DOCUMENTATION 1ης ΕΡΓΑΣΙΑΣ - ΑΝΑΠΤΥΞΗ ΛΟΓΙΣΜΙΚΟΥ ΓΙΑ ΑΛΓΟΡΙΘΜΙΚΑ ΠΡΟΒΛΗΜΑΤΑ

ΜΕΛΗ ΟΜΑΔΑΣ:
ΔΑΜΙΑΝΑΚΗΣ ΔΑΜΙΑΝΟΣ 1115201800306
ΛΑΚΕΣ ΑΘΑΝΑΣΙΟΣ 1115201800090

Το project έχει υλοποιηθεί σε γλώσσα C++ με compiler g++

Το project είναι χωρισμένο σε τρεις φακελους (lsh,hypercube και cluster) με μια main συναρτηση το καθενα,τα καταλληλα εργαλεια/δομες και το makefile.

Για την οργανωση δεδομενων έχουμε χρησιμοποιήσει συνδεδεμενη λιστα και hash tables.Τα εργαλεια αυτα ειναι χωρισμενα σε ξεχωριστα αρχεια .cpp συνοδευόμενα το καθένα με header file.Όσον αφορά τον έλεγχο του input και των command lines έχουμε φτιάξει κατάλληλη συνάρτηση στο αρχειο others.c.

Σε αυτό το αρχείο συμπεριλαμβάνονται διαφορες συναρτησεις που χρησιμοποιουνται σε ολο το ευρος του project οπως ο υπολογισμος της ευκλειδειας αποστασης, υψωμένης δύναμης ,initialization του k κοντινοτερου γειτονα κ.α. Επίσης για την καλύτερη οργάνωση του project έχουμε φτιάξει ένα ξεχωριστο αρχειο .h,το includes.h στο οποιο περιλαμβανονται όλες οι κατάλληλες βιβλιοθηκες και έτσι αποφευγεται η επαναληψη τους στα αρχεια. Επιπλεον στο αρχειο parsingInit.cpp περιλμβανεται η συναρτηση parsInit ,η οποια χρησιμοποιειται για το διαβασμα των αρχειων εισόδου,το input file και το query file.

> ****Να τονίσω πως λόγω έλλειψης χρόνου δεν έχουμε ασχοληθεί καθόλου με το deallocation της μνήμηςκαι θα παρατηρήσετε πως υπάρχουν memory leaks. Δεν τα ξεχάσαμε ούτε τα θεωρήσαμε αμελητέα, απλώς δεν είχαμε χρόνο να ασχοληθούμε με αυτά. Ακόμη, το README ίσως το βρείτε λίγο πρόχειρο. Αυτό συμβαίνει διότι το γράφω 20 λεπτά πριν λήξει η προθεσμία.**

- **LSH**
  Τον LSH αλγόριθμο θα τον βρείτε στον φάκελο LSH επιλέξαμε να χωρίσουμε τους αλγορίθμους σε φακέλους αφενός διότι κάνουν διαφορετικά πράγματα αφετέρου για να μην υπάρχει πολλή πληροφορία σε έναν μόνο φάκελο. Αυτό που θα μπορούσαμε ίσως να κάνουμε καλύτερα είναι να φτιάξουμε έναν ακόμη φάκελο ο οποίος θα περιείχε όλα τα κοινά header files των αλγορίθμων όπως το others.c ωστέ να μην τα χρησιμοποιούμε σε κάθε φάκελο.

  Όσον αφορά τις δομές του LSH αλγορίθμου, το hash table ακολουθεί την κλασσική λογική μιας τέτοια δομής με την διαφορά να έγκειται στα attributes του. Συγκεκριμένα εκεί θα βρείτε έναν πίνακα vAndT δομές που ουσιαστικά αναπαριστούν τα v

διανύσματα με τις αντίστοιχες t τιμές. Ακόμη υπάρχει ένα struct lshCon που περιέχει κάποιες σταθερές για τον LSH.

Κάθε bucket του hash table είναι μια λίστα από δομές linkeLlistNode* ενώ ακόμη για κάθε πινακα κατακερματισμού έχουμε προσθέσει και το αντίστοιχο R vector σε μορφή vector<int>.

Όσον αφορά τις συναρτήσεις ενός hashTable αντικειμένου, εκτός των προφανή helper methods έχουμε τις findKNeighbors, findKNeighborsTrue, findKNeighborsR οι οποίες υλοποιούν τις διαδικασίες αναζήτησης κοντινότερων γειτώνων με KNN, Brute Force και Range Search αντίστοιχα.

Μια ακόμη δομή που χρειάζεται αποσαφήνιση είναι εκείνη της KNearest. Το συγκεκριμένο struct σκοπό έχει να αναπαραστήσει τον πίνακα με τους κοντινότερους γείτονες που επέστρεψε μια διαδικασία αναζήτησης (range search, KNN, brute force). Προκειμένου να προσθέσουμε ακόμη περισσότερη πληροφορία στην έννοια του KNearest, επιλέξαμε να κάνουμε μια ξεχωριστή δομή στην οποία αποθηκεύουμε τόσο τα points (με δείκτες) που αποτελούν τους κοντινούς γείτονες όσο, ακόμη, και τις αποστάσεις τους από το συγκεκριμένο query.

Για την εύρεση του χρόνου υπολογισμού αποθηκεύουμε και το χρόνο που πήρε η διαδικασία εύρεσης κοντινότερων γειτόνων να τερματίσει και έτσι έχουμε και έναν πίνακα από `chrono::duration<double, std::micro>` αντικείμενα.

Για να τρέξετε το πρόγραμμα:

- **make**
- **make run**

- **HYPERCUBE**

Η δομή του HyperCube, ίσως και η πιο απαιτητική της εργασίας, ακολουθεί προφανώς τις ίδιες δομές δεδομένων (hashTable, linkedList) με την πρώτη, ωστόσο, να διαφοροποιείται για χάρη του αλγορίθμου.

Συγκεκριμένα θα βρείτε σε αυτήν ο,τι περιέχει ένα hashTable για τον LSH αλγόριθμο με την προσθήκη όμως ενός vector με στοιχεία 0 και 1 που αναπαριστούν την κορυφή του υπερκύβου.

Πάλι υπάρχουν τα helper methods και φυσικά οι findKNeighbors(x) (x = “ “ || “R” || “True”) οι οποίες εκτελούν τις ίδιες διαδικασίες. Υπάρχει ακόμη η προσθήκη της decToBin η οποία ορίσθηκε στα πλαίσια του ορισμού του διανυσματος με 0 και 1 που καθορίζει το bucket μου.

Προκειμένου να διατηρούμε τις τιμές των h_i και f, έχουμε προσθέσει ένα map της STL και έτσι αφού πρώτα περάσουμε την h από την zeroOrOne συνάρτηση για να παράξουμε το f(h()) έπειτα κρατάμε το pair<int, int>(h,f) και το αποθηκεύουμε στο hToF map.

Για να τρέξετε το πρόγραμμα:

1) **make**
2) **make run**

- **CLUSTER**

Αυτά που έχουμε αλλάξει στα αρχεία του cluster είναι αρκετά προφανή. Αρχικά ορίσαμε ξεχωριστά αρχεία cluster.h και cluster.cpp στα οποία ορίζουμε και υλοποιούμε αντίστοιχα όλες εκείνες τις συναρτήσεις που έχουν να κάνουν με clustering (lloyd, lsh reverse, hypercube reverse καθώς και διάφορες άλλες).

Ακόμη δημιουργήσαμε μια κλασση centroid που αντιπροσωπεύει το κεντροειδές μας. Αυτό γίνεται για τον απλούστατο λόγο οτι ένα centroid δεν είναι απλά ένα σημείο στο χώρο αλλά περιέχει και άλλες πληροφορίες όπως πχ τα σημεία που περιλαμβάνει. Προκειμένου να διατηρούμε και αυτή την πληροφορία, δημιουργήσαμε τη δομή centroid.

Κάτι ακόμη που πρέπει να αναφερθεί είναι οτι μέσα στο φάκελο cluster θα βρείτε τόσο ένα hashtable.h όσο και ένα hashtablecube.h και τα αντίστοιχα αρχεία πηγαίου κώδικα. Αυτό συμβαίνει διότι το lsh και το hypercube δεν χρησιμοποιούν ακριβώς την ίδια μορφή hashtable όπως είπαμε και παραπάνω.

Κάθε φορά που θέλουμε να ορίσουμε ένα hashTable με σκοπό να το χρησιμοποιοήσουμε σε hypercube αλγοριθμο, κάνουμε:

hashTableCube hashT…

Ενώ όταν θέλουμε να κάνουμε το αντίστοιχο για LSH αλγόριθμο, κάνουμε:

hashTable hashT…

Προκειμένου να τρέξετε τις διάφορες παραλαγές των cluster αλγορίθμων πρέπει να πληκτρολογίσετε μία από τις 3 αυτές εντολές:

make

1) LLoyd
   make runclassic
2) Reverse LSH
   make runlsh
3) Reverse Hypercube
   make runhyper

Στους reverse αλγορίθμους έχω βάλει max επαναλήψεις 100 και break από το loop μόλις το distance του προηγούμενου cluster από το καινούριο είναι μικρότερο του 85 (η τιμή γενικά μετά από πειραματισμούς δουλεύει). Ωστόσο αυτή η συνθήκη πρέπει να γίνει αληθείς 3 φορές στη σειρά ωστέ να συμβεί αυτό (θέλεο λίγο ψάξιμο αλλά… χρόνος!).

Τέλος στο cluster.cpp θα βρείτε ακόμη την συνάρτηση που κάνει initialize τα centroids, αυτή που τα κάνει update, μια συνάρτηση για να εκτυπώνω στο output file αυτά που ζητούνται ενώ ακόμη υπάρχει και η insert rest of points που εισάγει όσοα στοιχεία δνε έχουν εισαχθεί στο cluster.

Έχει υλοποιηθεί και ο sillouette και για τα αρχεία που έχουν δοθεί οι τιμές του κυμαίνται από 0.3 σε 0.5.

Ευχαριστούμε.
Λακές Αθανάσιος.
Δαμιανάκης Δαμιανός.