Project Part 1,2,3

ΑΝΔΡΕΑΣ ΟΙΚΟΝΟΜΑΚΗΣ 115200800272
ΔΗΜΗΤΗΣ ΣΑΝΤΟΡΙΝΑΙΟΣ 115200800282

Το προγραμμα κάνει compile με την εντολή make και του δίνεται ως όρισμα ποιο part να εκτελέσει.
Πχ >make
gcc -o SD-2014-All_Parts SD-2014-All_Parts.c GraphLib.c graph_functions.c hfunctions.c Statistics.c Query1.c Query2.c Query3.c Query4.c Tread_Utilities.c Forum_graph.c Community_Graph_Utilities.c Clique_Utilities.c GN_Algorithm.c Parts.c CHECKS.c -pthread
./SD-2014-All_Parts 1 για το 1ο part
./SD-2014-All_Parts 1 για το 2ο part
./SD-2014-All_Parts 1 για το 3ο part

Σημειωση: Δεν είναι δυνατόν να σταλθούν και τα dataset των part 1 /2 λόγω μεγέθους οπότε θα πρεπει να τοποθετηθουν κατα την εξέταση στους αντιστοιχους φακέλους για το κάθε part : dataset2,dataset3

Αρχικά θα εξηγηθούν κάποια πράγματα τα οποία ήταν να υλοποιηθούν στα προηγούμενα parts και υλοποιήθηκαν μετά.

1) ReachNodeN/Resultset / next

```
struct resultSet {
	struct VQueue *Visited_Queue; //ουρά επισκεψης
	struct Cell *Visited_hash; //hash table που κρατα το id των κομβων που εχουμε επισκεφθεί
}resultSet;
```

Η ReachNodeN δημιουργει την δομή Resultset , αρχικοποιει τα Visited_Queue και Visited_hash επισκέπτεται όλους τους γέιτονες του ζητούμενου κόμβου και τους βαζει στην ουρά και στο hash table που προαναφέρθηκαν και τους δίνει μια αποσταση dist + 1 οπου το dist ειναι μέλος της δομής Node και εχει αρχικοποιηθεί με 0 για τον κομβο που ζητησαμε.

Η next εξάγει απο την ουρά τον τρεχον κομβο και επισκεπεται ολους τους γειτονες του που δεν εχουμε ηδη επισκεφθεί και τους βαζει στην ουρα και το hash table και στην συνέχεια επιστεφει στην δομή pair το ID και το distance. Η διαδικασία σταματά όταν η ουρά δεν εχει άλλους κομβους

2) Bidirectional Search για την ReachNode1

Βρισκεται στο αρχείο graph_functions.c στη γραμμή 559 και υλοποιεί την διαδικασία με 2 hash_table , ενα για τον κομβο εναρξης και ενα για τον κομβο τελους εκτελοντας διαδοχικα bfs απο την αρχή και μετά απο το τελος μεχρι να συναντηθουν σε κάποιο κομβο και επειτα να αθροίσει τις αποστασεις και να βρει το τελικο shortest_path

3)Query 4

Η ιδέα γενικότερα είναι ότι κάθε forum περιεχεται σε ενα hash table στον γράφο και εκράζεται ως ενας κομβος Fnode:

```
struct FNode{
	int id;   id του Forum
	char name[MAX_STRING_LENGTH]; ονομα του forum
```

struct GNodeList *Forum_Members; λιστα με μέλη του forum
struct Cell *likes_to_from; hash table με likes σε κάποιον απο καποιον
struct Cell *replies_to_from; hash table με replies σε κάποιον απο καποιον
}
Τα δυο αυτά hash table περίεχουν κομβους τυπου TNode των οποιων τα id αντιστοιχνουν στους κομβους του γραφου που εχουν δεχθει like / reply απο κάποιον και στη συνεχεια αυτοι οι κομβοι Tnode περιεχουν εναν δεικτη σε hash table που και αυτα με τη σειρά τους εχουν κομβους Tnode των οποιων τα ιd αντιστοιχουν σε κομβους γράφου οι οποιοι εχουν κανει like / reply και εχουν ενα δεικτη στις δομες like_counter / reply_counter αντιστοιχα και έτσι εχουμε την πληροφορία σε ποιον εχει γινει like / reply απο ποιον σε κάθε forum.

Για την ευρεση του trust χρησιμοποειται η συναρτηση estimateTrust η οποια για την περιήγηση στον γράφο χρησιμοποιει την Reachnode12 η οποια βρισκει τα συντομότερα μονοπάτια σε κατευθυνομενο γραφο μιας και δεν υπαρχει εξασφάληση οτι η ακμή είναι διπλή.

Περιγραφή Part 3

→ Χρήση των Threads

Για τη χρήση νημάτων έχουμε επιλέξει να υλοποιησουμε ένα threadpooll και οι υλοποιησεις που χρησιμοποιουν τα threads υπάρχουν στο αρχει Tread_Utilities. Ακολουθεί μια συντομη περιγραφή για το πως λειτουργει η διαδικασία ευρεσης και δημιουργίας γράφων για τα top forums συμφωνα με το μέγεθος των μελών τους.

Τα threads ξεκινουν να δουλεουν με το που μπει μια δουλεια στην ουρά του threadpoll μεσω της Add_Job_To_ThreadPool η οποία περναει την συναρτηση με τα ορισματα σε μια δομη tsk_args για να τα χρησημοποιήσουνε τα threads.

Για την ευρεση των Nforums χρησιμοποειται μια λιστα στην οποία γράφουν τα threads τα Nforums. Η λιστα αυτή λειτουργει συμφωνα με 2 μεταβλήτες οι οποιες κρατούν το μεγιστο αριθμό των μελών των forum που εχουν μπει και τον αντιστοιχο ελάχιστο έτσι ώστε να ξερει το προγραμμα αν θα πρεπει να μπει το forum στη λιστα αυτη ή οχι. Στην περιπτωση που μπει (η λιστα εισαγει τα στοιχεια ταξινομημένα) αφαιρείται το τελευταιο στοιχειο της λιστας και η ελαχιστη τιμή των μελων του forum αποκτα την τιμή του αμέσως μικρότερου.

Τα threads για την καταμέτρηση και εισαγωγη στη λίστα με τα Nforums χρησιμοποιουν την συνάρτηση Find_Nforums.

Τα threads για την δημιουργία των γράφων χρησιμποιουν την συνάρτηση Create_F_Graphs.

Το main thread κανει suspend χρησημοποιοντας mutex/cond_var αρχικά για την ευρεση των Nforums και στην συνέχεια για την δημιουργια των γράφων για καθε forum.

Η εγγραφή στις κοινες δομές (critical section) προστατευεται και αυτή με mutex semaphores.

->Clique Percolation Method

Ο Αλγόριθμος προσεγγιστηκε ως εξής:
Καθε κομβος του γράφου του γράφου είναι υποψήφια 1-clique εκτως απο αυτους που εχουν λιγότερους απο k-1 γειτονες.

Όλες αυτες οι υποψήφιες κλίκες αποθηκευονται σε μια λίστα και μέσω αυτης της λίστας παράγονται οι 2-cliques μέσω κριτηριων ελεγχου και η διαδικασία συνεχίζει δημιουργοντας μια άλλη λίστα με τις k+1 cliques μεχρι να φτάσουμε στο επιθημητο μεγεθος κλίκας.
Κάθε κλίκα που δημιουργείται έχει ενα μοναδικο id το οποιο παραγεται απο τον αριθμο εισαγωγης (ins_num)του κόμβου. Πχ έστω οι 3 κομβοι με τα id και τα ins_num τους

id 5 ins_num 1
id 8 ins_num 2
id 2 ins_num 3

εαν αυτοι οι κομβοι αποτελεσουν μια κλίκα [2,5,8] το id της θα είναι το 123 το οποιο παραγεται απο την συναρτηση Fix_Clique_id . Τα ins num ταξινομουνται οποτε εαν εχει ηδη βρεθεί η παραπανω κλίκα ,η κλίκα [5,8,2] θα αγνοηθεί αφου είναι στην ουσια η ίδια κλίκα με το ιδιο id 123.

Στη συνέχεια οι κλίκες που βρεθηκαν εισάγωνται σε ενα υπεργραφο (SuperGraph) και συνδέονται συμφωνα με τα κριτήρια του αλγοριθμου.

Τα communites είναι καθε συνεκτικο γράφημα του υπεργράφου και εισαγωνται στην δομή Communities.

->Αλγοριθμος Girvan-Newman

Ο γράφος κρατάει ενα hash table με ιds όλων των ακμων οι οποιοι παράγονται πάλι απο τα ins_num πχ 2 κομβοι με τα ins_num τους

id 89 ins_num 1
id 900 ins_num 2

οποτε η ακμη τους θα εχει ιd 12 είτε είναι η 89-900 είται ειναι η 900-89.

Το edge betweenness υπολογίζεται περιπου όπως και το betweenness centrality ενός κόμβου μόνο που κρατείται ενα hash table το οποίο αντιστοιχει τα ids των ακμών με τον αριθμό που έχουν εμπλακεί ως γεφυρες οι συγκεκριμένες ακμές. (edge_bet_counter). Το ποιες ακμές εχουν εμπλακει μας το δείχει μια λίστα Engaged_Edges_List που περιεχει ακμές στην οποία βάζει τιμες η e_find_All_possible_path
η οποία εσωτερικά κάνει report το μονοπάτι που ακολουθήθηκε. Τα edge_bet_counter αρχικοποιουνται οταν αλλαζει ενα state το οποιο παιρνει τιμές όσες φορες εχει τρέξει ο αλγοριθμος. Τελος ο αλγοριθμος σταματαει όταν το modularity αρχιζει να φθίνει ή οταν παρει την μεγιστη τιμή του.