Αναστασίου Σπύρος 1115201300007

Κατηφόρης Ελευθέριος 1115201300065

Ραφαήλ Ηλίας 1115201300149

Time					
	Small Graph	Medium-Static Graph	Medium-Dynamic Graph	Large-Static Graph	Large-Dynamic Graph
Part 1	2 sec	-	38 sec	-	-
Part 2	1 sec	5 sec	42 sec	-	-
Part 3	1 sec	5 sec	20 sec	280 sec	1140 sec

Για τις μετρήσεις της μνήμης έγινε χρήση των εντολών valgrind --tool=massif ./out και ms_print massif.out.<pid> | less .

Memory					
	Small Graph	Medium-Static Graph	Medium-Dynamic Graph	Large-Static Graph	Large-Dynamic Graph
Part 1	99.3 Mb	-	286.5 Mb	-	-
Part 2	97.6 Mb	710 Mb	526.2 Mb	-	-
Part 3	174.4 Mb	710 Mb	754.7 Mb	2.3 Gb	1.4 Gb

~~~~~~

- List_node:

Σε έναν πίνακα σταθερού μεγέθους Ν κρατάμε τους γείτονες του κόμβου.

Σε έναν πίνακα σταθερού μεγέθους Ν κρατάμε τα versions για κάθε ακμή του κόμβου.

Σαν offset κρατάμε τον δείκτη στο επόμενο list_node του κόμβου.

Σαν last_neighbor κρατάμε σε ποιά θέση του πίνακα των γειτόνων έγινε η τελευταία εισαγωγή γείτονα.

- Buffer:

Σε έναν δυναμικό πίνακα κρατάμε όλα τα list_nodes του γράφου.

Σαν size κρατάμε το μέγεθος του buffer.

Σαν last κρατάμε σε ποιά θέση του πίνακα των list_nodes έγινε η τελευταία εισαγωγή.

- NodeIndex:

Σε έναν δυναμικό πίνακα (nodes) κρατάμε για κάθε κόμβο τον αντίστοιχο δείκτη/offset στον πίνακα buffer.

Σε έναν δυναμικό πίνακα (last_bucket) κρατάμε το offset του πίνακα buffer στο οποίο έγινε η τελευταία εισαγωγή γείτονα για κάθε κόμβο.

Σε έναν δυναμικό πίνακα (count) κρατάμε τον αριθμό των γειτόνων του κάθε κόμβου.

- Graph:

Κρατάμε το NodeIndex και τον αντίστοιχο Buffer για τις εισερχόμενες και τις εξερχόμενες ακμές του γράφου.
~~~~~~

~~~~~~

- Components:

Σε έναν δυναμικό πίνακα (node_ids) κρατάμε τα id των κόμβων που ανήκουν στο συγκεκριμένο component.

Σαν component_id κρατάμε το id του component.

Σαν nodes_count κρατάμε το πλήθος των κόμβων του συγκεκριμένου component.

Σαν size κρατάμε το μέγεθος του πίνακα node_ids.

- SCC:

Σε έναν δυναμικό πίνακα (components) κρατάμε όλα τα components.

Σαν components_count κρατάμε το πλήθος των components που έχουν δημιουργηθεί.

Σαν size κρατάμε το μέγεθος του πίνακα components.

Σε έναν δυναμικό πίνακα (id_belongs_to_component) κρατάμε σε ποιό component ανήκει κάθε κόμβος.

- GrailIndex:

Σε 2 πίνακες σταθερού μεγέθους κρατάμε τα rank και min_rank του κάθε component.

~~~~~~

- CC:

Σε έναν δυναμικό πίνακα (ccindex) κρατάμε για κάθε κόμβο σε ποιό component ανήκει.

Σε ένα HashList (updateIndex) κρατάμε ποιά components ενώθηκαν όταν γίνει κάποια προσθήκη κόμβου στο γράφο. Αρχικά το updateIndex το είχαμε υλοποιήσει σε μορφή γράφου το οποίο δεν ήταν τόσο αποδοτικό.

~~~~~~

Γενικά:

Η υλοποίηση καλύπτει όλα τα ζητούμενα όπως αυτά αναφέρονται στις εκφωνήσεις του εκάστοτε part .Η αναζήτηση των queries γίνεται μέσω του αλγορίθμου BBFS ,ο οποίος
~~~~~~

αναπτύσσει την μεριά που έχει λίγοτερους κόμβους ώστε να είναι πιο αποδοτικός . Στους static γράφους δημιουργείται το GrailIndex ,που παρέχει πληροφορίες για το αν 2 nodes ανήκουν στο ίδιο Strong Component . Η υλοποίηση μας έχει 2 τέτοια Indexes ,ενώ έχει τη δυνατότητα για δημιουργεία επιπλέον.Ο BBFS ελέγχει σε κάθε βήμα μόνο κόμβους είναι grail-reachable .Στους dynamic γράφους δημιουργείται το CC-index που παρέχει πληροφορίες για τα Connected Components του γράφου.

Επιπροσθέτως , το πρόγραμμα κάνει χρήση των threads .Υπάρχει ο JobScheduler ο οποίος δημιουργεί τα threads , και τους αναθέτει δουλειές όταν αυτό είναι εφικτό.

Κάθε thread παίρνει πλήθος απο jobs/queries , τα οποία εκτελεί και τα καταχωρεί σε πίνακα του main-thread , το οποίο τα εκτυπώνει όταν έχουν υπολογιστεί όλα τα ερωτήματα.Στο dynamic χρησιμοποιούνται τα versions του κάθε query ώστε να εκτελέιται το κάθε query βάση της κατάστασης του γράφου όταν διαβάστηκε και όχι της στιγμής εκτέλεσης του.

Επίσης δοκιμάστηκαν διάφοροι συνδιασμοί στις παραμέτρους (μέγεθος πινάκων , μετρική rebuild ,πλήθος threads,πλήθος grailIndexes κτλπ) , και οι τελικοί χρόνοι υπολογίστηκαν πάνω στις καλύτερες τιμές.

Οι μετρήσεις γίνανε σε linux 16.04 με 8 GB Ram και Cpu i7 στα 2.4 GHz (στα 8 threads).