Αναφορά Εργασίας

Υλοποίηση

Η εργασία αφορούσε την υλοποίηση δυο αλγόριθμων σχετικά με την διατήρηση των bridge-connected components και biconnected components σε μη κατευθυνόμενα γραφήματα. Αρχικά παρουσιάζεται ο αλγόριθμος για bridge-connected components.

Για την υλοποίηση του αρχικά ορίστηκε το αντικείμενο του κόμβου του γραφήματος ΄΄Node΄΄, όπου και διατηρεί σχετικές πληροφορίες που χαρακτηρίζουν τους κόμβους. Συγκεκριμένα ένα πεδίο name, αναγνωριστικό για την ονομασία του κόμβου, ένα πεδίο label όπου κρατάει το όνομα του label των στρογγυλών κόμβων του γραφήματος, ένα πεδίο type, που δηλώνει τον τύπο του κόμβου σε κυκλικό ή τετράγωνο, ένα πεδίο Node* όπου ο κόμβος κρατάει μια αναφορά για τον πατέρα του και ένα πεδίο vector<Node*> όπου ο κόμβος διατηρεί μια λίστα με τους κόμβους που συνδέονται μαζί του. Η ονομασία των κόμβων για να είναι μοναδική έγινε μέσω ένα static counter όπου αυξανόταν κάθε φορά που δημιουργούταν ένας νέος κόμβος και όριζε το όνομα του. Κάθε κόμβος έχει επίσης την δυνατότητα να τυπώνει πληροφορίες που σχετίζονται με αυτόν όπως τον τύπο του κόμβου και τον πατέρα του αν αυτός υπάρχει. Κάθε κόμβος κατά την δημιουργία του δέχεται σαν όρισμα το label και τον τύπου του κόμβου που θα δημιουργηθεί.

Αρχικά για την προσθήκη νέων τετράγωνων κόμβων πρέπει να δημιουργήσουμε δύο κόμβους, έναν τετράγωνο και ένα στρογγυλό διότι δεν μπορεί να υπάρξει τετράγωνος κόμβος χωρίς να είναι συνδεμένος με ένα στρογγυλό κόμβο. Για την διατήρηση των κόμβων χρησιμοποιούμε μια δομή δεδομένων τύπου vector για να καλύψουμε τις δυναμικές ανάγκες του αλγόριθμου σχετικά με την προσθήκη καινούργιων κόμβων. Για την ευκολότερη διαπίστωση αν ένας κόμβος ανήκει στο ίδιο bridge-connected component ή όχι υλοποιούμε ένα Disjoint Data Set Union Structure μέσω ενός unordered map και των συναρτήσεων του Union, Find και make_set. Οι συναρτήσεις χρησιμοποιούνται για να κάνουν ένωση δύο subset, εύρεση της ρίζας ενός στοιχείο και προσθήκη ενός καινούργιου στοιχείου στον πίνακα αντίστοιχα.

Κατά την δημιουργία ενός κόμβου αν δοθεί σαν όρισμα κάποιο label πέρα του μηδενός τότε δημιουργούμε έναν κυκλικό κόμβο και τον προσθέτου στο vector του γραφήματος και στο Disjoint Set. Αντίθετα αν το label κατά την δημιουργία του είναι 0 τότε προχωράμε στην δημιουργία ενός τετράγωνου κόμβου. Όταν όμως δημιουργούμε ένα τετράγωνο κόμβο τότε ταυτόχρονα δημιουργούμε και ένα κυκλικό κόμβο. Στην συνέχεια προχωράμε στη σύνδεση των δύο νέων κόμβων μέσω της συνάρτησης link η οποία ενώνει αυτός τους δύο κόμβους με την σχέση πατέρα παιδιού και ενημερώνει τα αντίστοιχα πεδία του καθενός. Μετά εφαρμόζουμε την συνάρτηση union για να ενώσουμε αυτά τα δύο subsets σε ένα και ενημερώνει και το rank τους.

Για την εύρεση του μονοπατιού μεταξύ δύο κόμβων χρησιμοποιούμε την συνάρτηση findpath. Αν οι δύο κόμβου για τους οποίους ψάχνουμε το μονοπάτι μεταξύ τους είναι ίδιοι τότε επιστρέφουμε μόνο των γονέα τους. Αλλιώς

δημιουργούμε αρχικά δυο vectors ένα για κάθε κόμβο που πήραμε σαν όρισμα. Κάθε vector αρχικοποιείται με έναν γονέα από τους κόμβους που δόθηκαν σαν όρισμα. Το κάθε μονοπάτι που ψάχνουμε τώρα θα ξεκινάει από τον γονέα ενός κόμβου μέχρι να φτάσει στην ρίζα. Μόλις και οι τα δύο μονοπάτια φτάσουν στην ρίζα τότε προχωράμε στο επόμενο βήμα.

Έπειτα για κάθε κόμβο του ενός μονοπατιού που αποθηκεύσαμε ελέγχουμε αν εμφανίζεται στο μονοπάτι του δεύτερου. Μόλις βρεθεί αυτός ο κόμβος τον κρατάμε ως τον κοντινότερο κοινό πρόγονο των δύο αυτών κόμβων. Αποθηκεύουμε σε μια ξεχωριστή λίστα το μονοπάτι μιας λίστας μέχρι το σημείο που βρέθηκε ο κοντινότερος κοινός πρόγονος και έπειτα προσθέτουμε στην ίδια λίστα την δεύτερη λίστα μέχρι να συναντήσουμε πάλι τον κοινό κόμβο και επιστρέφουμε το μονοπάτι αυτό.

Μια ακόμα συνάρτηση απαραίτητη για την διατήρηση της δομής του Bridge Component Forest είναι η evert η οποία κάνει τον κόμβο που παίρνει σαν όρισμα την ρίζα του δέντρο στο οποίο βρίσκεται. Αρχικά για να προχωρήσουμε ελέγχουμε αν ο κόμβος που μας δόθηκε ήταν ήδη ρίζα του του δέντρου του ελέγχοντας αν έχει πατέρα. Η απώλεια πατέρα μας βεβαιώνει ότι είναι η ρίζα. Εδώ θα μπορούσε να χρησιμοποιηθεί και η συνάρτηση find, η οποία θα επέστρεφε την ρίζα του δέντρο που βρίσκεται ο κόμβος, συγκρίνοντας το αποτέλεσμα της με το όνομα του κόμβου. Αν είναι ίδιο τότε αυτός ο κόμβος είναι η ρίζα. Αν ο κόμβος στο όρισμα ήταν κυκλικός τότε αυτός ορίζεται σαν ρίζα, αν είναι τετράγωνος τότε κάνουμε ρίζα τον πατέρα αυτού του κόμβου.

Έπειτα προχωράμε στην διαδικασία εναλλαγής των δεικτών γονέα-πατέρα όλων των κόμβων μέχρι την προηγούμενη ρίζα. Αρχικά καθαρίζουμε τον πατέρα της

νέας ρίζας σε null και προσθέτουμε στα παιδία του τον προηγούμενο πατέρα του. Μετά αφαιρούμε την καινούργια ρίζα από την λίστα παιδιών του προηγούμενου του πατέρα. Και έτσι με αυτόν τρόπο συνεχίζουμε σε κάθε κόμβου που συναντάμε στο αντίστροφο μονοπάτι, αλλάζοντας των γονέα του επόμενου κόμβου με τον προηγούμενο, προσθέτοντας τον προηγούμενο πατέρα τους στα παιδιά τους και αφαιρώντας από τα παιδία του προηγούμενου πατερά τους τον κόμβο αυτό.

Η εύρεση της ταυτότητας του block όπου ανήκει ένα κόμβος γίνεται με την συνάρτηση find block. Η συνάρτηση κάνει χρήση της ήδη ορισμένη συνάρτησης find path όπου της δίνουμε σαν όρισα δύο φορές τον ίδιο κόμβο για να μας επιστρέψει τον πατέρα του κόμβου. Από αυτών τον κόμβο αποκτούμε πρόσβαση στο πεδίο label, που είναι το label που ανήκει ο κόμβος αυτός.

Μια άλλη βασική συνάρτηση είναι η condense path, η οποία ενώνει τους κόμβους ενός μονοπατιού σε ένα καινούριο κόμβο και παίρνει σαν παιδία του όλα τα παιδιά που είχαν οι κυκλικοί κόμβοι σε αυτό το μονοπάτι. Αρχικά δημιουργούμε ένα νέο κυκλικό κόμβο με το label που δώσαμε σαν όρισμα στην συνάρτηση. Έπειτα ελέγχουμε αν το μονοπάτι μεταξύ των δύο κόμβων που μας δόθηκε έχει την ρίζα ως τον πλησιέστερο κοινό κόμβο του. Αν ναι τότε και ο κόμβος που μόλις δημιουργήσαμε θα γίνει η καινούργια η ρίζα του δέντρου. Αλλιώς ο καινούργιος κόμβος που δημιουργήσαμε θα γίνει παιδί του πατέρα του πλησιέστερου κοινού κόμβου του μονοπατιού. Στην συνέχεια προσθέτουμε τον κόμβο που δημιουργήσαμε στο disjoint set και ξεκινάμε να διατρέξουμε τους κόμβους του μονοπατιού. Για κάθε κόμβο διατρέχουμε την λίστα των παιδιών του και κάθε τετράγωνο κόμβο τον συνδέουμε με τον καινούργιο κόμβο μας.

Τέλος για κάθε κυκλικό κόμβο στο μονοπάτι ενημερώνουμε το Disjoint Union Set ώστε οι κόμβοι να μην δείχνουν πουθενά μιας και οι κόμβου αυτοί δεν είναι πια έγκυροι.

Τέλος έχουμε την συνάρτηση με την οποία υλοποιούμε τις ακμές στην δομή δεδομένων μας. Αρχικά ελέγχουμε σε ποια από τις δύο περιπτώσεις βρισκόμαστε, αν δηλαδή οι δύο κόμβοι βρίσκονται στο ίδιο component ή όχι.

Αν βρίσκονται στο ίδιο component όπου το ελέγχουμε χρησιμοποιώντας την συνάρτηση find στους δύο κόμβους. Δημιουργούμε ένα καινούργιο label και μετά εφαρμόζουμε την συνάρτηση condense path σε αυτούς τους δυο κόμβους επειδή η καινούργια ακμή δημιουργεί ένα κύκλο στο γράφημα οδηγώντας στην σύμπτυξη των κόμβων ανάμεσα στο μονοπάτι τους.

Αν όμως βρίσκονται σε δύο διαφορετικά components τότε θα βρούμε το μέγεθος του κάθε component χρησιμοποιώντας το disjoint set από το unordered map και ελέγχοντας το ranking κάθε component. Έπειτα ελέγχουμε ποιο είναι το μικρότερο component ώστε να εφαρμόσουμε στον κόμβο που ανήκει σε αυτό το component την συνάρτηση evert ώστε να τον κάνουμε την ρίζα του μικρότερου component για να μπορέσουμε μετά να ενώσουμε τα δύο αυτά component κάνοντας link τους δύο κόμβους ενώνοντας έτσι τα δύο components.

Διαφοροποιήσεις του Δεύτερου αλγόριθμου

Αρχικά η πρώτη διαφοροποίηση έρχεται στην συνάρτηση make vertex αφού τώρα πιά δεν χρειάζεται να κατασκευάσουμε και ένα στρογγυλό κόμβο για κάθε τετράγωνο κόμβο που θέλουμε να φτιάξουμε.

Η επόμενη αλλαγή έρχεται στην συνάρτηση condense path. Τώρα χρειάζεται να της δώσουμε και ένα ακόμα όρισμα που θα χρησιμοποιηθεί σαν το label για τον καινούργιο στρογγυλό κόμβο που θα κατασκευάσουμε για να συμπτύξουμε όλου

τους στρογγυλούς κόμβους που βρίσκονται στο μονοπάτι προς συμπύκνωση. Επίσης ο καινούργιος κόμβος που δημιουργούμε τώρα πια μπορεί να προκύψει και παιδί κάποιου τετράγωνου κόμβου, πατέρα του κοντινότερου κοινού προγόνου στο μονοπάτι μεταξύ των δύο κόμβων που πραγματοποιούμε την συμπύκνωση.

Τέλος στην περίπτωση δημιουργίας ακμής μεταξύ δύο κόμβων διαφορετικού component τότε αφού βρούμε το μεγαλύτερο component δημιουργούμε έναν καινούργιο στρογγυλό κόμβο πάνω στον οποίο θα προσδεθεί το μικρότερο component σαν παιδί του αφού πρώτα έχει γίνει η διαδικασία evert με όρισμα τον κόμβο που αφορά η ακμή και έπειτα αυτός ο καινούργιος κόμβος θα γίνει παιδί του δεύτερου κόμβου που αφορά την ακμή.