Ανάπτυξη Λογισμικού για Αλγοριθμικά Προβλήματα
Project 2

Ονοματεπώνυμο:Χατζηδάκης Ραφαήλ
Αριθμός Μητρώου:1115201400248

Εισαγωγή

- Η εργασία αναπτύχθηκε σε περιβάλλον ***ubuntu 16.04.3***
- Χρησιμοποιήθηκε η **STL** της **C++ 11 (-std=c++11) .**
- Απο STL containers εχει χρησιμοποιθεί μόνο:
  1. Vector
  2. Unordered_set.
  3. Unordered_map.
- Για τους χρόνους χρησιμοποιήθηκε η βιβλιοθήκη **chrono.**
- Υπάρχει η δυνατότητα μεταγλώτισης με Makefile.Με την εντολή make δημιουργείται 1 εκτελέσιμο:
  1. ./cluster
- Η εκτέλεση γίνεται όπως τις οδηγίες της άσκησης :
  1. **./cluster –i <input file> –c <configuration file> –o <output file> -d <metric>**
- ‘Όλα τα ορίσματα πρέπει να δωθουν.
- Σε περίπτωση κάπιου λάθους με τα ορίσματα εμφανίζονται όλα τα λάθοι που εντοπίστηκαν και το πρόγραμμα τερματίζει.
- Στο αρχείο param.h υπάρχουν σταθερες και τύποι μεταβλητών που μπορουν να αλλάζουν ευκολα για τις ανάγκες της εργασίας .
- Χρησιμοποιήθηκε github με gitkraken (https://github.com/d1rafaelol/project_2.git).

Ανάπτυξη Λογισμικού για Αλγοριθμικά Προβλήματα
Project 2
Ονοματεπώνυμο:Χατζηδάκης Ραφαήλ
Αριθμός Μητρώου:1115201400248

Αρχεία

Μερικές απο τις συναρτήσεις είναι ίδιες σε αρκετά αρχεία .Αυτές θα περιγράφονται μόνο μια φορά.

check_info.h

- **void print_err(int)**

  Εκτυπώνει μήνυμα λάθους ανάλογα με τον αριθμό .

- **int check_info_cluster(int ,char**,std::ifstream *,std::ifstream *,std::ofstream *,std::string *)**

  Ελέγχει τα ορίσματα ,ανοίγει τα αρχεία και δινει τιμές σε μεταβλητές του προγράμματος .

Eucl_Cos.h

- **Euclidean(int ,int);**

  Δίνει τυχαίες τιμες στα coordinates του διανύσματος vector_V και στο t.

- **int Euclidean ::h_func(Vector &)**

  Υπολογίζει το h(p) από την σελίδα 16 στο nnLSH.pdf.

- **Cosine(int)**

  Δίνει τυχαίες τιμες στα coordinates του διανύσματος vector_R.

- **int Cosine ::h_func(Vector &)**

  Υπολογίζει το h(p) από την σελίδα 22 στο nnLSH.pdf.

Ανάπτυξη Λογισμικού για Αλγοριθμικά Προβλήματα
Project 2

Ονοματεπώνυμο:Χατζηδάκης Ραφαήλ
Αριθμός Μητρώου:1115201400248

General_functions.h

- **int mod (long long int , unsigned long long int)**

  Δίνει το σωστό modulo ,όπως έχει αναφερθεί εδώ(https://eclass.uoa.gr/modules/forum/viewtopic.php?course=DI352&topic=20690&forum=30954&fbclid=IwAR1U1ah0d6sLyAuB_a-Wnwplp7b4Wkv8Yal1Kf-lyKp7EAJ2QI11u49rlf0).

- **void read_info(std::ifstream *,std::vector<Vector*> *)**

  Διαβάζει το αρχείο csv με τα διανύσματα και τα προσθέτει ολα σε ένα vector.

- **int check_parameters(std::ifstream * ,int * ,int * ,int *,int*,int*)**

  Τσεκάρει όλες τις παραμέτρους,και τους δίνει τιμές.(όλες έχουν και μια default τιμή σε περίπτωση που δεν δωθει καποια, εκτός από τον αριθμό των clusters)

- **int check_dimensions(std::vector<Vector*> &)**

  Τσεκάρει αν όλα τα διανύσματα έχουν τον ίδιο αριθμό διαστάσεων.

- **int check_cluster_num(std::vector<Vector*> & ,int)**

  Ελέγχει τον αριθμό των clusters να μην είναι πάνω από ένα οριο.Το οποίο μπορεί να αλλάξει από την μεταβλητή CLUSTERS_DIV στο param.h.

Ανάπτυξη Λογισμικού για Αλγοριθμικά Προβλήματα
Project 2

Ονοματεπώνυμο:Χατζηδάκης Ραφαήλ
Αριθμός Μητρώου:1115201400248

Hash.h

- **int_Euclidean_Hash::hashFunction(Vector *)**

  Υπολογίζει το φ(p) από την σελίδα 19 στο nnLSH.pdf.

- **int Euclidean_Hash::g_fun(Vector *,Vector *)**

  Ελέγχει αν δυο διανύσματα θα έπρεπε να ειναι στο ίδιο bucket με την hashfunction g(p) στην σελίδα 19 στο nnLSH.pdf.

  Υπολογίζει την h() για καθε διάνυσμα και κοιταει για ισότητες αν βρει τόσες όσες και οι k διαφορετικές h() τότε θα έμπαιναν στο ίδιο bucket οπότε επιστρέφει 1, διαφορετικά επιστρέφει 0

- **int Cosine_Hash::hashFunction(Vector *)**

  Υπολογίζει το φ(p) από την σελίδα 23 στο nnLSH.pdf. Η g(x) είναι ένα int το οποίο είναι ίσο με το δυαδικό αριθμό πχ(h1(x)=1,h2(x)=0,h3(x)=1-- 101=5)

- **int Cube_Hash::hashFunction(Vector *)**

  Βρίσκει την τιμή για κάθε h().Αν αυτή η τιμη υπαρχει στο map τοτε περνουμε την αντίστοιχη τιμή(0 1) απο εκεί αν οχι την υπολογίζει και την προσθέτει στο map.Αυτές οι τιμές δημιουργούν εναν αριθμό (απο 0 έως 2^k-1) ο οποίος ειναι το bucket που πρέπει το διάνυσμα να μπει.

- **void insert_Vector(Vector*)**

  Βαζει ενα διάνυσμα στο hash table στο κατάλληλο bucker χρησιμοποιώντας την hashFunction.

- **std::vector<Vector*> * get_similar_Vectors(Vector*)**

  Επιστρέφει το bucket που θα έμπαινε το διάνυσμα.

- **int get_bucket_num()**

  Επιστρέφει τον αριθμό των buckets.

- **std::vector<Vector*> * get_bucket(int)**

  Επιστρέφει bucket ανάλογα με τον αριθμό.

Ανάπτυξη Λογισμικού για Αλγοριθμικά Προβλήματα
Project 2

Ονοματεπώνυμο:Χατζηδάκης Ραφαήλ
Αριθμός Μητρώου:1115201400248

- ~~Neighbour_Namespaces.h~~ **Distance.h (Αλλαγη απο 1ο project).**

  Λειτουργεί με κλάση αντί για namespace

  - **void Range_N(std::vector<std::string > &,std::vector<Vector*> &,Vector * ,double)**

    Βαζει σε vector όλα τα διανύσματα που εχουν απόσταση λιγότερο από R το οποίο δίνεται στο αρχείο με τα querys.

  - **void Range_bet_N(std::vector<Vector*> &,std::vector<Vector*> & ,Vector * ,long double,long double)**

    Βαζει σε vector όλα τα διανύσματα που εχουν απόσταση ανάμεσα στο σε δύο ακτίνες

  - **void Nearest_N(std::vector< std::pair <std::string,double> > &,std::vector<Vector*> & ,Vector *)**

    Βαζει σε vector όλα το δίανυσμα με την μικρότερη απόσταση από το διάνυσμα του query

Η απόσταση υπολογίζεται με διαφορετικό τρόπο ανάλογα με το namespace .

double cosi(Vector &,Vector &)

σελίδα 21 στο nnLSH.pdf. Για τον υπολογισμό της απόστασης cosine 1 – cosi

double dist(Vector & x,Vector & y)

σελίδα 14 στο nnLSH.pdf. Για k=2

Ανάπτυξη Λογισμικού για Αλγοριθμικά Προβλήματα
Project 2

Ονοματεπώνυμο:Χατζηδάκης Ραφαήλ
Αριθμός Μητρώου:1115201400248

Vector.h

- **Vector(std::string)**

  Δίνει τιμή στο id του διανύσματος.

- **void add_coordinate(coordinate)**

  Προσθέτει ένα καινούργιο coordinate στο vector του διανύσματος.

- **std::string get_identity(void)**

  Επιστρέφει το id του διανύσματος.

- **std::vector<coordinate> *get_coordinates(void)**

  Επιστρέφει τις συντεταγμένες του διανύσματος.

Cluster.h

- **void add_Vector(Vector *)**
  Βάζει ενα Vector στο cluster.

- **int update_Centroid(Vector *)**
  Αλλάζει το centroid με τον αλγόριθμο k-means.

- **std::vector<Vector *> * get_cluster_vectors()**
  Επιστρέφει ένα vector με τα διανύσματα που είναι μέσα στο cluster εκτός του centroid.

Cluster_Group.h

- **void k_unique_rand_init(std::vector<Vector*> &)**
  Παίρνει τυχαιά k clusters από τα διαθέσιμα vectors.

- **void k_means_plus_plus_init(std::vector<Vector*> &)**
  Παίρνει k clusters από τα διαθέσιμα vectors σύμφωνα με το k-means++.

Ανάπτυξη Λογισμικού για Αλγοριθμικά Προβλήματα
Project 2

Ονοματεπώνυμο:Χατζηδάκης Ραφαήλ
Αριθμός Μητρώου:1115201400248

- **void k_means_average_init(std::vector<Vector*> &)**

  Παιρνει τυχαία ενα cluster και μετά υπολογίζει την μέση τιμή των αποστάσεων ολών των μέχρι τώρα επιλεγμένων clusters και παίρνει με πιυανότητα όπως και στο k-means_plus_plus αυτο με την μεγαλύτερη μέση τιμή για επόμενο centroid.

- **void Lloyd_assignment(std::vector<Vector*> &)**

  Βάζει όλα τα διανύσματα σε κάποιο cluster σύμφωνα με τον αλγόριθμο Lloyd's .

- **void Range_search_assignment_LSH(std::vector<Vector*> &, int ,int);**
- **void Range_search_assignment_Hypercube(std::vector<Vector*> &,int,int,int);**

  Range search κάνουν το ίδιο αλλά με διαφορετικό hash table και με την διαφορα ότι το hypercube κοιτάει και τα probes.Λειτουργουν σύμφνα με τον αλγόριθμο range search.Στην αρχή ψάχνουμε όλα τα vectors που πρεπει να ελεχθούν απο range search από κάθε hash table και κοιτάμε για ίδια διανύσματα στην αρχή ,μετά ο αλγόριθμος συνεχίζεται κανονικά.

- **int k_means_update()**

  Δημιουργεί k καινούργια clusters που δεν ανήκουν στο αρχείο εισόδου σύμφωνα με το αλγόριθμο k-means.

- **int PAM_improvement_update()**

  Αλλάζει τα clusters με τα medoids .

Το αρχειο cluster.conf παιρνει 2 παραπάνω παραμετρους το 1) Number of probes και 2) number_of_vectors. Τα οποία παιρνουν default τιμες απο το param.h αν δεν δωθουν στο αρχειο.