ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΑΝΑΠΤΥΞΗ ΛΟΓΙΣΜΙΚΟΥ ΓΙΑ ΑΛΓΟΡΙΘΜΙΚΑ ΠΡΟΒΛΗΜΑΤΑ

1η Εργασία: Αναζήτηση και συσταδοποίηση διανυσμάτων στη C/C++

ΒΕΚΡΑΚΗ ΑΘΗΝΑ 1115201400020

ΠΟΛΥΧΡΟΝΑΚΗΣ ΣΑΒΒΑΣ 1115201200150

ΠΕΡΙΕΧΟΜΕΝΑ

Αρχεία με βοηθητικές συναρτήσεις και υπολογισμούς:

help_functions.cpp	help_functions.h
calculations.cpp	calculations.h

Για τον αλγόριθμο LSH:

lsh.cpp	
calculations_lsh.cpp	calculations_lsh.h

Για τον αλγόριθμο Hypercube:

cube.cpp	
calculations_cube.cpp	calculations_cube.h

Για το Clustering:

cluster.cpp	
calculations_cluster.cpp	calculations_cluster.h

καθώς και τα αρχεία results_lsh.txt, results_cube.txt και results_cluster.txt για τα output των αλγορίθμων, το cluster.conf για τις ρυθμίσεις παραμέτρων για το clustering, το dataset train-images-idx3-ubyte, το query file t10k-images-idx3-ubyte και Makefile.

ΓΕΝΙΚΑ

LSH: Το πρόγραμμα αποθηκεύει τα δεδομένα του dataset σε L HashTables χρησιμοποιώντας ως hash-function τη συνάρτηση g, η οποία προκύπτει απο concatenation k h συναρτήσεων, στη συνέχεια διαβάζει το query file και ψάχνει στα HashTables για να βρει τον προσεγγιστικά κοντινότερο γείτονα της κάθε εικόνας, τους N προσεγγιστικά κοντινότερους γείτονες, τους γείτονες εντος ακτίνας R και τους πραγματικούς κοντινότερους γείτονες και εκτυπώνει τα αποτελέσματα σε αρχείο κειμένου.

HyperCube: Το πρόγραμμα αποθηκεύει τα δεδομένα του dataset σε HashTable χρησιμοποιώντας ως hash-function τη συνάρτηση p, η οποία μετατρέπει τις συναρτήσεις h σε ακολουθίες απο 0 και 1 και αντιπροσωπεύουν τις κορυφές του υπερκύβου. Στη συνέχεια διαβάζει το query file και ψάχνει στο HashTable μέχρι M εικόνες και μέχρι probes διαφορετικές κορυφές του υπερκύβου για να βρει τον προσεγγιστικά κοντινότερο γείτονα της κάθε εικόνας, τους N προσεγγιστικά κοντινότερους γείτονες, τους γείτονες εντος ακτίνας R και τους πραγματικούς κοντινότερους γείτονες και εκτυπώνει τα αποτελέσματα σε αρχείο κειμένου.

Clustering: Το πρόγραμμα αρχικοποιεί τα κεντροειδή με την τεχνική k-Means++ και τα ενημερώνει με τον υπολογισμό του διάμεσου (median) διανύσματος. Στη συνέχεια, ανατίθονται στα clusters ανάλογα με τη μέθοδο που έχει επιλέξει ο χρήστης (Classic/LSH/Hypercube). Στην πρώτη περίπτωση χρησιμοποιείται ο ακριβής αλγόριθμος Lloyd’s, στη δεύτερη η αντίστροφη ανάθεση (reverse) μέσω Range search με LSH και στην τρίτη η αντίστροφη ανάθεση (reverse) μέσω Range search με τυχαία προβολή στο Hypercube.

ΣΥΝΑΡΤΗΣΕΙΣ

- *help_functions.cpp*

void read_data: Διαβάζει το δοσμένο dataset και ενημερώνει magic_number, number_of_images, n_rows, n_cols και αποθηκεύει σε vector τις συντεταγμένες κάθε εικόνας

void read_inputLSH: Διαβάζει την είσοδο του χρήστη απο τη γραμμή εντολών για τον LSH αλγόριθμο και σε περίπτωση που ο χρήστης δε δώσει κάποια παράμετρο χρησιμοποιεί default τιμές

void read_inputCube: Διαβάζει την είσοδο του χρήστη απο τη γραμμή εντολών για τον αλγόριθμο του Hypercube και σε περίπτωση που ο χρήστης δε δώσει κάποια παράμετρο χρησιμοποιεί default τιμές

void read_inputCluster: Διαβάζει την είσοδο του χρήστη απο τη γραμμή εντολών για το Clustering και σε περίπτωση που ο χρήστης δε δώσει κάποια παράμετρο χρησιμοποιεί default τιμές

void read_confFile: Διαβάζει το αρχείο cluster.conf για το Clustering και ενημερώνει τις αντίστοιχες μεταβλητές

void quicksort: Αλγόριθμος ταξινόμησης quicksort που χρησιμοποιείται για την εύρεση του median διανύσματος στην ενημέρωση των κεντροειδών του Clustering

int partition: Χρησιμοποιείται για να βρεθεί το pivot στον Quicksort

- *calculations.cpp*

int modular_pow: Βρίσκει το modular exponentiation

vector<int> get_s: Επιστρέφει vector απο τυχαία ομοιόμορφα κατανεμημένα s στο διάστημα [0,w)

vector<int> calculate_a: Υπολογίζει τα a δεδομένων των s και w

int calculate_h: Υπολογίζει τις k h functions που θα χρησιμοποιηθούν για την εύρεση της g

unsigned int manhattan_dist: Υπολογίζει τη manhattan distance μεταξύ δύο σημείων

vector<distanceNode> actual_nearest_neighbor: Βρίσκει τον ακριβή κοντινότερο γείτονα με τη μέθοδο Brute Force

float get_x: Επιλέγει τυχαία έναν float σε ένα διάστημα για να βρει το index του επόμενου κεντροειδούς στον αλγόριθμο k-means++

- *calculations_lsh.cpp*

unsigned int calculate_g: Κάνει concatenation των k h functions

void create_hashtables_LSH: Αρχικοποιεί τα L HashTables μεγέθους N/16, υπολογίζει τα τυχαία s για κάθε συνάρτηση h και τα αντίστοιχα a για να βρει τις συναρτήσεις h, τις οποίες στη συνέχεια κάνει concatenate για να βρει τη συνάρτηση g και να κάνει εισαγωγή βάσει αυτής στο κάθε HashTable

vector<distanceNode> approximate_nearest_neighbor: Υπολογίζει κι επιστρέφει σε vector τους N προσεγγιστικά πλησιέστερους γείτονες του δωσμένου query ψάχνοντας στο bucket pos των L HashTables

vector<distanceNode> approximate_range_search: Υπολογίζει κι επιστρέφει σε vector τους προσεγγιστικά πλησιέστερους γείτονες του δωσμένου query σε ακτίνα R ψάχνοντας στο bucket pos των L HashTables

- *calculations_cube.cpp*

unsigned int calculate_p: Κάνει concatenation των f functions

vector<int> hamming_dist: Υπολογίζει κορυφές του υπερκύβου με hamming distance 1 και επιστρέφει vector με probes τέτοιες κορυφές

int get_f: Αντιστοιχίζει κάθε συνάρτηση h σε 0 ή 1 με τυχαίο τρόπο

void create_hashtable_cube: Αρχικοποιεί το HashTable μεγέθους 2^κ, υπολογίζει τα τυχαία s για κάθε συνάρτηση h και τα αντίστοιχα a για να βρει τις συναρτήσεις h, τις οποίες στη συνέχεια αντιστοιχίζει σε 0 ή 1 και με concatenation τους βρίσκει την p που αντιστοιχεί σε κορυφή του υπερκύβου, η οποία είναι index για την εισαγωγή στο HashTable

vector<distanceNode> approximate_nearest_neighbor_cube: Υπολογίζει κι επιστρέφει σε vector τους N προσεγγιστικά πλησιέστερους γείτονες του δωσμένου query ψάχνοντας στο bucket p του HashTable. Ελέγχει μέχρι M διαφορετικά σημεία και μέχρι probes διαφορετικές κορυφές του υπερκύβου

vector<distanceNode> approximate_range_search_cube: Υπολογίζει κι επιστρέφει σε vector τους προσεγγιστικά πλησιέστερους γείτονες του δωσμένου query σε ακτίνα R ψάχνοντας στο bucket p του HashTable. Ελέγχει μέχρι M διαφορετικά σημεία και μέχρι probes διαφορετικές κορυφές του υπερκύβου

- *calculations_cluster.cpp*

void k_means_init: Βρίσκει τυχαία το index του πρώτου κεντροειδούς. Στη συνέχεια για κάθε μη κεντροειδές βρίσκει την ελάχιστη απόσταση απο τα υπάρχοντα κεντροειδή, φτιάχνει εναν πίνακα με αυτές, βρίσκει τυχαίο αριθμό x και βρίσκει το index του επόμενου κεντροειδούς

void lloyds_assignment: Αναθέτει τα σημεία σε clusters υπολογίζοντας την ελάχιστη απόσταση απο τα κεντροειδή

void update_centroids_median: Ταξινομεί τις συντεταγμένες της κάθε διάστασης των εικόνων για να βρει το median και να φτιάξει με αυτά νέο σημείο το οποίο θα είναι το νέο κεντροειδές

vector<distanceNode> approximate_range_search_clusterLSH: Βρίσκει τους προσεγγιστικά κοντινότερους γείτονες του κεντροειδούς σε ακτίνα R ,τους μαρκάρει και τους αναθέτει στο αντίστοιχο cluster, ενώ αν είναι ήδη μαρκαρισμένοι τότε υπολογίζει ποιο είναι το κοντινότερο κεντροειδές απο τα δυο και κάνει την κατάλληλη ανάθεση

vector<distanceNode> approximate_range_search_clusterCube: Βρίσκει τους προσεγγιστικά κοντινότερους γείτονες του κεντροειδούς σε ακτίνα R ,τους μαρκάρει και τους αναθέτει στο αντίστοιχο cluster, ενώ αν είναι ήδη μαρκαρισμένοι τότε υπολογίζει ποιο είναι το κοντινότερο κεντροειδές απο τα δυο και κάνει την κατάλληλη ανάθεση. Ελέγχει μέχρι M διαφορετικά σημεία και μέχρι probes διαφορετικές κορυφές του υπερκύβου

void silhouette: Βρίσκει τη μέση απόσταση απο τα σημεία του ίδιου cluster καθώς και τη μέση απόσταση απο τα σημεία του αμέσως επόμενου κοντινότερου cluster και εφαρμόζοντας τον τύπο κρίνει την αποτελεσματικότητα του clustering και εκτυπώνει τα αποτελέσματα

ΟΔΗΓΙΕΣ ΜΕΤΑΓΛΩΤΤΙΣΗΣ

Τα προγράμματα μεταγλωττίζονται με την εντολή make

ΟΔΗΓΙΕΣ ΧΡΗΣΗΣ

LSH: ./lsh –d <input file> –q <query file> –k <int> -L <int> -o <output file> -N <number of nearest> -R <radius> , όπου:

–d : η επόμενη παράμετρος είναι το dataset
–q : η επόμενη παράμετρος είναι το αρχείο με τα queries
–k : η επόμενη παράμετρος είναι το πλήθος των συναρτήσεων h
-L : η επόμενη παράμετρος είναι ο αριθμός των HashTables
-o : η επόμενη παράμετρος είναι το αρχείο εξόδου με τα αποτελέσματα του αλγορίθμου
-N : η επόμενη παράμετρος είναι ο αριθμός των πλησιέστερων γειτόνων
-R : η επόμενη παράμετρος είναι η ακτίνα αναζήτησης

Αν ο χρήστης δε δώσει παραμέτρους, τότε χρησιμοποιούνται default τιμές.

Hypercube: ./cube –d <input file> –q <query file> –k <int> -M <int> -probes <int> -o <output file> -N <number of nearest> -R <radius> , όπου:

–d : η επόμενη παράμετρος είναι το dataset
–q : η επόμενη παράμετρος είναι το αρχείο με τα queries
–k : η επόμενη παράμετρος είναι η διάσταση d’ στην οποία προβάλλονται τα σημεία
-M : η επόμενη παράμετρος είναι το μέγιστο επιτρεπόμενο πλήθος υποψήφιων σημείων που θα ελεγχθούν
-probes : η επόμενη παράμετρος είναι το μέγιστο επιτρεπόμενο πλήθος κορυφών του υπερκύβου που θα ελεγχθούν
-o : η επόμενη παράμετρος είναι το αρχείο εξόδου με τα αποτελέσματα του αλγορίθμου
-N : η επόμενη παράμετρος είναι ο αριθμός των πλησιέστερων γειτόνων
-R : η επόμενη παράμετρος είναι η ακτίνα αναζήτησης

Αν ο χρήστης δε δώσει παραμέτρους, τότε χρησιμοποιούνται default τιμές.

Clustering: ./cluster –i <input file> –c <configuration file> -o <output file> -m <method: Classic OR LSH or Hypercube> , όπου:

–i : η επόμενη παράμετρος είναι το dataset
–c : η επόμενη παράμετρος είναι το αρχείο ρύθμισης παραμέτρων
-o : η επόμενη παράμετρος είναι το αρχείο εξόδου με τα αποτελέσματα του αλγορίθμου
-m : η επόμενη παράμετρος είναι η μέθοδος που θα χρησιμοποιηθεί

Αν ο χρήστης δε δώσει παραμέτρους, τότε χρησιμοποιούνται default τιμές.

ΔΟΚΙΜΕΣ-ΠΑΡΑΔΟΧΕΣ

Δοκιμάσαμε διαφορετικές τιμές στο w για τον LSH και τον Hypercube και συγκεκριμένα για w = 400 παρατηρήθηκε μικρή μείωση του χρόνου εκτέλεσης του αλγορίθμου, αλλά ήταν λιγότερο αποτελεσματικός σε σχέση με την εκτέλεση του με w = 40000.

Η τιμή του m για τον υπολογισμό του modular exponentiation έχει γίνει define με την τιμή 107, η οποία βρίσκεται μεταξύ του $max_i a_i$ και του M/2.

Η μεταβλητή NforTable που γίνεται define στην αρχή των προγραμμάτων είναι για το μέγεθος των HashTables στον LSH και δοκιμάστηκε με τιμές 8 και 16.