Τεχνολογίες Υλοποίησης Αλγορίθμων

1η εργασία

Ερώτημα Α:

Αλγόριθμος Πιστοποίησης

Γίνεται η δημιουργία συνάρτησης με το όνομα: my_BFS, η οποία παίρνει τέσσερα ορίσματα. Ένα γράφημα, ένα κόμβο, ο οποίος θα είναι ο αρχικός, ένα πίνακα κόμβων για να αποθηκεύονται τα επίπεδα των κόμβων με βάση τον BFS αλγόριθμο και τέλος, έναν πίνακα ακμών, ο οποίος θα περιλαμβάνει τους προκατόχους των ακμών. Αρχικά, ορίζονται όλες οι κορυφές στο επίπεδο -1, πράγμα που σημαίνει ότι δεν έχουν ανακαλυφθεί ακόμα από τον αλγόριθμο BFS. Ορίζεται μόνο ο αρχικός κόμβος στο επίπεδο 0, από όπου και ξεκινάμε. Δημιουργούμε μια ουρά, στην οποία προσθέτουμε τις κορυφές που ο BFS δεν έχει βρει ακόμα, καθώς και μια λίστα η οποία θα περιλαμβάνει τις κορυφές που έχει ανακαλύψει ο αλγόριθμος. Όσο η ουρά δεν είναι άδεια παίρνουμε την πρώτη κορφή, την προσθέτουμε στην λίστα και ελέγχουμε τις ακμές της κορυφής για να βρούμε τις κορυφές προορισμού. Αν το επίπεδο της κορυφής είναι -1, τότε την προσθέτουμε στην ουρά και ενημερώνουμε τους πίνακες επιπέδου και προκατόχου ακμής. Τέλος, η συνάρτηση επιστρέφει την λίστα με τις κορυφές.

Ελεγκτής Διμερότητας

Γίνεται η δημιουργία της συνάρτησης με όνομα my_bipar_checker, η οποία παίρνει τρία ορίσματα. Ένα γράφημα και δύο πίνακες κορυφών. Αρχικά, διαλέγουμε μια τυχαία κορυφή για να ξεκινήσουμε. Εφαρμόζουμε τον αλγόριθμο BFS που έχει υλοποιηθεί παραπάνω και χωρίζουμε τις κορυφές σε χρώματα, ανάλογα με το επίπεδο τους. Αν το επίπεδο είναι περιττό, το χρώμα θα είναι μπλε, ενώ αν το επίπεδο είναι άρτιο, τότε το χρώμα θα είναι πράσινο. Έπειτα, για κάθε κόμβο του γραφήματος ελέγχουμε αν υπάρχουν δύο διπλανές κορυφές με το ίδιο χρώμα. Αν υπάρχουν, τότε, δημιουργούμε μια λίστα, στην οποία θα τοποθετούμε τις κορυφές του κύκλου. Για κάθε ακμή βρίσκουμε εισερχόμενες και εξερχόμενες ακμές και προσθέτουμε κορυφές στις λίστες χρησιμοποιώντας συνθήκες για το αν υπάρχει ακμή προέλευσης και ακμή προορισμού. Ψάχνουμε στην λίστα, αφαιρώντας κόμβους, μέχρι να βρούμε τον κοινό κόμβο. Η συνάρτηση επιστρέφει την τιμή false, καθώς το γράφημα δεν είναι διμερές. Από την άλλη, αν δεν βρεθούν δύο διαδοχικοί κόμβοι με ίδιο χρώμα, τότε χωρίζουμε στους πίνακες κόμβων τις κορυφές ανάλογα με το χρώμα τους και η συνάρτηση επιστρέφει true, γιατί το γράφημα είναι διμερές.

Πολυπλοκότητα

Ο BFS απαιτεί χρόνο O(n). Ο έλεγχος των ακμών απαιτεί O(m) χρόνο, αφού στην χειρότερη περίπτωση θα πρέπει να ελεγχθούν 2m ακμές. Σύγκριση χρωμάτων κορυφών O(m) χρόνος. Εισαγωγή κύκλου στην λίστα O(n) χρόνος. Άρα η πολυπλοκότητα του ελεγκτή διμερότητας θα είναι O(n+m).

Ερώτημα Β:

Γράφημα Εμφωλιασμένων Τετραγώνων:

Δημιουργήθηκε η συνάρτηση Embbeded_Square_Graph, η οποία έχει τέσσερα ορίσματα. Ένα γράφημα, μια λίστα κόμβων, μια int μεταβλητή για τον αριθμό των κορυφών και μια int μεταβλητή η οποία μετράει τις κορυφές που βάζουμε στο γράφημα. Αρχικά, δημιουργούμε τέσσερις κόμβους, όσες οι γωνίες του τετραγώνου και τέσσερις ακμές για να ενώσουμε τις κορυφές. Επειδή θέλουμε το γράφημα μας να είναι μη κατευθυνόμενο, ορίζουμε και τις ανάποδες ακμές από αυτές που έχουν ήδη δημιουργηθεί. Ελέγχουμε αν η λίστα με τις ακμές είναι άδεια. Αν δεν είναι τότε θα υπάρχει ήδη εμφωλιασμένο τετράγωνο, οπότε παίρνουμε τις κορυφές του και τις ενώνουμε με ακμές (κανονικές και αντίστροφες) με τις κορυφές του νέου τετραγώνου. Προσθέτουμε στον counter τις κορυφές που εισαγάγαμε . Αλλιώς, αν η λίστα είναι άδεια, τότε δεν υπάρχει άλλο τετράγωνο, οπότε ο counter θα είναι ίσος με τέσσερα, δηλαδή τις κορυφές που μόλις προσθέσαμε. Βάζουμε στην λίστα των κόμβων τις κορυφές αυτές και τέλος, ελέγχουμε αν οι κορυφές που έχουμε είναι λιγότερες από τις κορυφές που θέλουμε. Αν αυτό ισχύει, τότε καλούμε αναδρομικά την συνάρτηση.

Γράφημα Δακτυλίου

Δημιουργήθηκε η συνάρτηση Ring_Graph, η οποία έχει δύο ορίσματα. Ένα γράφημα και μια μεταβλητή int με τον αριθμό των κορυφών που θέλουμε να έχει το γράφημα. Αρχικά, δημιουργούμε τις δύο πρώτες κορυφές και τις ενώνουμε μεταξύ τους με μια ακμή (κανονική και ανάποδη). Μετά, μέχρι να τελειώσει ο αριθμός των κορυφών που θέλουμε να βάλουμε προσθέτουμε κορυφές στο γράφημα μετά την τελευταία ήδη υπάρχουσα κορυφή και ενώνουμε πάλι με ακμές. Θέτουμε την νέα κορυφή ως τελευταία για να συνεχίσει από εκεί η προσθήκη νέων κόμβων. Τέλος, ενώνουμε με ακμή την πρώτη και την τελευταία κορυφή του γραφήματος.

Πειραματική Αξιολόγηση

Για κάθε γράφημα υπάρχουν 5 διαφορετικά πλήθη κορυφών που δοκιμάστηκαν.
Κάθε μέτρηση έγινε 10 φορές και ο τελικός χρόνος βγήκε με τον μέσο όρο.

Σημείωση:

Το γράφημα 4 επιπέδων δεν έχει υλοποιηθεί.

Γράφημα	# Κορυφών	My Checker(sec)	LEDA Checker(sec)
Εμφωλιασμένων Τετραγώνων	10000	0.002	0.001
Εμφωλιασμένων Τετραγώνων	50000	0.01	0.009
Εμφωλιασμένων Τετραγώνων	80000	0.01	0.013
Εμφωλιασμένων Τετραγώνων	100000	0.024	0.018
Εμφωλιασμένων Τετραγώνων	150000	0.038	0.033
Δακτυλίου	10000	0.04	0.00099
Δακτυλίου	50000	1.023	0.004
Δακτυλίου	80000	2.698	0.0069
Δακτυλίου	100000	4.562	0.011
Δακτυλίου	150000	10.085	0.016

Διαγράμματα χρόνων εκτέλεσης

Παρατηρούμε ότι η δική μου υλοποίηση είναι πιο αργή από την υλοποίηση της LEDA, καθώς της LEDA είναι βέλτιστη. Ειδικά στα γραφήματα δακτυλίου, όπου η υλοποίηση της LEDA μένει κοντά στο μηδέν, ενώ η δική μου ξεπερνάει τα 10 δευτερόλεπτα για μεγάλο πλήθος ακμών. Στον άξονα y βλέπουμε τον χρόνο εκτέλεσης, ενώ στον άξονα x, βλέπουμε το πλήθος των κορυφών του γραφήματος.