Κ23γ: Ανάπτυξη Λογισμικού για Αλγοριθμικά Προβλήματα

2η Προγραμματιστική Εργασία - Γραφοθεωρητική αναζήτηση πλησιέστερων γειτόνων στη C++

Μαρία Σιακαβέλλα - 1115201400181

Τσάμης Αντώνιος - 1115201300183

Περιγραφή του προγράμματος

Το παρόν README αποτελεί συνέχεια το προηγούμενο (της 1ης εργασίας). Περιέχει επιπλέον την αναφορά μας για τις επιδόσεις των αλγορίθμων (ερώτημα Γ).

Github link: mariasiak/project_2023 (github.com)

Κατάλογος των αρχείων κώδικα / επικεφαλίδων και περιγραφή τους

Στα αρχεία της 1ης εργασίας προστέθηκαν τα παρακάτω:

gnns.cpp & gnns.h
Περιέχει την υλοποίηση του gnns με μεθόδους για την αρχικοποίηση της δομής και την εκτέλεση αναζήτησης κοντινών γειτόνων.

mrng.cpp & mrng.h
Περιέχει την υλοποίηση του mrng με μεθόδους για την αρχικοποίηση της δομής και την εκτέλεση αναζήτησης κοντινών γειτόνων.

graph_search_main.cpp
Η main για την δημιουργία του εκτελέσιμου graph_search για τους αλγορίθμους mrng και gnns.

Οδηγίες μεταγλώττισης του προγράμματος

Η μεταγλώττιση του προγράμματος γίνεται με το εργαλείο make. Παρακάτω παραθέτουμε λίστα των εντολών και των λειτουργιών τους.

all: κάνει compile όλα τα εκτελέσιμα
lsh: compile για lsh
lsh_run: εκτέλεση lsh με συγκεκριμένα ορίσματα
cube: compile για hypercube

cube_run: εκτέλεση hypercube με συγκεκριμένα ορίσματα
cluster: compile για clustering
cluster_run: εκτέλεση clustering με συγκεκριμένα ορίσματα
graph_search: compile για graph search
gnns_run: εκτέλεση clustering με συγκεκριμένα ορίσματα
mrng_run: εκτέλεση clustering με συγκεκριμένα ορίσματα
clear: αφαίρεση εκτελέσιμων και αρχείων εξόδου (τελειώνουν σε .out)

Οδηγίες χρήσης του προγράμματος

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

./graph_search -d <input_file> -q <query_file> -k <int> -E <int> -R <int> -N <int> -l <int, only for search on graph> -m <1 for GNNS, 2 for MRNG> -o <output_file>

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.

Σύγκριση επιδόσεων αλγορίθμων

LSH			
(L,k) =	**5, 5**	**5, 25**	**5, 100**
tAA (μs)	4625	5105	6857
MAF	2,4553	2,6953	2,6960
(L,k) =	**25, 5**	**25, 25**	**25, 100**
tAA (μs)	22621	25072	34125
MAF	2,3274	1,7032	2,2261
(L,k) =	**100, 5**	**100, 25**	**100, 100**
tAA (μs)	92302	100901	135906
MAF	1,8182	1,5325	1,4510

Hypercube		
(k,probes) =	**5, 0**	**5, 1**
tAA (μs)	27502	165692
MAF	1,6581	1,3118
(k,probes) =	**10, 0**	**10, 1**
tAA (μs)	1027	10081
MAF	4,8566	1,7658
(k,probes) =	**15, 2**	**15, 4**
tAA (μs)	5302	60208
MAF	2,2261	1,6188

GNNS	
tAA (μs)	9366
MAF	2,9697

MRNG	
tAA (μs)	4419
MAF	2,9385

tAverageTrue
873155

Πιο πάνω παρουσιάζουμε στατιστικά που συλλέξαμε από εκτελέσεις των αλγορίθμων για διάφορες παραμέτρους. Για τον LSH δοκιμάσαμε διαφορετικές τιμές του L (αριθμός πινάκων κατακερματισμού) και k (αριθμός συναρτήσεων h ανά συνάρτηση g). Για τον hypercube δοκιμάσαμε διαφορετικές τιμές k (διαστάσεις υπερκύβου) και probes (αριθμός γειτονικών κουβάδων κατά την αναζήτηση κοντινότερου γείτονα). Για τους GNNS και MRNG χρησιμοποιήσαμε τις default τιμές της εκφώνησης. Στις δοκιμές μας χρησιμοποιήσαμε και τα 60000 train σημεία για την κατασκευή των δομών και τα 100 πρώτα test σημεία για queries. Όλοι οι χρόνοι είναι σε microseconds. Με tAA σημειώνουμε τον χρόνο tAverageApproximate. Παραθέτουμε για σύγκριση και τον χρόνο tAverageTrue που ήταν σε όλες τις εκτελέσεις περίπου ίδιος (περίπου 870 milliseconds ή 0,87 δευτερόλεπτα). Παρακάτω παραθέτουμε τις παρατηρήσεις και τα συμπεράσματά μας από τις εκτελέσεις των αλγορίθμων.

LSH

Για τον LSH παρατηρούμε ότι τα αποτελέσματα γίνονται καλύτερα όσο αυξάνεται το L αλλά γραμμικά αυξάνεται και ο χρόνος απάντησης (αν διπλασιάσουμε το L διπλασιάζεται και ο χρόνος απάντησης). Αυτό είναι αναμενόμενο διότι αυξάνονται οι πίνακες κατακερματισμού στους οποίους εκτείνεται η αναζήτηση. Η τιμή του k φαίνεται να επηρεάζει τον χρόνο των ερωτημάτων ελαφρά και να συνεισφέρει καλύτερα στην εύρεση καλύτερου γείτονα όταν είναι ίσο με το L.

Hypercube

Οι παράμετροι k και probes, και συγκεκριμένα ο συνδιασμός τους, επηρεάζουν σημαντικά τον χρόνο εκτέλεσης και την ποιότητα του αποτελέσματος. Μικρά k δημιουργούν λίγους και μεγάλους κουβάδες με συνέπεια η αύξησης της παραμέτρου probes να αυξάνει σημαντικά τον χρόνο αναζήτησης και η διαδικασία να προσεγγίζει την εξαντλητική αναζήτηση. Καθώς μεγαλώνει το k η παράμετρος probes μπορεί να πάρει μεγαλύτερες τιμές χωρίς να γίνεται απαγορευτικός ο χρόνος αναζήτησης. Μάλιστα για μεγαλύτερα k είναι σημαντικό να επιλεχθεί και κατάλληλη τιμή για την παράμετρο probes γιατί οι κουβάδες έχουν όλο και λιγότερα στοιχεία και αν δεν ελεγχθούν αρκετοί δεν θα λάβουμε καλή απάντηση στο query.

Σύγκριση Αλγορίθμων

Οι GNNS και MRNG δεν δίνουν γρήγορα αποτελέσματα αλλά η μετρική MAF είναι μεγάλη σε σχέση με τους LSH και Hypercube. Στις εκτελέσεις που κάναμε μετρούσαμε και την μέση τιμή του κλάσματος distance_approximate/distance_true και ενδεχομένως η μεγάλη τιμή του MAF να οφείλεται σε κάποιες κακές περιπτώσεις queries. Από όλες τις εκτελέσεις των αλγορίθμων αυτό που φαίνεται είναι ότι, σχετικά με την ποιότητα του αποτελέσματος για τον χρόνο που χρειάζεται, καλύτερα συμπεριφέρονται ο LSH και ο Hypercube για συντηρητικές τιμές των παραμέτρων (L,k=25,25 για τον LSH και k,probes=10,1 για τον Hypercube).