Τεχνολογίες Υλοποίησης Αλγορίθμων

Αλγόριθμοι Συντομότερων Διαδρομών Δυναμικών Γραφημάτων

Φλώρος Παναγιώτης 1047137

CEID
University of Patras
2018-2019

Υλοποίηση

Επιλογή βιβλιοθήκης

Για την υλοποίηση της εργασίας επιλέχθηκε χρήση της βιβλιοθήκης Boost, η οποία περιέχει αλγόριθμους συντομότερων διαδρομών όπως οι αλγόριθμοι Johnson και Floyd Warshall, οι οποίοι χρησιμοποιήθηκαν για την πειραματική σύγκριση.

Δομές

Ο αλγόριθμος της εργασίας [1] χρησιμοποιεί 5 δομές δεδομένων:

1. D: ένα μητρώο με δείκτες τους συνδιασμούς των κόμβων, του οποίου τα στοιχεία είναι το κόστος της συντομότερης διαδρομής απο το πρώτο δείκτη στον δεύτερο.
2. FORWARD: παρόμοιο μητρώο με το D αλλά αντί να αποθηκεύει συντομότερες αποστάσεις, αποθηκέυει τη θέση του δεύτερου δείκτη στο δέντρο DESC του πρώτου.
3. BACKWARD: παρόμοιο μητρώο με το FORWARD αλλά αποθηκέυει τη θέση του δεύτερου δείκτη στο δέντρο ANC του πρώτου.
4. DESC: δομή που αντιστοιχίζει κάθε κόμβο σε δέντρο συντομότερων διαδρομών με τους απογόνους του με ρίζα το συγκεκριμένο κόμβο.
5. ANC: ομοίως με DESC αλλά αντιστοιχίζει με δέντρο προς τους προγόνους του κόμβου.

Οι παραπάνω δομές υλοποιήθηκαν όλες ως map της standard cpp library για λόγους ευκόλιας και διότι το map αποτελεί δυναμική δομή. Άρα δεν χρειάζεται εξ΄ αρχής ορισμό του μεγέθους. Κάτι που θα χρειαζόταν ένα c array, το οποίο ίσως να ήταν πιο αποδοτικό, αλλά απορρίφθηκε για αυτο το λόγο.

Δέντρα Όπως αναφέρθηκε παραπάνω, ύπαρχουν οι δομές *ANC* και *DESC* οι οποίες αποθηκεύουν δέντρα για κάθε κόμβο. Αυτά τα δέντρα υλοποιήθηκαν ώς γράφοι της *Boost* όπου κάθε κόμβος αποθηκεύει την τιμή του στο γράφημα, και την τιμή του πατέρα του στο γράφημα. Ο ορισμός που χρησιμοποιήθηκε δηλώνει την αποθήκευση τον κόμβων του δέντρου σε vector. Κάθε κόμβος του δέντρου έχει δικό του μοναδικό id/index το οποίο

είναι ανεξάρτητο απο το κόμβο που αντιστοιχεί στο αρχικό γράφημα. Πράγμα το οποίο δημιουργεί την ανάγκη για την αποθήκευση της αντίστοιχης τιμής του γράφου στο κάθε κόμβο του δέντου καθώς και του αντίστοιχου πατέρα.

Αλγόριθμος

Ο αλγόριθμος σε μορφή ψευδοκώδικα ο οποίος περιλαμβάνεται στην εργασία [1] είναι καλός για να βοηθήσει τον αναγνώστη να κατανοήσει το πως λειτουργεί ο αλγόριθμος αλλά στην φάση της υλοποίησης χρειάζεται να γίνουν αρκετές αλλαγές. Για παράδειγμα, στον ψευδοκώδικα σε κάθε κλήση της *UpdateForward* γίνεται ένθεση κόμβου στο δέντρο T αλλά στην πράξη πρέπει να γίνουν έλεγχοι για το αν ήδη υπάρχει αυτός ο κόμβος και στην περίπτωση που υπάρχει πρέπει να διαγραφεί η ακμή που τον συνδέει με τον προηγούμενο πατέρα αλλιώς υπάρχει κίνδυνος να σχηματιστεί κύκλος στο δέντρο και να μην είναι σωστα τα αποτελέσματα μετά την εκτέλεση του αλγορίθμου.

Μια ακόμα διαφοροποιήση υπάρχει κατά την αρχικοποίηση του γραφου. Σύμφωνα με την εργασία [1] το γράφημα ξεκινά κενό, αλλα κατά την υλοποίηση στον constructor μπορεί να ληφθεί γράφημα με ακμές καθώς ο constractor προσπελαύνει τις ακμές του γραφήματος και τις κάνει add στον αλγόριθμο. Με αυτό το τρόπο η υλοποίηση προσφέρει περισσότερη ευελιξία στο χρήστη.

Επιπλέον, η υλοποίηση δεν κρατά στην μνήμη της κάποιο reference, π.χ. με μορφή pointer, ή κάποιο αντίγραφο του γραφήματος. Το μόνο που χρειάζεται είναι αρχικοποίηση και μετά σύμφωνα με κάθε αλλαγή στο γράφημα να καλείται και η αντίστοιχη συνάρτηση της υλοποίησης.

Οι συναρτήσεις της υλοποίησης είναι οι εξης:

- **add(Vertex x, Vertex y, int weight)** προσθέτει ακμή στα δεδομένα του αλγορίθμου, όχι στο γράφο.
- **decrease(Vertex x, Vertex y, int init_cost, int diff)** μειώνει το βάρος της ακμής (x, y) κατά diff στα δεδομένα του αλγορίθμου, όχι στο γράφο.
- **length(Vertex x, Vertex y)** επιστρέφει το μήκος της συντομότερης διαδρομής απο το x στο y, ο χρήστης που καλεί τον αλγόριθμο δεν έχει πρόσβαση στον D, αυτή η συνάρτηση του προφέρει read-access.
- **minpath(Vertex x, Vertex y)** επιστρέφει του κόμβους οι οποίοι ανήκουν στο συντομότερο μονοπάτι απο x στο y.

Οπτικοποίηση

Η οπτικοποίηση γίνεται χρησιμοποιώντας την write_graphviz της Boost η οποία γράφει το γράφημα σε μορφή dot αρχείο το οποίο μπορεί να διαβαστεί απο ένα dot file viewer, ακόμα και online, και να εμφανιστεί γραφική αναπαράσταση του γραφήματος.

Πειραματική αξιολόγηση

Ανάλυση

Οι αλγόριθμοι οι οποίοι χρησιμοποιούνται για την πειραματική αξιολόγηση είναι οι αλγόριθμοι Johnson και Floyd Warshall, υλοποιημένοι απο την Boost. Οι δύο αυτοί αλγόριθμοι είναι στατικοί, δηλαδή δεν υπάρχει τρόπος για ενημέρωση του distance matrix κάθε φορά που γίνεται κάποια αλλαγή στο αρχικό γράφημα, όποτε σε αυτή την περίπτωση πρέπει να επανακληθούν. Ο αλγόριθμος της εργασίας [1] είναι δυναμικός, οπότε μπορεί να ενημερωθεί για τις αλλαγές και έτσι να πετύχει κόστος κατά κάποια αλλαγή μικρότερο απο την επανάκληση κάποιου στατικού αλγορίθμου, ωστόσο η αρχικοποίηση του αλγορίθμου είναι πολύ πιο κοστοβόρα καθώς απαιτεί αρχικοποίηση πολλών δομών δεδομένων, όπως παρατηρείται παρακάτω.

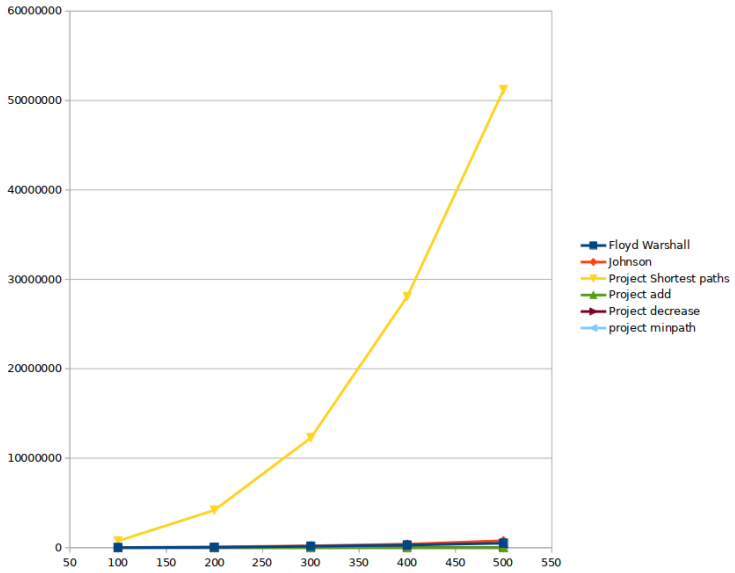
Μετρήσεις

Για τις μετρήσεις χρησιμοποιήθηκαν τυχαία γραφήματα μεγέθους 100, 200, 300, 400 και 500, με τυχαίο αριθμό ακμών στο εύρος (N, N^2), ώστε να μπορούν να παραχθούν και αραιά αλλά και πυκνά γραφήματα. Όλες οι παρακάτω μετρήσεις είναι σε μsec.

Πίνακας 1: Μετρήσεις, Μέσοι όροι 10 εκτελέσεων.

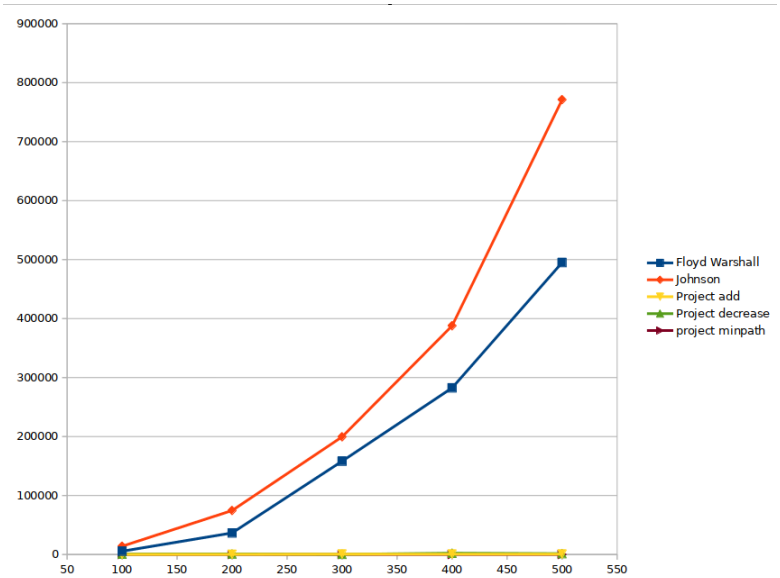
Αριθμός κόμβων	100	200	300	400	500
Floyd Warshall	5483.8	36523.5	158217.1	282519.5	494984.5
Johnson	13942.6	74538.6	199520.3	387870.3	771227
Project Shortest paths	745921.1	4188066.3	12281343	28067072.1	51201124.8
Project add	73.3	141.6	554	493.2	532.4
Project decrease	654.1	860.7	325.2	2243	1457
Project minpath	4.8	8.5	11.2	14.4	18.9

Φαίνεται ότι ο χρόνος αρχικοποίησης του αλγορίθμου της εργασίας είναι τόσο μεγάλος που δεν φαίνονται οι υπόλοιπες γραφικές παραστάσεις.

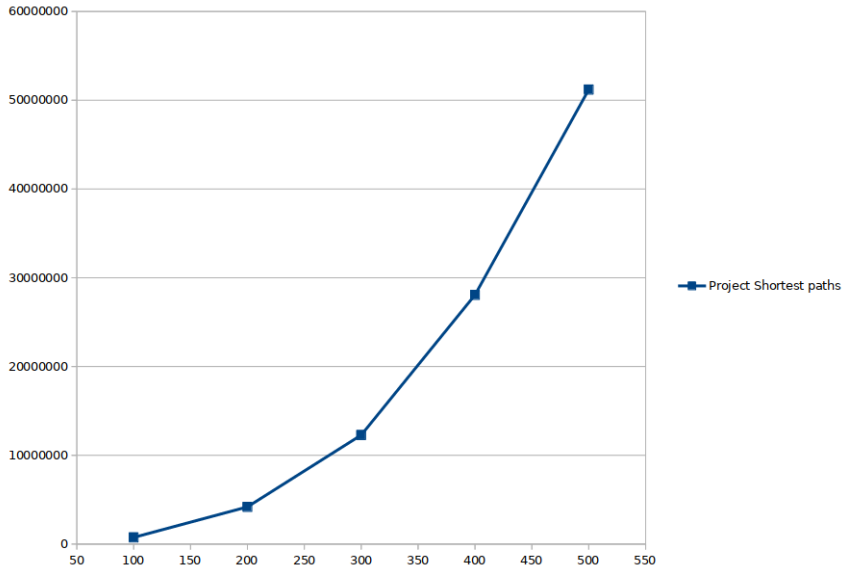


Σχήμα 1: Συγκρίσεις μέσων χρόνων εκτέλεσης.

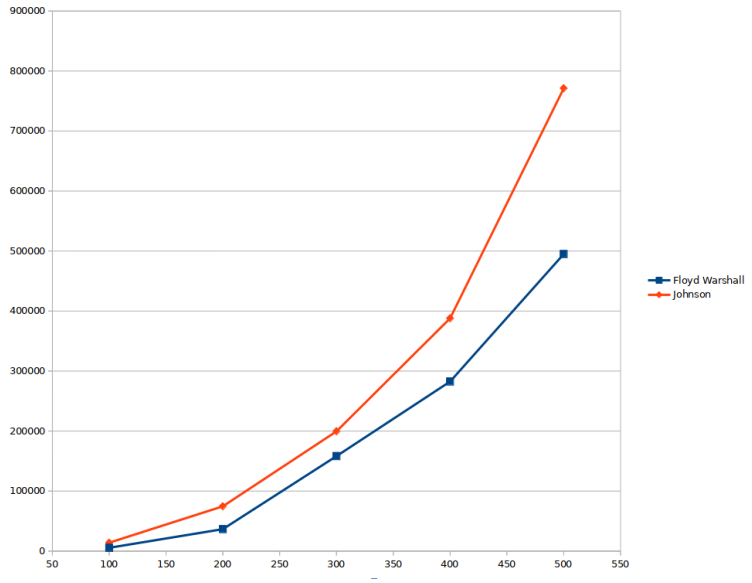
Ομοίως οι χρόνοι εκτέλεσεις των *Johnson, Floyd Warshall* είναι τόσο μεγάλοι που δεν φαίνονται οι παραστάσεις των πράξεων του αλγορίθμου της εργασίας.



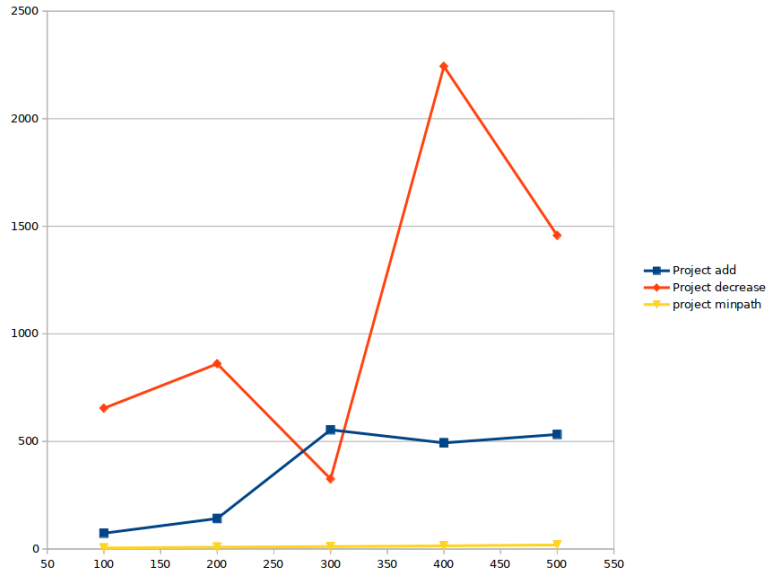
Σχήμα 2: Συγκρίσεις μέσων χρόνων εκτέλεσης. Χωρίς αρχικοποίηση του project.



Σχήμα 3: Μέσος χρόνος εκτέλεσης αρχικοποίησης αλγορίθμου εργασίας.



Σχήμα 4: Συγκρίσεις μέσων χρόνων εκτέλεσης *Johnson, Floyd Warshall.*



Σχήμα 5: Παραστάσεις μέσων χρόνων εκτέλεσης πράξεων του αλγορίθμου.

Οι χρόνοι εκτέλεσεις των πράζεων παραμένουν μικροί. Υπενθυμίζεται ότι για κάθε πράξη add ή decrease στον αλγόριθμο, οι στατικοί θα πρέπει να επανεκτελεσθούν. Όμως, ο χρόνος αρχικοποίησης του αλγορίθμου είναι υπερβολικά μεγάλος καθώς απετεί πάρα πολλές αρχικοποιήσεις πολλών δομών δεδομένων που καταλαβάνουν υπερβολικά πολύ μνήμη.

Συμπεράσματα

Ο αλγόριθμος της εργασίας πετυχένει καλύτερους χρόνους σε ενημερώσεις απο ότι κοστίζει η επανάκληση των στατικών αλγορίθμων που αναφέρθηκαν παραπάνω, πράγμα που τον καθιστά χρονικά αποδοτικότερο αλλά ο αλγόριθμος έχει μεγάλο μειονέκτημα στην χωρική πολυπλοκότητα καθώς δεσμέυει υπερβολικά πολύ χώρο για κάθε κόμβο, πράγμα το οποίο ίσως σημαίνει ότι στη πράξη δεν μπορεί να χρησιμοποιηθεί παρα σε λίγες συγκεκριμένες εφαρμογές.

Βιβλιογραφία

[1] G. Ausiello, G.F. Italiano, A. Marchetti-Spaccamela, and U. Nanni. *Incremental algorithms for minimal length paths.* Journal of Algorithms 12 (1991), pp.615-638.