του module για να έχουν πρόσβαση στα δεδομένα μέσω της θέσης των σημείων στο dataset, με την χρήση δηλαδή ενός int i. Να σημειωθεί ότι αρχικά δεν μπορούσαμε να βρούμε το ubyte αρχείο του mnist γιατί δεν είχαμε τον κωδικό που ζητάει το link της εκφώνησης στα windows. Για αυτό αρχικά αναπτύξαμε τον κώδικα με ένα dataset του mnist που ήταν αποθηκευμένο ως csv και είχε και labels για τις εικόνες. Υπάρχουν μέθοδοι για την ανάγνωση και csv και ubyte, αλλά στο παραδοτέο χρησιμοποιούνται οι ubyte μέθοδοι.

hash.hpp
Περιέχει ορισμούς και υλοποιήσεις για συναρτήσεις κατακερματισμού. Συγκεκριμένα τις h, g, f όπως περιγράφονται στις διαφάνειες για την δημιουργία των δομών του αλγορίθμου lsh και του αλγορίθμου hypercube.

hypercube.cpp & hypercube.h
Περιέχει την υλοποίηση του hypercube με μεθόδους για την αρχικοποίηση της δομής και την εκτέλεση αναζήτησης κοντινών γειτόνων (έναν, K, εντός ακτίνας).

lsh.cpp & lsh.h
Περιέχει την υλοποίηση του lsh με μεθόδους για την αρχικοποίηση της δομής και την εκτέλεση αναζήτησης κοντινών γειτόνων (έναν, K, εντός ακτίνας).

my_rand.cpp & my_rand.h
Δεν περιέχει κάποια υλοποίηση, αλλά διευκολύνει και οργανώνει την δημιουργία τυχαίων αριθμών με χρήση καθιερωμένων μεθόδων παραγωγής τυχαίων αριθμών στη C++.

nearest_neighbor.cpp & nearest_neighbor.h
Περιέχει εξαντλητικές αναζητήσεις κοντινότερου γείτονα και K-κοντινότερων γειτόνων.

test_main
Δεν χρησιμοποιείται πλέον. Κατά την ανάπτυξη του κώδικα καλούσε δοκιμαστικές main από το επόμενο module.

testing.cpp & testing.h
Περιέχονται δοκιμαστικές main που χρησιμοποιήθηκαν κατά την ανάπτυξη του κώδικα για τον έλεγχο και την υλοποίηση των διαφόρων δομών και αλγορίθμων. Δεν παίζει κάποιο ρόλο στην παρούσα φάση της ανάπτυξης αλλά λειτουργεί σαν ένα ημερολόγιο της εξέλιξης του κώδικα.

vec_math.cpp & vec_math.h
Περιέχει διάφορες μεθόδους για πράξεις με διανύσματα.

Οδηγίες μεταγλώττισης του προγράμματος

Η μεταγλώττιση του προγράμματος γίνεται με το εργαλείο make. Υπάρχουν τρεις main, μια για τον LSH, μια για τον Hypercube και μια για το clustering. Η μεταγλώττιση της κάθε main και των modules γίνεται αντίστοιχα με τις εντολές:

- make lsh
- make cube
- make cluster

Με την εντολή make all (ή σκέτο make) γίνεται η μεταγλώττιση και των τριών αρχείων.

Οδηγίες χρήσης του προγράμματος

Η χρήση του προγράμματος γίνεται σύμφωνα με τις προδιαγραφές της εκφώνησης.

Για τον lsh:
./lsh -d <input file> -q <query file> -k <int> -L <int> -o <output file> -N <number of nearest> -R <radius>

Για τον hypercube:
./cube -d <input_file> -q <query file> -k <int> -M <int> -probes <int> -o <output file> -N <number of nearest> -R <radius>

Για το clustering:
./cluster -i <input file> -c <configuration file> -o <output file> -complete <optional> -m <method: Classic or LSH or Hypercube>

Για το clustering to configuration file χρησιμοποιείται όπως προβλέπει η εκφώνηση και πρέπει να έχει την μορφή:

```
number_of_clusters:<int>                  // K of K-medians
number_of_vector_hash_tables: <int>       // default: L=3
number_of_vector_hash_functions: <int>    // k of LSH for vectors, default: 4
max_number_M_hypercube: <int>             // M of Hypercube, default: 10
number_of_hypercube_dimensions: <int>     // k of Hypercube, default: 3
number_of_probes: <int>                   // probes of Hypercube, default: 2
```

Επίσης στο αρχείο makefile υπάρχουν και οι εντολές lsh_run, cube_run, cluster_run με τις οποίες μπορεί να γίνει εκτέλεση του αντίστοιχου αλγορίθμου με κάποιες δοσμένες τιμές. Μπορείτε να τροποποιήσετε το makefile και τις εντολές αυτές για να εκτελέσετε τους αλγορίθμους με τις παραμέτρους που επιθυμείτε. Επίσης θα βρείτε και ένα αρχείο cluster.conf με την μορφή του αρχείου παραμέτρων για το clustering (όπως φαίνεται παραπάνω), το οποίο μπορείτε να τροποποιήσετε για να εκτελέσετε την συσταδοποίηση.

Επιπλέον όλα τα προγράμματα εκτυπώνουν διάφορα ενημερωτικά στοιχεία κατά την εκτέλεσή τους και στο τέλος δημιουργούν αρχεία εξόδου σύμφωνα με τις προδιαγραφές της εκφώνησης.

ΣΗΜΑΝΤΙΚΟ Στο αρχείο source/common.h περιέχονται τα #define MAX_DATA_SIZE και MAX_QUERY_SIZE τα οποία δηλώνουν πόσα από τα στοιχεία του dataset θα χρησιμοποιηθούν για τον αλγόριθμο και πόσα από τα στοιχεία ελέγχου θα χρησιμοποιηθούν ως queries. Κατά την ανάπτυξη της εργασίας οι αριθμοί αυτοί ήταν σχετικά μικροί (~10000 για data size, ~5 για query size). Στο παραδοτέο τα έχουμε θέσει σε 60000 (για να χρησιμοποιείται όλο το dataset) και 100. Σε περίπτωση που το query file σας περιέχει περισσότερα από 100 στοιχεία παρακαλούμε αλλάξτε τον αριθμό αυτό. Ειδικά ο αριθμός query size ήταν αναγκαίος για την ανάπτυξη του κώδικα καθώς χρησιμοποιήσαμε το test dataset του mnist με τα 10000 σημεία.