Πανεπιστήμιο Θεσσαλίας

Τμήμα Μηχανικών Η/Υ Τηλεπικοινωνιών και Δικτύων

Αλγόριθμοι γραφημάτων στην δευτερεύουσα μνήμη

Λιώλη Διονυσία
e-mail:dilioli@inf.uth.gr

Διπλωματική Εργασία στα πλαίσια του Προπτυχιακού Προγράμματος Σπουδών του Τμήματος Μηχανικών Η/Υ Τηλεπικοινωνιών και Δικτύων

Επιβλέποντες Καθηγητές:

Μποζάνης Παναγιώτης
Κατσαρός Δημήτριος

Σεπτέμβριος 2008

ΕΥΧΑΡΙΣΤΙΕΣ

Ευχαριστώ τον επιβλέποντα καθηγητή μου κ. Μποζάνη Παναγιώτη για την καθοδήγησή του κατά την εκπόνηση της διπλωματικής μου εργασίας. Επίσης ευχαριστώ θερμά την οικογένειά μου και τους φίλους μου που με στήριξαν και με βοήθησαν καθ' όλη τη διάρκεια των σπουδών μου.

Περιεχόμενα

Εισαγωγή.....1
Βασικά στοιχεία του μοντέλου μας.....3
Βασικές εφαρμογές γραφημάτων.....3

1.Αναπαράσταση γραφημάτων.....6
1.1 Εισαγωγή.....6
1.2 Ανάπτυξη συστημάτων για την αναπαράσταση δικτύων και γραφημάτων.....6
1.3 LEDA-SM.....8
1.4 Τεχνική ανάθεσης ετικετών.....9

2. BFS.....12
2.1 Εισαγωγή.....12
2.2 Munagala & Ranade.....15
2.3 Η νέα προσέγγιση-Γρήγορος BFS.....17
2.3.1 Φάση προεπεξεργασίας(με συνθήκες τυχαιότητας).....17
2.3.2 Φάση BFS.....19
2.3.3 Ντετερμινιστική έκδοση.....22
2.4 Ημί-εξωτερικός BFS σε κατευθυνόμενα Euler-ιανά γραφήματα.....24
2.5 Συμπεράσματα.....27

3. DFS.....28
3.1 Εισαγωγή.....28
3.2 DFS χρησιμοποιώντας απλό κύκλο διαχωριστή.....29
3.3 Βρίσκοντας τον απλό κύκλο διαχωριστή.....34
3.3.1 Έλεγχος για μεγάλες όψεις.....37
3.3.2 Έλεγχος για μεγάλα υποδέντρα.....37
3.3.3 Διαμοιράζοντας ένα μεγάλο υποδέντρο.....41
3.4 Ελαττώνοντας τον DFS σε BFS.....46
3.5 Συμπεράσματα.....54

4. MST.....55
4.1 Εισαγωγή.....55
4.2 Αλγόριθμος εύρεσης ελάχιστου επικαλύπτοντος δέντρου.....55
4.3 Αλγόριθμος ελάττωσης κορυφών.....59
4.3.1 Αλγόριθμος συνένωσης κορυφών.....60
4.3.2 Αλγόριθμος υπερφάσης.....63

5. SSSP.....66
5.1 Εισαγωγή.....66
5.2 Πολλαπλών τρόπων διαχωρισμός επίπεδων γραφημάτων.....66
5.2.1 Υποδιαιρώντας ένα επίπεδο γράφημα με επικαλύπτον δέντρο με όριο ύψους.....68
5.2.2 Υποδιαιρώντας επίπεδα γραφήματα.....71
5.3 SSSP.....77

6. APSP.....82
6.1 Εισαγωγή.....82
6.2 Εξωτερικός APSP σε γενικά αραιά γραφήματα.....83
6.2.1 Tournament Tree.....83
6.2.2 Πολλαπλά tournament δέντρα και παράλληλοι SSSP υπολογισμοί.....85
6.3 AP-BFS σε γενικά γραφήματα.....87
6.3.1 Γρήγορος BFS.....87
6.3.2 Γρήγορος AP-BFS.....88
6.3.3 Πολυπλοκότητα I/O πράξεων του γρήγορου AP-BFS.....89
6.4 APSP για επίπεδα κατευθυνόμενα γραφήματα.....90
6.4.1 Εισαγωγή.....90
6.4.2 Αποδοτικός APSP αλγόριθμος για επίπεδα γραφήματα.....91

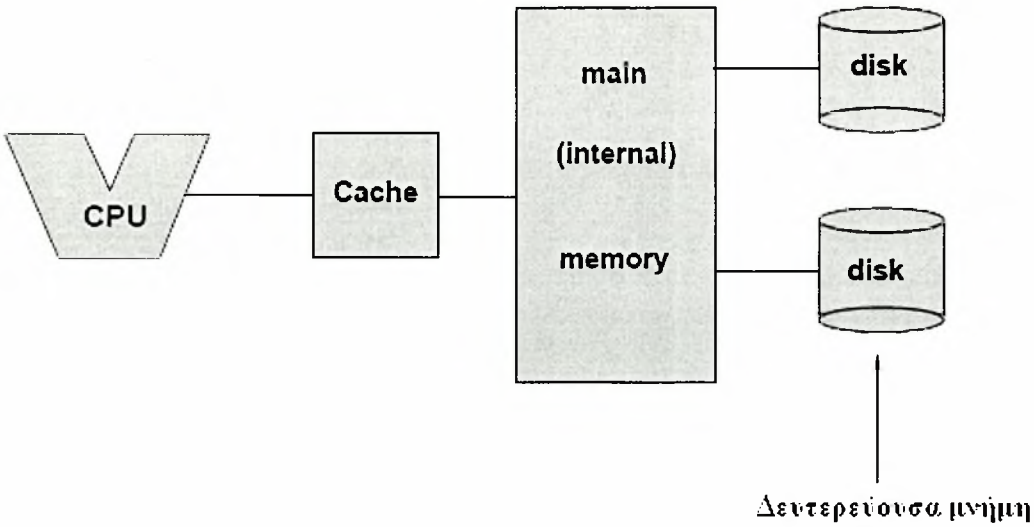
Βιβλιογραφία.....94

Εισαγωγή

Κατά τον σχεδιασμό αλγορίθμων και δομών δεδομένων ο κόσμος έχει στο νου του την ελαχιστοποίηση του χρόνου υπολογισμού όπως επίσης και του χώρου για την λύση. Θεωρητικά σχεδιάζουν αλγορίθμους για ένα μοντέλο μηχανής τυχαίας προσπέλασης (random access machine-RAM)[AHU74]. Ένα από τα κύρια χαρακτηριστικά αυτού του μοντέλου είναι το γεγονός ότι η μνήμη είναι άπειρα μεγάλη και η πρόσβαση σε διαφορετικά κελιά της μνήμης είναι μοναδιαίου κόστους. Επίσης ,στην πράξη ,τα περισσότερα λογισμικά είναι γραμμένα για ένα μοντέλο σαν το RAM όπου γίνεται η υπόθεση ότι υπάρχει μία τεράστια (σχεδόν άπειρη) μνήμη και όπου τα ξεχωριστά κελιά μνήμης μπορούν να προσπελαστούν με μοναδιαίο κόστος.

Οι σημερινές αρχιτεκτονικές υπολογιστών αποτελούνται από διαφορετικά επίπεδα μνήμης όπου κάθε επίπεδο έχει διαφορετικά χαρακτηριστικά ,μεγέθη και χρόνους πρόσβασης. Πλησιέστερα στο επεξεργαστή (CPU) είναι η μικρή ,κρυφή μνήμη (cache) που μπορεί συχνά να προσπελαύνεται σε έναν ή δύο κύκλους μηχανής. Το μέγεθος της κρυφής μνήμης κυμαίνεται μεταξύ λίγων kilobytes και κάποιων Megabytes.(ένα kilobyte ισούται με 1024 bytes ,ένα Megabyte ισούται με 1024 kilobytes).Η μνήμη cache σώζει αντίγραφα της κύριας μνήμης και έτσι επιτρέπει στην CPU να προσπελαύνει τα δεδομένα ή τον κώδικα του προγράμματος πολύ πιο γρήγορα. Η κύρια(εσωτερική) μνήμη είναι περίπου εκατό φορές πιο αργή από την μνήμη cache αλλά το μέγεθός της μπορεί να φτάσει να είναι ίσο με κάποια Gigabytes. Η κύρια μνήμη είναι η κεντρική μνήμη αποθήκευσης του υπολογιστή για τον κώδικα του προγράμματος και για τα δεδομένα του. Η μεγαλύτερη αλλά και η πιο αργή μνήμη είναι η εξωτερική-δευτερεύουσα μνήμη που σήμερα εξασφαλίζεται με τους σκληρούς δίσκους. Έτσι σε αντίθεση με όλα τα υπόλοιπα επίπεδα μνήμης ,οι σκληροί δίσκοι είναι μηχανικές συσκευές ενώ όλα τα υπόλοιπα επίπεδα μνήμης έχουν κατασκευαστεί από ηλεκτρονικές συσκευές. Οι σκληροί δίσκοι κατά ένα παράγοντα ίσο με το εκατό πιο αργοί στον χρόνο προσπέλασης από ότι η κύρια μνήμη και κάθε δίσκος μπορεί να σώσει παραπάνω από 100 Gigabytes.

Πρόχειρα υπολογίζεται πως σήμερα υπάρχει ένας παράγοντας της τάξεως των εκατό χιλιάδων έως του ενός εκατομμυρίου στην διαφορά των χρόνων προσπέλασης στα διαφορετικά επίπεδα της μνήμης. Έτσι το να υποθέτουμε ότι είναι μοναδιαίου κόστους ο χρόνος προσπέλασης στον σχεδιασμό ενός αλγορίθμου είναι αμφίβολο. [1]



[13] Στα πρόσφατα χρόνια ένας συνεχώς αυξανόμενος αριθμός εφαρμογών γραφημάτων συνδέονται με μεγάλα γραφήματα και έτσι τα δεδομένα που πρέπει να επεξεργαστούν γίνονται ολοένα και μεγαλύτερα. Ιδιαίτερα μεγάλες εφαρμογές μπορούμε να συναντήσουμε στα γεωγραφικά πληροφοριακά συστήματα (GIS)όπου για αποστολές όπως το σύστημα παρακολούθησης εδάφους της NASA(EOS) ή το space radar topography mission(SRTM) χρησιμοποιούνται δεδομένα μεγέθους της τάξεως των terabytes, στην υπολογιστική βιολογία (DNA και αμινοξεϊκές βάσεις δεδομένων), στις μηχανές αναζήτησης του διαδικτύου. Υπάρχουν πολλές ακόμα εφαρμογές που πρέπει να διαχειριστούν και να δουλέψουν με σύνολα δεδομένων που είναι τόσο μεγάλα που δεν μπορούν να χωρέσουν στην κύρια –εσωτερική μνήμη και έτσι ανήκουν στην δευτερεύουσα. Αυτά τα μεγάλα ποσά δεδομένων θέτουν διαφορετικές απαιτήσεις στους αλγορίθμους και στις δομές δεδομένων. Σε αυτές τις εφαρμογές η επικοινωνία μεταξύ κύριας και δευτερεύουσας μνήμης είναι σημαντική καθώς τα δεδομένα θα πρέπει να ανταλλάσσονται μεταξύ των δύο αυτών επιπέδων μνήμης. Αυτή η ανταλλαγή δεδομένων καλείται πράξη εισόδου/εξόδου ή I/O πράξη. Μία απλή I/O πράξη περικλείει το διάβασμα(ή το γράψιμο) ενός μπλοκ δεδομένων από (στο) δίσκο στην (από) την εσωτερική μνήμη.

Ο χρόνος που ξοδεύεται για τις I/O πράξεις είναι το κύριο εμπόδιο για τον υπολογισμό εφαρμογών που εκτελούν πολλές τέτοιες πράξεις ή για εφαρμογές που δεν είναι σχεδιασμένες να δουλεύουν καλά στην δευτερεύουσα μνήμη. Για να αντιμετωπιστεί αυτό το εμπόδιο η αλγοριθμική κοινωνία ξεκίνησε να αναπτύσσει δομές δεδομένων και αλγόριθμους που υπολογίζουν την I/O επικοινωνία. Αυτοί οι αλγόριθμοι που αποκαλούνται αλγόριθμοι εξωτερικής μνήμης ή για συντομία εξωτερικοί αλγόριθμοι μελετούν την μνήμη ως διαμοιρασμένη σε μία εσωτερική μνήμη περιορισμένου μεγέθους και σε έναν αριθμό εξωτερικών συσκευών. Τα κελία της κύριας μνήμης μπορούν να προσπελαύνονται με μοναδιαίο κόστος ενώ η πρόσβαση στην εξωτερική μνήμη είναι πιο ακριβή και πάντα μεταφέρει ένα συνεχόμενο μπλοκ δεδομένων. Στο μοντέλο της εξωτερικής μνήμης η εκτέλεση ενός

αλγορίθμου μετριέται με (i)τον αριθμό των CPU πράξεων, (ii) τον αριθμό των προσπελάσεων της εξωτερικής μνήμης ,και (iii) μετρώντας τον χρησιμοποιούμενο χώρο στην εξωτερική μνήμη.

Τα βασικά σημεία του μοντέλου μας:

Το ακριβές μοντέλο της δευτερεύουσας μνήμης και των οδηγών δίσκων που χρησιμοποιούμε είναι το [VS94b] .Είναι από τα βασικά μοντέλα εξωτερικής μνήμης και χρησιμοποιεί τις ακόλουθες παραμέτρους:

N=Ο συνολικός αριθμός αντικειμένων της εισόδου της εφαρμογής.
M=Ο αριθμός των αντικειμένων που χωράνε στην κύρια μνήμη.
B=Ο αριθμός των αντικειμένων που χωράνε σε ένα μπλοκ δίσκου.
D=Ο αριθμός των σκληρών δίσκων (σε πολλές περιπτώσεις σε αυτό το κείμενο υποθέτουμε χάριν ευκολίας και χωρίς βλάβη της γενικότητας πως D=1).

Όπου M<N και $1 \leq B \leq M/2$.Ο αριθμός των I/Os που χρειάζονται για να διαβαστούν N συνεχόμενα στοιχεία από τον δίσκο είναι $scan(N) = \Theta(\frac{N}{B})$. Ο αριθμός των I/Os που απαιτούνται για να ταξινομηθούν N στοιχεία είναι $sort(N) = \Theta(\frac{N}{B} \log_{\frac{M}{B}} \frac{N}{B})$. Για ρεαλιστικές τιμές των N,M και B έχουμε scan(N) < sort(N) << N. Άρα η διαφορά των χρόνων εκτέλεσης ενός αλγορίθμου που εκτελεί N I/O πράξεις και ενός που εκτελεί scan(N) ή sort(N) I/O πράξεις μπορεί να είναι πολύ σημαντική.

Βασικές εφαρμογές γραφημάτων:

[9] Στους αλγορίθμους που θα περιγραφούν χρησιμοποιούμε ήδη αποδεδειγμένες O(sort(N)) I/O λύσεις για κάποια βασικά προβλήματα γραφημάτων .Παραθέτουμε αυτά τα προβλήματα παρακάτω. Οι περισσότεροι από τους βασικούς υπολογισμούς απαιτούν μία πλήρης ταξινόμηση του συνόλου κορυφών V και του συνόλου ακμών E του γραφήματος G=(V,E). Για το σύνολο των κορυφών V, μία πλήρης ταξινόμηση εξασφαλίζεται με μία μοναδική απόδοση αριθμών στις κορυφές του G. Για το σύνολο των ακμών υποθέτουμε πως μία ακμή {u,w} σώζεται ως ένα ζευγάρι (u,w), u<w ,και ορίζουμε (u,w)<(x,y) για ακμές (u,w) και (x,y) στο E εάν είτε u<x ή u=x και w<y. Ονομάζουμε αυτή την ταξινόμηση λεξικογραφική ταξινόμηση του E. Μία άλλη ταξινόμηση που την ονομάζουμε ανάστροφη λεξικογραφική ταξινόμηση του E ορίζει (u,w)<(x,y) εάν είτε w<y ή w=y και u<x.

(Α) Διαχωρισμός συνόλων

Ακόμη και αν ο διαχωρισμός συνόλων δεν είναι μία πράξη γραφημάτων ,συχνά τη χρησιμοποιούμε σε ένα σύνολο κορυφών και ακμών ενός γραφήματος. Για να υπολογίσουμε των διαχωρισμό X \ Y δύο συνόλων X και Y αρχικά ταξινομούμε και το X και το Y. Μετά σαρώνουμε τις δύο παραγόμενες ταξινομημένες λίστες ταυτοχρόνως με παρόμοιο τρόπο που θα γινόταν αν ήταν ενωμένες σε μία ταξινομημένη λίστα .Ωστόσο στοιχεία του Y δεν αντιγράφονται στην λίστα εξόδου και ένα στοιχείο του X αντιγράφεται μόνο αν δεν είναι ίδιο με το τρέχον στοιχείο στο Y. Αυτό παίρνει καθαρά O(sort(N)) I/O όπου N=|X|+|Y|.

(Β) Αφαίρεση διπλότυπων

Δοθείσας μίας λίστας $X<x_1,\ldots,x_N>$ με κάποιες εισαγωγές που ενδεχομένως να εμφανίζονται παραπάνω της μίας φοράς η αφαίρεση διπλότυπων υπολογίζει μία λίστα $Y=<y_1,\ldots,y_q>$ τέτοια ώστε $\{x_1,\ldots,x_N\}=\{y_1,\ldots,y_q\}$. Η λίστα Y περιέχει την πρώτη εμφάνιση όλων των στοιχείων του X σε ταξινομημένη σειρά. Για να υπολογίσουμε την Y σε O(sort(N)) I/Os σαρώνουμε την X και αντικαθιστούμε κάθε στοιχείο x_i με ένα ζευγάρι (x_i,i). Ταξινομούμε την παραγόμενη λίστα X' λεξικογραφικά. Τώρα σαρώνουμε την X' και αποβάλλουμε για κάθε x ,όλα τα ζευγάρια που έχουν το x σαν πρώτη συνιστώσα τους εκτός από το πρώτο τέτοιο ζευγάρι. Η λίστα Y μπορεί τώρα να επιτευχθεί ταξινομώντας τα εναπομείναντα ζευγάρια (x,y) βάσει των δεικτών τους y, και σαρώνοντας την παραχθείσα λίστα για να αντικαταστήσουμε κάθε ζευγάρι (x,y) με το μοναδικό στοιχείο x.

(Γ) Υπολογισμός των προσκείμενων ακμών

Δοθέντος ενός συνόλου V κορυφών και ενός συνόλου E ακμών ,η πράξη του υπολογισμού προσκείμενων ακμών υπολογίζει ένα σύνολο E' ακμών $\{u,w\}\in E$ τέτοιες ώστε $u \in V$ και $w \notin V$. Για να υπολογίσουμε το E' σε O(sort(N)) I/Os όπου N=|V|+|E|, ταξινομούμε το V με αύξουσα σειρά και το E με λεξικογραφική σειρά. Σαρώνουμε το V και το E και μαρκάρουμε κάθε ακμή στο E που έχει το πρώτο της άκρο στο V. Ταξινομούμε το E με ανάστροφη λεξικογραφική σειρά και σαρώνουμε το V και το E ξανά για να μαρκάρουμε κάθε ακμή στο E που έχει το δεύτερό της άκρο στο V. Τέλος σαρώνουμε το E και αφαιρούμε όλες τις ακμές που δεν έχουν μαρκαριστεί ή έχουν μαρκαριστεί δυο φορές.

(Δ) Αλγόριθμοι για λίστες και για δέντρα

Δοθείσας μίας λίστας ως μια μη ταξινομημένη ακολουθία ακμών {(u,next(u))} η ταξινόμησή της είναι το πρόβλημα του καθορισμού για κάθε κορυφή u της λίστας ,του αριθμού των ακμών από το u μέχρι το τέλος της λίστας. Η ταξινόμηση λίστας μπορεί να λυθεί σε O(sort(N)) I/Os χρησιμοποιώντας τεχνικές όμοιες με αυτές που χρησιμοποιούνται στους αποδοτικούς αλγορίθμους ταξινόμησης παράλληλων λιστών. Χρησιμοποιώντας την ταξινόμηση λίστας και τεχνικές PRAM ,O(sort(N)) I/O αλγόριθμοι μπορούν επίσης να αναπτυχθούν στα περισσότερα προβλήματα των δέντρων, συμπεριλαμβανομένων, του υπολογισμού του κύκλου του Euler, μετρήσεων BFS και DFS,και ερωτημάτων λιγότερων κοινών προγόνων. Κάθε υπολογισμός που μπορεί να δηλωθεί σαφώς με διαπέραση «επίπεδου-επίπεδου» ενός δέντρου ,όπου η τιμή της κάθε κορυφής είναι υπολογισμένη είτε από τις τιμές των παιδιών της είτε από την τιμή του πατέρα της ,μπορεί επίσης να εκτελεστεί σε O(sort(N)) I/Os.

(Ε) Αλγόριθμοι για επίπεδα γραφήματα

Ακόμα και αν κανένας O(sort(N)) I/O αλγόριθμος για BFS και DFS σε επίπεδα γραφήματα δεν έχει αναπτυχθεί υπάρχουν O(sort(N)) I/O λύσεις για πολλά αλλά προβλήματα στα επίπεδα γραφήματα ,όπως ο υπολογισμός συνεκτικών και δισυνεκτικών συνιστωσών, επικαλυπτόντων δέντρων και ελάχιστων επικαλυπτόντων δέντρων. Όλοι αυτοί οι αλγόριθμοι είναι βασισμένοι στη σύγκριση ακμών ,όμοια με τους PRAM αλγορίθμους αυτών των προβλημάτων. Κάνουμε εκτεταμένη χρήση αυτών των αλγορίθμων στους DFS αλγορίθμους μας.

Κεφάλαιο 1

Αναπαράσταση γραφημάτων

1.1 Εισαγωγή:

Η αναπαράσταση γραφημάτων έγινε αποδέκτης μεγάλης προσοχής από την αρχή της μελέτης της θεωρίας των υπολογιστών .Αυτό που γενικά απαιτείται είναι μία καλή και αποδοτική αναπαράσταση: καλή για την αποδοτικότητα των αλγορίθμων και αποδοτική και από την άποψη του χώρου για την αποθήκευση των δεδομένων και για τον χρόνο υπολογισμού που χρειάζεται για την παραγωγή της αναπαράστασης.

Στην εσωτερική μνήμη υπάρχουν δύο κύριες δομές για την αναπαράσταση γραφημάτων, ο πίνακας γειτνιάσεως και η λίστα γειτνιάσεως. Υπάρχει και μία τρίτη ,η αναπαράσταση με πίνακα ακμών αλλά υστερεί από τις δύο προηγούμενες και παρουσιάζει λιγότερο ενδιαφέρον.

[1] Στην εξωτερική μνήμη οι αλγόριθμοι γραφημάτων μπορούν να κατηγοριοποιηθούν είτε σε πλήρως-εξωτερικούς είτε σε ημί-εξωτερικούς. Οι πλήρως-εξωτερικοί αλγόριθμοι γραφημάτων υποθέτουν ότι δεν είναι δυνατό να σώσουμε πληροφορία μεγέθους $\Theta(|V|)$ ή $\Theta(|E|)$ στην εσωτερική – κύρια μνήμη. Στην πράξη όμως οι πλήρως-εξωτερικοί αλγόριθμοι δεν είναι πολύ χρήσιμοι. Ακόμα και για πολύ μεγάλα γραφήματα όπως τα γραφήματα για τηλεφωνικές γραμμές είναι δυνατό να σώσουμε πληροφορία μεγέθους $O(|V|)$ στην εσωτερική μνήμη και να έχουν έναν αποδοτικό ημί-εξωτερικό αλγόριθμο.

1.2 Ανάπτυξη συστημάτων για την αναπαράσταση δικτύων και γραφημάτων:

Με την ανάπτυξη των υπολογιστών, της τεχνολογίας δικτύων και των αντίστοιχων λειτουργικών συστημάτων, μία πληθώρα προϊόντων ,αρχικά για το DOS και έπειτα για τα WINDOWS και τα LINUX, αναπτύχτηκαν για την βελτιστοποίηση των δικτύων και των γραφημάτων. Κάποια γνωστά συστήματα που αφορούν αυτά τα προβλήματα είναι τα :GIDEN, GRIN, GOBLIN, LEDA, WinQBS.

[3] Το GIDEN προσανατολίζεται στην βελτιστοποίηση δικτύων .Έχει μία πολύ καλά ανεπτυγμένη γραφικά διεπαφή χρήστη. Το GRIN κατευθύνεται κυρίως στην επίλυση γραφικών προβλημάτων και για μόνο ένα περιορισμένο σύνολο προβλημάτων σε δίκτυα. Η γραφική διεπαφή σε αυτό το σύστημα είναι καλή. Η πινακοειδής αναπαράσταση πραγματοποιείται με πίνακες γειτνίασης ,όπου μόνο μία παράμετρος μπορεί να οριστεί ,το οποίο όμως δεν αρκεί για την αναπαράσταση δικτύων. Το GOBLIN είναι ένα πολύ δυνατό εργαλείο για την επίλυση προβλημάτων δικτύων. Αρχικά αναπτύχτηκε για τα LINUX και αργότερα μεταφέρθηκε και στο περιβάλλον των WINDOWS. Έχει επίσης μία καλή γραφική διεπαφή για τον χρήστη. Αυτό που προσφέρει το LEDA είναι μία ολοκληρωμένη συλλογή καλά υλοποιημένων δομών δεδομένων. Το WinQBSυποστηρίζει ένα σύστημα το οποίο προσφέρει έναν πίνακα δυναμικών εργαλείων που βοηθούν την επίλυση προβλημάτων και καθορίζουν επιτυχημένες επιχειρηματικές αποφάσεις. Τα δεδομένα των γραφημάτων θα πρέπει να εισάγονται δομημένα σε έναν πίνακα γειτνίασης.

Η αναπαράσταση των δικτύων είναι πολύ σημαντική για την επιτυχή επίλυση πολλών προβλημάτων σε πραγματικές συνθήκες και έτσι ο κάθε ένας που θα πρέπει να πάρει μία απόφαση για το δίκτυο θα πρέπει να γνωρίζει την αναπαράσταση του δικτύου ώστε να καταλήξει σε ποιο μοντέλο και σε ποιες μεθόδους επίλυσης πάνω σε αυτό να στραφεί.

Ένα γράφημα G είναι μία δομή(ένα σύνολο) που ορίζεται ως G=(V,E), όπου V είναι ένα πεπερασμένο σύνολο κορυφών(κόμβων) με δείκτες φυσικούς αριθμούς από το 1 μέχρι το n, E είναι ένα πεπερασμένο σύνολο ζευγών κορυφών ,που ανήκουν στο V.Αυτά τα ζεύγη ονομάζονται ακμές, |E|=m. Εάν οι ακμές δεν είναι ταξινομημένες ,δηλαδή αν για $i \in V, j \in V$, η ακμή (i,j) και η (j,i) ανήκουν και οι δύο στο E, τότε το γράφημα ονομάζεται μη κατευθυνόμενο. Ενώ αν κάθε ακμή (i,j) είναι ταξινομημένη τότε οι κόμβοι ορίζονται να αρχίζουν από την i και να τελειώνουν στην j. Σε αυτή την περίπτωση το γράφημα ονομάζεται κατευθυνόμενο ή διάγραμμα. Οι κατευθυνόμενες ακμές συχνά ονομάζονται τόξα. Ένα μονοπάτι είναι μία ακολουθία ακμών (i_1,i_2), (i_2,i_3),…, (i_{k-1},i_k) που ανήκουν στο E και οι οποίες συνδέουν τους κόμβους i_1 και i_k που ανήκουν στο V. Ένα γράφημα είναι συνδεδεμένο εάν υπάρχει ένα μονοπάτι μεταξύ οποιωνδήποτε δύο κορυφών του. Σε διαφορετική περίπτωση ονομάζεται μη συνδεδεμένο. Εάν μία ή και περισσότερες παράμετροι ορίζονται πάνω στα τόξα του γραφήματος G τότε το γράφημα καλείται δίκτυο. Αυτές οι παράμετροι μπορεί να είναι βάρος ακμής, χωρητικότητα, κέρδος, κόστος κ.α.

Τα γραφήματα και τα δίκτυα αναπαριστούνται επισήμως μαθηματικά είτε με πίνακες γειτνίασης είτε με πίνακες ακμών.

Ο πίνακας γειτνίασης είναι ένας nxn πίνακας. Κάθε μη μηδενικό στοιχείο του πίνακα φανερώνει την ύπαρξη μίας ακμής μεταξύ των κορυφών της αντίστοιχης γραμμής και κορυφής. Εάν η ακμή χαρακτηρίζεται και από κάποια παράμετρο, όπως βάρος ,χωρητικότητα, κόστος κτλ , τότε το

αντίστοιχο στοιχείο του πίνακα θα είναι μία εγγραφή που θα περιέχει αυτή την τιμή της παραμέτρου. Ο πίνακας γειτνίασης ενός μη κατευθυνόμενου γραφήματος είναι συμμετρικός, ενώ των κατευθυνόμενων δεν είναι.

Ο πίνακας ακμών είναι ένας nxm πίνακας. Σε αυτόν τον πίνακα το στοιχείο (i,j) είναι ίσο με 1 εάν το i είναι η αρχή του τόξου (i,j) και το στοιχείο (j,i) είναι ίσο με -1.

Αυτοί οι πίνακες είναι πολύ σημαντικοί καθώς με πράξεις πινάκων πάνω σε αυτούς πολλοί άλλοι πίνακες , που αφορούν διαφορετικές ιδιότητες των γραφημάτων μπορούν εύκολα να παραχθούν, όπως για παράδειγμα κύκλοι, επικαλύπτοντα δέντρα κτλ.

Για τα συνδεδεμένα γραφήματα συνήθως ισχύει $m \geq n$. Εάν $m = n^2$ τότε το γράφημα ονομάζεται πλήρως συνεκτικό γράφημα. Οι πίνακες σε πραγματικές εφαρμογές είναι αραιοί και για να σωθούν απαιτούν πολύ εξωτερικό χώρο αποθήκευσης. Για παράδειγμα ,ένα γράφημα 100 κόμβων εάν αναπαρασταθεί με πίνακα γειτνίασης απαιτεί 10.000 εγγραφές, οι περισσότερες από τις οποίες θα είναι μηδέν. Σε αυτή την περίπτωση του παραδείγματος εάν η αναπαράσταση γίνει με πίνακα ακμών τότε θα είναι ακόμα χειρότερα ,γιατί το m είναι μεγαλύτερο του n και εκτός αυτού και ο προσδιορισμός των παραμέτρων είναι πιο δύσκολος.

Η αναπαράσταση των γραφημάτων στους υπολογιστές είναι κάτι πιο πολύπλοκο και εξαρτάται κυρίως από το μέγεθος και την πυκνότητα των πινάκων. Η αναπαράσταση με μία λίστα που θα στηρίζεται στην διασύνδεση των κόμβων αποδεικνύεται ο καλύτερος τρόπος για την εξωτερική αποθήκευση ενός δικτύου καθώς από αυτή μπορούμε εύκολα να αποκτήσουμε εάν χρειαστεί και τον πίνακα γειτνίασης και των πίνακα ακμών, και έτσι και όποιον άλλον πίνακα μας είναι απαραίτητος. Αυτή η πράξη έχει πολυπλοκότητα O(n) και σε χρόνο και σε μνήμη.

Η λίστα γειτνίασης έχει ως εξής:

Μελετούμε ένα δίκτυο n κόμβων ,άρα n γραμμών. Κάθε στοιχείο της γραμμής i περιέχει μία εγγραφή της ακόλουθης δομής $[(j),p_{(i,j)}(1)\ldots p_{(i,j)}(k)]$,όπου j είναι ένας κόμβος που συνδέεται με ένα τόξο (που έχει ως αρχή το i και τέλος το j) με το i και $p_{(i,j)}(.)$ είναι οι παράμετροι του τόξου. Εάν έχουμε στη διάθεση μας την αναπαράσταση του γραφήματος σε λίστα γειτνίασης κάθε πίνακας μπορεί να κατασκευαστεί.

1.3 LEDA-SM:

[2] Η LEDA-SM πρόκειται για μία C++ βιβλιοθήκη που χρησιμοποιείται για υπολογισμούς στην δευτερεύουσα μνήμη κατά έναν αποδοτικό και εύκολο τρόπο. Θα εξετάσουμε τις δομές δεδομένων που είναι άμεσα διαθέσιμες στην LEDA-SM. Θεωρητικά τα όρια των πράξεων εισόδου/εξόδου θα δίνονται για το κλασσικό μοντέλο εξωτερικής μνήμης το [VS94b], στο οποίο το M συμβολίζει το μέγεθος της κύριας μνήμης ,το B το μέγεθος ενός μπλοκ δίσκου και το N το

μέγεθος του γραφήματος εισόδου. Προς χάρη της απλότητας υποθέτουμε πως το D,που συμβολίζει τον αριθμό των δίσκων ,είναι ίσο με ένα.

Στην LEDA-SM υπάρχουν κλάσεις για την κατασκευή και πινάκων διανυσμάτων(διαστάσεων δηλαδή 1xn) και πινάκων διαστάσεων nxm.

Πίνακες: Οι πίνακες είναι η πιο ευρέως διαδεδομένη δομή δεδομένων στην κύρια μνήμη. Το κύριο όμως μειονέκτημα της δομής των πινάκων όταν αυτή χρησιμοποιείται στην εξωτερική μνήμη είναι το γεγονός ότι δεν μπορεί να ελεγχθεί η σελιδοποίηση. Η δική μας δομή δεδομένων για πίνακες στην εξωτερική μνήμη αποτελείται από μία συνεχόμενη συλλογή μπλοκ δίσκου και μία δομή δεδομένων εσωτερικής μνήμης σταθερού μεγέθους που υλοποιεί μια πολύ-τμηματική κρυφή μνήμη (cache). Όταν θέλουμε να προσπελάσουμε έναν στοιχείο i,j του πίνακα A ,πρώτα ελέγχουμε αν το αντικείμενο A[i][j] υπάρχει στην κρυφή μνήμη. Εάν δεν υπάρχει τότε φορτώνουμε ένα μπλοκ B αντικειμένων ,συμπεριλαμβανομένου και του A[i][j] στην κρυφή μνήμη cache. Καθώς η κρυφή μνήμη έχει σταθερό μέγεθος ,τότε πιθανόν να χρειάζεται να αφαιρέσουμε ένα μπλοκ από αυτή. Αυτή η διαδικασία γίνεται με τη χρήση ενός αλγορίθμου αντικατάστασης σελίδας. Υποστηρίζονται αρκετές τεχνικές αντικατάστασης σελίδων όπως η LRU(least recently used),η τυχαία κ.α. Ο χρηστής μπορεί επίσης να υλοποιήσει την δική toy τεχνική αντικατάστασης σελίδας. Η κρυφή μνήμη έχει πολλά πλεονεκτήματα: το σταθερό της μέγεθος επιτρέπει τον έλεγχο την χρήσης της εσωτερικής μνήμης για τον εξωτερικό πίνακα. Ο τρόπος αντικατάστασης των μπλοκ επιτρέπει την σάρωση ενός εξωτερικού πίνακα μέσα σε ιδανικό αριθμό πράξεων εισόδου/εξόδου και επιπλέον ,καθώς η cache είναι πολύ-τμηματική, είναι πιθανό να ενθέσουμε διαφορετικές περιοχές του εξωτερικού πίνακα χρησιμοποιώντας διαφορετικά τμήματα της cache για κάθε περιοχή.

Εάν με G συμβολίσουμε ένα κατευθυνόμενο γράφημα, με V το σύνολο των κόμβων του και $E \subseteq VxV$ το σύνολο των ακμών του τότε ορίζουμε με |V| το πλήθος του συνόλου των κόμβων του γραφήματος και με |E|, αντίστοιχα ,το πλήθος του συνόλου των ακμών του. Συχνά για την αναπαράσταση γραφημάτων χρησιμοποιούνται οι λίστες γειτνίασης. Αυτή η αναπαράσταση αποτελείται από |V| λίστες. Η καθεμία από αυτές τις λίστες έχει την ετικέτα x όταν σώζει όλες τις ακμές που είναι γειτονικές στην x (μία ακμή είναι γειτονική στις κορυφές των άκρων της). Στην LEDA-SM χρησιμοποιείται η αναπαράσταση γραφημάτων με λίστα γειτνίασης.

1.4 Τεχνική ανάθεσης ετικετών:

[6] Το πώς θα αναπαραστήσουμε ένα γράφημα στην μνήμη είναι ένα βασικό πρόβλημα δομών δεδομένων. Στις συνήθεις αναπαραστάσεις γραφημάτων σώζονται ρητώς όλες οι κορυφές και

όλες οι ακμές του γραφήματος. Τα ονόματα που εκχωρούνται στις κορυφές χρησιμοποιούνται μόνο για να κωδικοποιήσουν τις ακμές και δεν συμβολίζουν τίποτα για την δομή του ίδιου του γραφήματος και έτσι είναι χαμένος χώρος .Παρακάτω θα παρουσιάσουμε ένα γενικό πλαίσιο εργασίας ανάθεσης ετικετών σε κάθε γράφημα έτσι ώστε ή γειτνίαση μεταξύ δύο οποιωνδήποτε δοθέντων κορυφών να μπορεί να ελέγχεται σε σταθερό χρόνο. Με το σχήμα της ανάθεσης ετικετών εκχωρείται σε κάθε κορυφή x ενός γενικού γραφήματος μία ετικέτα $O(\delta(x)\log^3 n)$ bit, όπου n είναι ο αριθμός των κορυφών και δ(x) είναι ο βαθμός της κορυφής x. Ο έλεγχος γειτνίασης μπορεί να εκτελεστεί σε 5 βήματα και η ανάθεση ετικετών γίνεται σε πολυωνυμικό χρόνο. Αυτή η αναπαράσταση αντιπαραβάλλεται αυστηρά με τις συνήθεις αναπαραστάσεις των γραφημάτων, δηλαδή με την αναπαράσταση με πίνακα γειτνίασης και με λίστα γειτνίασης ,οι οποίες απαιτούν ετικέτες O(n log n)bit ανά κορυφή και σταθερό χρόνο για τον έλεγχο γειτνίασης ,και ετικέτες O(δ(x) log n) bit ανά κορυφή και O(log δ(x)) βήματα για τον έλεγχο γειτνίασης ,αντίστοιχα. Επιπλέον στο σχήμα ετικετών δεν χρειάζεται η χρήση δεικτών.

Η αναπαράσταση με το σχήμα ετικετών είναι ευρέως αποδεκτή ως ένας καλός τρόπος αναπαράστασης καθώς οι δομές δεδομένων προσδίδουν στον εαυτό τους μία συνεχόμενη αποθήκευση που δεν απαιτεί δείκτες ,έτσι εξασφαλίζεται ένας τρόπος αποθήκευσης των δεδομένων χωρίς σπατάλη χώρου για τους δείκτες. Επιπλέον οι αλγόριθμοι είναι πιο εύκολο να υλοποιηθούν και συχνά πιο αποδοτικοί. Επιπλέον αυτός ο τρόπος αναπαράστασης ικανοποιεί την ιδιότητα της τοπικότητας ,δηλαδή τα δεδομένα σώζονται ανά κόμβο, έτσι εξασφαλίζεται και μία αποδοτική αναπαράσταση γραφημάτων για κατανεμημένα συστήματα και για αποθήκευση στη δευτερεύουσα-εξωτερική μνήμη.

Αρκετοί μελετητές ασχολήθηκαν με αυτό το πρόβλημα. Ο Breuer και ο Folkman μελέτησαν το πρόβλημα της ανάθεσης ετικετών στις κορυφές έτσι ώστε η γειτνίαση των κορυφών να καθορίζεται από την απόσταση Hamming μεταξύ των ετικετών. Το σχήμα τους όμως είναι πολύ περιορισμένο για γενικά γραφήματα και το μήκος των ετικετών μπορεί να είναι O(n log n). Ο Turan και ο Kannan μελέτησαν το πρόβλημα της αναπαράστασης γραφημάτων όσο πιο λακωνικά γινόταν .Ωστόσο ,απέδωσαν μία αποδοτική αναπαράσταση της λίστας γειτνίασης του γραφήματος μόνο για περιορισμένες τάξεις γραφημάτων. Το ίδιο πρόβλημα μελετήθηκε για την δρομολόγηση μηνυμάτων σε κατανεμημένα συστήματα. Μελετήθηκε ,δηλαδή, πώς να σώζεται η διαδρομή της πληροφορίας καθώς μεταφέρεται ανάμεσα στις κορυφές ενός κατανεμημένου δικτύου. Όμως το πρόβλημα πάλι λύθηκε για μόνο για μια περιορισμένη τάξη γραφημάτων, όπως το δέντρα, οι δακτύλιοι ,τα πλήρη γραφήματα, τα επίπεδα γραφήματα ή για ορισμένες τοπολογίες δικτύων όπως οι υπερκύβοι.

Σε αυτή την ενότητα όμως προτείνουμε ένα σχήμα απόδοσης ετικετών σε k βήματα που στηρίζεται σε ένα σχήμα k αμοιβαία παραγόμενων συναρτήσεων, που η καθεμία αποτιμάται σε ένα βήμα. Με σκοπό τον έλεγχο γειτνίασης οι συναρτήσεις αποτιμούνται διαδοχικά, έτσι η γειτνίαση μπορεί να ελεγχεί σε k βήματα. Με αυτό το σενάριο πετυχαίνουμε τα ακόλουθα αποτελέσματα:

1. Ένα σχήμα ανάθεσης ετικετών σε 3 βήματα για συμμετρικά διμερή γραφήματα βαθμού δ. Αναθέτει ετικέτα $O(\delta \log^2 n)$ bit σε κάθε κορυφή.
2. Ένα σχήμα ανάθεσης ετικετών σε 5 βήματα για γενικά γραφήματα. Αναθέτει ετικέτα $O(\delta(x) \log^3 n)$ bit σε κάθε κορυφή x.

Σημειώνουμε πως διμερές ονομάζεται ένα γράφημα του οποίου οι κορυφές μπορούν να διαιρεθούν σε δύο ανεξάρτητα σύνολα Α και Β τέτοια ώστε κάθε ακμή του γραφήματος να ενώνει μία κορυφή του Α με μία κορυφή του Β.

Η τακτική απόδειξης προέρχεται έτσι: i) το πρόβλημα ελέγχου γειτνίασης για ένα γενικό γράφημα ελαττώνεται σε ένα ισοδύναμο πρόβλημα πάνω σε ένα γενικό διμερές γράφημα. ii) το ίδιο πρόβλημα για ένα γενικό διμερές γράφημα ελαττώνεται σε ένα ισοδύναμο πρόβλημα για μία συλλογή συμμετρικών διμερών γραφημάτων. iii) το πρόβλημα για ένα συμμετρικό διμερές γράφημα ελαττώνεται έπειτα σε ένα ισοδύναμο πρόβλημα για μία συλλογή διμερών γραφημάτων με όριο βαθμού.

Χωρίς βλάβη της γενικότητας υποθέτουμε πως το γράφημα G=(V,E) είναι συνεκτικό, διαφορετικά όλα τα αποτελέσματα εφαρμόζονται σε κάθε μία συνεκτική συνιστώσα ξεχωριστά χωρίς να υπάρχει καμία αλλαγή στα όρια πολυπλοκότητας.

Κεφάλαιο 2

BFS

2.1 Εισαγωγή:

[10] Η αναζήτηση κατά πλάτος είναι μία βασική τεχνική εξερεύνησης γραφημάτων. Σε αυτό το κεφάλαιο θα παρουσιάσουμε τον πρώτο αλγόριθμο εξωτερικής μνήμης για αραιά μη κατευθυνόμενα γραφήματα με πολυπλοκότητα πράξεων εισόδου/εξόδου που αυξάνεται πιο αργά απ' ότι το μέγεθος του προβλήματος. Ο προηγούμενος καλύτερος αλγόριθμος απαιτεί $\Theta(n + (n+m)/(DB) \log_{M/B}((n+m)/B))$ I/Os σε ένα γράφημα με n κόμβους και m ακμές. Εμείς όμως θα παρουσιάσουμε την νέα προσέγγιση που είναι αρκετά απλή και πρακτική .Επίσης θα παρουσιάσουμε και έναν βελτιωμένο ημί-εξωτερικό αλγόριθμο για αναζήτηση κατά πλάτος και σε κατευθυνόμενα Euler-ιανά γραφήματα.[1] Τον όρο ημί-εξωτερικός αλγόριθμος τον χρησιμοποιούμε για να δηλώσουμε πως υποθέτουμε ότι είτε η πληροφορία των κόμβων ,είτε η πληροφορία των ακμών μπορεί να σωθεί στην κύρια μνήμη, ενώ με τον όρο εξωτερικός αλγόριθμος που χρησιμοποιούμε στις περισσότερες περιπτώσεις δηλώνουμε πως ούτε η πληροφορία των κόμβων ούτε των ακμών δεν μπορεί να σωθεί στην κύρια μνήμη αλλά μόνο στην εξωτερική.

Η αναζήτηση κατά πλάτος (BFS) αναλύει το γράφημα εισόδου των n κόμβων και των m ακμών σε το πολύ n επίπεδα όπου το επίπεδο i αποτελείται από όλους τους κόμβους στους οποίους μπορούμε να φτάσουμε από την πηγή s μέσω ενός μονοπατιού i ακμών. Ο BFS χρησιμοποιείται ως υπορουτίνα σε πολλούς αλγόριθμους γραφημάτων και είναι επίσης θεμελιώδης για τους υπολογισμούς συντομότερων μονοπατιών. Σε αυτό το κεφάλαιο θα εστιάσουμε την προσοχή μας στην αναζήτηση κατά πλάτος γενικών μη κατευθυνόμενων γραφημάτων και αραιών κατευθυνόμενων Euler-ιανών γραφημάτων(δηλαδή γραφημάτων με έναν κύκλο που διαπερνά

κάθε ακμή ακριβώς μία φορά.) Σε γραφήματα n κόμβων και m ακμών η ημί-εξωτερική μνήμη υποθέτει c $n \leq M < m$ για κάποια κατάλληλη σταθερά $c \geq 1$.

[17] Ανακαλούμε τον στάνταρ BFS αλγόριθμο εσωτερικής μνήμης χρόνου O(n+m) που επισκέπτεται τις κορυφές του γραφήματος εισόδου G μία προς μία διατηρώντας σε μία ουρά FIFO Q τους κατάλληλους υποψήφιους κόμβους για την επόμενη κορυφή που θα επισκεφτεί. Όταν μία κορυφή υ εξάγεται από την ουρά Q τότε εξετάζεται το σύνολο των γειτόνων της στο G με σκοπό την ανανέωση της Q. Στην Q εισάγονται οι γείτονες κόμβοι που δεν έχουν επισκεφτεί ακόμα. Τρέχοντας αυτόν τον αλγόριθμο στην εξωτερική μνήμη θα έχουμε ως αποτέλεσμα Θ(n+m)I/Os. Σε αυτό το όριο ο όρος Θ(n) έρχεται ως αποτέλεσμα των ανοργάνωτων προσπελάσεων στις λίστες γειτνίασης και ο όρος Θ(m) προκαλείται από τα m ανοργάνωτα ερωτήματα για να ανακαλύψουμε πότε οι γείτονες κόμβοι έχουν ήδη επισκεφτεί.

[7] Ο καλύτερος γνωστός εξωτερικός αλγόριθμος BFS ,ο Munagala και Ranade αλγόριθμος, υπερνικά το τελευταίο πρόβλημα και απαιτεί Θ(n+sort(n+m)) I/Os σε γενικά μη κατευθυνόμενα γραφήματα. Ωστόσο ο Munagala και Ranade αλγόριθμος χρειάζεται μία πράξη εισόδου/εξόδου για κάθε κόμβο.

Προηγούμενα αποτελέσματα:

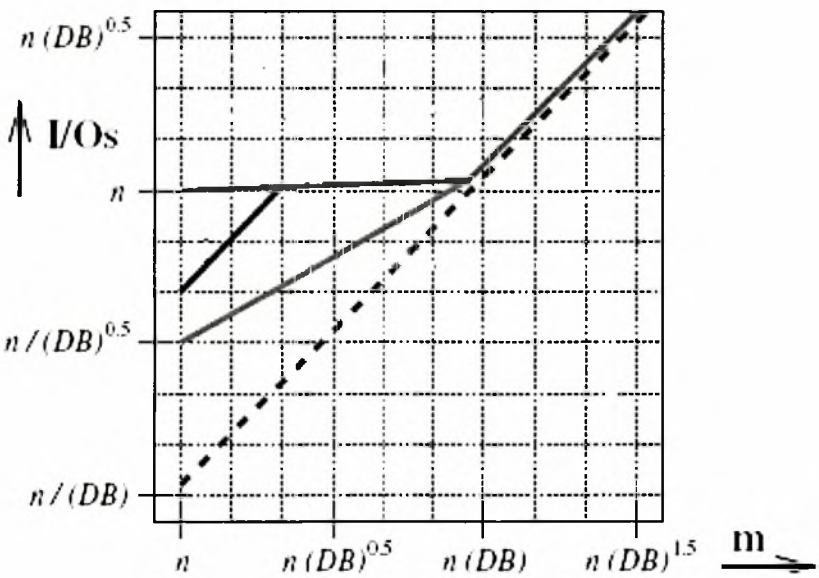
Ο μέχρι στιγμής γρηγορότερος BFS αλγόριθμος για γενικά μη κατευθυνόμενα γραφήματα χρειάζεται O(n+sort(n))I/Os. Το καλύτερο γνωστό όριο για εξωτερικό αλγόριθμο BFS είναι $O(\min\{n+\frac{n}{M}\frac{m}{DB}, (n+\frac{m}{DB})\log_2\frac{n}{DB}\})$I/Os το οποίο αποφέρει έναν $O(n+\frac{m}{DB})$ I/O BFS αλγόριθμο ημί-εξωτερικής μνήμης.

Είναι γνωστοί, ταχύτεροι αλγόριθμοι για ειδικούς όμως τύπους γραφημάτων. O(sort(n)) πράξεις εισόδου/εξόδου είναι αρκετές για να επιλύσουμε το πρόβλημα του εξωτερικού BFS σε δέντρα, πλέγματα, εξωτερικά επίπεδα γραφήματα και γραφήματα με δέντρο με όριο πλάτους. Ο Maheshwari και ο Zeh πρότειναν αλγόριθμος με ευνοϊκή πολυπλοκότητα πράξεων εισόδου/εξόδου σε επίπεδα γραφήματα. Ειδικότερα έδειξαν πώς να κάνουμε αναζήτηση κατά πλάτος σε επίπεδα γραφήματα χρησιμοποιώντας O(sort(n)) I/Os. Το χαμηλότερο όριο για τον BFS είναι $\Omega(\min\{n,\text{sort}(n)\}+\frac{n+m}{DB})$ I/Os και έπεται από το αντίστοιχο χαμηλότερο όριο του προβλήματος ταξινόμησης λίστας.

Νέα αποτελέσματα:

Θα παρουσιάσουμε έναν καινούριο BFS αλγόριθμο εξωτερικής μνήμης για μη κατευθυνόμενα γραφήματα .Παρουσιάζεται με δύο εκδόσεις (έκδοση με συνθήκες τυχαιότητας και

ντετερμινιστική έκδοση) και απαιτεί $O(\sqrt{\frac{n(n+m)}{DB}} + sort(n+m))$ I/Os (αναμενόμενη και χειρότερη περίπτωση αντίστοιχα). Για αραιά γραφήματα με m=O(n) και με ρεαλιστικές παραμέτρους μηχανής στο παραπάνω όριο ο όρος $O(\sqrt{\frac{n(n+m)}{DB}})$θα είναι ο κύριος, σε αυτή την περίπτωση η προσέγγισή μας βελτιώνει την πολυπλοκότητα σε πράξεις εισόδου/εξόδου του προηγούμενου καλύτερου αλγορίθμου κατά έναν παράγοντα $\Omega(\sqrt{DB})$. Πιο γενικά ο νέος αλγόριθμος είναι πολύ πιο καλός από τον παλιό αλγόριθμο για $m=o(\frac{DBn}{\log\frac{M}{B}n})$. Για πυκνά γραφήματα και οι δύο προσεγγίσεις απαιτούν O(sort(n+m)) I/O πράξεις. Μετά από μία μετατροπή πετυχαίνουμε έναν βελτιωμένο αλγόριθμο BFS για ημί-εξωτερική μνήμη για αραιά κατευθυνόμενα Euler-ιανά γραφήματα που πετυχαίνει $O((n+m)/(DB)^{1/3} + sort(n+m)\log n)$ I/Os. Μία παρουσίαση της σύγκρισης των BFS αλγορίθμων φαίνεται στην παρακάτω εικόνα.



Ημι-εξωτερικά κατευθυνόμενα Euler-ιανά γραφήματα
Καλύτερος προηγούμενος εξωτερικός αλγόριθμος
Νέος αλγόριθμος
sort(n+m)

Εικόνα 2.1: Σύγκριση: Παρουσίαση των πράξεων εισόδου/εξόδου των νέων BFS αλγορίθμων.

2.2 Ο αλγόριθμος των Munagala και Ranade:

[16] Θα παρουσιάσουμε τον αλγόριθμο των Munagala και Ranade ή για συντομία τον MR_BFS. Επικεντρώνουμε την προσοχή μας στον υπολογισμό του BFS επιπέδου του κάθε κόμβου υ, δηλαδή τον ελάχιστο αριθμό ακμών που χρειάζονται για να φτάσουμε από την πηγή στην υ. Για μη κατευθυνόμενα γραφήματα το αντίστοιχο BFS δέντρο ή οι αντίστοιχοι BFS αριθμοί μπορούν να αποκτηθούν αποδοτικά: Αποδεικνύεται πως κάθε μία από τις επόμενες μετατροπές μπορεί να γίνει χρησιμοποιώντας O(sort(n+m)) I/Os : BFS αριθμοί → BFS δέντρο → BFS επίπεδα → BFS αριθμοί .

Η μετατροπή BFS αριθμοί → BFS δέντρο γίνεται έτσι: Για κάθε κόμβο υ ,ο πατέρος του υ στο BFS δέντρο είναι ο κόμβος υ' με BFS αριθμό bfsnum(υ')= $\min_{(\upsilon,w)\in E}(w)$,όπου Ε είναι το σύνολο των ακμών του γραφήματος. Οι λίστες γειτνίασης μπορούν να αυξηθούν με τους ταξινομημένους BFS αριθμούς. Ακόμα μία σάρωση αρκεί για να εξάγουμε τις ακμές του BFS δέντρου.

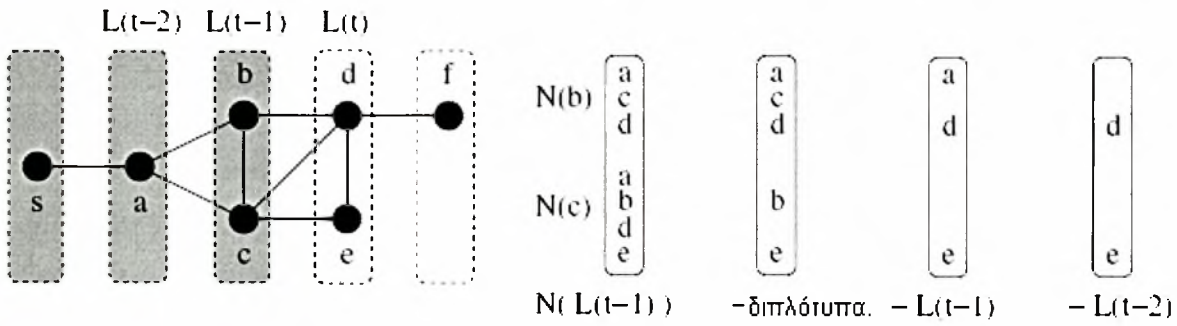
Για την μετατροπή BFS δέντρο → BFS επίπεδα ένας κύκλος του Euler γύρω από το μη κατευθυνόμενο BFS δέντρο μπορεί να κατασκευαστεί και να επεξεργαστεί χρησιμοποιώντας βήματα σάρωσης και ταξινόμησης λίστας. Οι ακμές του κύκλου του Euler που κατευθύνονται προς τα φύλλα προσδιορίζονται με ένα βάρος +1 ενώ οι ακμές που κατευθύνονται προς την κορυφή παίρνουν βάρος -1. Ένας διαδοχικός υπολογισμός του αθροίσματος των βαρών του κύκλου του Euler αποδίδει τα κατάλληλα επίπεδα.

Η τελευταία μετατροπή BFS επίπεδα → BFS αριθμοί επεξεργάζεται τα επίπεδα το ένα μετά το άλλο. Έχοντας υπολογίσει σωστά τους αριθμούς του επιπέδου i η σειρά (δηλαδή οι BFS αριθμοί) των κόμβων στο επίπεδο i+1 δίνονται ως εξής: κάθε κόμβος του επιπέδου i+1 πρέπει να είναι παιδί (στο BFS δέντρο) του γειτονικού κόμβου με τον μικρότερο BFS αριθμό του επιπέδου i. Αφού ταξινομήσουμε τους αριθμούς του επιπέδου i και τις ακμές μεταξύ των επιπέδων i και i+1 ,μία σάρωση μας δίνει τις λίστες γειτνίασης των κόμβων του επιπέδου i+1 με την απαραίτητη πληροφορία.

Ορίζουμε ως L(t) το σύνολο των κόμβων του BFS επιπέδου t και ως A(t)=N(L(t-1)) το σύνολο των γειτόνων των κόμβων του επιπέδου L(t-1). Ο MR_BFS κατασκευάζει το L(t) ως εξής: Το A(t) κατασκευάζεται με | L(t-1) | προσπελάσεις στις λίστες γειτνίασης όλων των κόμβων του επιπέδου L(t-1). Αυτό προκαλεί O(| L(t-1) | + | A(t) | / (DB)) I/Os. Παρατηρούμε πως O(1+x/(DB))I/Os χρειάζονται για να διαβάσουμε μία λίστα μήκους x. Έπειτα ο αλγόριθμος αφαιρεί τους διπλότυπους κόμβους από το A(t). Αυτό μπορεί να γίνει με μία ταξινόμηση του A(t) με βάση τους δείκτες των κόμβων ακολουθούμενη από μία φάση σάρωσης και συμπίεσης, έτσι η αφαίρεση των διπλότυπων απαιτεί O(sort(|A(t)|))I/Os. Το σύνολο A'(t)που παίρνουμε ως αποτέλεσμα είναι και αυτό ακόμα ταξινομημένο.

Τώρα ο MR_BFS υπολογίζει το L(t)=A'(t)\{L(t-1)∪L(t-2)}.Η αφαίρεση των κόμβων που ήδη περιέχονται στις ταξινομημένες λίστες L(t-1) και L(t-2) μπορεί να γίνει με παράλληλη σάρωση.

Άρα αυτό το βήμα μπορεί να γίνει χρησιμοποιώντας O((|A(t)|+|L(t-1)|+|L(t-2)|) / DB) I/Os. Καθώς Σ_t |A(t)| = O(m) και Σ_t |L(t)| = O(n) ,ο MR_BFS απαιτεί O(n+sort(n+m)) I/Os. Οι Θ(n) I/O πράξεις είναι αποτέλεσμα των ανοργάνωτων προσβάσεων στις n λίστες γειτνίασης.



Εικόνα 2.2: Παράδειγμα αλγορίθμου Munagala και Ranade

Η ορθότητα αυτού του BFS αλγορίθμου κυρίως στηρίζεται στο ότι το γράφημα εισόδου είναι μη κατευθυνόμενο. Υποθέτουμε αρχικά ότι όλα τα L(0),….,L(t-1) επίπεδα έχουν ήδη υπολογιστεί σωστά και μελετούμε έναν γείτονα υ ενός κόμβου u του επιπέδου L(t-1). Η απόσταση από τον κόμβο πηγή s μέχρι τον υ είναι το λιγότερο t-2 γιατί σε αντίθετη περίπτωση η απόσταση από τον u θα ήταν μικρότερη από t-1. Έτσι η υ ανήκει στο L(t-2) ∪ L(t-1) ∪ L(t) και έτσι είναι σωστό να αναθέτουμε στο L(t) τους κόμβους του A'(t)\{ L(t-1)∪L(t-2)}.

ΘΕΩΡΗΜΑ 2.1:

Ο αλγόριθμος BFS για μη κατευθυνόμενα γραφήματα απαιτεί O(n+sort(n+m)) I/O πράξεις.

2.3 Η νέα προσέγγιση-Γρήγορος BFS:

Σε αυτή την ενότητα θα δείξουμε πως μπορεί ο αλγόριθμος Munagala και Ranade της προηγούμενης ενότητας να επιταχυνθεί. Αναφερόμαστε στον αλγόριθμο στον οποίο θα καταλήξουμε σε αυτή την ενότητα ως γρήγορο_BFS. Ο γρήγορος_BFS λειτουργεί σε δύο φάσεις. Στην πρώτη φάση προεπεξεργάζεται το γράφημα εισόδου και στην δεύτερη φάση εκτελεί τον BFS χρησιμοποιώντας την πληροφορία στην οποία κατέληξε στην πρώτη φάση. Θα υποθέσουμε ότι το γράφημα εισόδου είναι συνεκτικό (στην αντίθετη περίπτωση θα εκτελέσουμε τον O(sort(n+m)) I/O πράξεων αλγόριθμο συνεκτικών συνιστωσών και θα κρατήσουμε μόνο τους κόμβους και τις ακμές της συνιστώσας C_s που περιέχει τον κόμβο πηγή, όλοι οι υπόλοιποι κόμβοι που δεν ανήκουν στην C_s θα ανατεθούν σε άπειρο BFS επίπεδο και ο υπολογισμός του BFS θα συνεχιστεί στην C_s). Θα ξεκινήσουμε με την φάση προεπεξεργασίας με συνθήκες τυχαιότητας, έπειτα θα περιγράψουμε την φάση εκτέλεσης του BFS και τέλος θα παρουσιάσουμε την φάση προεπεξεργασίας με την ντετερμινιστική της έκδοση.

2.3.1.Φαση προεπεξεργασίας(με συνθήκες τυχαιότητας)

Διαχωρισμός του γραφήματος σε υπογραφήματα μικρών αποστάσεων

Σαν πρώτο βήμα ,ο γρήγορος_BFS, αναδομεί τις λίστες γειτνίασης της αναπαράστασης του γραφήματος. Αναπτύσσει ξεχωριστά συνεκτικά υπογραφήματα S_i ,$0 \leq i \leq K$ με μικρές διαμέτρους από τυχαία επιλεγμένους κόμβους s_i και σώζει τις λίστες γειτνίασης όλων των κόμβων που ανήκουν στο υπογράφημα S_i σε ένα εξωτερικό αρχείο F_i ,και όλα μαζί τα αρχεία F_i συγκροτούν το εξωτερικό αρχείο $F=F_0F_1\ldots F_i\ldots F_{K-1}$. Ο κόμβος s_i ονομάζεται κύριος κόμβος του υπογραφήματος S_i. Ένας κόμβος επιλέγεται να είναι ο κύριος κόμβος ενός υπογραφήματος με πιθανότητα $\mu=\min\{1, \sqrt{\frac{(n+m)}{nDS}}\}=\min\{1, \sqrt{\frac{|V|+|E|}{|V|DS}}\}$. Αυτή η επιλογή του μ ελαχιστοποιεί το κόστος του αλγορίθμου όπως θα δούμε στην συνέχεια. Επιπροσθέτως βεβαιωνόμαστε ότι ο κόμβος πηγή s θα είναι ο κύριος κόμβος του υπογραφήματος S_0. Αναθέτουμε στην παράμετρο K να είναι ο αριθμός των κύριων κόμβων έτσι $E[K]=1+\mu n$.

Ο διαχωρισμός του αρχικού γραφήματος γίνεται παράλληλα εκτελώντας έναν τοπικό BFS αλγόριθμο από κάθε κύριο κόμβο παράλληλα. Σε κάθε γύρο κάθε κύριος κόμβος s_i προσπαθεί να καλύψει όλους τους γείτονες του τρέχοντος υπογραφήματος S_i που δεν έχουν επισκεφτεί ακόμα. Εάν διαφορετικοί κύριοι κόμβοι θέλουν να συμπεριλάβουν στο υπογράφημά τους έναν συγκεκριμένο κόμβο υ τότε πετυχαίνει ένας τυχαίος κύριος κόμβο από αυτούς.

Στο ξεκίνημα κάθε φάσης οι λίστες γειτνίασης των κόμβων που βρίσκονται στο όριο των μέχρι στιγμής υπογραφημάτων θεωρούνται ενεργοί (γιατί για την δημιουργία της επόμενης φάσης θα χρειαστούμε τους γείτονες των τελευταίων κόμβων που συμπεριλήφθηκαν σε κάθε υπογράφημα) και κουβαλούν πληροφορία για το ποιος είναι ο κύριος κόμβος τους. Μία σάρωση του συνόλου των λιστών γειτνίασης απομακρύνει τις ήδη προσδιορισμένες λίστες γειτνίασης και δημιουργεί ένα σύνολο αιτήσεων για τους υποψήφιους κόμβους του επόμενου επιπέδου. Στη συνέχεια οι αιτήσεις ταξινομούνται .Μία παράλληλη σάρωση των ταξινομημένων αιτήσεων και των τμημάτων εκείνων της αναπαράστασης του γραφήματος που δεν έχουν επισκεφτεί ακόμα μας επιτρέπει να ταυτοποιήσουμε τους νέους κόμβους ορίου (και τις λίστες γειτνίασης τους). Κάθε λίστα γειτνίασης είναι ενεργή κατά τη διάρκεια το πολύ μίας φάσης. Η διαδικασία διαχωρισμού σταματάει μόλις δεν υπάρχουν πια άλλες ενεργές λίστες γειτνίασης.

ΛΗΜΜΑ 2.1:

Ορίζουμε αυθαίρετα τον κόμβο υ που ανήκει στο αρχικό γράφημα G. Ο υ μπορεί να εκχωρηθεί σε κάποιο υπογράφημα (και έτσι να αφαιρεθεί από την αναπαράσταση του γραφήματος) ύστερα από έναν αναμενόμενο αριθμό γύρων που είναι το πολύ $1/\mu$.

Απόδειξη:

Μελετούμε το συντομότερο μονοπάτι $P=<s,u_j,\ldots,u_2,u_1,\upsilon>$ από τον κόμβο πηγή s στον κόμβο υ του γραφήματος G. Ορίζουμε τον k, $1 \leq k \leq j$,να είναι ο μικρότερος δείκτης έτσι ώστε ο u_k να είναι ο κύριος κόμβος. Κατόπιν ο κόμβος υ θα προσδιοριστεί σε ποιο υπογράφημα ανήκει κατά τη διάρκεια ή πριν τον k-οστό γύρο. Εξ' αιτίας της τυχαίας επιλογής των κύριων κόμβων έχουμε $E[k] \leq 1/\mu$.

ΠΟΡΙΣΜΑ 2.1:

Μελετώντας έναν τυχαίο κόμβο υ που ανήκει στο υπογράφημα S_i και έστω ότι s_i είναι ο κύριος κόμβος του υπογραφήματος S_i , η αναμενόμενη απόσταση συντομότερου μονοπατιού μεταξύ του υ και του s_i μέσα στο υπογράφημα S_i είναι το πολύ $1/\mu$.

Από το ΛΗΜΜΑ 2.1 η αναμενόμενη συνολική ποσότητα δεδομένων που επεξεργαζόμαστε από την αναπαράσταση των λιστών γειτνίασης κατά τη διάρκεια του παράλληλου διαχωρισμού οριοθετείται από το $X=O(\Sigma_{\upsilon \in V}\ 1/\mu(1+degree(\upsilon))) = O((n+m)/\mu) = O((|V|+|E|)/\mu)$. Ωστόσο η ταξινόμηση αφορά μόνο τις ενεργές λίστες γειτνίασης έτσι η φάση προεπεξεργασίας απαιτεί $O((n+m)/(\mu DB) + sort(n+m))$ πράξεις εισόδου/εξόδου.

Ύστερα από τη φάση προεπεξεργασίας κάθε κόμβος γνωρίζει τον δείκτη του υπογραφήματος στο οποίο ανήκει. Με έναν σταθερό αριθμό πράξεων σάρωσης και ταξινόμησης μπορούμε να διαχωρίσουμε τις λίστες γειτνίασης στη μορφή $F_0F_1...F_i...F_{|S|-1}$, όπου το F_i περιέχει τις λίστες γειτνίασης των κόμβων που ανήκουν στο S_i. Μία εγγραφή $(\upsilon,w,S(w),f_{S(w)})$ από την λίστα του $\upsilon \in F_i$ αντιπροσωπεύει την ακμή (υ,w) και μας παρέχει την επιπρόσθετη πληροφορία ότι ο κόμβος w ανήκει στο υπογράφημα S(w) του οποίου το υποαρχείο $F_{S(w)}$ ξεκινάει στη θέση $f_{S(w)}$ στο συνολικό αρχείο F. Οι εγγραφές ακμών σε κάθε F_i είναι λεξικογραφικά ταξινομημένες. Συνολικά το εξωτερικό αρχείο F κατανέμεται σε $O((n+m)/B)$ μπλοκ της εξωτερικής μνήμης .Το F αποτελείται από K υποαρχεία με $E[K]=1+\mu n$. Το μέγεθος των υποαρχείων έχει ευρύ φάσμα. Κάποια κατανέμονται σε διαφορετικά μπλοκ του δίσκου και κάποια μοιράζονται το ίδιο μπλοκ. Το επόμενο λήμμα συναθροίζει τα παραπάνω.

ΛΗΜΜΑ 2.2:

Η προεπεξεργασία του γρήγορου_BFS με συνθήκες τυχαιότητας απαιτεί $O(\frac{n+m}{\mu DS} + sort(n+m))$ I/Os.

2.3.2.Η BFS φάση:

Στην δεύτερη φάση κατασκευάζουμε τα BFS επίπεδα ένα-ένα όπως στο αλγόριθμο Munagala και Ranade(MR_BFS).Το καινούριο χαρακτηριστικό και η κύρια διαφορά όμως του αλγορίθμου μας είναι η διατήρηση ενός ταξινομημένου εξωτερικού αρχείου Η που αποτελείται από τα αχρησιμοποίητα μέρη των υποαρχείων F_i που περιέχουν έναν κόμβο στο τρέχον επίπεδο L(t-1). Αρχικοποιούμε το Η με το υποαρχείο F_0. Έτσι αρχικά το Η θα περιέχει την λίστα γειτνίασης του κόμβου πηγή s του επιπέδου L(0). Οι κόμβοι του κάθε BFS επιπέδου που κατασκευάζεται θα κουβαλούν και την πληροφορία του δείκτη του υποαρχείου τους F_i του αντίστοιχού τους υπογραφήματος S_i.

Κατά την κατασκευή του επιπέδου L(t) που στηρίζεται στα L(t-1) και L(t-2) ο γρήγορος_BFS δεν προσπελαύνει απλές λίστες γειτνίασης όπως κάνει ο MR_BFS.Αντίθετα εκτελεί μία παράλληλη σάρωση των ταξινομημένων λιστών του L(t-1) και του H. Όταν το κάνει αυτό εξάγει τις λίστες γειτνίασης όλων των κόμβων v_j που ανήκουν στο L(t-1) και μπορούν να βρεθούν στο H. Ορίζουμε ως V_1 το υποσύνολο των κόμβων του L(t-1) οι λίστες γειτνίασης των οποίων μπορούν να βρεθούν με αυτόν τον τρόπο. Σαν ένα δεύτερο βήμα ο γρήγορος_BFS εξάγει από το L(t-1) τους αναγνωριστές των διαμερίσεων των κόμβων που ανήκουν στο V_2= L(t-1)\ V_1. Αφού ταξινομήσουμε αυτούς τους αναγνωριστές και αφού διαγράψουμε τους διπλότυπους ο γρήγορος_BFS γνωρίζει ποια υποαρχεία F_i του F περιέχουν τις λίστες γειτνίασης που λείπουν. Τα αντίστοιχα υποαρχεία εισάγονται αλυσιδωτά σε ένα προσωρινό αρχείο F' και έπειτα ταξινομούνται. Έπειτα οι λίστες γειτνίασης των κόμβων του V_2 που έλειπαν μπορούν να εξαχθούν με μία απλή πράξη σαρώσεως του αρχείου F' και οι υπόλοιπες λίστες γειτνίασης μπορούν να ενωθούν με το ταξινομημένο σύνολο H σε ένα απλό βήμα.

Αφού αποκτήσαμε τις λίστες γειτνίασης των κόμβων του επιπέδου L(t-1) το σύνολο N(L(t-1)) μπορεί να παραχθεί με μία απλή σάρωση. Σε αυτό το σημείο η επαυξημένη μορφή των λιστών γειτνίασης χρησιμοποιείται με σκοπό να επισυνάψουμε την πληροφορίας της διαμερίσεως σε κάθε κόμβο του N(L(t-1)) .Στη συνέχεια ο γρήγορος_BFS προχωρά όπως ο MR_BFS,δηλαδή απομακρύνει τους διπλότυπους κόμβους από το σύνολο N(L(t-1)) και επίσης διαγράφει και όλους τους κόμβους που έχουν ήδη προσδιοριστεί να ανήκουν στα επίπεδα L(t-1) και L(t-2).Οι υπόλοιποι κόμβοι συνιστούν το L(t). Τα παραγόμενα επίπεδα γράφονται στην εξωτερική μνήμη ως συνεχόμενα δεδομένα, έτσι απαιτούνται O(n/(DB)) μπλοκ από τους D δίσκους.

Καθώς ο γρήγορος_BFS είναι μία βελτίωση του MR_BFS , η ορθότητά του είναι δεδομένη. Έχουμε μόνο να επανεξετάσουμε τα όρια των πράξεων εισόδου/εξόδου.

ΛΗΜΜΑ 2.3:

Η BFS φάση του γρήγορου_BFS απαιτεί $O(\mu n + \frac{n+m}{\mu DB} + sort(n+m))$ πράξεις εισόδου/εξόδου.

Απόδειξη:

Θα πρέπει κυρίως να υπολογίσουμε την συνολική ποσότητα πράξεως I/O που απαιτούνται για να διατηρήσουμε την δομή δεδομένων H. Για την κατασκευή των επιπέδων L(t) ,τα περιεχόμενα των ταξινομημένων συνόλων H,L(t-1) και L(t-2) θα σαρωθούν έναν σταθερό αριθμό φορών. Τα πρώτα DB μπλοκ αυτών των συνόλων κρατούνται πάντα στην κύρια μνήμη. Έτσι η σάρωση αυτών των δεδομένων δεν απαιτεί απαραίτητα πράξη I/O για κάθε επίπεδο. Η προσπέλαση της εξωτερικής μνήμης είναι απαραίτητη μόνο όταν το πλήθος τον δεδομένων είναι Ω(DB) .Σε αυτή

την περίπτωση ο αριθμός των I/Os που χρειάζονται για να σαρώσουμε x δεδομένα καθ' όλη τη διάρκεια της εκτέλεσης του γρήγορου_BFS οριοθετείται από O(x/(DB)).

Ανοργάνωτες I/O πράξεις συμβαίνουν όταν το H γεμίζει με την ένωση υποαρχείων F_i με τα τρέχοντα περιεχόμενά του. Για ένα BFS επίπεδο μπορούν να προστεθούν δεδομένα στο H από διαφορετικά υποαρχεία F_i. Ωστόσο τα δεδομένα καθενός ξεχωριστού F_i εισάγονται στο H το πολύ μία φορά. Έτσι ο αριθμός των I/Os που χρειάζονται για να εκτελέσουμε τις ενώσεις μπορεί να μοιραστεί σε (α)οι λίστες γειτνίασης που φορτώνονται από το F και (β)αυτές που ήδη υπάρχουν στο H. Το όριο των I/O πράξεων για το (α) είναι : $O(\Sigma_i(1+\frac{|F_i|}{DB} \log_{M/B}\frac{n+m}{B}))$ = O(K+sort(n+m)) I/Os.

Όσον αφορά το (β) παρατηρούμε πρώτα ότι η λίστα γειτνίασης A_v ενός τυχαίου κόμβου $v \in S_i$ παραμένει στο H για το πολύ 2/μ γύρους. Αυτό έπεται από το γεγονός ότι η αναμενόμενη απόσταση συντομότερου μονοπατιού μεταξύ δύο κόμβων του υπογραφήματος είναι το πολύ 2/μ. Αναθέτουμε στο L(t') να είναι το BFS επίπεδο για το οποίο το F_i (άρα και η A_v)εισήχθη στο H (δεν ανήκει οπωσδήποτε ο v στο L(t') αλλά αν ανήκει οποιοσδήποτε άλλος κόμβος του ίδιου υπογραφήματος προκαλεί την εισαγωγή των λιστών γειτνίασης όλων των κόμβων του υπογραφήματος στο H). Συνεπώς θα πρέπει να υπάρχει κάποιος κόμβος v' $\in S_i$ που ανήκει στο BFS επίπεδο L(t').O s_i είναι ο κύριος κόμβος του υπογραφήματος S_i και το d(x,y) ορίζει των αριθμό των ακμών του συντομότερου μονοπατιού μεταξύ των κόμβων x και y στο S_i .Καθώς το γράφημα είναι μη κατευθυνόμενο το BFS επίπεδο του v θα είναι μεταξύ των L(t') και L(t'+d(v',s_i)+d(s_i,v)) (δεν είναι σίγουρα δεν προηγούμενο από το L(t') επίπεδο γιατί το F_i εισάγεται στο η μόνο μία φορά ,άρα αν ήταν σε προηγούμενο το F_i θα είχε ενωθεί πιο νωρίς στο H). Όταν ο κόμβος v προσδιορίζεται να ανήκει σε ένα BFS επίπεδο η αντίστοιχη λίστα γειτνίασης του A_v διαγράφεται από το H(μόνο η A_v όχι όλο το υποαρχείο F_i). Από το ΠΟΡΙΣΜΑ 1.1 παίρνουμε ότι E[d(v',s_i)+d(s_i,v)] ≤ 2/μ. Με άλλα λόγια η κάθε λίστα γειτνίασης είναι μέρος του H για O(2/μ) BFS επίπεδα. Έτσι ο αναμενόμενος συνολικός όγκος δεδομένων για το (β) οριοθετείται από O((n+m)/μ). Αυτό έχει ως αποτέλεσμα O((n+m)/(μDB)) πράξεις I/O για την σάρωση του H κατά τη διάρκεια των πράξεων ενώσεων. Με την ίδια επιχειρηματολογία συμπεραίνουμε και ότι κάθε λίστα γειτνίασης στο H παίρνει μέρος το πολύ σε O(1/μ) βήματα σάρωσης για την δημιουργία των N(L(·)) και L(·) . Παρομοίως με τον MR_BFS αλγόριθμο η σάρωση και η ταξινόμηση όλων των BFS επιπέδων και των συνόλων N(L(·)) χρειάζεται O(sort(n+m)) I/Os. ¤

Συνενώνοντας τα λήμματα 2.2 και 2.3 και κάνοντας σωστή επιλογή για την τιμή του μ έχουμε:

ΘΕΩΡΗΜΑ 2.2:

Ο BFS αλγόριθμος εξωτερικής μνήμης για τυχαία μη κατευθυνόμενα γραφήματα μπορεί να λυθεί χρησιμοποιώντας $O(\sqrt{\frac{n(n+m)}{DB}} + sort(n+m))$ I/Os.

Απόδειξη:

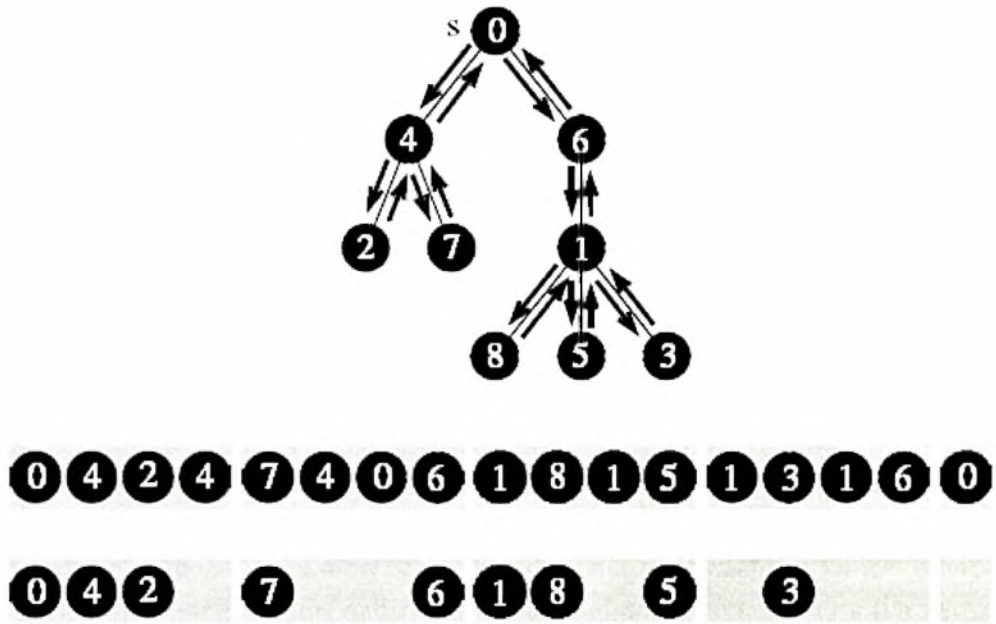
Από αυτά τα λήμματα ο αναμενόμενος αριθμός για πράξεις I/O οριοθετείται από $O(\mu n+\frac{n+m}{\mu DB}+\text{sort}(n+m))$. Αυτό το όριο ελαχιστοποιείται για μ^2 n D B = n + m. Άρα επιλέγοντας $\mu=\min\{1,\sqrt{(nDB)/(n+m)}\}$ παίρνουμε το παραπάνω όριο.

2.3.3.Η ντετερμινιστική έκδοση:

Με σκοπό να πετύχουμε το αποτέλεσμα του θεωρήματος ΘΕΩΡΗΜΑ 2.2 και στην χειρότερη περίπτωση ,αρκεί να τροποποιήσουμε την φάση προεπεξεργασίας που παρουσιάστηκε στην ενότητα 1 αυτού του κεφαλαίου. Αντί να αναπτύσσουμε υπογραφήματα γύρω από τυχαία επιλεγμένους κύριους κόμβους , η ντετερμινιστική έκδοση εξάγει τα υποαρχεία F_i από έναν κύκλο του Euler γύρω από ένα επικαλύπτον δέντρο της συνεκτικής συνιστώσας C_s που περιέχει τον κόμβο πηγή s. Την συνεκτική συνιστώσα C_s μπορούμε να την αποκτήσουμε με τον ντετερμινιστικό αλγόριθμο συνεκτικών συνιστωσών που χρησιμοποιεί $O((1+\text{loglog}(DBn/m))\cdot\text{sort}(n+m))$ I/Os.Το ίδιο σύνολο I/O πράξεων αρκεί για να υπολογίσουμε ένα (ελάχιστο) επικαλύπτον δέντρο T_s για την C_s.

Αφού σχηματίσουμε το T_s η φάση προεπεξεργασίας κατασκευάζει έναν κύκλο Euler γύρω του, χρησιμοποιώντας έναν σταθερό αριθμό βημάτων ταξινόμησης και σάρωσης. Ο κύκλος σταματά στον κόμβο πηγή. Τα αντικείμενα της παραγόμενης λίστας σώζονται συνεχόμενα χρησιμοποιώντας τον ντετερμινιστικό αλγόριθμο ταξινόμησης λίστας .Αυτό χρειάζεται O(sort(n)) I/Os. Στη συνέχεια ο κύκλος του Euler μπορεί να κοπεί σε τμήματα μεγέθους 1/μ με ένα απλό βήμα σάρωσης. Αυτά τα τμήματα του κύκλου Euler αποτελούν τα υπογραφήματα S_i ,με την ιδιότητα ότι η απόσταση μεταξύ δύο οποιωνδήποτε κόμβων του S_i στο αρχικό γράφημα G είναι το πολύ ίση με 1/μ – 1. Παρατηρούμε ότι ένας κόμβος υ βαθμού d μπορεί να είναι μέρος Θ(d) διαφορετικών υπογραφημάτων S_i. Ωστόσο με έναν σταθερό αριθμό βημάτων ταξινόμησης μπορούμε να αφαιρέσουμε τους διπλότυπους κόμβους και έτσι να είμαστε βέβαιοι ότι κάθε κόμβος της συνεκτικής συνιστώσας C_s είναι μέρος ακριβώς ενός υπογραφήματος S_i,για

παράδειγμα είναι μέρος του υπογραφήματος με τον μικρότερο δείκτη και γενικά $s \in S_0$. Τελικά, τα μειωμένα υπογραφήματα S_i χρησιμοποιούνται για να δημιουργήσουμε τα αναδιατεταγμένα αρχεία F_i λιστών γειτνίασης F_i. Αυτό γίνεται όπως και στην φάση προεπεξεργασίας υπό συνθήκες τυχαιότητας και χρειάζεται άλλες O(sort(n+m)) πράξεις I/O. Σημειώνουμε ότι τα μειωμένα υπογραφήματα S_i μπορεί να μην είναι πια συνδεδεμένα, αυτό όμως δεν είναι πρόβλημα γιατί η προσέγγισή μας απαιτεί μόνο οποιοιδήποτε δύο κόμβοι σε ένα υπογράφημα να είναι σχετικά κοντά στο αρχικό γράφημα εισόδου.



Εικόνα 2.3: Χρησιμοποιώντας τον κύκλο του Euler γύρω από ένα επικαλύπτον δέντρο του γραφήματος εισόδου ,με σκοπό να πετύχουμε έναν διαμοιρασμό για τον ντετερμινιστικό BFS αλγόριθμο.

Η BFS φάση του αλγορίθμου παραμένει αμετάβλητη. Η τροποποιημένη φάση προεπεξεργασίας όμως εγγυάται ότι κάθε λίστα γειτνίασης θα είναι μέρος του εξωτερικού συνόλου H για το πολύ 1/μ BFS επίπεδα. Εάν ένα υποαρχείο F_i ενσωματωθεί στο H για το BFS επίπεδο L(t) ,τότε το BFS επίπεδο οποιουδήποτε κόμβου υ του S_i είναι το πολύ L(t) + 1/μ -1. Το όριο του συνολικού αριθμού πράξεων εισόδου/εξόδου έπεται από το γεγονός ότι $O((1+\text{loglog}(DBn/m))\cdot sort(n+m)) = O(\sqrt{\frac{n(n+m)}{DB}} + sort(n+m))$.

ΘΕΩΡΗΜΑ 2.3:

Υπάρχει ένας ντετερμινιστικός αλγόριθμος που λύνει το πρόβλημα του BFS για μη κατευθυνόμενα γραφήματα στην εξωτερική μνήμη χρησιμοποιώντας $O(\sqrt{\frac{n(n+m)}{DB}} + sort(n+m))$ πράξεις εισόδου/εξόδου.

2.4 Ημί-εξωτερικός BFS σε κατευθυνόμενα Euler-ιανά γραφήματα:

[7] Η ανάλυση του γρήγορου_BFS όπως αναφέρθηκε στις προηγούμενες ενότητες δεν μπορεί να ανταποκριθεί και στην περίπτωση των κατευθυνόμενων γραφημάτων. Αυτό συμβαίνει για δύο κυρίως λόγους. (i)Κατά την κατασκευή ενός BFS επιπέδου L(t) δεν αρκεί να ελέγξουμε μόνο τους κόμβους των επιπέδων L(t-1) και L(t-2). Ένας κόμβος του A'(t)(δηλαδή γείτονες του L(t) που ούτε είναι διπλότυποι ούτε ανήκουν στα L(t-1) και L(t-2))μπορεί να έχει ήδη εμφανιστεί σε κάποιο από τα προηγούμενα επίπεδα ,δηλαδή σε κάποιο από τα L(0),…,L(t-1). (ii)Εάν το υπογράφημα S_i δεν είναι ισχυρά συνεκτικό δεν είναι βέβαιο πως όταν ένας κόμβος υ που ανήκει στο S_i και είναι μέρος του BFS επιπέδου L(t), τότε όλοι οι υπόλοιποι κόμβοι υ' που ανήκουν στο ίδιο υπογράφημα S_i θα ανήκουν σε κάποια BFS επίπεδα L(t') με $t'<t+|S_i|$.Με άλλα λόγια οι λίστες γειτνίασης των κόμβων του S_i μπορεί να χρειαστεί να μείνουν στη δομή δεδομένων Η για μεγάλο διάστημα.

Το πρόβλημα (i) μπορεί να αντιμετωπιστεί στην ημί-εξωτερική μνήμη κρατώντας έναν πίνακα στοιχείων στην εσωτερική μνήμη που θα μας βοηθά στην διαδικασία ψαξίματος των κόμβων όλων των επιπέδων. Δυστυχώς όμως δεν υπάρχει μία γενική λύση για το πρόβλημα (ii). Παρ' όλα αυτά έχουμε πετύχει ένα βελτιωμένο όριο πράξεων εισόδου/εξόδου για τον ημί-εξωτερικό BFS αλγόριθμο για τα αραιά κατευθυνόμενα Euler-ιανά γραφήματα(Ένα κατευθυνόμενο γράφημα είναι συνεκτικό τότε είναι και Euler-ιανό εάν ο βαθμός εισόδου της κάθε κορυφής υ είναι ίσος με το βαθμό εξόδου της, εάν δηλαδή ισχύει ότι indegree(υ)=outdegree(υ)). Η φάση προεπεξεργασίας είναι περίπου όμοια με αυτή της ντετερμινιστικής έκδοσης για μη κατευθυνόμενα γραφήματα που παρουσιάστηκε στην ενότητα 3. Ωστόσο αντί να ομαδοποιούμε τις λίστες γειτνίασης βασισμένοι σε έναν κύκλο του Euler γύρω από ένα επικαλύπτον δέντρο ,τις ομαδοποιούμε λαμβάνοντας υπ' όψιν μας ένα κύκλωμα του Euler για ολόκληρο το γράφημα:

Υπάρχει αλγόριθμος PRAM που παράγει το κύκλωμα του Euler σε O(log n) χρόνο, χρησιμοποιώντας O(n+m) επεξεργαστές και O(n+m) χώρο .Αυτός ο παράλληλος αλγόριθμος μπορεί να μετατραπεί σε έναν αλγόριθμο εξωτερικής μνήμης ο οποίος θα απαιτεί O(log n ·

sort(n+m)) I/Os. Αναθέτουμε στο $<v_0,v_1,\ldots,v_{m-1},v_m>$ να ορίζει την σειρά των κόμβο στο Euler κύκλωμα του G ξεκινώντας από μία εμφάνιση του κόμβου πηγή ,v_0=s .Ορίζουμε το υπογράφημα S_i να περιέχει του κόμβους του εξής πολύ-συνόλου $\{v_{i(DB)^{\frac{1}{3}}},\ldots,v_{(i+1)(DB)^{\frac{1}{3}}-1}\}$. Όπως και στην

ντετερμινιστική προεπεξεργασία των μη κατευθυνόμενων γραφημάτων ,ένας κόμβος υ μπορεί να είναι μέρος διαφορετικών υπογραφημάτων S_i,όμως η λίστα γειτνίασης του θα κρατείται μόνο σε ακριβώς ένα υποαρχείο F_i. Επίσης εκμεταλλευόμαστε έναν επιπλέον περιορισμό: Τα υποαρχεία F_i σώζουν μόνο τις λίστες γειτνίασης των κόμβων που έχουν βαθμό εξόδου το πολύ $(DB)^{1/3}$.Αυτοί οι κόμβοι αποκαλούνται ελαφριοί κόμβοι ,κόμβοι με βαθμό εξόδου μεγαλύτερο του $(DB)^{1/3}$ αποκαλούνται βαριοί. Οι λίστες γειτνίασης των βαριών κόμβων κρατούνται σε μία τυπική αναπαράσταση λιστών γειτνίασης.

Η φάση BFS των κατευθυνόμενων γραφημάτων διαφέρει από την προσέγγιση για τα μη κατευθυνόμενων σε δύο σημεία. (i)Το BFS επίπεδο L(t) κατασκευάζεται ως A'(t)\{ L(0) ∪ L(1) ∪ … ∪ L(t-1)} όπου τα L(0),L(1),…,L(t-1) κρατούνται στην εσωτερική μνήμη σε έναν πίνακα. (ii) Η λίστα γειτνίασης του κάθε βαρίου κόμβου υ προσπελαύνεται ξεχωριστά χρησιμοποιώντας O(1 + outdegree(υ)/(DB)) I/Os τη στιγμή που η λίστα πρέπει να διαβαστεί. Κάθε τέτοια λίστα γειτνίασης θα προσπελαστεί το πολύ μία φορά. Καθώς υπάρχουν το πολύ $m/(DB)^{1/3}$ βαριοί κόμβοι αυτό έχει ως αποτέλεσμα $O(m/(DB)^{1/3})$ επιπλέον I/Os.

ΘΕΩΡΗΜΑ 2.4:

Ο ημί-εξωτερικός BFS αλγόριθμος σε κατευθυνόμενα Euler-ιανά γραφήματα απαιτεί $O((n+m)/(DB)^{1/3} + sort(n+m) \cdot \log n)$I/Os στη χειρότερη περίπτωση.

Απόδειξη:

Όπως έχει ήδη συζητηθεί προηγουμένως η τροποποιημένη φάση προεπεξεργασίας μπορεί να υλοποιηθεί χρησιμοποιώντας O(log n · sort(n+m)) I/Os.Η ποσότητα δεδομένων που κρατείται σε κάθε υποαρχείο F_i οριοθετείται από $O((DB)^{2/3})$.Έτσι η προσπέλαση και η ένωση όλων των $m/(DB)^{1/3}$ υποαρχείων F_i στο Η κατά τη διάρκεια της BFS φάσης παίρνει $O(m/(DB)^{1/3} + sort(m))$ I/Os(χωρίς να υπολογίζουμε τις I/O πράξεις για την σάρωση δεδομένων που υπάρχουν ήδη στο Η).

Ένα υποαρχείο F_i αναφέρεται ως κανονικό αν καμία από τις λίστες γειτνίασης του δεν παραμένει στο Η για περισσότερα από $2\cdot(DB)^{1/3}$ συνεχόμενα BFS επίπεδα. Αλλιώς το F_i

αναφέρεται ως υποαρχείο με καθυστέρηση. Η συνολική ποσότητα δεδομένων που κρατείται και σαρώνεται στο Η από κανονικά υποαρχεία είναι το πολύ $O(m/(DB)^{1/3}\cdot(DB)^{2/3}\cdot(DB)^{1/3}) = O(m\cdot(DB)^{2/3})$. Αυτό προκαλεί $O(m/(DB)^{1/3})$ I/O πράξεις.

Για τα υποαρχεία με καθυστέρηση έχουμε: Ορίζουμε ως $D=\{F_{i0}, F_{i1},...,F_{ik}\}, k \le m/(DB)^{1/3}$,το σύνολο όλων των υποαρχείων με καθυστέρηση και επιπλέον ορίζουμε ως t_{ij} το χρόνο (BFS επίπεδο) κατά τον οποίο το F_{ij} εισήχθη στο Η, παρομοίως ορίζουμε ως t'_{ij} τον χρόνο (BFS επίπεδο) μετά τον οποίο όλα τα δεδομένα του F_{ij} αφαιρούνται από το Η.

Ανακαλούμε ότι ο κόμβος πηγή s είναι ο πρώτος κόμβος στο Euler κύκλωμα $<\upsilon_0,\upsilon_1,\ldots,\upsilon_{m-1},\upsilon_m,\upsilon_0>$. Έτσι ,ο κόμβος υ_i ανήκει στο BFS επίπεδο με αριθμό το πολύ ίσο με i. Επιπλέον αν ο υ_i ανήκει σε BFS επίπεδο $x \le i$ τότε ο διαδοχικός κόμβος υ_{i+1} του κυκλώματος Euler θα ανήκει σε BFS επίπεδο με αριθμό το πολύ ίσο με x+1. Καθώς το υποαρχείο F_{i0} περιέχει λίστες γειτνίασης ελαφριών κόμβων στο σύνολο $\{\upsilon_{i_0(DB)^{2/3}},\ldots,\upsilon_{(i_0+1)(DB)^{\frac{2}{3}}-1}\}$, βρίσκουμε ότι $t_{i0}' \le (i_0+1)(DB)^{1/3}$ και γενικότερα $t_{ij}' \le t_{ij-1} + (i_j-i_{j-1}+1)(DB^{1/3})$.

Το ΘΕΩΡΗΜΑ 2.4 λαμβάνει υπ' όψιν του και την παρακάτω παρατήρηση: Όταν όλα τα δεδομένα του F_i εισαχθούν στη δομή δεδομένων Η, τα δεδομένα του υποαρχείου F_{i+1} θα έχουν τελείως επεξεργαστεί το αργότερο μετά από τα επόμενα $(1+1)\cdot(DB)^{1/3}$ BFS επίπεδα. Επειδή κάθε υποαρχείο F_i περιέχει το πολύ $O((DB)^{2/3})$ δεδομένα, η συνολική ποσότητα δεδομένων που σαρώνεται στο Η από τα F_i υποαρχεία με καθυστέρηση είναι το πολύ $Z = \sum_{j=0}^{k}(t'_{ij} - t_{ij})(DB)^{2/3} \le (i_0+1)\cdot(DB)+\sum_{j=0}^{k}(t_{i_{j-1}} - t_{i_j} + (i_j - i_{j-1} + 1)\cdot(DB)^{1/3})\cdot(DB)^{2/3}$.

Το τελευταίο άθροισμα συμπτύσσεται και το Z οριοθετείται από $t_k\cdot(DB)^{2/3}+(i_k+k+1)\cdot(DB)$. Χρησιμοποιώντας k τέτοιο ώστε $i_k \le m/(DB)^{1/3}$ και $t_k \le n$ αυτό συνεπάγεται άλλες $O((n+m)/(DB)^{1/3})$ πράξεις εισόδου/εξόδου.
¤

Το ΘΕΩΡΗΜΑ 2.4 ισχύει ακόμη και για την ασθενή συνθήκη μνήμης $M=\Omega(n/(DB)^{2/3})$,αντί να διατηρεί έναν εσωτερικό δυαδικό πίνακα n κόμβων αρκεί να θυμάται τα υποσύνολα μεγέθους $\Theta(M)$ και να προσαρτά τις λίστες γειτνίασης στην εξωτερική μνήμη κάθε φορά που γεμίζει η δομή δεδομένων της εσωτερικής μνήμης. Αυτό μπορεί να συμβεί το πολύ $O((DB)^{2/3})$φορές. Κάθε προσάρτηση των λιστών γειτνίασης στην εξωτερική μνήμη μπορεί να γίνει χρησιμοποιώντας O((n+m)/(DB)) I/Os.

Το ΘΕΩΡΗΜΑ 2.4 επίσης ισχύει και για γραφήματα που είναι σχεδόν Euler-ιανά, δηλαδή όταν ισχύει $\Sigma_\upsilon|\text{indegree}(\upsilon)-\text{outdegree}(\upsilon)|=O(m/n)$.Μία απλή προεπεξεργασία μπορεί να συνδέσει τους κόμβους με τους μη ισορροπημένους βαθμούς μέσω μονοπατιών n ψεύτικων κόμβων .Το γράφημα G' που παίρνουμε ως αποτέλεσμα είναι Euler-ιανό, έχει μέγεθος O(n+m) και τα BFS επίπεδα των κόμβων που από το αρχικό γράφημα παραμένουν τα ίδια.

2.5 Συμπεράσματα:

Παρουσιάσαμε έναν νέο BFS αλγόριθμο εξωτερικής μνήμης. Για γενικά μη κατευθυνόμενα γραφήματα είναι καλύτερος από κάθε άλλη προηγούμενη προσέγγιση και μπορεί να βοηθήσει για αποδοτικές ,από πλευράς I/O πράξεων, λύσεις σε άλλα προβλήματα γραφημάτων όπως ο εντοπισμός συντομότερων μονοπατιών μοναδικής πηγής (SSSP). Ωστόσο δεν είναι ξεκάθαρο πότε αντίστοιχη αποδοτικότητα σε πράξεις εισόδου/εξόδου μπορεί να επιτευχθεί για τυχαία κατευθυνόμενα γραφήματα. Επιπλέον είναι ανοιχτό ερώτημα πότε θα μπορέσει να υπάρξει ένα καλύτερο χαμηλότερο όριο για την επίλυση του BFS στην εξωτερική μνήμη.

Κεφάλαιο 3

DFS

3.1 Εισαγωγή:

[10] Η αναζήτηση κατά πλάτος, BFS (Breadth First Search),και η αναζήτηση κατά βάθος ,DFS (Depth First Search) είναι οι δύο πιο θεμελιώδεις στρατηγικές αναζήτησης γραφημάτων. Χρησιμοποιούνται σε μεγάλο βαθμό σε πολλούς αλγόριθμους γραφημάτων. Οι λόγοι είναι ότι και οι δύο αυτές στρατηγικές α) μπορούν να υλοποιηθούν σε γραμμικό χρόνο στην εσωτερική μνήμη και β) αποκαλύπτουν σημαντική πληροφορία για την δομή του γραφήματος εισόδου.

Παρακάτω θα μελετήσουμε μία σημαντική κλάση αραιών γραφημάτων, που ονομάζονται μη κατευθυνόμενα χωρίς ακμές διασταύρωσης επίπεδα γραφήματα (undirected embedded planar graphs). Θα μελετήσουμε έναν βελτιωμένο I/O DFS αλγόριθμο για επίπεδα γραφήματα ο οποίος χρησιμοποιεί O(sort(N)logN) I/Os και γραμμικό χώρο. Για πρακτικές τιμές των B,M και N αυτός ο αλγόριθμος χρησιμοποιεί o(N) I/Os και είναι ο πρώτος από αυτούς τους αλγορίθμους που χρησιμοποιεί γραμμικό χώρο. Ο αλγόριθμος αυτός βασίζεται στην προσέγγιση του «διαίρει και βασίλευε».Χρησιμοποιεί έναν νέο O(sort(N)) I/O αλγόριθμο για να βρει έναν απλό κύκλο σε ένα δισυνεκτικό επίπεδο γράφημα (δηλ. σε γράφημα στο οποίο για δύο οποιεσδήποτε κορυφές του υπάρχουν διαζευγμένα ως προς τις κορυφές τους μονοπάτια που τις συνδέουν) τέτοιον ώστε κανένα υπογράφημα ούτε μέσα ούτε έξω από αυτόν να περιέχει περισσότερες από έναν σταθερό παράγοντα κορυφές του γραφήματος. Μετά θα δείξουμε πως αυτός ο I/O DFS αλγόριθμος μπορεί να ελαττωθεί σε I/O BFS για επίπεδα γραφήματα μέσω αποδοτικού O(sort(N)) I/O τρόπου.

[9] Οι καλύτεροι ,έως πρόσφατα, γνωστοί αλγόριθμοι DFS σε μη κατευθυνόμενα γραφήματα χρησιμοποιούν $O((|V|+(|E|/B))\log_2|V|)$ I/Os ή $O(|V|+(|V|/M)(|E|/B))$ I/Os .Ενώ οι πιο γνωστοί γενικοί BFS αλγόριθμοι χρησιμοποιούν μόνο $O(|V|+(|E|/|V|)sort(|V|)) = O(|V|+sort(|E|))$ I/Os ,αυτό αποδεικνύει ότι στα μη κατευθυνόμενα γραφήματα ο DFS μάλλον είναι πιο δύσκολος από τον BFS. Για κατευθυνόμενα γραφήματα οι καλύτεροι γνωστοί αλγόριθμοι για BFS και για DFS

χρησιμοποιούν O((|V|+|E|/B)log(|V|/B)+sort(|E|)) I/Os. Γενικά όμως δεν μπορούμε να ελπίζουμε ότι θα κατασκευάσουμε αλγόριθμους που να εκτελούν λιγότερα από Ω(min(|V|,sort(|V|))) I/Os για κανένα από τα δύο προβλήματα. Για τα περισσότερα προβλήματα γραφημάτων αυτό είναι το χαμηλότερο όριο και επειδή για ρεαλιστικές τιμές των N , B , M , scan(N) < sort(N) << N αυτό το όριο γίνεται Ω (sort(|V|)).Για αραιά γραφήματα τέτοια πολυπλοκότητα σε I/Os μπορεί να επιτευχθεί πολύ ευκολότερα χρησιμοποιώντας τους στάνταρ αλγορίθμους εσωτερικής μνήμης.

3.2 DFS χρησιμοποιώντας απλό κύκλο διαχωριστή

Κύρια σημεία αλγορίθμου:

Ο O(sort(N)logN) I/O και γραμμικού χώρου αλγόριθμός μας για τον υπολογισμό ενός DFS δέντρου ενός επίπεδου γραφήματος βασίζεται στην προσέγγιση του «διαίρει και βασίλευε». Παρουσιάζουμε λίγη ορολογία που θα χρησιμοποιηθεί. Μία κορυφή αποκοπής ενός γραφήματος G είναι μία κορυφή της οποίας η απόσβεση επαυξάνει των αριθμό των συνεκτικών συνιστωσών του γραφήματος, με άλλα λόγια αποσυνδέει το γράφημα . Ένα συνεκτικό γράφημα G είναι δισυνεκτικό όταν δεν έχει κανένα σημείο αποκοπής. Σε περίπτωση που το γράφημα δεν είναι δισυνεκτικό τότε ως δισυνεκτικές συνιστώσες ορίζονται τα μέγιστα υπογραφήματα του που διαθέτουν την ιδιότητα της δισυνεκτικότητας. Πρέπει να σημειωθεί πως μία κορυφή μπορεί να ανήκει σε περισσότερες από μία δισυνεκτικές συνιστώσες. Ένας απλός κύκλος α-διαχωριστής C ενός επίπεδου γραφήματος G χωρίς ακμές διασταύρωσης είναι ένας απλός κύκλος τέτοιος ώστε κανένα υπογράφημα είτε μέσα είτε έξω από αυτόν να περιέχει περισσότερες από α|V| κορυφές .Ένας τέτοιος κύκλος υπάρχει εγγυημένα εάν το γράφημα είναι δισυνεκτικό.

Η κύρια ιδέα του αλγορίθμου μας είναι να χωρίσουμε το G χρησιμοποιώντας έναν απλό κύκλο α-διαχωριστή C, περιοδικά να υπολογίζουμε τα DFS δέντρα των συνεκτικών συνιστωσών του G\C και να τα ενώσουμε για να κατασκευάσουμε ένα DFS δέντρο για ολόκληρο το γράφημα. Εάν κάθε περιοδικό βήμα εκτελείται σε O(sort(N)) I/Os, έπεται ότι ολόκληρος ο αλγόριθμος θα χρησιμοποιεί O(sort(N)log(N/M)) I/Os επειδή τα μεγέθη των δύο υπογραφημάτων του G που επιστρέφονται μειώνονται με γεωμετρική πρόοδο και σταματούμε να εκτελούμε τα περιοδικά βήματα μόλις το τρέχον γράφημα χωράει στην κύρια μνήμη. Παρακάτω μελετάμε τον αλγόριθμο με περισσότερες λεπτομέρειες.

Δοθέντος ενός δισυνεκτικού επίπεδου γραφήματος G χωρίς ακμές διασταύρωσης και μίας κορυφής s ∈ G ,κατασκευάζουμε ένα DFS δέντρο T του G με ρίζα την κορυφή s ακολουθώντας τα παρακάτω βήματα:

1. Υπολογίζουμε έναν απλό κύκλο $\frac{2}{3}$ – διαχωριστή C του G.

   Σε παρακάτω ενότητα θα δείξουμε πως γίνεται αυτό σε O(sort(N)) I/Os.

2. Βρίσκουμε ένα μονοπάτι P από την s σε μία κορυφή υ στον C.

   Για να το κάνουμε αυτό υπολογίζουμε ένα αυθαίρετο επικαλύπτον δέντρο T' του G που έχει ρίζα στην s και βρίσκουμε μία κορυφή υ ∈ C της οποίας η απόσταση από την s στο T' είναι η ελάχιστη. Το μονοπάτι P είναι ένα μονοπάτι από την s στην υ στο T'. Το επικαλύπτον δέντρο T' μπορεί να υπολογιστεί σε O(sort(N)) I/Os. Δοθέντος του δέντρου T' η κορυφή υ μπορεί εύκολα να βρεθεί σε O(sort(N)) I/Os χρησιμοποιώντας μία BFS διαπέραση στο T'. Αυτό χρειάζεται O(sort(N)) I/Os στάνταρ υπολογισμούς δέντρων.

3. Επεκτείνουμε το P σε ένα μονοπάτι P' που περιέχει όλες τις κορυφές του P και του C.
   Για να το κάνουμε αυτό εντοπίζουμε τον αντίστροφα με τους δείχτες του ρολογιού γείτονα w ∈ C της υ και αναθέτουμε στο C' να είναι το μονοπάτι που δημιουργείται από την απόσβεση της ακμής {υ,w} από τον C. Μετά το μονοπάτι P' είναι η ένωση του P (πριν το επεκτείνουμε) και του C'.Αυτός ο υπολογισμός μπορεί εύκολα να εκτελεστεί σε O(sort(N)) I/Os :Αρχικά σαρώνουμε την λίστα ακμών του C και διαγράφουμε την πρώτη ακμή που βρίσκουμε που έχει την υ ως ένα άκρο της. Μετά συνδέουμε αλυσιδωτά την παραχθείσα λίστα ακμών του C' και την λίστα ακμών του P.

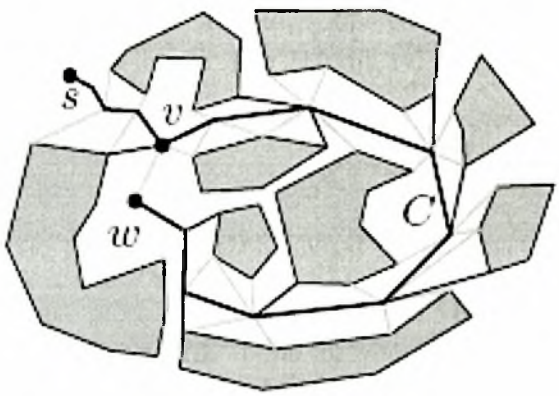
4. Υπολογίζουμε τις συνεκτικές συνιστώσες $H_1,\ldots..,H_k$ του G\P'. Για κάθε συνιστώσα H_i

   βρίσκουμε μία κορυφή $\upsilon_i \in P'$,την πιο μακρινή δυνατή από την s κατά μήκος του P' τέτοια ώστε να υπάρχει μία ακμή $\{u_i,\upsilon_i\}$, $u_i \in H_i$. Οι συνεκτικές συνιστώσες $H_1,\ldots..,H_k$ μπορούν να υπολογιστούν σε O(sort(N)) I/Os. Βρίσκουμε τις κορυφές $\upsilon_1,\ldots\ldots,\upsilon_k$ σε O(sort(N)) I/Os έτσι: Αρχικά σημειώνουμε κάθε κορυφή του P' μαζί με την απόστασή της από την s κατά μήκος του P'. Αυτές οι αποστάσεις μπορούν να υπολογισθούν σε O(sort(N)) I/Os με την τεχνική του κύκλου του Euler και την ταξινόμηση λίστας. Μετά χρησιμοποιούμε την πράξη (Γ),υπολογισμός προσκείμενων ακμών, στο V(P') και στο

E(G) για να βρούμε όλες τις ακμές στο E(G)\E(P') που προσπίπτουν στο P'(δηλαδή θα καταλήξουμε να έχουμε όλες τις ακμές του γραφήματος που προσπίπτουν στο P' εκτός από αυτές που ανήκουν στο P'). Ταξινομούμε το παραχθέν σύνολο ακμών έτσι ώστε ακμή $\{υ,w\}, υ \in H_i$, $w \in P'$ να προηγείται ακμής $\{x,y\}, x \in H_j, y \in P'$,εάν είτε $i<j$ ή $i=j$ και η απόσταση από την s στην w δεν είναι μεγαλύτερη από την απόσταση από την s στην y. Ισοβαθμίες αντιμετωπίζονται τυχαία. Σαρώνουμε την λίστα στην οποία καταλήξαμε και εξάγουμε για κάθε συνιστώσα H_i την τελευταία ακμή $\{u_i,υ_i\}$, $u_i \in H_i$ αυτής της λίστας.

5 . Υπολογίζουμε, επαναληπτικά, τα $T_1,\ldots T_κ$, DFS δέντρα για τις συνεκτικές συνιστώσες

$H_1,\ldots,H_k$ των οποίων οι ρίζες είναι οι κορυφές $u_1,\ldots,u_k$ αντίστοιχα και κατασκευάζουμε το συνολικό DFS δέντρο T για το γράφημα G ενώνοντας τα δέντρα $T_1,\ldots T_κ$,το μονοπάτι P' και τις ακμές $\{u_i,υ_i\}$, $1 \leq i \leq k$.Σημειώνουμε πως οι συνιστώσες $H_1,\ldots,H_k$ δεν είναι κατ' ανάγκη και δισυνεκτικές.



Εικόνα 3.1 : Το μονοπάτι P' είναι σημειωμένο με την σκούρα γραμμή. Οι συνεκτικές συνιστώσες του G\P' είναι σκιαγραφημένες με σκούρο γκρι. Οι μεσαίες ακμές είναι οι $\{u_i,υ_i\}$.Οι πολύ ανοιχτόχρωμες ακμές είναι οι ακμές που δεν ανήκουν στο δέντρο T.

Για να αποδείξουμε την ορθότητα του αλγορίθμου μας πρέπει να αποδείξουμε ότι το T ,είναι πράγματι ένα DFS δέντρο για το γράφημα G .Για να το κάνουμε αυτό η επόμενη ταξινόμηση των ακμών E(G) \ E(T) είναι χρήσιμη : Μία ακμή e = (u,υ) στο E(G) \ E(T) καλείται οπισθοακμή εάν η u είναι ένας πρόγονος της υ στο T ή αντίθετα στην άλλη περίπτωση η ακμή e καλείται ακμή διασταύρωσης. Αποδεικνύεται ότι ένα επικαλύπτον δέντρο T ενός γραφήματος G είναι ένα DFS δέντρο του G ένα και μόνο αν όλες οι ακμές στο E(G) \ E(T) είναι οπισθοακμές.

ΛΗΜΜΑ 3.1:

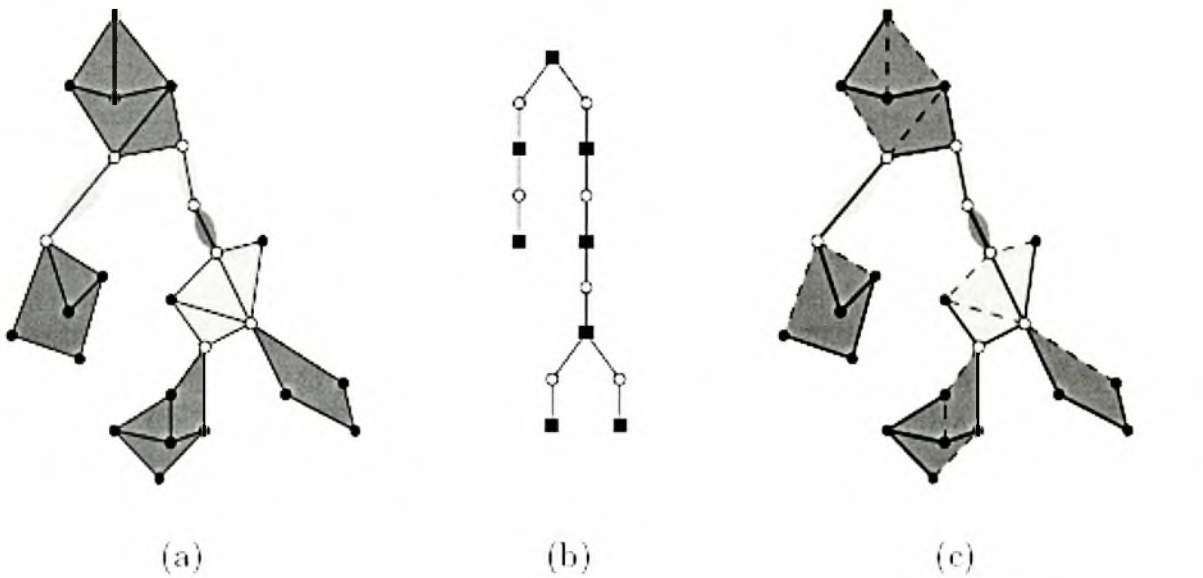
Το δέντρο T που κατασκευάστηκε από τον παραπάνω αλγόριθμο είναι ένα DFS δέντρο του γραφήματος G.

Απόδειξη:

Είναι εύκολο να κατανοήσουμε ότι το T είναι ένα επικαλύπτον δέντρο του G. Για να αποδείξουμε ότι το T είναι ένα DFS δέντρο πρέπει να δείξουμε ότι όλες οι ακμές που δεν ανήκουν στο δέντρο είναι οπισθοακμές. Αρχικά σημειώνουμε πως δεν υπάρχουν ακμές μεταξύ των συνιστωσών $H_1,\ldots..,H_k$.Όλες οι ακμές που δεν ανήκουν στο δέντρο T και έχουν και τα δύο άκρα τους στην ίδια συνιστώσα H_i είναι οπισθοακμές γιατί το δέντρο T_i είναι DFS δέντρο της H_i (το T_i είναι DFS δέντρο της H_i άρα όσες ακμές δεν ανήκουν σε αυτό και ανήκουν στην H_i είναι οπισθοακμές για το T_i και κατά συνέπεια είναι οπισθοακμές και για το T.). Όλες οι ακμές που δεν ανήκουν στο δέντρο T και έχουν και τα δύο άκρα τους στο P' είναι οπισθοακμές γιατί το P' είναι μονοπάτι. Για κάθε ακμή που δεν ανήκει στο T $\{\upsilon,w\}$ με $\upsilon \in P'$ και $w \in H_i$, η w είναι απόγονος της ρίζας u_i του DFS δέντρου T_i .Το δέντρο T_i συνδέεται με το P' μέσω της ακμής $\{\upsilon_i,u_i\}$. Από την επιλογή της κορυφής υ_i ,η υ είναι πρόγονος της υ_i και έτσι είναι και πρόγονος των u_i και w. Έτσι η ακμή $\{\upsilon,w\}$ είναι οπισθοακμή. ¤

Στην παραπάνω περιγραφή βασιστήκαμε στην υπόθεση ότι το γράφημά μας ήταν δισυνεκτικό. Στην περίπτωση που κάτι τέτοιο δεν ισχύει ,βρίσκουμε τις δισυνεκτικές συνιστώσες του G, υπολογίζουμε τα DFS δέντρα όλων των δισυνεκτικών συνιστωσών και ενώνουμε αυτά τα δέντρα στις κορυφές αποκοπής του γραφήματος. Ακριβέστερα υπολογίζουμε το δισυνεκτικό-με κορυφές αποκοπής-δέντρο (bicomp-cutpoint-tree) T_G του G που περιέχει όλες τις κορυφές αποκοπής του γραφήματος και μία κορυφή υ(C) για κάθε δισυνεκτική συνιστώσα C .Υπάρχει μία ακμή μεταξύ μίας κορυφής αποκοπής υ και μιας κορυφής δισυνεκτικής συνιστώσας υ(C) εάν η υ περιέχεται στην συνιστώσα C. Επιλέγουμε την κορυφή δισυνεκτικής συνιστώσας $\upsilon(C_r)$ που

αναφέρεται στην δισυνεκτική συνιστώσα C_r που περιέχει την κορυφή s σαν ρίζα του δέντρου T_G. Ο πατέρας κορυφή αποκοπής (parent cutpoint) μιας δισυνεκτικής συνιστώσας $C \neq C_r$ είναι ο πατέρας p(υ(C)) της υ(C) στο T_G. Το T_G μπορεί να κατασκευαστεί σε O(sort(N)) I/Os χρησιμοποιώντας αλγορίθμους που παρουσιάζονται ως βασικές εφαρμογές γραφημάτων στην εισαγωγή. Υπολογίζουμε ένα DFS δέντρο της C_r με ρίζα την κορυφή s. Για κάθε δισυνεκτική συνιστώσα $C \neq C_r$, υπολογίζουμε το DFS δέντρο με ρίζα τον πατέρα κορυφή αποκοπής της C. Η ένωση των παραχθέντων DFS δέντρων είναι το DFS δέντρο για το γράφημα G με ρίζα την s αφού δεν υπάρχουν ακμές μεταξύ διαφορετικών δισυνεκτικών συνιστωσών. Έτσι πετύχαμε το πρώτο μας ολοκληρωμένο αποτέλεσμα.



Εικόνα 3.2 : (a)Ένα συνεκτικό γράφημα G με σημειωμένες τις δισυνεκτικές συνιστώσες του. Οι κορυφές αποκοπής είναι οι άσπρες. Οι άλλες κορυφές είναι μαύρες. (b)Το bicomp-cutpoint δέντρο του G. Οι δισυνεκτικές κορυφές είναι οι τετράγωνες. (c)Το DFS δέντρο του G,ενώνοντας τα DFS δέντρα των δισυνεκτικών συνιστωσών του.

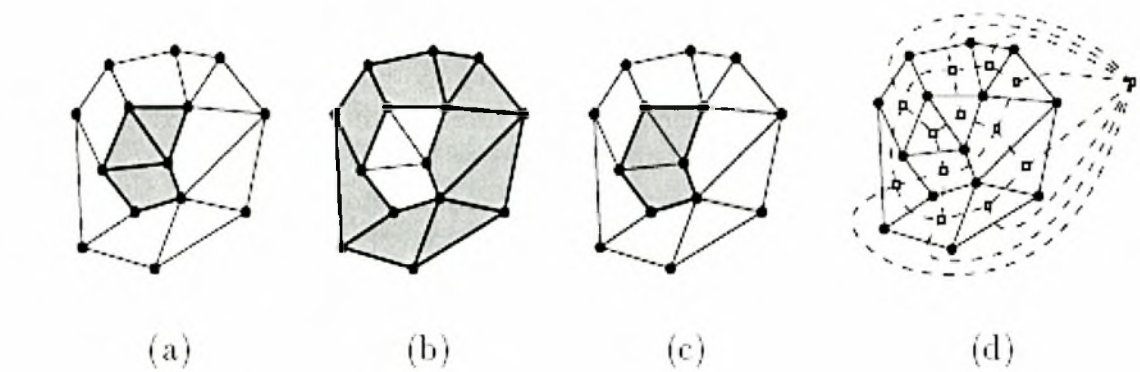
ΘΕΩΡΗΜΑ 3.1:

Ένα DFS δέντρο ενός επίπεδου χωρίς ακμές διασταύρωσης γραφήματος μπορεί να υπολογιστεί σε O(sort(N)log(N/M)) I/O πράξεις και σε γραμμικό χώρο.

3.3 Βρίσκοντας τον απλό κύκλο διαχωριστή

Σε αυτή την ενότητα θα παρουσιάσουμε πώς να υπολογίζουμε έναν απλό κύκλο $\frac{2}{3}$ – διαχωριστή ενός επίπεδου χωρίς ακμές διασταύρωσης γραφήματος. Αρχίζουμε παρουσιάζοντας την απαραίτητη ορολογία.

Δοθέντος ενός επίπεδου χωρίς ακμές διασταύρωσης γραφήματος G, οι όψεις (faces) του G είναι συνδεδεμένες περιοχές του $R^2 \backslash G$. Όταν ένα γράφημα είναι σχεδιασμένο χωρίς γραμμές διασταύρωσης τότε κάθε κύκλος που περικλείει μία περιοχή χωρίς καμία ακμή να ξεκινάει από τον κύκλο και να πηγαίνει προς το εσωτερικό της περιοχής ονομάζεται face,όψη. Χρησιμοποιούμε το γράμμα F για να ορίσουμε το σύνολο των όψεων του γραφήματος. Το όριο μίας όψης f είναι το σύνολο των ακμών που περιέχονται στην κλειστότητα της f. Για ένα σύνολο F' των όψεων του G αποδίδουμε στο $G_{F'}$ να είναι το υπογράφημα του G που ορίζεται ως η ένωση των ορίων των όψεων που ανήκουν στο F'. Το συμπλήρωμα $\overline{G_{F'}}$ του $G_{F'}$ είναι το γράφημα που προκύπτει από την ένωση των ορίων όλων των faces του F\F'.Το όριο του $G_{F'}$ είναι η τομή μεταξύ του $G_{F'}$ και του συμπληρώματός του. Το διπλό (dual) G^* του G είναι το γράφημα που περιέχει μία κορυφή f^* για κάθε face $f \in F$, και μία ακμή μεταξύ των κορυφών f_1^* και f_2^* εάν οι όψεις f_1 και f_2 μοιράζονται μία ακμή στο γράφημα G. Το dual G^* ενός επίπεδου γραφήματος είναι επίσης επίπεδο γράφημα και μπορεί να υπολογιστεί σε O(sort(N)) I/Os.



Εικόνα 3.3: (a)Ένα γράφημα G και ένα σύνολο όψεων F' σκιαγραφημένο γκρι. Οι ακμές του $G_{F'}$ είναι οι σκούρες γραμμές. (b)Οι σκιαγραφημένες όψεις είναι όλες οι φάσεις του F\F'.Οι σκούρες ακμές είναι οι ακμές που ανήκουν στο συμπλήρωμα του $G_{F'}$. (c)Το όριο του $G_{F'}$ φαίνεται με τις σκούρες γραμμές. (d)Το G^* παριστάνεται με τα άσπρα τετράγωνα και τις διακεκομμένες γραμμές.

Η ιδέα του αλγορίθμου μας είναι να βρούμε ένα σύνολο των faces F' υποσύνολο του F τέτοιο ώστε το όριο του $G_{F'}$ να είναι ένας απλός κύκλος $\frac{2}{3}$ – διαχωριστής. Η κύρια δυσκολία είναι το να βεβαιωθούμε ότι το όριο του $G_{F'}$ είναι ένας απλός κύκλος .Υπολογίζουμε το F'έτσι:

1. Έλεγχος για μεγάλες όψεις.

   Ελέγχουμε εάν υπάρχει μια απλή όψη της οποίας το όριο να έχει μέγεθος τουλάχιστον $\frac{|V|}{3}$. Εάν βρούμε μία τέτοια όψη αναφέρουμε το όριό της ως τον διαχωριστή C. Αυτό γίνεται γιατί δεν υπάρχει καμία κορυφή μέσα στον C και ο ίδιος ο C έχει τουλάχιστον $\frac{|V|}{3}$ κορυφές, άρα έξω από τον κύκλο θα υπάρχουν το πολύ τα άλλα $\frac{2}{3}$ του |V|.

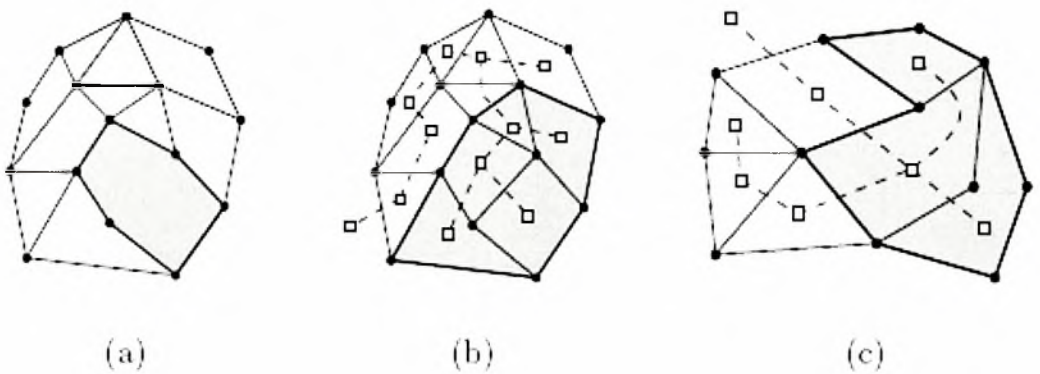
2. Έλεγχος για μεγάλα υποδέντρα.

   Εάν δεν υπάρχει καμία μεγάλη όψη, αν το βήμα 1 δηλαδή αποτύχει, υπολογίζουμε ένα επικαλύπτον δέντρο T^* του G^* και επιλέγουμε αυθαίρετα μία κορυφή r ως την κορυφή του. Κάθε κορυφή $\upsilon \in T^*$ ορίζει ένα υποδέντρο $T^*(\upsilon)$ του T^* το οποίο περιέχει την υ και όλους τους απογόνους της. Οι κορυφές αυτού του υποδέντρου αναφέρονται σε ένα σύνολο όψεων του G των οποίων τα όρια ορίζουν ένα γράφημα G(υ). Το όριο του G(υ) είναι απλός κύκλος. Προσπαθούμε να βρούμε μία κορυφή υ τέτοια ώστε $\frac{1}{3}|V| \leq |G(\upsilon)| \leq \frac{2}{3}|V|$, όπου |G(υ)| είναι το πλήθος των κορυφών στο G(υ). Εάν επιτύχουμε τότε αναφέρουμε το όριο του G(υ) σαν τον κύκλο διαχωριστή C.

3. Διαμοιράζοντας ένα μεγάλο υποδέντρο.

   Εάν τα βήματα 1 και 2 αποτύχουν να παράγουν έναν απλό κύκλο $\frac{2}{3}$ – διαχωριστή του γραφήματος G ,βρισκόμαστε σε μία κατάσταση όπου για κάθε φύλλο $l \in T^*$ ισχύει $|G(l)| < \frac{1}{3}|V|$(εφόσον έχει αποτύχει το βήμα 1). Για την ρίζα r του T^* έχουμε $|G(r)|=|V|$(εφόσον το T^* περιέχει μία κορυφή για κάθε φάση, το αντίστοιχο υπογράφημα θα περιέχει όλες τις φάσεις άρα όλες τις κορυφές.). Για κάθε άλλη κορυφή $\upsilon \in T^*$, είτε $|G(\upsilon)| < \frac{1}{3}|V|$, είτε $|G(\upsilon)| > \frac{2}{3}|V|$,(εφόσον έχει αποτύχει το βήμα 2). Έτσι θα πρέπει να

υπάρχει μία κορυφή υ με $|G(\upsilon)| > \frac{2}{3}|V|$ και $|G(w_i)| < \frac{1}{3}|V|$,για όλα τα παιδία $w_1,\ldots,w_k$ της υ. Υπολογίζουμε ένα υπογράφημα G'του G(υ) που αποτελείται από το όριο της όψης υ^* και ένα υποσύνολο των γραφημάτων $G(w_1),\ldots\ldots,G(w_k)$ τέτοιο ώστε $\frac{1}{3}|V| \leq |G'| \leq \frac{2}{3}|V|$,και το όριο του G' είναι απλός κύκλος.



Εικόνα 3.4: (a)Μία μεγάλη όψη. (b)Ένα μεγάλο υποδέντρο. (c)Διαμοιράζοντας ένα μεγάλο υποδέντρο.

Όλα τα παραπάνω βήματα μπορούν να εκτελεστούν σε O(sort(N)) I/Os.

ΘΕΩΡΗΜΑ 3.2:

Ένας απλός κύκλος $\frac{2}{3}$ – διαχωριστής ενός δισυνεκτικού επίπεδου γραφήματος χωρίς ακμές διασταύρωσης μπορεί να υπολογιστεί σε O(sort(N)) I/O πράξεις και γραμμικό χώρο.

Για τα παραπάνω βήματα ,αναλυτικά ,έχουμε:

3.3.1 Έλεγχος για μεγάλες όψεις

Για να ελέγξουμε πότε υπάρχει μία όψη f στο G που έχει όριο με μέγεθος το λιγότερο $\frac{1}{3}|V|$, αναπαριστούμε κάθε όψη του G ως μία λίστα με τις κορυφές που βρίσκονται κατά μήκος του ορίου τους. Ο υπολογισμός μιας τέτοιας αναπαράστασης χρειάζεται O(sort(N)) πράξεις εισόδου/εξόδου. Στη συνέχεια σαρώνουμε αυτές τις λίστες για να βρούμε κάποια από αυτές που να έχει μήκος το λιγότερο $\frac{1}{3}|V|$. Συνολικά αυτό το βήμα απαιτεί O(sort(N)) I/Os.

3.3.2 Έλεγχος για μεγάλα υποδέντρα

Πρώτα θα αποδείξουμε ότι το όριο του G(υ) ,που ορίζεται από τις κορυφές του $T^*(υ)$ είναι ένας απλός κύκλος. Μελετούμε ένα υποσύνολο F' των όψεων του αρχικού επίπεδου γραφήματος G, και ορίζουμε ως H το υπογράφημα του G που είναι η ένωση των ορίων των όψεων του F'. Το H^* είναι το dual του H. Αποκαλούμε το γράφημα H σταθερό εάν το H^* είναι συνεκτικό. Καθώς για κάθε κορυφή $υ \in T^*$ ισχύει πως τα $T^*(υ)$ και $T^* \setminus T^*(υ)$ είναι συνεκτικά συμπεραίνουμε πως τα G(υ) και το συμπλήρωμά του είναι και τα δύο σταθερά .Χρησιμοποιώντας το παρακάτω λήμμα συνεπάγεται πως το όριο του G(υ) είναι ένας απλός κύκλος.

ΛΗΜΜΑ 3.2:

Το G' είναι ένα υπογράφημα του δισυνεκτικού επίπεδου γραφήματος G.Το όριο του G' είναι ένας απλός κύκλος αν και μόνο αν το G' και το συμπλήρωμά του είναι και τα δύο σταθερά.

Η κύρια δυσκολία για να βρούμε μία κορυφή $υ \in T^*$ τέτοια ώστε $\frac{1}{3}|V| \leq |G(υ)| \leq \frac{2}{3}|V|$, είναι ο υπολογισμός των μεγεθών |G(υ)| των γραφημάτων G(υ) για όλες τις κορυφές $υ \in T^*$. Όταν αυτή η πληροφορία υπολογιστεί τότε μία απλή σάρωση των κορυφών του συνόλου T^* είναι αρκετή για να αποφασίσουμε πότε υπάρχει μία κορυφή $υ \in T^*$ με $\frac{1}{3}|V| \leq |G(υ)| \leq \frac{2}{3}|V|$. Καθώς $|T^*| =$ O(N) για τον εντοπισμό αυτό χρειαζόμαστε O(scan(N)) I/Os. Δοθείσας μίας κορυφής υ, οι κορυφές $T^*(υ)$ μπορούν να αναφερθούν σε O(sort(N)) I/Os χρησιμοποιώντας στάνταρ υπολογισμούς δέντρων. Όταν έχουμε αυτές τις κορυφές μπορούμε να εφαρμόσουμε την πράξη

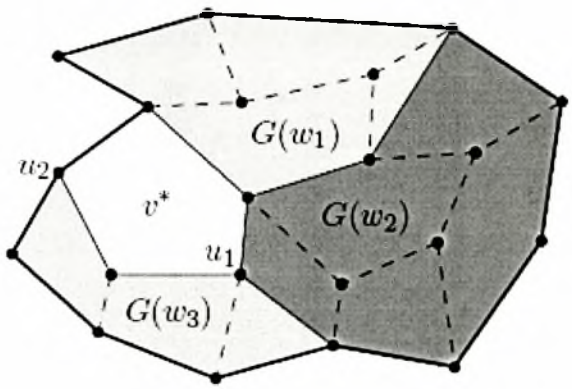
(Γ)-υπολογισμός προσκείμενων ακμών από τις βασικές εφαρμογές γραφημάτων για να βρούμε το σύνολο Ε' των ακμών στο G^* που έχουν ακριβώς ένα άκρο στο $T^*(υ)$. Το σύνολο $\{e^*:e \in E'\}$είναι το όριο του G(υ) .Αυτό που απομένει είναι να περιγράψουμε πως υπολογίζουμε τα μεγέθη των γραφημάτων G(υ) αποδοτικά.

Υποθέτουμε ότι κάθε κορυφή $υ \in T^*$ σώζει έναν αριθμό $|υ^*|$ κορυφών στο όριο της φάσης $υ^*$.Η βασική ιδέα του αλγορίθμου μας για τον υπολογισμό των |G(υ)| είναι να αθροίσουμε τα $|w^*|$ για όλους τους απογόνους w του υ στο T^*. Αυτό μπορεί να γίνει με επεξεργασία του T^* ανά επίπεδο από κάτω προς τα πάνω και υπολογίζοντας για κάθε κορυφή υ, την τιμή $|G(υ)|=|υ^*| +\sum_{i=1}^{k} |G(w_i)|$, όπου $w_1....w_k$ είναι τα παιδία της υ. Ωστόσο ,κάνοντας αυτό κάποιες κορυφές υπολογίζονται περισσότερες από μία φορές. Παρακάτω θα δείξουμε πώς να μετατρέψουμε την παραπάνω ιδέα έτσι ώστε να είμαστε βέβαιοι ότι κάθε κορυφή έχει υπολογιστεί μόνο μία φορά.

Ορίζουμε τον χαμηλότερο κοινό πρόγονο (Lowest common ancestor–LCA(e)) μιας ακμής $e \in G$, να είναι ο χαμηλότερος κοινός γείτονας των άκρων της διπλής της ακμής στο T^*. Για κάθε κορυφή $υ \in T^*$,ορίζουμε με Ε(υ) το σύνολο των ακμών στο G των οποίων οι διπλές ακμές έχουν την υ σαν τον χαμηλότερο κοινό τους γείτονα. Για μία κορυφή υ με παιδιά τα $w_1,w_2,....,w_k$, το Ε(υ) αποτελείται από όλες τις ακμές στο όριο μεταξύ των $υ^*$ και των γραφημάτων $G(w_1),G(w_2),...,G(w_k)$, όπως επίσης και από τις ακμές στο όριο μεταξύ των γραφημάτων $G(w_1),G(w_2),...,G(w_k)$. Κάθε άκρο κάθε τέτοιας ακμής περιέχεται σε περισσότερα από ένα υπογραφήματα του G(υ) ,και έτσι υπολογίζεται περισσότερες από μία φορές με την παραπάνω προσέγγιση. Η ιδέα στην μετατροπή μας είναι να ορίσουμε έναν υπερμετρητή $c_{υ,u}$,για κάθε άκρο u μίας ακμής του Ε(υ), ο οποίος θα είναι κατά ένα μικρότερος από τον αριθμό των φορών που η κορυφή u υπολογίστηκε στο άθροισμα $S=|υ^*| +\sum_{i=1}^{k} |G(w_i)|$. Έπειτα το άθροισμα αυτών των υπερμετρητών αφαιρείται από το S έτσι ώστε να έχουμε τη σωστή τιμή για το |G(υ)|.

Αναθέτουμε στο V(υ) να ορίζει το σύνολο των άκρων των ακμών στο Ε(υ). Μία κορυφή u που ανήκει στο V(υ) μετριέται μία φορά για καθένα από τα υπογραφήματα $\{υ^*,G(w_1),G(w_2),...,G(w_k)\}$ που έχουν την u στο όριο τους. Ορίζουμε με l τον αριθμό των ακμών του Ε(υ) που είναι προσκείμενες στην u. Κάθε τέτοια ακμή είναι μέρος του ορίου μεταξύ δύο υπογραφημάτων από τα $υ^*,G(w_1),G(w_2),...,G(w_k)$. Έτσι αν η u είναι μία εσωτερική κορυφή του G(υ) (δηλαδή δεν είναι στο όριό του) τότε υπάρχουν l τέτοια υπογραφήματα και η u υπολογίζεται l φορές. Στην αντίθετη περίπτωση ,δηλαδή αν η u είναι στο όριο του G(υ) ,έπεται από την σταθερότητα του G(υ) και του συμπληρώματος του ,ότι δύο από τις ακμές του G(υ) ,που είναι προσκείμενες στην υ είναι στο όριο του G(υ). Έτσι l+1 από τα υπογραφήματα $υ^*,G(w_1),G(w_2),...,G(w_k)$ περιέχουν την u και η u υπολογίζεται l+1 φορές. Άρα ο υπερμετρητής $c_{υ,u}$ για την κορυφή $u \in V(υ)$ ορίζεται ως εξής:

$$c_{v,u} = \begin{cases} l-1, & \text{αν όλες οι προσκείμενες ακμές της } u \text{ έχουν τον } LCA \text{ τους στο } T^*(v) \\ l, & \text{αλλιώς} \end{cases}$$



Εικόνα 3.4: Το όριο του G(υ) είναι σχεδιασμένο με σκούρα γραμμή. Οι LCAs αυτών των ακμών είναι πρόγονοι της υ στο T^*. Οι λεπτές ενιαίες ακμές είναι αυτές στο όριο μεταξύ των γραφημάτων $υ^*$,$G(w_1)$,$G(w_2)$ και $G(w_3)$.Ο LCA αυτών των ακμών στο T^* είναι η υ. Οι LCAs των διακεκομμένων ακμών είναι απόγονοι της $υ^*$. Η κορυφή u_1 υπολογίζεται τρείς φορές στο άθροισμα $S=|υ^*|+|G(w_1)|+|G(w_2)|+|G(w_k)|$ επειδή είναι στα $υ^*$, $G(w_2)$, $G(w_3)$. Έχει τρείς προσκείμενες κορυφές με LCA την υ και όλες οι ακμές οι προσκείμενες στην u_1 έχουν τον LCA τους στο $T^*(υ)$. Έτσι ,ο υπερμετρητής c_{v,u_1} θα είναι 2, έτσι ώστε με την αφαίρεση του c_{v,u_1} από το S η κορυφή u_1 υπολογίζεται μία φορά. Η κορυφή u_2 υπολογίζεται στο S δύο φορές. Επειδή είναι στο $υ^*$ και στο $G(w_3)$. Έχει μία προσκείμενη ακμή με LCA την υ, αλλά δεν έχουν όλες οι προσκείμενες ακμές της τον LCA τους στο $T^*(υ)$ (είναι στο όριο του G(υ)). Έτσι ,ο υπερμετρητής c_{v,u_2} θα είναι 1 έτσι ώστε με την αφαίρεση του c_{v,u_2} από το S η κορυφή u_2 υπολογίζεται μόνο μία φορά.

ΛΗΜΜΑ 3.3:

Για κάθε κορυφή $\upsilon \in T^*$,

$|G(\upsilon)|=$

$$\begin{cases} |\upsilon^*| + \sum_{i=1}^{k} |G(w_i)| - \sum_{u \in V(\upsilon)} c_{\upsilon,u}, & \text{εάν η υ είναι εσωτερική κορυφή με παιδιά w1, ..., wk} \\ |\upsilon^*| & \text{, εάν η υ είναι φύλλο} \end{cases}$$

Απόδειξη:

Το λήμμα προφανώς ισχύει για τα φύλλα του T^* .Για να αποδείξουμε ότι το λήμμα ισχύει και για μία εσωτερική κορυφή υ του T^* ,πρέπει να δείξουμε ότι υπολογίζουμε κάθε κορυφή στο G(υ) ακριβώς μία φορά στο άθροισμα $|\upsilon^*| + \sum_{i=1}^{k} |G(w_i)| - \sum_{u \in V(\upsilon)} c_{\upsilon,u}$. Μία κορυφή στο G(υ)\V(υ)υπολογίζεται μία φορά ,καθώς περιέχεται σε μόνο ένα από τα γραφήματα υ^*,$G(w_1)$,$G(w_2)$,…,$G(w_k)$. Μία κορυφή u ∈ V(υ) συμπεριλαμβάνεται στο άθροισμα $|\upsilon^*|+\sum_{i=1}^{k} |G(w_i)|$ μια φορά για κάθε γράφημα υ^* ή $G(w_i)$ που την περιέχει. Εάν όλες οι ακμές που είναι προσκείμενες στην u έχουν τον LCA τους στο $T^*(\upsilon)$, τότε όλες οι όψεις γύρω από την u είναι στο G(υ).Έτσι η u είναι μία εσωτερική κορυφή του G(υ). Όπως συζητήθηκε παραπάνω ,η u υπολογίζεται l φορές σε αυτή την περίπτωση, όπου l είναι ο αριθμός των ακμών στο E(υ) που είναι προσκείμενες στην u . Έτσι ο υπερμετρητής είναι l-1 και πετυχαίνουμε την ακριβή μέτρηση αφαιρώντας το $c_{\upsilon,u}$= l-1. Αλλιώς η u είναι στο όριο του G(υ) και όπως συζητήθηκε παραπάνω υπολογίζεται l-1 φορές. Έτσι πετυχαίνουμε τον σωστό υπολογισμό αφαιρώντας το $c_{\upsilon,u}$= l. ¤

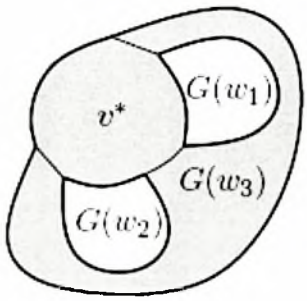
Τώρα είμαστε έτοιμοι να δείξουμε πως θα υπολογίσουμε με αποδοτικό τρόπο από πλευράς πράξεων εισόδου/εξόδου τα |G(υ)| για κάθε υ ∈ T^*. Υποθέτοντας ότι κάθε κορυφή, υ ∈ T^*, σώζει τα $|\upsilon^*|$ και $\sum_{u \in V(\upsilon)} c_{\upsilon,u}$, τα μεγέθη των γραφημάτων |G(υ)| , με υ ∈ T^*, μπορούν να υπολογιστούν χρησιμοποιώντας O(sort(N)) I/Os. Για τα φύλλα του T^* ,αρχικοποιούμε $|G(\upsilon)|=|\upsilon^*|$. Έπειτα επεξεργαζόμαστε το T^* ανά επίπεδο, από τα φύλλα προς τη ρίζα και υπολογίζουμε για κάθε εσωτερική κορυφή υ με παιδιά $w_1,\ldots,w_k$, $|G(\upsilon)|= |\upsilon^*| + \sum_{i=1}^{k} |G(w_i)| - c_\upsilon$. Απομένει να δείξουμε πως υπολογίζουμε τα c_υ, υ ∈ T^*.

Από τον ορισμό των υπερμετρητών $c_{v,u}$, $c_v = 2|E(v)| - |V'(v)|$, όπου $V'(v)$ είναι το σύνολο των κορυφών u ∈ V(υ) έτσι ώστε όλες οι ακμές στο G που είναι προσκείμενες στις u να έχουν τους LCAs στο $T^*(v)$. Για να υπολογίσουμε τα σύνολα E(υ), για όλες τις υ ∈ T^* ,υπολογίζουμε τους LCAs για όλες τις ακμές στο G. Αυτό μπορεί να γίνει σε O(sort(N)) I/Os επειδή υπάρχουν O(N) ακμές στο G και O(N) κορυφές στο T^*. Ταξινομώντας τις ακμές του G βάσει των LCAs τους πετυχαίνουμε την αλληλουχία των λιστών E(υ), υ ∈ T^*, την οποία και σαρώνουμε για να προσδιορίσουμε τα |E(υ)| ,για όλες τις υ ∈ T^*. Για να υπολογίσουμε τα σύνολα V'(υ) για όλες τις υ ∈ T^*, βρίσκουμε για κάθε κορυφή u ∈ G, την προσκείμενη ακμή στην u της οποίας ο LCA είναι κοντινότερος στην ρίζα. Αποκαλούμε τον LCA αυτής της κορυφής ως τον μέγιστο_LCA της u. Ταξινομώντας τις κορυφές στο G βάσει των μέγιστων_LCAs τους αποκτούμε την αλληλουχία των λιστών V'(υ) , υ ∈ T^*, την οποία σαρώνουμε για να προσδιορίσουμε τα |V'(υ)|,για όλες τις υ ∈ T^*.

3.3.3 Διαμοιράζοντας ένα μεγάλο υποδέντρο

Εάν κανένα από τα προηγούμενα δύο βήματα δεν καταφέρει να προσδιορίσει έναν απλό κύκλο $\frac{2}{3}$-διαχωριστή του G, τότε θα πρέπει να διαχειριστούμε την περίπτωση όπου καμία κορυφή υ ∈ T^*, δεν ικανοποιεί τη σχέση: $\frac{1}{3}|V| \leq |G(v)| \leq \frac{2}{3}|V|$.Σε αυτή την περίπτωση θα πρέπει να υπάρχει μία κορυφή υ ∈ T^*, με παιδία $w_1,\ldots,w_k$, τέτοια ώστε $|G(v)| > \frac{2}{3}|V|$ και $|G(w_i)| < \frac{1}{3}|V|$, για $1 \leq i \leq k$. Ο στόχος μας είναι να υπολογίσουμε ένα υπογράφημα του G(υ) που να αποτελείται από το όριο του v^* και από ένα υποσύνολο των γραφημάτων $G(w_i)$, τέτοιο ώστε το μέγεθός του να είναι μεταξύ $\frac{1}{3}|V|$ και $\frac{2}{3}|V|$ και το όριο του να είναι ένας απλός κύκλος C.

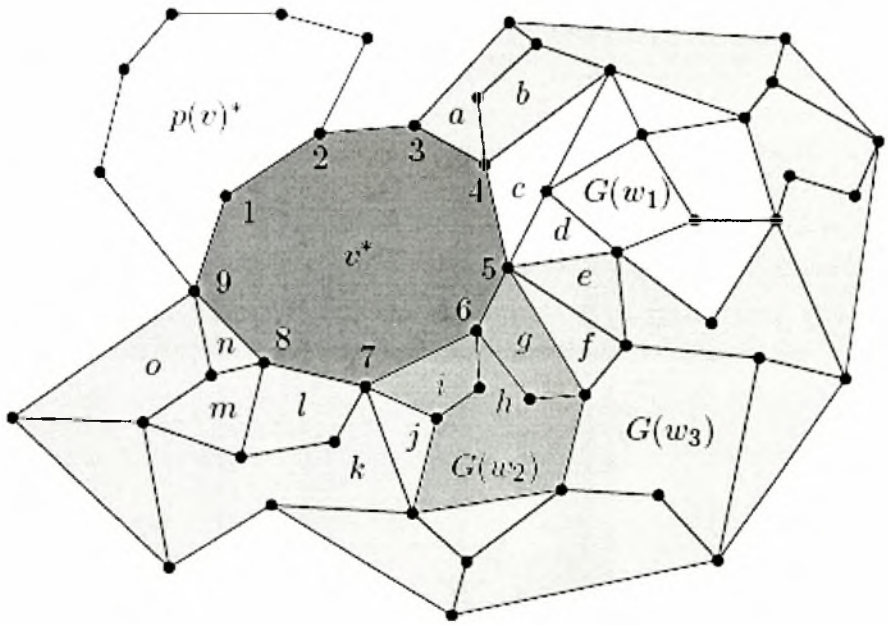
Υπάρχει η πεποίθηση ότι το όριο του γραφήματος που ορίζεται από το v^* και από ένα οποιοδήποτε υποσύνολο των γραφημάτων $G(w_i)$ είναι ένας απλός κύκλος. Δυστυχώς, όμως ,όπως μπορούμε να δούμε και στην παρακάτω εικόνα κάτι τέτοιο δεν ισχύει γενικά σε όλες τις περιπτώσεις.



Εικόνα 3.5: Το όριο του $\upsilon^* \cup G(w_3)$ δεν είναι ένας απλός κύκλος.

Ωστόσο ,όπως θα δείξουμε παρακάτω, μπορούμε να υπολογίσουμε μία αντιμετάθεση σ:[1,k]→[1,k] τέτοια ώστε το όριο κάθε γραφήματος που αποκτούμε με τη σταδιακή πρόσθεση των γραφημάτων $G(w_{\sigma(1)}),\ldots, G(w_{\sigma(k)})$ στην όψη υ^* να είναι ένας απλός κύκλος. Τυπικότερα ,ορίζουμε τα γραφήματα $H_\sigma(1),\ldots, H_\sigma(k)$ ως $H_\sigma(i)= \upsilon^* \cup \bigcup_{j=1}^{i} G(w_{\sigma(j)})$. Έπειτα θα δείξουμε ότι τα $H_\sigma(i)$ και $\overline{H\sigma(i)}$ είναι και τα δύο σταθερά για όλα τα $1 \leq i \leq k$. Έτσι από το ΛΗΜΜΑ 3.2 συνεπάγεται ότι το όριο του $H_\sigma(i)$ είναι ένας απλός κύκλος. Καθώς έχουμε ήδη υπολογίσει τα μεγέθη $|\upsilon^*|$ των όψεων υ^*, και τα μεγέθη $|G(w_1)|,\ldots| G(w_k)|$ των γραφημάτων $G(w_1),\ldots, G(w_k)$, τα μεγέθη $|H_\sigma(1)|,\ldots,|H_\sigma(k)|$ των γραφημάτων $H_\sigma(1),\ldots, H_\sigma(k)$ μπορούν να υπολογιστούν εφαρμόζοντας μία διαδικασία παρόμοια με αυτή που χρησιμοποιήθηκε στις προηγούμενες ενότητες για τον υπολογισμό των μεγεθών $|G(\upsilon)|$ των γραφημάτων $G(\upsilon)$, $\upsilon \in T^*$. Καθώς $|G(\upsilon)| > \frac{2}{3}|V|$ και $|G(w_i)|< \frac{1}{3}|V|$, για $1 \leq i \leq k$, θα πρέπει να υπάρχει γράφημα $H_\sigma(i)$ τέτοιο ώστε $\frac{1}{3}|V| \leq |H_\sigma(i)| \leq \frac{2}{3}|V|$. Απομένει να δείξουμε πως υπολογίζουμε αποδοτικά από πλευράς πράξεων εισόδου/εξόδου την πράξη αντιμετάθεσης σ.

Για να κατασκευάσουμε την σ εξάγουμε το $G(\upsilon)$ από το G και προσδιορίζουμε κάθε όψη του $G(w_i)$ με i, και όλες τις υπόλοιπες όψεις του $G(\upsilon)$ με 0.Αυτός ο προσδιορισμός μπορεί να γίνει με O(sort(N)) πράξεις εισόδου/εξόδου με την επεξεργασία του $T^*(\upsilon)$ από τη ρίζα προς τα φύλλα. Έπειτα οι ετικέτες (προσδιορισμοί)των δύο όψεων κάθε πλευράς μιας ακμής του $G(\upsilon)$ γίνονται η ετικέτα της συγκεκριμένης ακμής .Δοθέντων των παραπάνω ετικετών των όψεων του $G(\upsilon)$ (ή των κορυφών του $T^*(\upsilon)$) ο προσδιορισμός των ετικετών των ακμών του $G(\upsilon)$ μπορεί να γίνει σε O(sort(N)) πράξεις εισόδου/εξόδου με τη βοήθεια του διπλού γραφήματος $G^*(\upsilon)$ του $G(\upsilon)$. Τώρα μελετούμε τις κορυφές $\upsilon_1,\ldots,\upsilon_t$ του ορίου του υ^* με τη σειρά που εμφανίζονται σύμφωνα με τη φορά των δεικτών του ρολογιού γύρω από το υ^*, αρχίζοντας από ένα άκρο μίας ακμής που μοιράζεται στο υ^* και στην όψη που αναφέρεται στο πατέρα της υ, $p(\upsilon)$στο T^*.



Εικόνα 3.6: Το γράφημα G(υ) είναι σκιαγραφημένο. Ορίζουμε σ(i)=i. Οι διαφορετικές σκιές στα υπογραφήματα αναπαριστούν τα διαφορετικά υπογραφήματα $G(w_1)$, $G(w_2)$ και $G(w_3)$ του G(υ).Οι κορυφές στο $υ^*$, είναι αριθμημένες σύμφωνα με τη φορά των δεικτών του ρολογιού γύρω από το $υ^*$, ξεκινώντας από το άκρο μίας ακμής που μοιράζεται στο $υ^*$ και στο $p^*(υ)$. Οι όψεις στο G(υ) που είναι προσκείμενες στο όριο του $υ^*$, έχουν ως ετικέτα μικρά γράμματα.

Μπορούμε να υπολογίσουμε αυτή τη σειρά με O(sort(N)) πράξεις εισόδου/εξόδου χρησιμοποιώντας την τεχνική του κύκλου του Euler ή την ταξινόμηση λίστας. Για κάθε κορυφή $υ_i$ κατασκευάζουμε μία λίστα L_i με τις ακμές γύρω από την $υ_i$ με τη σειρά που εμφανίζονται δεξιόστροφα, ξεκινώντας από μία ακμή $\{υ_{i-1},υ_i\}$ και τελειώνοντας με μία ακμή $\{υ_i,υ_{i+1}\}$. Αυτές οι λίστες μπορούν να εξαχθούν από την αναπαράσταση του γραφήματος G χωρίς ακμές που διασταυρώνονται ,χρησιμοποιώντας O(sort(N)) πράξεις εισόδου/εξόδου. Αναθέτουμε στην L να είναι η αλληλουχία των λιστών $L_1,L_2,\ldots,L_t$. Για μία ακμή e στην L που είναι προσκείμενη σε μία κορυφή $υ_i$ ορίζουμε με f_1 και f_2 τις δύο όψεις που βρίσκονται από τη μία και την άλλη πλευρά της e ,όπου η f_1 προηγείται της f_2 στην δεξιόστροφη σειρά γύρω από την $υ_i$. Κατασκευάζουμε μία λίστα F με τις ετικέτες των όψεων από την L μελετώντας τις ακμές στην L με τη σειρά που εμφανίζονται και προσαρτούμε τις μη μηδενικές ετικέτες των όψεων f_1 και f_2 με αυτή τη σειρά στην F.(Ανακαλούμε ότι οι όψεις στο $G(w_i)$ έχουν ετικέτες των αριθμών i) Αυτό χρειάζεται O(scan(N)) I/Os.Η λίστα F αποτελείται από ακεραίους μεταξύ του 1 και του k. Κάποιοι από τους ακεραίους μπορεί να εμφανίζονται περισσότερες από μία φορές, και οι εμφανίσεις του ακεραίου

i δεν είναι απαραιτήτως συνεχόμενες. (Αυτό συμβαίνει εάν η ένωση των υ^* με ένα υπογράφημα $G(w_i)$ εσωκλείει και ένα άλλο υπογράφημα $G(w_j)$.) Για το γράφημα $G(\upsilon)$ της εικόνας 2.6 έχουμε:

F={ 3,3 , 3,3,3,3,1,1 , 1,1,1,1,3,3,3,3,2,2 , 2,2,2,2,2,2 , 2,2,3,3,3,3,3,3 , 3,3,3,3,3,3 ,
κορυφή 3 κορυφή 4 κορυφή 5 κορυφή 6 κορυφή 7 κορυφή 8
3,3,3,3 }
κορυφή 9

Κατασκευάζουμε μία τελική λίστα S αφαιρώντας όλες τις εμφανίσεις πλην της τελευταίας για κάθε ακέραιο από την F. (Διαισθητικά αυτό εγγυάται πως αν η ένωση των υ^* με ένα υπογράφημα $G(w_i)$ εσωκλείει και ένα άλλο υπογράφημα $G(w_j)$, τότε το j εμφανίζεται πριν το i στην S. Για το προηγούμενο παράδειγμα έχουμε S={1,2,3}). Η λίστα S μπορεί να κατασκευαστεί από την λίστα L σε O(sort(N)) I/Os χρησιμοποιώντας την πράξη (B) Αφαίρεση διπλότυπων (από τις βασικές εφαρμογές γραφημάτων). Η λίστα S περιέχει καθέναν από τους ακεραίους 1 μέχρι k ακριβώς μία φορά ,έτσι ορίζει μία πράξη αντιμετάθεσης $\sigma:[1,k]\rightarrow[1,k]$, όπου $\sigma(i)$ ισούται με το i-οστό στοιχείο στην S. Απομένει να αποδείξουμε το παρακάτω λήμμα.

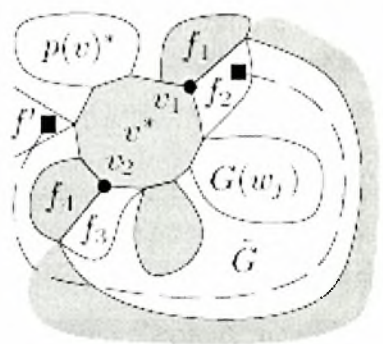
ΛΗΜΜΑ 3.4:

Για όλα τα i ,$1 \leq i \leq k$, τα γραφήματα $H_\sigma(i)$ και $\overline{H\sigma(i)}$ είναι και τα δύο σταθερά.

Απόδειξη:

Πρώτα σημειώνουμε ότι κάθε γράφημα $H_\sigma(i)$ είναι σταθερό γιατί κάθε υπογράφημα $G(w_j)$ είναι σταθερό και υπάρχει μία ακμή μεταξύ των υ και w_j στο G^*, για $1 \leq j \leq k$. Στη συνέχεια θα δείξουμε πως κάθε $\overline{H\sigma(i)}$ είναι σταθερό. Για να το κάνουμε αυτό θα πρέπει να δείξουμε πως το διπλό κάθε $\overline{H\sigma(i)}$, δηλαδή κάθε $\overline{H\sigma(i)}^*$ είναι συνεκτικό. Παρατηρούμε πως, το $\overline{G(\upsilon)}$ είναι σταθερό και κάθε γράφημα $G(w_j)$ είναι σταθερό. Το γράφημα $\overline{H\sigma(i)}$ είναι η ένωση ενός υποσυνόλου αυτών των γραφημάτων ,είναι δηλαδή $\overline{G(\upsilon)} \subseteq \overline{H\sigma(i)}$. Έτσι για να αποδείξουμε ότι το $\overline{H\sigma(i)}^*$ είναι συνεκτικό, αρκεί να δείξουμε πως για όλα τα $i \leq j \leq k$, υπάρχει ένα μονοπάτι στο $\overline{H\sigma(i)}^*$ που συνδέει μία κορυφή του $G(w_j)^*$ με μία κορυφή του $\overline{G(\upsilon)}^*$.Αυτό θα το αποδείξουμε με την εις άτοπον απαγωγή. Θα υποθέσουμε δηλαδή ότι υπάρχει ένα γράφημα $G(w_j)^*$, $i \leq j \leq k$, τέτοιο ώστε να μην υπάρχει κανένα μονοπάτι από μία κορυφή του $G(w_j)^*$ προς

μία κορυφή του $\overline{G(\upsilon)}^*$ στο $\overline{H\sigma(i)}^*$. Αναθέτουμε στο $\tilde{G}$ να είναι η σταθερή συνιστώσα του $\overline{H\sigma(i)}$ που περιέχει το $G(w_j)$ και C να είναι ο κύκλος ορίου του $\tilde{G}$.Ορίζουμε με P το μονοπάτι που αποκτούμε αφαιρώντας όσες ακμές μοιράζονται στο υ^* και στο $p^*(\upsilon)$ από τον κύκλο ορίου του υ^*. Ορίζουμε με υ_1 την πρώτη κορυφή του C που καταμετράται κατά τη διάρκεια μιας δεξιόστροφης πορείας κατά μήκος του P και ορίζουμε με υ_2 την τελευταία τέτοια κορυφή. Ορίζουμε με P' το μονοπάτι που αποκτούμε ακολουθώντας μία δεξιόστροφη πορεία γύρω από το $\tilde{G}$ ξεκινώντας από την υ_1 και καταλήγοντας στην υ_2. Με e_1 και e_2 συμβολίζουμε την πρώτη και την τελευταία ακμή του P' αντίστοιχα. Η ακμή e_1 διαχωρίζει δύο όψεις $f_1 \in H_\sigma(i)$ και $f_2 \in \overline{H\sigma(i)}$. Παρομοίως η ακμή e_2 διαχωρίζει δύο όψεις $f_3 \in \overline{H\sigma(i)}$ και $f_4 \in H_\sigma(i)$. Συμβολίζουμε με j_1 , j_2 , j_3 και j_4 τις ετικέτες αυτών των όψεων. Αποδεικνύουμε ότι η ετικέτα j_2 εμφανίζεται πριν την ετικέτα j_4 στην τελική λίστα S. Έπειτα θα πρέπει να υπάρχει μία όψη f ' με ετικέτα j_2 που να εμφανίζεται μετά την όψη f_4 αν ακολουθήσουμε μία πορεία σύμφωνα με τη κατεύθυνση της φοράς των δεικτών του ρολογιού γύρω από το υ^*. Ειδικότερα η όψη f ' είναι στο εξωτερικό του κύκλου C ,ενώ η όψη f_2 βρίσκεται στο εσωτερικό του. Καθώς το $G^*(w_{j_2})$ είναι συνεκτικό θα πρέπει να υπάρχει ένα μονοπάτι από την f'^*στην f_2^* μέσα στο $G^*(w_{j_2})$. Αλλά αυτό δε μπορεί να είναι εφικτό καθώς κάθε μονοπάτι από την f_2^*στην f'^* θα πρέπει να περιέχει μία ακμή e^* , για κάποια ακμή $e \in C$ και η ακμή e^* δεν μπορεί να υπάρχει μέσα στο $G^*(w_{j_2})$ επειδή ένα από τα άκρα της θα είναι στο $H_\sigma^*(i)$. Άρα έπεται ότι η ετικέτα j_2 πριν την ετικέτα j_4 στην S. Αλλά αυτό σημαίνει ότι η f_2 προστίθεται στο $H_\sigma(i)$ πριν την f_4, κάτι τέτοιο όμως έρχεται σε αντίθεση με την υπόθεση ότι η $f_2 \in \overline{H\sigma(i)}$ και η $f_4 \in H_\sigma(i)$.

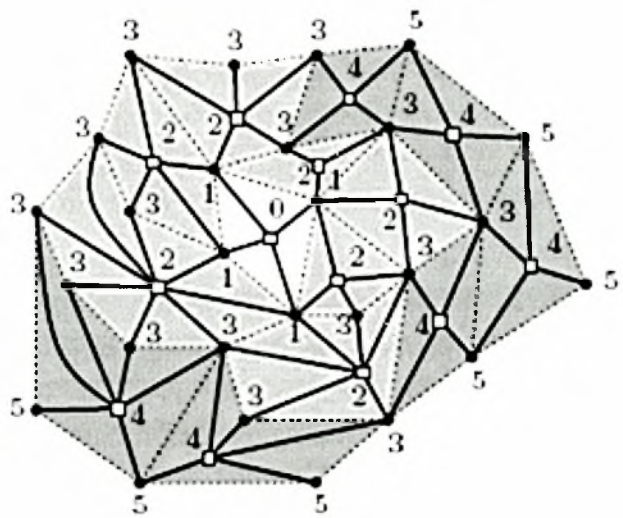


Εικόνα 3.7: Οι σκιαγραφημένες περιοχές ανήκουν στο $H_\sigma(i)$.

3.4 Ελαττώνοντας την αναζήτηση σε βάθος (DFS) σε αναζήτηση κατά πλάτος (BFS)

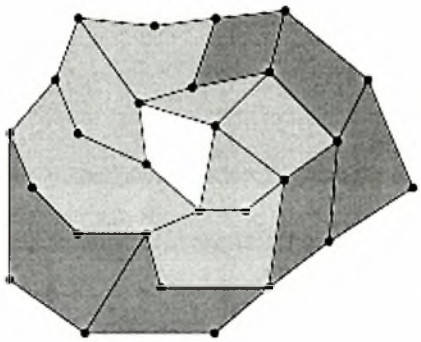
Σε αυτή την ενότητα θα παρουσιάσουμε μία αποδοτική σε πράξεις I/O προσέγγιση για ελάττωση του DFS ενός επίπεδου γραφήματος χωρίς ακμές διασταυρώσεως, σε BFS πάνω σε ένα γράφημα που το ονομάζουμε «κορυφή-σε-όψη» γράφημα(vertex-on-face graph) του αρχικού. Η κύρια ιδέα είναι να χρησιμοποιήσουμε τον BFS για να διαχωρίσουμε τις όψεις του αρχικού γραφήματος G σε επίπεδα γύρω από μία όψη-πηγή που περιέχει την κορυφή-πηγή s του DFS στο όριό της, και έπειτα να αναπτύξουμε το DFS δέντρο επίπεδο-επίπεδο γύρω από αυτή την όψη.

Για να πετύχουμε έναν διαχωρισμό των όψεων του αρχικού γραφήματος G σε επίπεδα γύρω από μία όψη-πηγή, ορίζουμε ένα γράφημα το οποίο το αποκαλούμε «κορυφή-σε-όψη» γράφημα του G και το συμβολίζουμε με G^+. Όπως και προηγουμένως ορίζουμε το διπλό γράφημα του αρχικού G ως $G^*(V^*,E^*)$ και ανακαλούμε ότι κάθε κορυφή f^* στο V^* αντιστοιχεί σε μία όψη f στο G.Το σύνολο κορυφών του «vertex-on-face» γραφήματος G^+ είναι $V \cup V^*$ και το σύνολο ακμών περιέχει μία ακμή (v, f^*) αν η κορυφή v ανήκει στο όριο της f.



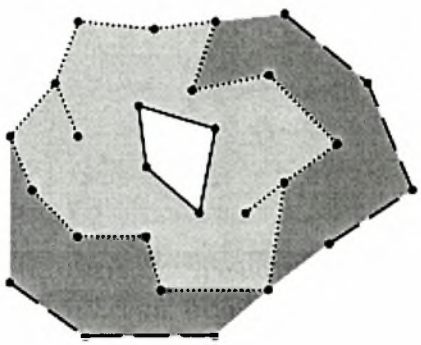
Εικόνα 3.8: Το γράφημα G^+ σχηματισμένο με σκούρες γραμμές. Οι αριθμοί αντιπροσωπεύουν τα BFS βάθη του G^+.

Θα δείξουμε πως ένα BFS δέντρο του G^+ μπορεί να χρησιμοποιηθεί για να πετύχουμε έναν διαχωρισμό των όψεων του G έτσι ώστε η όψη πηγή να είναι στο επίπεδο 0, όλες οι όψεις που μοιράζονται μία κορυφή με την όψη πηγή να είναι στο επίπεδο 1, όλες οι όψεις που μοιράζονται μία κορυφή με μία όψη του επιπέδου 1 αλλά καμία με την όψη πηγή να είναι στο επίπεδο 2,και να συνεχίζει κατά αυτό τον τρόπο.



Εικόνα 3.9: Ένα γράφημα G με τις όψεις του χρωματισμένες ανάλογα με το επίπεδο στο οποίο ανήκουν. Η όψη του επιπέδου 0 είναι λευκή, οι όψεις του επιπέδου 1 είναι ανοιχτό γκρι και οι όψεις του επιπέδου 2 είναι σκούρο γκρι.

Ορίζουμε με G_i το υπογράφημα του G που προσδιορίζεται από την ένωση των ορίων των όψεων που βρίσκονται το πολύ μέχρι το επίπεδο i και ορίζουμε $H_i = G_i \setminus G_{i-1}$, για $i > 0$. Η διαφορά του $G_i \setminus G_{i-1}$ με τα γραφήματα G_i και G_{i-1} είναι το υπογράφημα του G_i με σύνολο κορυφών $V(G_i) \setminus V(G_{i-1})$ και του οποίου το σύνολο ακμών περιέχει όλες τις ακμές του G_i που έχουν και τα δύο άκρα τους στο $V(G_i) \setminus V(G_{i-1})$. Για i=0 ορίζουμε $H_0 = G_0$.

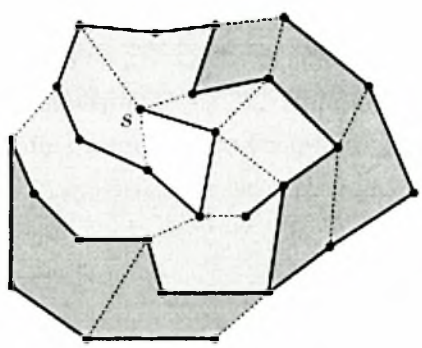


Εικόνα 3.10: Το γράφημα H_0:με την συνεχόμενη γραμμή. Το γράφημα H_1:με την στικτή γραμμή. Το γράφημα H_2:με την διακεκομμένη γραμμή.

Αποκαλούμε τις κορυφές και τις ακμές του H_i,κορυφές και ακμές επιπέδου i. Μία ακμή {υ,w} που συνδέει δύο κορυφές $\upsilon \in H_i$ και $w \in G_{i-1}$ ονομάζεται ακμή επικόλλησης του H_i.Οι ακμές του G_{i-1} και του H_i μαζί με τις ακμές επικόλλησης του H_i σχηματίζουν έναν διαχωρισμό των ακμών του G_i. Η βασική ιδέα του αλγορίθμου μας είναι να αναπτύξουμε το DFS δέντρο ακολουθώντας μία δεξιόστροφη πορεία από την s γύρω από την όψη του επιπέδου $0,G_0$ μέχρι να καταλήξουμε στον αντίθετο ,σύμφωνα με τη φορά των δεικτών του ρολογιού, γείτονα της s.Το μονοπάτι που παίρνουμε ως αποτέλεσμα είναι ένα DFS δέντρο T_0 του G_0. Έπειτα κατασκευάζουμε το DFS δέντρο του H_1 και το επικολλούμε στο T_0 μέσω μίας ακμής επικόλλησης του H_1με τρόπο τέτοιο ώστε να μην παρουσιαστούν ακμές διασταυρώσεως. Έτσι το αποτέλεσμα είναι ένα DFS δέντρο T_1 του G_1. Επαναλαμβάνουμε τη διαδικασία μέχρι να επεξεργαστούμε όλα τα επίπεδα $H_0,\ldots,H_r$ και να αποκτήσουμε ένα DFS δέντρο T για το G. Το κλειδί της αποδοτικότητας του αλγορίθμου βρίσκεται στην απλή δομή των γραφημάτων $H_0,\ldots,H_r$. Παρακάτω θα δώσουμε λεπτομέρειες για τον αλγόριθμό μας και θα αποδείξουμε το ακόλουθο θεώρημα.

ΘΕΩΡΗΜΑ 3.3:

Αν το G είναι ένα μη κατευθυνόμενο επίπεδο γράφημα χωρίς ακμές διασταυρώσεως, G^+ είναι το «vertex-on-face» γράφημά του και f_s μια όψη του G που περιέχει την κορυφή πηγή s ,τότε, δοθέντος του BFS δέντρου του G^+ με ρίζα την f_s^* ,ένα DFS δέντρο του G με ρίζα την s μπορεί να υπολογιστεί σε O(sort(N))πράξεις εισόδου/εξόδου και γραμμικό χώρο.



Εικόνα 3.11: Το DFS δέντρο του G.

Πρώτα μελετούμε τον υπολογισμό των γραφημάτων $G_1,\ldots,G_r$ και $H_0,\ldots,H_r$. Ξεκινάμε από τον υπολογισμό του γραφήματος G^+ σε O(sort(N))πράξεις εισόδου/εξόδου: Πρώτα υπολογίζουμε μία αναπαράσταση του G ως μία λίστα κορυφών με τη σειρά που εμφανίζονται δεξιόστροφα γύρω από κάθε όψη του G. Μία τέτοια αναπαράσταση μπορεί να υπολογιστεί σε O(sort(N))πράξεις εισόδου/εξόδου. Στη συνέχεια εισάγουμε μία κορυφή όψεως f^* ,για κάθε μία όψη του G,και συνδέουμε την f^* με όλες τις κορυφές της λίστας κορυφών που αναπαριστούν την f. Αυτό απαιτεί μία απλή σάρωση στις λίστες κορυφών που αναπαριστούν τις όψεις του G. Τώρα μπορούμε να αποκτήσουμε τα επίπεδα των όψεων του G από ένα BFS δέντρο του «vertex-on-face»γραφήματος ,G^+,με ρίζα την f_s^*,που αφορά την όψη f_s που περιέχει την s. Κάθε κορυφή του G βρίσκεται σε ένα περιττό επίπεδο στο BFS δέντρο και κάθε κορυφή του διπλού γραφήματος που αναφέρεται σε μία όψη του G βρίσκεται σε άρτιο επίπεδο(βλ. Εικόνα 2.8) .Το επίπεδο μίας όψης είναι ίσο με το επίπεδο της αντίστοιχης κορυφής στο BFS δέντρο διηρημένο δια δύο. Το σύνολο κορυφών $V(H_i)$ του γραφήματος H_i περιέχει όλες τις κορυφές του G που βρίσκονται σε απόσταση $2\cdot i+1$ από την f_s^* στο G^+ (για παράδειγμα στην Εικόνα 2.8 οι κορυφές με αριθμό 3 ανήκουν στο H_1:$3=2\cdot i+1=2\cdot 1+1$). Έτσι μπορούμε να αποκτήσουμε έναν διαχωρισμό του V(G) σε σύνολα κορυφών $V(H_1),\ldots, V(H_r)$,ταξινομώντας τις κορυφές του V(G) βάσει των αποστάσεών τους από την f_s^* στο G^+.Το σύνολο κορυφών $V(G_i)$ του γραφήματος G_i είναι $V(G_i)=\cup_{j=0}^{i} V(H_j)$. Μία ακμή $e \in G$ βρίσκεται στο H_i εάν και τα δύο της άκρα βρίσκονται σε απόσταση ίση με $2\cdot i+1$ από την f_s^* στο G^+. Για μία ακμή επικόλλησης {υ,w} του H_i η υ βρίσκεται σε απόσταση $2\cdot i+1$ από την f_s^* στο G^+ (αφού ανήκει στο H_i) και η w βρίσκεται σε απόσταση $2\cdot i-1$(περιττός αριθμός πιο εσωτερικού επιπέδου) από την f_s^* στο G^+. Έτσι μπορούμε να πετύχουμε έναν διαχωρισμό του E(G) σε σύνολα $E(H_0),\ldots,E(H_r)$ και σε σύνολα ακμών επικόλλησης των γραφημάτων $H_0,\ldots,H_r$, ταξινομώντας τις ακμές του E(G) με ανάστροφη λεξικογραφική σειρά που ορίζεται από τις αποστάσεις των άκρων τους από την f_s^* στο G^+.

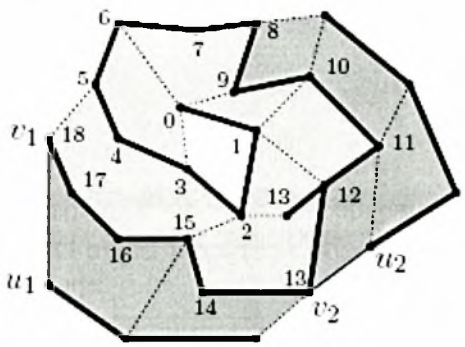
Στη συνέχεια θα παρουσιάσουμε μερικές απλές ιδιότητες των γραφημάτων G_i και H_i, τις οποίες χρησιμοποιούμε για να αποδείξουμε την ορθότητα του αλγορίθμου μας. Για κάθε ακμή του G_{i-2}, όπως επίσης και για κάθε ακμή επικόλλησης του H_{i-1}(είναι ακμές που προσπίπτουν στις κορυφές του G_{i-2}) ,οι δύο όψεις και στις δύο πλευρές της ακμής είναι το πολύ μέχρι το επίπεδο i-1.Έτσι δεν μπορούν να είναι ακμές στο όριο του G_{i-1}. Έπεται ότι οι ακμές ορίου του G_{i-1} ανήκουν στο σύνολο $E(H_{i-1})$. Συνεπώς όλες οι κορυφές ορίου του G_{i-1} είναι στο σύνολο $V(H_{i-1})$. Καθώς το G_{i-1} είναι μία ένωση όψεων, το όριό του θα αποτελείται από ένα σύνολο κύκλων, που το αποκαλούμε κύκλοι ορίου του G_{i-1}. Το γράφημα H_i βρίσκεται εντελώς στο εξωτερικό του ορίου του G_{i-1} ,βρίσκεται δηλαδή στο $\overline{G_{i-1}}$.Έτσι όλες οι ακμές επικόλλησης του H_i συνδέονται μόνο με τις κορυφές ορίου του G_{i-1} ,δηλαδή με τις κορυφές του H_{i-1}. Τέλος ,σημειώνουμε ότι το γράφημα G_i είναι σταθερό. Αυτό μπορεί να αποδειχτεί έτσι: Το γράφημα G_i στα πρώτα 2i επίπεδα του BFS δέντρου του G^+ .Για μία όψη f_1 του επιπέδου (i-1) και για μία όψη f_2 του επιπέδου i που μοιράζονται μία κορυφή υ, το γράφημα G_i περιέχει όλες τις όψεις που πρόσκεινται στην υ. Έτσι υπάρχει ένα μονοπάτι από την f_1^* προς την f_2^* στο G_i^*. Εφαρμόζοντας αυτό το επιχείρημα

επαγωγικά, εξασφαλίζουμε ότι υπάρχει ένα μονοπάτι στο G_i^* από την f_s^* προς κάθε κορυφή του G_i^* το οποίο δηλώνει πως το G_i είναι σταθερό. Αντίθετα το γράφημα G_{i-1} και έτσι και το H_i ,δεν είναι απαραιτήτως σταθερά.

Τώρα είμαστε έτοιμοι για να περιγράψουμε τις λεπτομέρειες του αλγορίθμου μας για την κατασκευή του DFS δέντρου του G αναπτύσσοντας επαναληπτικά ένα DFS δέντρο T_i για το G_i, από ένα DFS δέντρο T_{i-1} για το G_{i-1},ξεκινώντας με ένα DFS δέντρο T_0 για το G_0. Κατά τη διάρκεια του αλγορίθμου ισχύουν τα παρακάτω προτάσεις:

(i)Κάθε κύκλος ορίου C του G_{i-1} περιέχει ακριβώς μία ακμή e που δεν είναι στο T_{i-1}.Το ένα από τα δύο άκρα αυτής της ακμής είναι ένας πρόγονος για όλες τις άλλες κορυφές στον C.

(ii)Το βάθος κάθε κορυφής στο G_{i-1},ορίζεται ως η απόσταση από την s στο T_{i-1}.



Εικόνα 3.12: T_1,H_2 και οι ακμές επικόλλησης $\{u_i,v_i\}$.Οι κορυφές στο T_1 έχουν την ετικέτα με το DFS βάθος τους.

Υποθέτουμε ότι έχουμε κατασκευάσει ένα DFS δέντρο T_{i-1} για το G_{i-1}.Ο στόχος μας είναι να κατασκευάσουμε ένα DFS δάσος για το H_i και να το συνδέσουμε με το T_{i-1} μέσω ακμών επικόλλησης του H_i χωρίς να εισάγουμε ακμές διασταυρώσεως, ώστε να αποκτήσουμε ένα DFS δέντρο T_i για το G_i. Εάν μπορούμε να υπολογίσουμε ένα DFS δάσος για το H_i σε $O(sort(|H_i|))$ I/Os και να το συνδέσουμε με το T_{i-1} σε $O(sort(|H_{i-1}|+|H_i|))$ I/Os ,τότε ο συνολικός υπολογισμός

ενός DFS δέντρου για το G χρησιμοποιεί $O(sort(|H_0|)+\sum_{i=1}^{r} sort(|Hi-1|+|Hi|)) = O(\sum_{i=0}^{r} \frac{2|H_i|}{B} \log_{M/B} \frac{N}{B}) = O(sort(N))$ I/Os. Στη συνέχεια θα δείξουμε πώς να εκτελέσουμε και τους δύο υπολογισμούς με τον επιθυμητό αριθμό πράξεων εισόδου/εξόδου.

Ορίζουμε ως $H_1', \ldots, H_k'$ να είναι οι συνεκτικές συνιστώσες του H_i. Αυτές υπολογίζονται σε $O(sort(|H_i|))$ I/Os. Για κάθε συνιστώσα H_j' βρίσκουμε την κορυφή v_j στο όριο του G_{i-1} με το μεγαλύτερο βάθος ,τέτοια ώστε να υπάρχει μία ακμή επικόλλησης $\{u_j,v_j\}$ του H_i ,με $u_j \in H_j'$. Έπειτα υπολογίζουμε ένα DFS δέντρο T_j' του H_j' με ρίζα την u_j και συνδέουμε το T_j' με το T_{i-1} χρησιμοποιώντας την ακμή $\{u_j,v_j\}$. Το T_i θα είναι το δέντρο που θα πάρουμε ως αποτέλεσμα.

ΛΗΜΜΑ 3.5:

Το δέντρο T_i είναι ένα DFS δέντρο του G_i.

Απόδειξη:

Το T_i είναι ένα επικαλύπτον δέντρο του G_i, καθώς το T_{i-1} είναι ένα DFS δέντρο του G_{i-1}, τα δέντρα $T_1', \ldots, T_k'$ είναι DFS δέντρα των συνεκτικών συνιστωσών του H_i, και κάθε δέντρο T_j' είναι συνδεδεμένο με το T_{i-1} μέσω μίας μοναδικής ακμής. Τώρα ορίζουμε μία ακμή $\{v,w\}$ που δεν ανήκει στο δέντρο του G_i. Καθώς δεν υπάρχουν ακμές μεταξύ διαφορετικών συνεκτικών συνιστωσών του H_i στο G_i, είτε $v,w \in H_j'$,για $1 \leq j \leq k$,$v,w \in G_{i-1}$, είτε $v \in H_j'$,για $1 \leq j \leq k$, και $w \in G_{i-1}$. Στις δύο πρώτες περιπτώσεις η ακμή $\{v,w\}$ είναι μία οπισθοακμή ,καθώς τα δέντρα T_{i-1} και T_j' είναι DFS δέντρα για το G_{i-1} και το H_j' ,αντίστοιχα. Στην τελευταία περίπτωση η $\{v,w\}$ είναι μία οπισθοακμή επειδή η v είναι ένας απόγονος την u_j ,και από την πρόταση (i) η w θα πρέπει να είναι πρόγονος της u_j στον κύκλο ορίου του G_{i-1} που περιλαμβάνει και το H_j'. ¤

Μπορούμε να υπολογίσουμε το δέντρο T_i από το δέντρο T_{i-1} σε $O(sort(|H_{i-1}|+|H_i|))$ I/Os: Πρώτα εντοπίζουμε τις ακμές επικόλλησης $\{u_1,v_1\},\ldots,\{u_k,v_k\}$που συνδέουν τα γραφήματα $H_1', \ldots, H_k'$ με το G_{i-1}. Καθώς οι ακμές επικόλλησης του H_i είναι ακμές ενός επίπεδου γραφήματος με σύνολο κορυφών το $V(Hi_{-1}) \cup V(Hi)$ αυτή η διαδικασία χρειάζεται $O(sort(|H_{i-1}|+|H_i|))$ I/Os.Αυτό που απομένει τώρα είναι να δείξουμε πώς να υπολογίσουμε ένα DFS δέντρο T_j' με ρίζα την

u_j,για κάθε συνεκτική συνιστώσα H_j' του H_i.Το κλειδί για να το κάνουμε αυτό αποδοτικά από πλευράς πράξεων εισόδου/εξόδου είναι να ακολουθήσουμε το επόμενο λήμμα το οποίο δείχνει ότι το H_i έχει μία απλή δομή.

ΛΗΜΜΑ 3.6:

Οι μη τετριμμένες δισυνεκτικές συνιστώσες(δηλαδή δισυνεκτικές συνιστώσες που αποτελούνται από περισσότερες τις μίας ακμής) του H_i είναι οι κύκλοι ορίου του G_i.

Απόδειξη:

Μελετούμε έναν κύκλο C στο H_i. Όλες οι όψεις, οι προσκείμενες στον C είναι στο επίπεδο i ή σε μεγαλύτερο .Έτσι ,καθώς το G_{i-1} είναι σταθερό ,όλες οι όψεις του θα είναι είτε εντός είτε εκτός του κύκλου. Υποθέτουμε ότι το G_{i-1} είναι εντός του C. Τότε ,καμία από τις όψεις που είναι εκτός του C δεν μοιράζεται καμία κορυφή μια μία όψη επιπέδου (i-1). Έτσι ,όλες οι όψεις που είναι έξω από τον C θα πρέπει να είναι σε ένα επίπεδο τουλάχιστον (i+1), το οποίο σημαίνει ότι ο C είναι ένας κύκλος ορίου για το G_i.

Κάθε δισυνεκτική συνιστώσα που δεν είναι κύκλος περιέχει το λιγότερο τρία διαζευγμένα μονοπάτια P_1,P_2,P_3 ,με τα ίδια άκρα υ και w. Καθώς έχουμε μόλις δείξει ότι ,το γράφημα C_1= $P_1 \cup P_3$ είναι ένας κύκλος ορίου του G_i, όπως και το γράφημα C_2= $P_1 \cup P_2$. Αναθέτουμε στην {υ,x} να είναι η πρώτη ακμή του P_2 και {y,w} να είναι η τελευταία ακμή του P_2. Καθώς ο C_1 είναι ένας κύκλος ορίου του G_i, το G_i είναι είτε εντελώς εντός είτε εντελώς εκτός του C_1. Καθώς ο C_1 είναι ένα υπογράφημα του H_i,όλες οι όψεις ,οι προσκείμενες του C_1 που είναι από την ίδια μεριά του C_1 όπως το G_i, είναι στο επίπεδο i επειδή όλες οι όψεις της άλλης πλευράς του C_1 είναι το λιγότερο στο επίπεδο (i+1) .Έτσι ,αν το P_2 είναι στην ίδια μεριά του C_1 όπως το G_i, οι τέσσερις όψεις που είναι προσκείμενες στις ακμές {υ,x} και {y,w} είναι στο επίπεδο i, το οποίο αντικρούει το γεγονός πως το γράφημα C_2 είναι ένας κύκλος ορίου για το G_i. Εάν το P_2 είναι στην άλλη μεριά του C_1, οι τέσσερις όψεις που είναι προσκείμενες στις ακμές {υ,x} και {y,w} είναι το λιγότερο στο επίπεδο (i+1) , το οποίο αντικρούει το γεγονός πως οι ακμές {υ,x} και {y,w} είναι στο επίπεδο i. Έτσι κάθε δισυνεκτική συνιστώσα του H_i αποτελεί έναν μοναδικό κύκλο ορίου ¤

Για να υπολογίσουμε ένα DFS δέντρο του H_j' με ρίζα την u_j, πρώτα διαχωρίσουμε το H_j' στις δισυνεκτικές του συνιστώσες. Αυτό χρειάζεται O(sort($|H_j'|$)) I/Os .Έπειτα κατασκευάζουμε το δισυνεκτικό-με κορυφές αποκοπής-δέντρο (bicomp-cutpoint-tree) του H_j' με ρίζα την δισυνεκτική συνιστώσα που περιέχει την v_j. Για κάθε δισυνεκτική συνιστώσα K, καθορίζουμε την κορυφή αποκοπής-πατέρα x. Εάν η K είναι μία τετριμμένη δισυνεκτική συνιστώσα (δηλαδή μία δισυνεκτική συνιστώσα που αποτελείται από μία μοναδική ακμή) , το DFS δέντρο T_K της K αποτελείται από την μοναδική ακμή της K. Διαφορετικά από το ΛΗΜΜΑ 3.6 παίρνουμε ότι η K είναι ένας κύκλος. Ορίζουμε με y ένα γείτονα της x στην K. Αυτόν τον γείτονα μπορούμε να τον αποκτήσουμε με μία σάρωση στο σύνολο ακμών της K. Για να αποκτήσουμε ένα DFS δέντρο T_K της K με ρίζα την x, αφαιρούμε την ακμή{x,y} από την K. Το DFS δέντρο T_j' του H_j' είναι η ένωση των DFS δέντρων T_K για όλες τις δισυνεκτικές συνιστώσες K του H_j'. Σημειώνουμε ότι το T_K είναι ένα μονοπάτι από την x στην y και όλες οι κορυφές κατά μήκος αυτού του μονοπατιού είναι απόγονοι της x. Καθώς οι μη τετριμμένες δισυνεκτικές συνιστώσες του H_i είναι οι κύκλοι ορίου του G_i η πρόταση (i) ισχύει αφού προσαρτήσουμε τα παραγόμενα DFS δέντρα $T_1', \ldots, T_k'$ στο T_{i-1}.

Τέλος για να ισχύει η πρόταση (ii) πρέπει να προσδιορίσουμε το βάθος κάθε κορυφής στο T_i. Το βάθος των κορυφών $T_{i-1} \subseteq T_i$ δεν αλλάζει προσθέτοντας το δέντρα $T_1', \ldots, T_k'$ στο T_{i-1}.Τα βάθη των κορυφών του H_i μπορούν να υπολογιστούν έτσι: Κάθε κορυφή u_j έχει βάθος κατά ένα περισσότερο από το βάθος που έχει η κορυφή $v_j \in T_{i-1}$. Τα βάθη όλων των υπόλοιπων κορυφών στο T_j' μπορούν να υπολογιστούν από το βάθος της u_j χρησιμοποιώντας O(sort($|T_j'|$)) πράξεις εισόδου/εξόδου ,εκτελώντας μία DFS διαπέραση στο T_j'. Έτσι αυτός ο υπολογισμός χρειάζεται O(sort($|H_i|$)) I/Os για όλα τα δέντρα του DFS δάσους του H_i.

Με αυτό τελειώνουμε την περιγραφή της ελάττωσης της επίπεδης αναζήτησης σε βάθος σε επίπεδη αναζήτηση κατά πλάτος και την απόδειξη του θεωρήματος 3.3 .Το επόμενο πόρισμα είναι άμεση συνέπεια του θεωρήματος 3.3.

ΠΟΡΙΣΜΑ 3.1:

Ένα DFS δέντρο ενός επίπεδου γραφήματος χωρίς ακμές διασταυρώσεως μπορεί να υπολογιστεί σε O(sort(N)) I/O πράξεις και γραμμικό χώρο.

3.5 Συμπεράσματα:

Σε αυτό το κεφάλαιο αν αναπτύξαμε τον πρώτο o(N) I/O και γραμμικού χώρου αλγόριθμο για την αναζήτηση σε βάθος(DFS) ενός επίπεδου γραφήματος χωρίς ακμές διασταυρώσεως. Επίσης περιγράψαμε μία O(sort(N)) I/O ελάττωση της επίπεδης αναζήτησης σε βάθος σε επίπεδη αναζήτηση κατά πλάτος, αποδεικνύοντας ότι η επίπεδη αναζήτηση σε βάθος στην δευτερεύουσα μνήμη δεν είναι δυσκολότερη από την αναζήτηση κατά πλάτος. Αυτό οδηγεί σε έναν αλγόριθμο που υπολογίζει ένα DFS δέντρο ενός επίπεδου γραφήματος χωρίς ακμές διασταυρώσεως σε O(sort(N)) I/O πράξεις.

Κεφάλαιο 4

MST

4.1 Εισαγωγή:

Σε αυτό το κεφάλαιο θα μελετήσουμε τον αλγόριθμο για την εύρεση ελάχιστου επικαλύπτοντος δέντρου σε γενικά μη κατευθυνόμενα βεβαρημένα γραφήματα. Η βασική ιδέα είναι ότι θα χρησιμοποιήσουμε έναν O(sort(E) log log (VB/E)) αλγόριθμο ώστε να ελαττώσουμε τον αριθμό των κορυφών από V σε O(E/B) και έπειτα θα χρησιμοποιήσουμε έναν O(V + sort(E)) αλγόριθμο εύρεσης ελάχιστου επικαλύπτοντος δέντρου (minimum spanning tree-MST) στο γράφημα που προέκυψε. Η συνολική πολυπλοκότητα πράξεων I/O λοιπόν θα είναι O(sort(E) log log (VB/E) + E/B + sort(E)) = O(sort(E) log log (VB/E)). Πρώτα θα περιγράψουμε τον O(V+ sort(E)) και έπειτα θα περιγράψουμε τον O(sort(E) log log (VB/E)) αλγόριθμο ελάττωσης κορυφών. Το αποτέλεσμα συνοψίζεται στο παρακάτω θεώρημα :

ΘΕΩΡΗΜΑ 4.1:

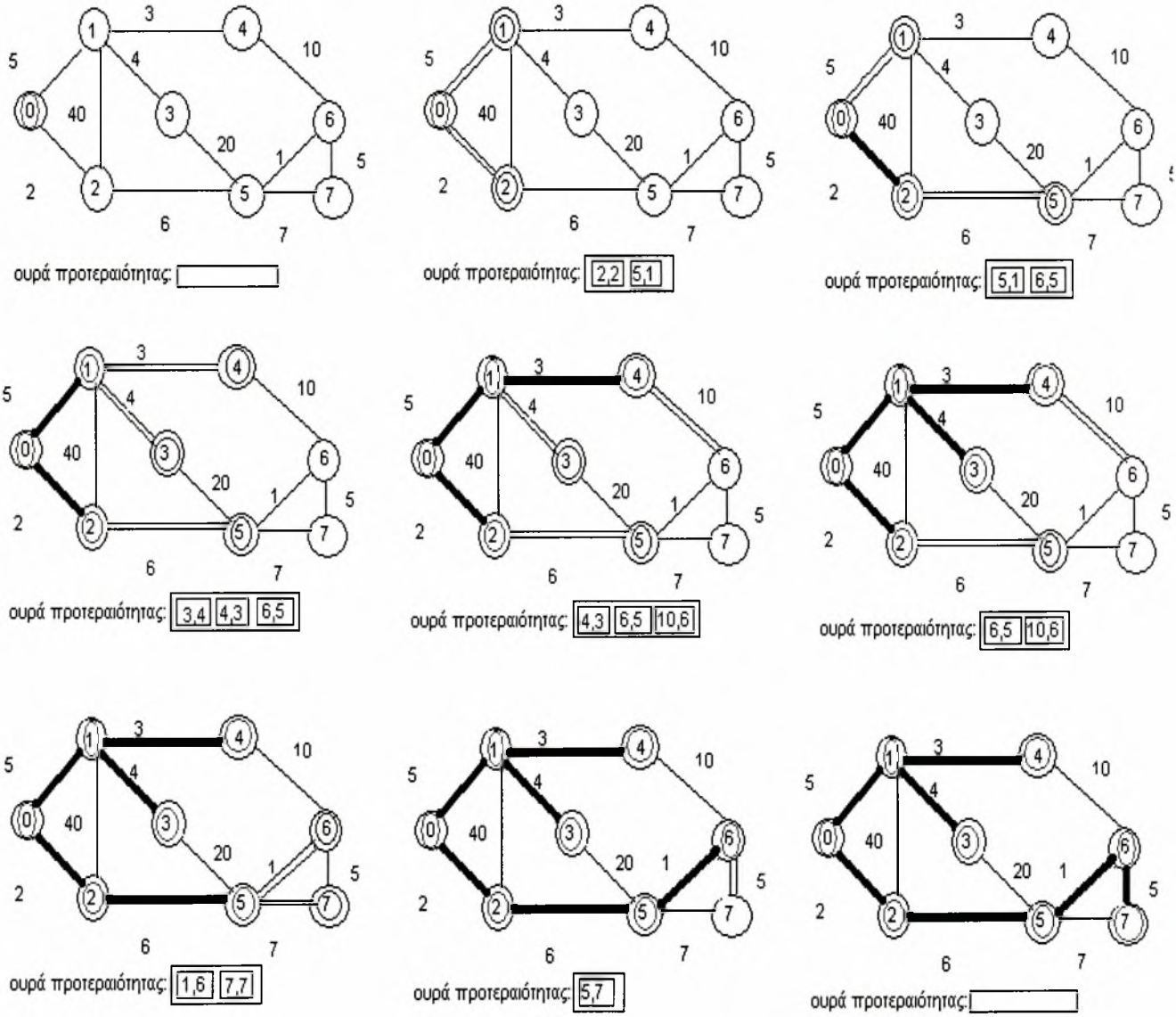
Ένα ελάχιστο επικαλύπτον δέντρο ενός μη κατευθυνόμενου βεβαρημένου γραφήματος G=(V,E) μπορεί να υπολογιστεί σε O(sort(E) log log (VB/E)) I/Os.

4.2 Ένας O(V + sort(E)) αλγόριθμος εύρεσης ελάχιστου επικαλύπτοντος δέντρου.

Ο αλγόριθμος αυτός είναι μία τροποποίηση του αλγορίθμου Prim για την εύρεση ελάχιστου επικαλύπτοντος δέντρου που χρησιμοποιείται στην εσωτερική μνήμη του υπολογιστή. [17] Η ιδέα του αλγορίθμου Prim ,γνωστός και ως αλγόριθμος Prim-Jarnik ,είναι να διατηρεί κάθε στιγμή ένα μοναδικό MST το οποίο και θα επεκτείνει ακμή προς ακμή. Ξεκινώντας από το στοιχειώδες δέντρο της μίας κορυφής και διατηρώντας μία ουρά προτεραιότητας για τις κορυφές που δεν έχουν συμπεριληφθεί ακόμα στο MST κάθε φορά επιλέγει ποια θα είναι η επόμενη κορυφή που θα εισαχθεί στο υπό κατασκευήν ελάχιστο επικαλύπτον δέντρο. Η προτεραιότητα της κάθε κορυφής είναι το βάρος της ελαφρύτερης ακμής που την συνδέει με το τρέχον MST. Ο

αλγόριθμος Prim σε κάθε βήμα του εξάγει από την ουρά προτεραιότητας την κορυφή ,υ , με την μικρότερη προτεραιότητα και την εισάγει στο MST και έπειτα ανανεώνει την προτεραιότητα των κορυφών u που είναι γειτονικές με την υ. Ειδικότερα το βάρος, w ,της ακμής (υ,u) συγκρίνεται με την προτεραιότητα της κορυφής u στην ουρά προτεραιότητας και γίνεται ανανέωση της προτεραιότητάς της εάν το w είναι μικρότερο από την τρέχουσα προτεραιότητα.

Παράδειγμα αλγόριθμου Prim:



Όμως ο αλγόριθμος του Prim δεν μπορεί να υλοποιηθεί αποδοτικά στην εξωτερική μνήμη , κυρίως επειδή η τρέχουσα προτεραιότητα της εκάστοτε κορυφής δεν μπορούμε γενικά να την εξασφαλίσουμε χωρίς να κάνουμε μία πράξη I/O. Με αυτό τον τρόπο μία τέτοια υλοποίηση θα οδηγούσε σε έναν Ω(E) I/O αλγόριθμο.

[12] Για αυτό τροποποιούμε τον αλγόριθμο Prim και πλέον η ουρά προτεραιότητας θα σώζει ακμές αντί για κορυφές. Κατά τη διάρκεια του αλγορίθμου η ουρά προτεραιότητας θα κρατάει το λιγότερο όλες τις ακμές που συνδέουν τις κορυφές του τρέχοντος ελάχιστου επικαλύπτοντος δέντρου με κορυφές που δεν ανήκουν ακόμα στο δέντρο, μπορεί επίσης να κρατάει και ακμές μεταξύ δύο κορυφών που έχουν ήδη εισαχθεί στο MST. Ο αλγόριθμος λειτουργεί έτσι:

Αρχικοποιούμε την ουρά ώστε να περιέχει όλες τις ακμές τις προσπίπτουσες στην κορυφή-πηγή. Εξάγουμε επαναληπτικά την ακμή (u,υ) που έχει το ελάχιστο βάρος από την ουρά προτεραιότητας.
Εάν η υ είναι ήδη στο MST τότε η ακμή απορρίπτεται. Αλλιώς η υ εισάγεται στο MST και όλες οι ακμές οι προσκείμενες στην υ εκτός της (υ,u) εισάγονται στην ουρά προτεραιότητας.

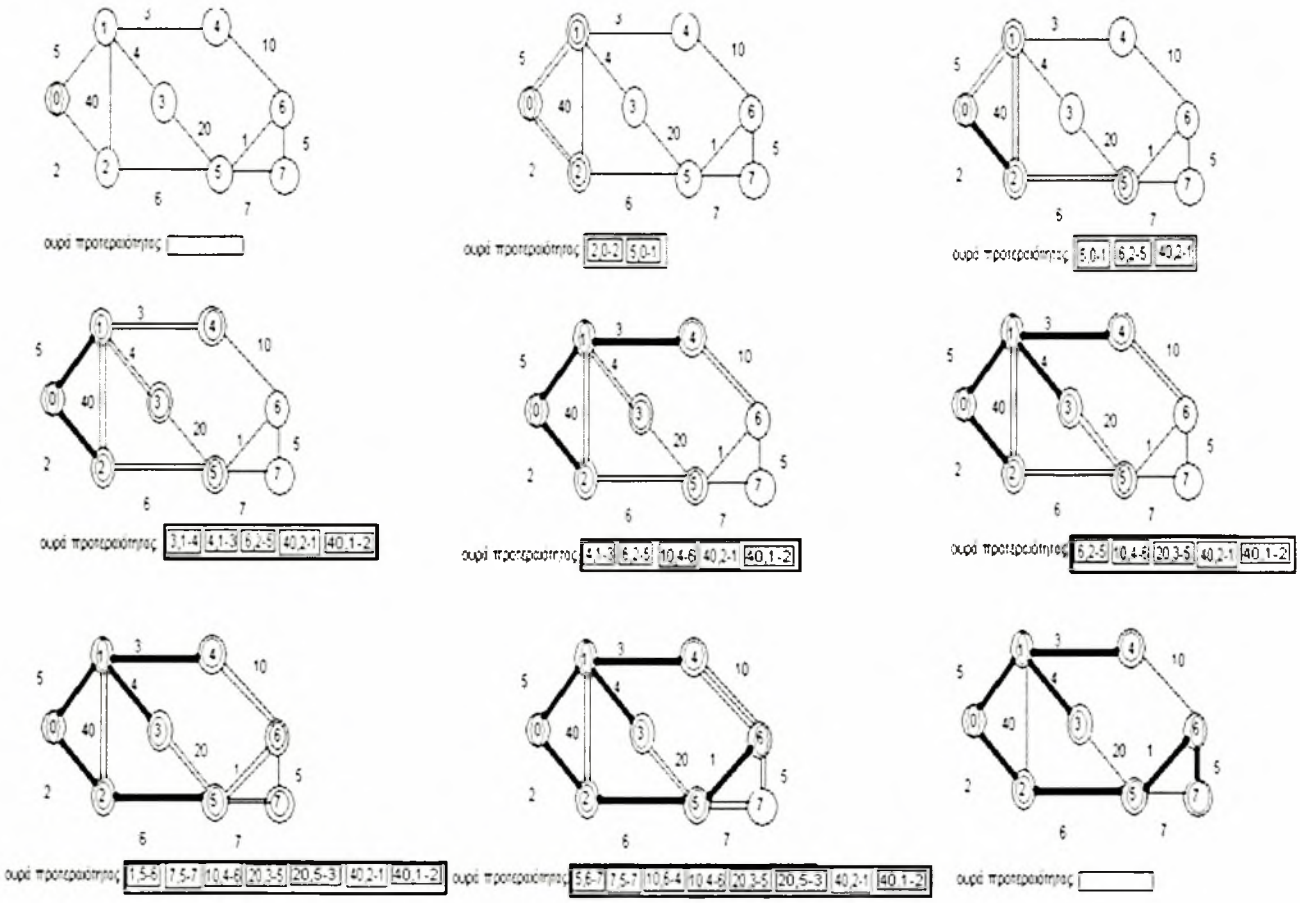
Η ορθότητα του αλγορίθμου έπεται άμεσα από την ορθότητα του αλγορίθμου Prim. Το κλειδί στην αποδοτικότητα των I/O πράξεών του είναι ο απλός τρόπος του για να καθορίσει εάν η υ έχει ήδη συμπεριληφθεί στο MST: Εάν και η u και η υ είναι στο MST όταν επεξεργαζόμαστε την ακμή e=(u,υ) η ακμή e θα έχει εισαχθεί στην ουρά προτεραιότητας δύο φορές. Έτσι μπορούμε να καθορίσουμε εάν η υ έχει ήδη συμπεριληφθεί στο MST απλά ελέγχοντας εάν η επόμενη ακμή με ελάχιστο βάρος στην ουρά προτεραιότητας είναι ίδια με την e. Για να πετύχει αυτό χρησιμοποιούμε την υπόθεσή μας ότι καμία ακμή δεν μπορεί να έχει το ίδιο βάρος με καμία άλλη.

Η παραλλαγή αυτή του Prim αλγορίθμου εκτελεί το λιγότερο μία πράξη I/O για κάθε κορυφή ώστε να διαβάσει τις γειτονικές της κορυφές ,ελέγχοντας την λίστα γειτνιάσεως της, όταν εισέρχεται στο ελάχιστο επικαλύπτον δέντρο. Με αυτόν τον τρόπο για να επεξεργαστούμε συνολικά όλες τις κορυφές και τις ακμές ο αλγόριθμος χρησιμοποιεί O(V+E/B) I/Os. Ο αλγόριθμος επίσης εκτελεί O(E) πράξεις στην ουρά προτεραιότητας. Χρησιμοποιώντας μία εξωτερική ουρά προτεραιότητας που υποστηρίζει O(N) πράξεις σε O(sort(N)) I/Os πετυχαίνουμε το παρακάτω:

ΛΗΜΜΑ 4.1:

Το ελάχιστο επικαλύπτον δέντρο ενός μη κατευθυνόμενου βεβαρημένου γραφήματος G=(V,E) μπορεί να υπολογιστεί σε O(V + sort(E)) I/Os.

Παράδειγμα τροποποιημένου αλγορίθμου Prim όπου η ουρά προτεραιότητας κρατάει ακμές αντί κορυφών:



Όπως μπορούμε να δούμε και από τα δύο παραδείγματα ,του αλγορίθμου Prim και της τροποποίησης του, διαφορές κατά το τρέξιμό τους εντοπίζουμε μόνο στην ουρά προτεραιότητας.

4.3 Αλγόριθμος ελάττωσης κορυφών

Ο MST αλγόριθμος ελάττωσης κορυφών στηρίζεται στη χρήση ιδεών από τον Munagala-Ranade αλγόριθμο συνεκτικών συνιστωσών (ο οποίος στηρίζεται στην συνένωση ακμών) και στην έννοια «περιοριστικών τιμών». Ο στάνταρ MST αλγόριθμος που στηρίζεται στην συνένωση ακμών εκτελείται σε O(logV) φάσεις ή βήματα Baruvka. Ο αλγόριθμος Baruvka διατηρεί ένα σύνολο από MST τα οποία στην πορεία του συνενώνονται σε ένα μεγάλο MST. Σε κάθε βήμα του αλγορίθμου επιλέγεται η ελαφρότερη ακμή για κάθε κορυφή και συμπεριλαμβάνεται στο ελάχιστο επικαλύπτον δέντρο. Οι κορυφές που διασυνδέονται με τις ακμές που επιλέχτηκαν δημιουργούν υπέρ-κορυφές. Δηλαδή μία υπέρ-κορυφή είναι ένα σύνολο κορυφών του αρχικού γραφήματος που διασυνδέονται μεταξύ τους με κάποιες από τις ακμές που επιλέχτηκαν. Το μέγεθος μίας υπέρ-κορυφής είναι το πλήθος των κορυφών που εμπεριέχει από το αρχικό γράφημα .Ύστερα από το i-οστό βήμα το μέγεθος της κάθε υπέρ-κορυφής είναι το λιγότερο 2^i και έτσι μετά από O(log(V/M)) βήματα το συνενωμένο πλέον γράφημα χωράει στην μνήμη. Κάθε βήμα του αλγόριθμου Baruvka μπορεί να εκτελεστεί σε O(sort(E))I/Os που έχει ως αποτέλεσμα έναν O(sort(E) log(V/M)) αλγόριθμο. Η ορθότητα αυτού του αλγορίθμου στηρίζεται στο παρακάτω θεώρημα, εάν θεωρήσουμε τις κορυφές του δέντρου ως το ένα σύνολο της διαμερίσεως και τις υπόλοιπες κορυφές ως το δεύτερο σύνολο της διαμερίσεως:

ΘΕΩΡΗΜΑ 4.2:

Έστω G=(V,E) συνεκτικό βεβαρημένο γράφημα και V_1,V_2 μία διαμέριση του V. Εάν e = (u , w) $\in$ E , u $\in$ V_1 ,w $\in$ V_2 , είναι η ακμή με το ελάχιστο βάρος μεταξύ όλων όσων διασυνδέουν κορυφές της διαμέρισης , τότε υπάρχει ένα ελάχιστο επικαλύπτον δέντρο που την εμπεριέχει.

Ο Kumar και ο Schwabe πέτυχαν ένα βελτιωμένο O(sort(E) logB + scan(E) logV) αλγόριθμο χρησιμοποιώντας ότι μετά από Θ(logB) φάσεις, όταν ο αριθμός των κορυφών έχει μειωθεί ,λόγω συνενώσεων , σε O(V/B) μία πράξη συνένωσης μπορεί να εκτελεστεί πιο αποδοτικά.

Σύμφωνα με όσα είπαμε λοιπόν θα χρησιμοποιήσουμε έναν αλγόριθμο συνένωσης για να μειώσουμε τον αριθμό των υπέρ-κορυφών σε E/B και μετά θα μπορούμε να χρησιμοποιήσουμε τον αλγόριθμο εύρεσης ελάχιστου επικαλύπτοντος δέντρου που περιγράψαμε στην προηγούμενη παράγραφο. Για να το πετύχουμε αυτό χρειάζεται να κάνουμε Θ(log(VB/E)) φάσεις συνένωσης και θα δείξουμε πως αυτές οι φάσεις μπορούν να εκτελούνται μέσα σε O(sort(E)loglog(VB/E)) I/Os (σε αντίθεση με τις O(sort(E)log(VB/E))I/Os) διαιρώντας τις Θ(log(VB/E)) φάσεις σε Θ(loglog(VB/E)) υπέρ-φάσεις που απαιτούν O(sort(E)) I/Os η κάθε μία. Με αυτόν τον τρόπο

αποκομίζουμε το παρακάτω λήμμα το οποίο σε συνδυασμό με το ΛΗΜΜΑ 4.1 αποδεικνύουν το ΘΕΩΡΗΜΑ 4.1.

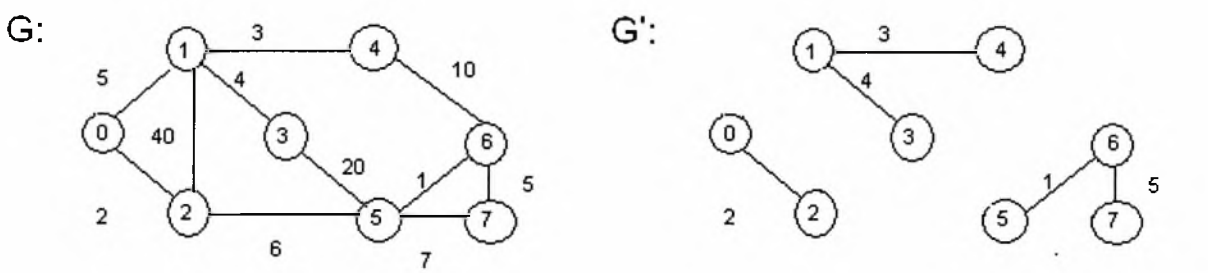
ΛΗΜΜΑ 4.2:

Το ελάχιστο επικαλύπτον δέντρο ενός μη κατευθυνόμενου βεβαρημένου γραφήματος G=(V,E) μπορεί να ελαττωθεί στο ίδιο πρόβλημα σε ένα γράφημα με O(E/B) κορυφές και O(E) ακμές σε O(sort(E)loglog(VB/E)) I/Os.

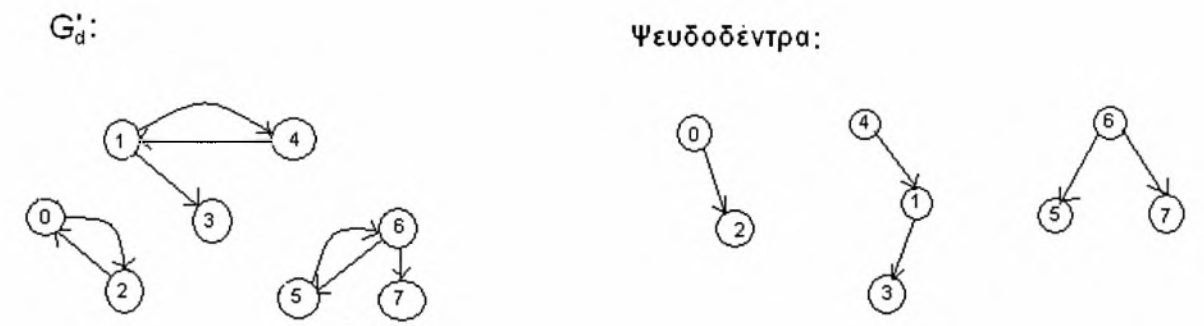
4.3.1 Αλγόριθμος συνένωσης κορυφών με πολυπλοκότητα I/O πράξεων O(sort(E))

Σε κάθε βήμα του αλγόριθμου Baruvka σε ένα γράφημα G=(V,E) επιλέγεται για κάθε κορυφή η ελαφρύτερη προσκείμενη ακμή και συνενώνει τις κορυφές και προκύπτουν υπέρ-κορυφές. Οι ελαφρύτερες προσκείμενες ακμές των κορυφών μπορούν εύκολα να συλλεχτούν μέσα σε O(E/B) I/Os με μία απλή σάρωση της λίστας ακμών που αναπαριστά το γράφημα. Διάφοροι αλγόριθμοι με πολυπλοκότητα I/O πράξεων O(sort(E)) έχουν αναπτυχθεί για συνενώνεις κορυφών αλλά εμείς θα περιγράψουμε τον απλούστερο από αυτούς.

Για κάθε κορυφή υ αναθέτουμε στο C(υ) να δηλώνει την ελαφρύτερη γείτονα κορυφή της υ (με άλλα λόγια η ακμή (υ,C(υ)) είναι η ελαφρύτερη προσκείμενη στην υ). Ορίζουμε ένα γράφημα G' να είναι το γράφημα που αποκτούμε την ακμή (υ,C(υ)) για κάθε κορυφή υ. Ο στόχος μας είναι να συνενώσουμε κάθε συνεκτική συνιστώσα που υπάρχει στο γράφημα G'. Αυτό θα γίνει ορίζοντας μία μοναδική κορυφή αντιπρόσωπο σε κάθε συνιστώσα και να αντικαταστήσουμε κάθε ακμή (υ.u) του αρχικού γραφήματος G με μία ακμή ($υ_r$,u_r) όπου $υ_r$ και u_r οι μοναδικές κορυφές αντιπρόσωποι των συνιστωσών που περιέχουν τις υ και u αντίστοιχα.



Για να υπολογίσουμε τις μοναδικές κορυφές αντιπροσώπους υποθέτουμε το κατευθυνόμενο γράφημα G_d' που το αποκτούμε κατευθύνοντας τις ακμές (υ,C(υ)) του G' από την C(υ) προς την υ. Σημειώνουμε πως κάθε κορυφή στο G_d' έχει βαθμό εισόδου ίσο με 1. Οι συνεκτικές συνιστώσες του G_d' αποτελούνται από ψευδο-δέντρα. Σε κάθε συνιστώσα δύο ακμές e_1=(u,υ) και e_2=(υ,u) θα πρέπει να έχουν το ίδιο (ελάχιστο) βάρος και να σχηματίζουν έναν κύκλο (σημειώνουμε πως κάθε ακμή έχει μοναδικό βάρος, άρα η ακμή σε μία συνιστώσα με το ελάχιστο βάρος είναι λογικό να είναι και για τις δύο κορυφές των άκρων της η ελαφρύτερη, άρα θα υπάρχει στο G_d' βελάκι και από την υ στην u αλλά και από την u στην υ). Οι e_1 και e_2 αναφέρονται στην ίδια ακμή e του G και η e είναι η ελαφρύτερη προσκείμενη ακμή και στην υ και στην u. Εάν μία από αυτέ τις δύο ακμές που σχηματίζουν τον κύκλο αφαιρεθεί τότε η προκύπτουσα συνιστώσα θα πρέπει να σχηματίζει ένα δέντρο (καθώς ο αριθμός των ακμών είναι κατά ένα μικρότερος από τον αριθμό των κορυφών) με ρίζα την υ ή την u ανάλογα με το ποία από τις δύο ακμές αφαιρεθεί. Σε αυτό το δέντρο κάθε κορυφή βρίσκεται σε ένα κατευθυνόμενο μονοπάτι από την ρίζα προς ένα φύλλο(γιατί κάθε κορυφή έχει βαθμό εισόδου ίσο με ένα)και τα βάρη των ακμών κατά μήκος του κατευθυνόμενου μονοπατιού αυξάνονται.



Η δομή των συνεκτικών συνιστωσών του G_d' μας επιτρέπει να υπολογίσουμε τις μοναδικές κορυφές αντιπροσώπους με αποδοτικό I/O τρόπο. Μπορούμε εύκολα να κατασκευάσουμε το G_d' και να εντοπίσουμε όλους τους κύκλους σε O(sort(N)) I/Os με κάποια βήματα ταξινόμησης και σάρωσης. Αφού αφαιρέσουμε την μία από τις ακμές σε κάθε κύκλο μας απομένουν κάποια δέντρα στα οποία έχουμε ορίσει τις ρίζες. Σε κάθε δέντρο επιλέγουμε την ρίζα να είναι η μοναδική κορυφή αντιπρόσωπος και διανέμουμε αυτή την πληροφορία στους υπόλοιπους κόμβους του δέντρου χρησιμοποιώντας την ιδέα του «time forward processing». Ορίζουμε την λίστα ακμών του G_d' ,L, ταξινομημένη με αύξουσα σειρά βάρους όπου μετά από κάθε ακμή (u,υ) σώζουμε ένα αντίγραφο όλων των υπόλοιπων ακμών που είναι προσκείμενες στην υ. Η λίστα L περιέχει κάθε ακμή του G_d' δύο φορές και μπορεί να κατασκευαστεί εύκολα με O(sort(V))I/Os με κάποια βήματα ταξινόμησης και σάρωσης. Επεξεργαζόμαστε τις ακμές με τη

σειρά που είναι ταξινομημένες στη λίστα L ενώ διατηρούμε μία ουρά προτεραιότητας. Αυτή η ούρα χρησιμοποιείται για να στέλνετε η πληροφορία των μοναδικών κορυφών αντιπροσώπων των ήδη επεξεργασμένων κορυφών στους άμεσους γείτονες τους. Ακριβέστερα η ουρά προτεραιότητας περιέχει κάθε κορυφή υ για την οποία η εισερχόμενη ακμή (u,υ) δεν έχει ακόμα επεξεργαστεί αλλά η εισερχόμενη ακμή της u έχει. Η προτεραιότητα μίας κορυφής υ στην ουρά είναι ίση με το βάρος της εισερχόμενης ακμής (u,υ) και η υ έχει πάρει την πληροφορία για την μοναδική κορυφή αντιπρόσωπο της u. Σημειώνουμε ότι όταν η υ εισέρχεται στην ουρά έχει προσδιοριστεί αντιπρόσωπός της και μπορούμε να εξάγουμε αυτή την πληροφορία. Αρχικά η ουρά προτεραιότητας περιέχει κάθε κορυφή που είναι άμεσα συνδεδεμένη με μία κορυφή που είναι ρίζα στο G_d'.Εάν η υ είναι άμεσα συνδεδεμένη με την ρίζα u τότε η ακμή e(u,υ) υπάρχει και ορίζουμε την προτεραιότητα της υ να είναι ίση με το βάρος της e και με ετικέτα την u. Μπορούμε εύκολα να κάνουμε αυτή την αρχικοποίηση σε O(sort(V)) I/Os απλά σαρώνοντας τις ακμές του G_d' και εισάγοντας τις σχετικές κορυφές. Για να επεξεργαστούμε την επόμενη ακμή e=(υ,w) από την λίστα L, πρώτα εξάγουμε από την ουρά την κορυφή με την μικρότερη προτεραιότητα. Καθώς τις ακμές τις επεξεργαζόμαστε κατά αύξουσα σειρά βάρους, η υ θα πρέπει να έχει ήδη επεξεργαστεί και η w εισάγεται στην ουρά προτεραιότητας με προτεραιότητα ίση με το βάρος της e. Επειδή όλες τις ακμές με βάρος μικρότερο της e τις έχουμε επεξεργαστεί ,η κορυφή w θα πρέπει να είναι η επόμενη κορυφή που θα εξάγουμε από την ουρά προτεραιότητας. Έτσι αποκτήσαμε την μοναδική κορυφή αντιπρόσωπο της w. Μπορούμε να επαναλάβουμε την διαδικασία εισάγοντας στην ουρά προτεραιότητας κάθε διάδοχο της w με την αντίστοιχη προτεραιότητά του και να του βάλουμε ετικέτα την μοναδική κορυφή αντιπρόσωπο της υ. Καθώς εκτελούμε O(V) πράξεις στην ουρά προτεραιότητας συνολικά χρησιμοποιούμε O(sort(V)) I/Os για να εκτελέσουμε τις πράξεις στην ουρά προτεραιότητας. Έτσι έχουμε καθορίσει τις μοναδικές κορυφές αντιπροσώπους σε O(E/B + sort(V)) I/Os.

L: {6-5} {5-6}
{5-6} {6-5}{6-7}
{0-2} {2-0}
{2-0} {0-2}
{4-1} {1-4}{1-3}
{1-4} {4-1}
{1-3}
{6-7}

Ουρά προτεραιότητας :
Αρχικά: 5,2,1,7

Εξάγουμε την κορυφή 5 αλλά δεν έχει άλλη κορυφή διάδοχο άρα δεν εισάγουμε κάποια άλλη κορυφή στην ούρα άρα εξάγουμε την 2 ομοίως συνεχίζουμε και εξάγουμε την 1 η οποία έχοντας διάδοχο την 3 έχει ως αποτέλεσμα την εισαγωγή της 3 στην ουρά με την αντίστοιχη προτεραιότητα της δηλαδή το βάρος της ακμής {1-3}.Εξάγεται από την ουρά και τέλος επεξεργαζόμαστε και την κορυφή 7.

Τελειώνοντας την συνένωση κορυφών χρειαζόμαστε να αντικαταστήσουμε κάθε ακμή (υ,u) του G με μία ακμή ($υ_r$,u_r) μεταξύ των κορυφών αντιπροσώπων $υ_r$ της υ και u_r της u. Δοθείσας μίας λίστας E ακμών (υ,u) και μίας λίστας R με τις κορυφές αντιπροσώπους (υ, $υ_r$) μπορούμε εύκολα να το κάνουμε μέσα σε O(sort(E)) I/Os. Αρχικά ταξινομούμε τις λίστες E και R με βάση το πρώτο συστατικό τους. Έπειτα σαρώνουμε τις δύο λίστες ταυτόχρονα αντικαθιστώντας κάθε ακμή (υ, u) στην E με την ($υ_r$, υ). Στη συνέχεια αντικαθιστούμε το δεύτερο συστατικό της E τις κορυφές αντιπροσώπους με παρόμοιο τρόπο ταξινομώντας την E βάσει του δεύτεροι συστατικού της και σαρώνοντας την E και την R ταυτόχρονα. Στο τέλος απομακρύνουμε τις διπλότυπες ακμές σε ένα απλό βήμα ταξινόμησης και σάρωσης .

E: 0-1 ----> 0-1 ----> 0-4
0-2 ----> 0-2 ----> 0-0
1-2 ----> 4-2 ----> 4-0
1-3 ----> 4-3 ----> 4-4
1-4 ----> 4-4 ----> 4-4
2-5 ----> 0-5 ----> 0-6
3-5 ----> 4-5 ----> 4-6
4-6 ----> 4-6 ----> 4-6
5-6 ----> 6-6 ----> 6-6
5-7 ----> 6-7 ----> 6-6

R: 0,0
1,4
2,0
3,4
4,4
5,6
6,6
7,6

ΛΗΜΜΑ 4.3:

Δοθέντος ενός μη κατευθυνόμενου βεβαρημένου γραφήματος G=(V,E) οι ελαφρύτερες ακμές προσκείμενες σε κάθε κορυφή μπορούν να συνενωθούν σε O(sort(E)) I/Os.

4.3.2 Αλγόριθμος υπέρ-φάσης

Σε αυτή την ενότητα θα δείξουμε πώς να εκτελούμε Θ(log(VB/E)) φάσεις συνένωσης σε ένα γράφημα G=(V,E) μειώνοντας τον αριθμό των κορυφών σε E/B μέσα σε O(loglog(VB/E)sort(E)) I/Os. Αυτό γίνεται εκτελώντας τις Θ(log(VB/E)) φάσεις σε Θ(loglog(VB/E)) υπέρ-φάσεις που απαιτούν O(sort(E)) I/Os η κάθε μία.

Ορίζουμε $N_i=2^{(3/2)^i}$, $N_{i+1}=N_i\sqrt{N_i}$.Η υπέρ-φάση i αποτελείται από $[\log_{\sqrt{N_i}}]$ φάσεις. Θα ισχυριστούμε ότι πριν από την υπέρ-φάση i ο αριθμός των υπέρ-κορυφών είναι το πολύ $2V/N_i$. Ορίζουμε το $G_i=(V_i,E_i)$ να είναι το γράφημα ακριβώς πριν την υπέρ-φάση i. Για πετύχουμε μεγαλύτερη αποδοτικότητα οι φάσεις στην υπέρ-φάση i ενεργούν μόνο σε ένα υποσύνολο E_i' των ακμών E_i. Για κάθε κορυφή υ, το σύνολο E_i' περιέχει τις $[\sqrt{N_i}]$ ελαφρύτερες προσκείμενες ακμές στην υ. Οι βαρύτερες προσκείμενες στην υ ακμές e=(υ,u) περιλαμβάνονται στο E_i' εάν η e

είναι μεταξύ των $[\sqrt{N_i}]$ ελαφρύτερων προσκείμενων ακμών της υ. Επιπλέον ορίζουμε την περιοριστική τιμή της υ να είναι το βάρος της $[\sqrt{N_i}]$ + 1)–οστής ελαφρύτερης προσκείμενης ακμής της υ. Σημειώνουμε ότι $E_i' \leq 2V_i$ $[\sqrt{N_i}]$ και επειδή $V_i \leq 2V/N_i$ έχουμε $E_i' \leq V_i$ $[\sqrt{N_i}] < 2V/\sqrt{N_i}$. Το σύνολο E_i' και η περιοριστικές τιμές μπορούν να υπολογιστούν σε $O(sort(E_i))$ I/Os χρησιμοποιώντας κάποια βήματα ταξινόμησης και σάρωσης.

Τώρα εκτελούμε $[\log_{\sqrt{N_i}}]$ φάσεις συνένωσης στο ελαττωμένο γράφημα $G_i'=(V_i,E_i')$. Μία φάση εκτελείται όπως στο βασικό αλγόριθμο ελάττωσης κορυφών : Για κάθε κορυφή υ εξετάζουμε την προσκείμενη κορυφή της, e=(υ,u) που ανήκει στο E_i', με το μικρότερο βάρος. Εάν το βάρος της e είναι μικρότερο από την περιοριστική τιμή της υ τότε επιλέγουμε την ακμή e για την συνένωση ενώ αν το βάρος της είναι μεγαλύτερο από την περιοριστική της τιμή τότε καμία ακμή δεν συλλέγεται για την υ μέχρις ότου μπορεί να υπάρξει κάποια ελαφρύτερη ακμή προσκείμενη στην υ που να ανήκει στο E_i-E_i'. Οι ακμές που επιλέχτηκαν συνενώνονται μέσα σε $O(sort(E_i'))$I/Os σσύμφωνα με το λήμμα ΛΗΜΜΑ 4.3 και με όσα είπαμε στην προηγούμενη ενότητα. Μετά από την συνένωση ορίζουμε την περιοριστική τιμή της υπέρ-κορυφής που προέκυψε ως την ελάχιστη περιοριστική τιμή των κορυφών που συνενώθηκαν. Επαγωγικά προκύπτει ότι οι υπόλοιπες ακμές του E_i' είναι οι προσκείμενες ακμές της υπέρ-κορυφής υ του E_i με βάρος μικρότερο από την περιοριστική τιμή της υ. Έτσι σωστά ο αλγόριθμος συνενώνει μόνο τις ακμές που πραγματικά ανήκουν στο ελάχιστο επικαλύπτον δέντρο του γραφήματος G.

Ότι ο αριθμός των υπέρ-κορυφών ύστερα από $[\log_{\sqrt{N_i}}]$ φάσεις είναι το πολύ $2V/N_{i+1}$ αποδεικνύεται έτσι: Εάν στην υπέρ-φάση i η περιοριστική τιμή κάποιας υπέρ-κορυφής υ μας εμποδίζει να συλλέξουμε μία ακμή για την υ ,τότε η υ θα πρέπει να είναι η συνένωση τουλάχιστον $\sqrt{N_i}$ κορυφών του V_i .Αυτό προκύπτει από το γεγονός ότι η περιοριστική τιμή της υ αντιστοιχεί στην περιοριστική τιμή κάποιας κορυφής u του συνόλου V_i και η υ πρέπει να περιέχει τις $\sqrt{N_i}$ προσκείμενες κορυφές της u στο E_i'. Εάν καμία περιοριστική τιμή δεν μας μποδίζει να επιλέξουμε ακμές για την υ τότε μετά από $[\log_{\sqrt{N_i}}]$ φάσεις η υ θα πρέπει να έχει μέγεθος τουλάχιστον $2^{\log_{\sqrt{N_i}}} = \sqrt{N_i}$.Έτσι μετά από $O(sort(E_i)+ sort(E_i') \log_{\sqrt{N_i}}) = O(sort(E)+ sort(V/\sqrt{N_i}) \log_{\sqrt{N_i}}) = O(sort(E))$ I/Os ο αριθμός των κορυφών έχει μειωθεί κατά έναν παράγοντα τουλάχιστον ίσο με $\sqrt{N_i}$,δηλαδή ο αριθμός των κορυφών έπειτα από $[\log_{\sqrt{N_i}}]$ φάσεις συνένωσης είναι το πολύ ίσος με $V_i/\sqrt{N_i} \leq 2V/(N_i\sqrt{N_i}) = 2V/N_{i+1}$.

Αφού εκτελέσουμε τις $[\log_{\sqrt{N_i}}]$ φάσεις συνένωσης στο G' (μελετώντας δηλαδή μόνο τις ακμές που ανήκουν στο E_i') χρειάζεται να ενσωματώσουμε ξανά τις ακμές (E_i-E_i') ώστε να περατώσουμε την υπέρ-φάση i. Η ακμή (υ,u) θα πρέπει να αντικατασταθεί με την (υ_s,u_s) όπου η υ_s και η u_s είναι οι υπέρ-κορυφές που περιέχουν τις υ,u αντίστοιχα. Για να το κάνουμε αυτό κατά τη διάρκεια των φάσεων συνένωσης διατηρούμε μία λίστα C που περιλαμβάνει για κάθε

κορυφή υ την τρέχουσα υπέρ-κορυφή που περιέχει την υ, έτσι η λίστα C περιέχει ζευγάρια της μορφής (υ, υ_s) .Μετά από κάθε φάση αποκτούμε μία λίστα αντίστοιχη με την L της προηγούμενης ενότητας με ζευγάρια (κορυφές , κορυφές-αντιπροσώπους)και χρειάζεται να ανανεώσουμε την C αναλόγως. Μπορούμε εύκολα να το κάνουμε αυτό μέσα σε O(sort(V_i)) I/Os ταξινομώντας την C βάσει του δεύτερου συστατικού της και την L βάσει του πρώτου συστατικού της και έπειτα σαρώνουμε τις δύο λίστες ταυτόχρονα καθώς αντικαθιστούμε κάθε ζευγάρι (υ, υ_s),(υ_s, $\upsilon_{s'}$) με (υ, $\upsilon_{s'}$). Συνολικά χρησιμοποιούμε O($\log_{\sqrt{N_i}}$ sort(Vi)) = O($\log_{\sqrt{N_i}}$ sort(V/ N_i)) = O(sort(V)) I/Os για να διατηρήσουμε την λίστα L. Δοθείσας της λίστας L μπορούμε να ενσωματώσουμε ξανά τις ακμές (E_i-E_i') σε O(sort(E)) I/Os με τον ίδιο τρόπο που ανανεώσαμε τις ακμές μετά από ένα απλή συνένωση στην προηγούμενη ενότητα.

Τέλος για να μειώσουμε τον αριθμό των κορυφών στο αρχικό γράφημα G σε O(E/B) είναι αρκετό να εκτελέσουμε i υπέρ-φάσεις τόσες ώστε $V/N_i \leq E/B$. Για αυτό είναι αρκετό να εκτελέσουμε O(loglog(VB/E)) υπέρ-φάσεις που χρειάζεται O(sort(E)) I/Os η κάθε μία με συνολικά O(sort(E) loglog(VB/E)) I/Os.

Κεφάλαιο 5

SSSP

5.1 Εισαγωγή:

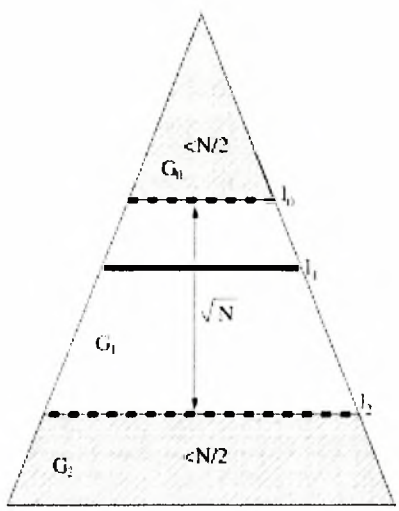
[17] Εάν G=(V,E) είναι ένα μη κατευθυνόμενο βεβαρημένο γράφημα και s και t είναι δύο κορυφές του τότε το συντομότερο μονοπάτι δ(s,t) από την s στην t είναι το μονοπάτι με το ελάχιστο μήκος μεταξύ όλων των μονοπατιών από την s στην τ στο G, όπου μήκος ενός μονοπατιού είναι το άθροισμα των βαρών των ακμών του. Κατά τον αλγόριθμο εύρεσης συντομότερων μονοπατιών μοναδικής πηγής εντοπίζουμε τα συντομότερα μονοπάτια από μια κορυφή προς όλες τις υπόλοιπες κορυφές του γραφήματος. Ο υπολογισμός των συντομότερων μονοπατιών ενός γραφήματος είναι από τα βασικότερα προβλήματα στην αλγοριθμική θεωρία γραφημάτων. Για παράδειγμα η έρευνα στο μοντέλο του διαδικτύου χρησιμοποιεί τα συντομότερα μονοπάτια σαν κύρια ρουτίνα για την εξέταση της δομής του διαδικτύου. Επιπλέον εφαρμογές συχνά εμφανίζονται και στα γεωγραφικά πληροφοριακά συστήματα (GIS).

[12] Προτού περιγράψουμε τον αλγόριθμο εύρεσης συντομότερων μονοπατιών μοναδικής πηγής ΣΜΣΠ θα πρέπει να περιγράψουμε την τεχνική διαχωρισμού ενός επίπεδου γραφήματος γιατί θα μας χρειαστεί κατά την ανάλυσή του.

5.2 Πολλαπλών τρόπων διαχωρισμός επίπεδων γραφημάτων

Σε αυτή την ενότητα θα δείξουμε πώς να διαχωρίσουμε ένα επίπεδο γράφημα G σε $\Theta(N/R)$ υπογραφήματα που το καθένα έχει $O(R)$ κορυφές και ένα σύνολο $O(sort(N))$ κορυφών διαχωριστών χρησιμοποιώντας $O(sort(N))$ I/Os.

Δοθέντος ενός BFS δέντρου T ενός επίπεδου γραφήματος G=(V,E) ο Hutchinson έδειξε πώς να υπολογίζουμε έναν $O(\sqrt{N})$-διαχωριστή του G με O(sort(N)) I/Os. Αυτός ο αλγόριθμος ακολουθεί πιστά τον αλγόριθμο των Lipton και Tarzan : Το BFS δέντρο έχει την ιδιότητα ότι καμία ακμή του G δεν διαπερνά δύο ή περισσότερα επίπεδα ,έτσι κάθε επίπεδο του T είναι ένας διαχωριστής του G. Το «μεσαίο» επίπεδο l_1 του T είναι το επίπεδο που περιέχει την κορυφή με τον αριθμό N/2 στην αρίθμηση του BFS .Το «μεσαίο» επίπεδο έχει την ιδιότητα ότι ο συνολικός αριθμός των κορυφών των επιπέδων που βρίσκονται από πάνω του αλλά και ο συνολικός αριθμός των κορυφών των επιπέδων που βρίσκονται από κάτω του είναι μικρότερος του N/2. Το πρόβλημα είναι ότι το επίπεδο l_1 μπορεί να περιέχει περισσότερες από $\sqrt{N}$ κορυφές (ενώ για να γίνει διαχωριστής πρέπει να περιέχει $O(\sqrt{N})$) .Ωστόσο ,υπάρχει ένα επίπεδο l_0 πιο πάνω από το l_1 και ένα επίπεδο l_2 πιο κάτω από το l_1 με $\sqrt{N}$ κορυφές το καθένα τέτοια ώστε l_2- $l_0 \leq \sqrt{N}$.Τα επίπεδα l_0 και l_2 χωρίζουν το G σε τρία υπογραφήματα , τα G_0 ,G_1 και G_2 που αποτελούνται από τις κορυφές των επιπέδων πάνω από το l_0 ,μεταξύ των l_0 και l_2 και κάτω από το l_2 αντίστοιχα, με τις ιδιότητες ότι τα G_0 και G_2 περιέχουν λιγότερες από N/2 κορυφές και το G_1 έχει ένα επικαλύπτον δέντρο με όριο ύψους $\sqrt{N}$.



Εικόνα 5.1: Τα επίπεδα G_0 και G_2 περιέχουν λιγότερες από N/2 κορυφές και το G_1 έχει ένα επικαλύπτον δέντρο με όριο ύψους $\sqrt{N}$.

Μπορούμε να αποδείξουμε ότι για να βρούμε έναν διαχωριστή του G αρκεί να βρούμε έναν διαχωριστή του G_1. Ένας τέτοιος διαχωριστής μπορεί να βρεθεί χρησιμοποιώντας τις ιδιότητες του διπλού γραφήματος του G_1. Το διπλό γράφημα $G_1^*=(V^*,E^*)$ ενός επίπεδου γραφήματος G_1 είναι ένα επίπεδο γράφημα που το αποκτούμε βάζοντας μία κορυφή για κάθε όψη του G_1 και

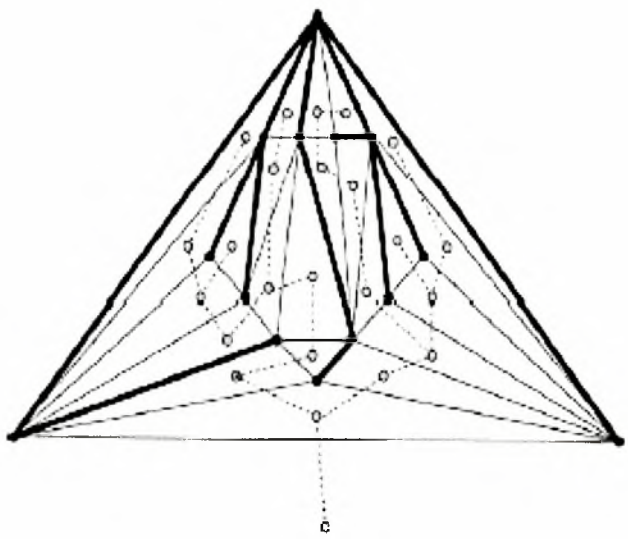
συνδέουμε δύο όψεις f_i και f_j ,που είναι γειτονικές και μοιράζονται μία κοινή ακμή e=(u,υ) του G_1 , με μία ακμή $e^*=(f_i, f_j)$ στο E^*. Η ακμή e^* στο G_1^* ονομάζεται διπλή ακμή της e του G_1. Αναθέτουμε στο Τ' να είναι ένα υποσύνολο των ακμών του G_1 .Είναι γνωστό ότι το Τ' είναι ένα επικαλύπτον δέντρο του G_1 αν και μόνο αν το $(E-T')^*$ είναι ένα επικαλύπτον δέντρο στο G^* .Εάν το Τ΄ είναι επικαλύπτον δέντρο με όριο ύψους $\sqrt{N}$ τότε προσθέτοντας οποιαδήποτε ακμή του (E-T) στο Τ' δημιουργείται κύκλος με το πολύ $2\sqrt{N}$ κορυφές. Υποθέτοντας χωρίς βλάβη της γενικότητας ότι το G είναι τριγωνοποιημένο, Ο Lipton και ο Tarzan απέδειξαν ότι υπάρχει μία ακμή e Є (E-T) τέτοια ώστε ο αριθμός των κορυφών εντός και εκτός του κύκλου που ορίζεται από την e να είναι $\leq$ 2N/3 και έδειξαν πως μπορεί αυτό να υπολογιστεί αποδοτικά χρησιμοποιώντας μία από κάτω προς τα πάνω διαπέραση του διπλού επικαλύπτοντος δέντρου $(E-T')^*$.Ο Hutchinson έδειξε πώς να εκτελέσουμε όλες αυτές τις πράξεις χρησιμοποιώντας O(sort(N)) I/Os με την προϋπόθεση ότι μας δίνεται το BFS δέντρο του G.

Ο πολλαπλός διαχωρισμός ενός επίπεδου γραφήματος είναι το να διαχωρίσουμε το επίπεδο γράφημα G σε Θ(N/R) υπογραφήματα με O(R) κορυφές το καθένα χρησιμοποιώντας ένα σύνολο κορυφών – διαχωριστών S. Ο αλγόριθμος του $O(\sqrt{N})$-διαχωριστή του Hutchinson που χωρίζει το γράφημα σε δύο μισά μπορεί να χρησιμοποιηθεί για να αναπτύξουμε έναν περιοδικά επαναλαμβανόμενο O(log (N/R sort(N)) πολλαπλό αλγόριθμο διαχωρισμού. Σε αυτή την ενότητα θα δείξουμε πως μπορούμε να βελτιώσουμε αυτό το όριο σε O(sort(N)) I/Os χωρίζοντας το G σε περίπου M/B υπογραφήματα (αντί για δύο) σε κάθε βήμα που υλοποιείται σε O(N/B) I/Os. Για να το κάνουμε αυτό βασιζόμαστε σε ιδέες παρόμοιες με τον Goodrich : Ορίζουμε περίπου M/B επίπεδα στο Τ που διαιρούν το G σε υπογραφήματα μεγέθους O(N/(M/B)). Έπειτα χρησιμοποιούμε αυτά τα επίπεδα για να βρούμε ένα σύνολο επιπέδων με κάποιες κορυφές που διαιρούν το G σε υπογραφήματα που το κάθε υπογράφημα είτε είναι μεγέθους O(N/(M/B)) είτε έχει ένα επικαλύπτον δέντρο με όριο ύψους $O(\sqrt{R})$.Υποδιαιρούμε τα γραφήματα με το επικαλύπτον δέντρο περιορισμένου ύψους σε γραφήματα μεγέθους O(R) χρησιμοποιώντας τις ιδιότητες των διπλών γραφημάτων και υποδιαιρούμε τα γραφήματα μεγέθους O(N/(M/B)) περιοδικά.

5.2.1 Υποδιαιρώντας ένα επίπεδο γράφημα με επικαλύπτον δέντρο με όριο ύψους

Σε αυτή την ενότητα θα δείξουμε πως σε O(sort(N)) I/Os μπορούμε να χωρίσουμε ένα επίπεδο γράφημα με ένα επικαλύπτον δέντρο Τ ύψους Η σε Θ(N/R)υπογραφήματα μεγέθους O(R) το καθένα χρησιμοποιώντας O((N/R) H) κορυφές – διαχωριστές.

Για ευκολία υποθέτουμε πως το G είναι τριγωνοποιημένο .(Εάν δεν είναι τότε μπορούμε να το τριγωνοποιήσουμε με O(sort(N))I/Os και να σημειώσουμε τις ακμές που προσθέσαμε έτσι ώστε να μπορούμε να τις αφαιρέσουμε μετά το τέλος του διαχωρισμού. Σημειώνουμε ότι το T εξακολουθεί να είναι επικαλύπτον δέντρο του G και μετά την τριγωνοποιήση.) Αναθέτουμε στο G^* να είναι το διπλό γράφημα του G και στο $T^+=(E-T)^*$ να είναι το επικαλύπτον δέντρο στο G^*. Κάθε ακμή στο T^+ είναι η διπλή μιας ακμής e=(u,υ) στο (E-T) και καθώς υπάρχει ένα μοναδικό μονοπάτι από την u στην υ στο T ,προσθέτοντας την e στο T δημιουργείται κύκλος. Επειδή το T έχει όριο ύψους ίσο με H αυτός ο κύκλος θα περιέχει το πολύ 2H-1 κορυφές. Με αυτόν τον τρόπο κάθε ακμή του T^+ προσδιορίζει έναν κύκλο στο G μεγέθους O(H), που διαχωρίζει το γράφημα σε κορυφές εντός και εκτός του κύκλου. Η κύρια ιδέα του αλγορίθμου μας είναι να εντοπίσουμε O(N/R) ακμές/κύκλους που να χωρίζουν το G σε υπογραφήματα μεγέθους O(R) .Παρακάτω θα δούμε πως μπορούμε να βρούμε O(N/R) ακμές του T^+ που η αφαίρεσή τους χωρίζει το T^+ σε υποδέντρα μεγέθους O(R) και μετά θα δούμε πως οι διπλές γραμμές αυτών των ακμών ορίζουν O(N/R) κύκλους στο G με τις επιθυμητές ιδιότητες.

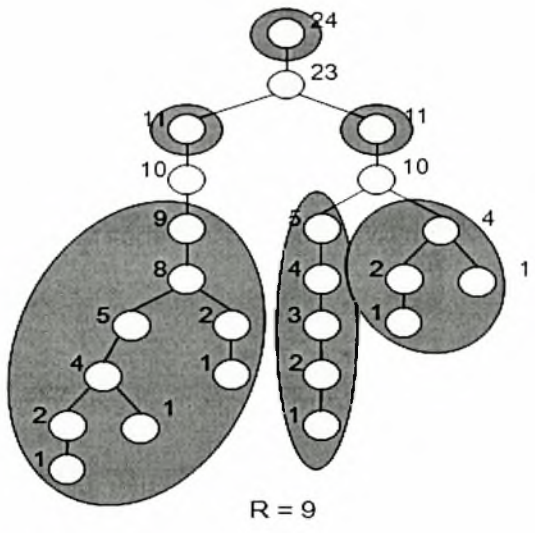


Εικόνα 5.2: Ένα τριγωνοποιημένο γράφημα G (συνεχείς γραμμές) με ένα επικαλύπτον δέντρο T (χοντρές γραμμές) και το διπλό επικαλύπτον δέντρο T^+ (διακεκομμένες γραμμές) .

Παράλληλοι αλγόριθμοι για την διάσπαση ενός δέντρου σε ανεξάρτητα υποδέντρα ίδιου περίπου μεγέθους μελετήθηκαν από τον Gazit. Ανασκοπούμε σύντομα τα κύρια σημεία και τα αποτελέσματα. Αναθέτουμε στο T^+ να είναι ένα δέντρο με N κορυφές. Το βάρος W(υ) μίας

κορυφής υ του δέντρου είναι ο αριθμός των κορυφών του υποδέντρου με ρίζα την υ. Μία κορυφή υ ονομάζεται R-κρίσιμη εάν η υ δεν είναι φύλλο του δέντρου και αν $\lceil w(\upsilon)/R \rceil > \lceil w(\upsilon')/R \rceil$ για όλα τα παιδιά υ' της υ. Ορίζουμε ένα σύνολο C να είναι υποσύνολο των κορυφών του T^+ .Δύο ακμές e και e' του T^+ ονομάζονται C-ισοδύναμες εάν υπάρχει μονοπάτι από την e στην e' που να αποφεύγει της κορυφές του C. Τα γραφήματα που δημιουργούνται από τις C-ισοδύναμες ακμές ονομάζονται γέφυρες του C. Οι εφαπτόμενες κορυφές μιας γέφυρας I είναι οι κορυφές της γέφυρας που είναι επίσης στο C. Οι R-γέφυρες του T^+ είναι οι γέφυρες του συνόλου των R-κρίσιμων κορυφών του T^+ .Ο Gazit απέδειξε τα ακόλουθα:

1. Ο αριθμός των R-κρίσιμων κορυφών δέντρου μεγέθους N είναι το πολύ 2N/R - 1.
2. Αν ο βαθμός του T^+ έχει όριο d τότε ο αριθμός των R-γεφυρών είναι το πολύ d(2N/R -1).
3. Ο αριθμός των κορυφών μίας R-γέφυρας είναι το πολύ R+1.
4. Μία R-γέφυρα έχει το πολύ δύο εφαπτόμενες κορυφές.



Εικόνα 5.3: Τα βάρη της κάθε κορυφής του T^+ .Οι κορυφές με βάρη 23 και 10 είναι οι R-κρίσιμες, άρα αποτελούν το σύνολο C και σκιαγραφημένες είναι οι R-γέφυρες του γραφήματος.

Χρησιμοποιώντας τις παραπάνω ιδιότητες μπορούμε εύκολα να βρούμε O(N/R) ακμές τέτοιες ώστε η αφαίρεσή τους να διαιρεί το T^+ σε O(N/R) υποδέντρα μεγέθους O(R) : Το T^+ είναι ένα δυαδικό δέντρο καθώς το G είναι τριγωνοποιημένο γράφημα και έτσι έχει το πολύ 4N/R R-γέφυρες μεγέθους R+1 η καθεμία. Οι δύο ακμές της κάθε R-γέφυρας που είναι προσκείμενες στις εφαπτόμενες κορυφές της ορίζουν δύο κύκλους στο G.Ο ένας από αυτούς τους κύκλους θα είναι μέσα στον άλλον και οι όψεις στο εσωτερικό του εξωτερικού κύκλου αλλά στο εξωτερικό του εσωτερικού κύκλου είναι οι όψεις που αναφέρονται στις κορυφές της γέφυρας. Καθώς η γέφυρα περιέχει το πολύ R+1 κορυφές (όψεις στο G), δύο ακμές (κύκλοι στο G) ορίζουν ένα υπογράφημα στο G μεγέθους το πολύ ίσο με 3(R+1). Γενικά επειδή κάθε κύκλος περιέχει O(H) κορυφές ,οι O(N/R) R-γέφυρες και οι αντίστοιχες γειτονικές ακμές ορίζουν O((N/R) H) κορυφές διαχωριστές που διαχωρίζουν το G σε O(N/R) υπογραφήματα των O(R) κορυφών. Δοθέντων των R-γεφυρών η διάσπαση του G μπορεί εύκολα να υπολογιστεί σε O(sort(N)) I/Os.

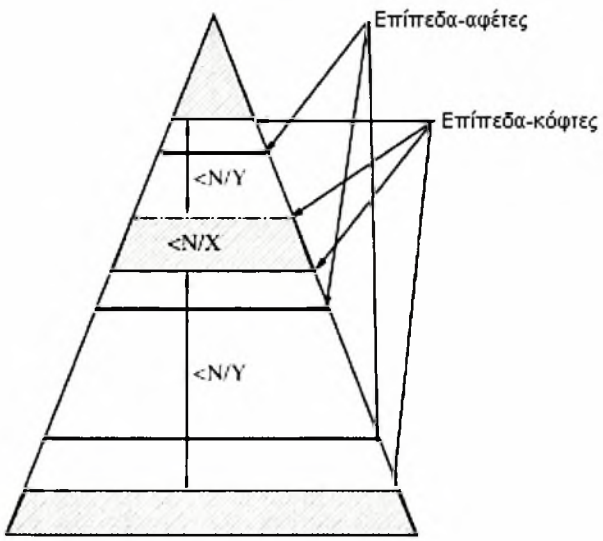
ΛΗΜΜΑ 5.1:

Ένα επίπεδο γράφημα G με ένα επικαλύπτον δέντρο ύψους H μπορεί να διαμοιραστεί σε Θ(N/R) υπογραφήματα μεγέθους O(R) χρησιμοποιώντας O((N/R)H) κορυφές διαχωριστές σε O(sort(N))I/Os.

5.2.2 Υποδιαιρώντας επίπεδα γραφήματα

Τώρα είμαστε έτοιμοι να περιγράψουμε τον αλγόριθμο διαχωρισμού πολλαπλών τρόπων με λεπτομέρειες. Ορίζουμε ένα επίπεδο γράφημα G=(V,E) που έχει δέντρο BFS T και ορίζουμε L(i) να είναι ο συνολικός αριθμός κορυφών των επιπέδων 0 έως i στο T. Δοθείσας μίας παραμέτρου X<N ορίζουμε ως επίπεδα-αφέτες τα επίπεδα i για τα οποία τα μεσοδιάστημα (L(i),L(i+1)) περιέχει ένα πολλαπλάσιο του $\lceil N/X \rceil$.Είναι εύκολο να διαπιστώσουμε ότι υπάρχουν το πολύ X επίπεδα-αφέτες και ότι ο αριθμός κορυφών μεταξύ δύο συνεχόμενων επιπέδων-αφετών είναι μικρότερος από $\lceil N/X \rceil$ Όπως και το επίπεδο l_1 στον αλγόριθμο του Lipton και Tarzan ,τα επίπεδα-αφέτες διαιρούν το γράφημα σε υπογραφήματα μικρού μεγέθους. Ωστόσο τα επίπεδα-αφέτες μπορεί να περιέχουν πολλές κορυφές. Επομένως εξετάζουμε το πρώτο επίπεδο πάνω από

κάθε επίπεδο-αφέτη όπως και το πρώτο επίπεδο κάτω από κάθε επίπεδο-αφέτη που περιέχει το πολύ Y κορυφές για μία παράμετρο Y<N. Ονομάζουμε αυτά τα επίπεδα επίπεδα-κόφτες. Τα επίπεδα-κόφτες διαιρούν το γράφημα G σε O(X) υπογραφήματα που αποτελούνται από τις κορυφές που βρίσκονται μεταξύ δύο συνεχόμενων επιπέδων-κοφτών. Εάν τα δύο επίπεδα-κόφτες που ορίζουν το υπογράφημα G_i βρίσκονται μεταξύ δύο συνεχόμενων επιπέδων-αφετών τότε το G_i έχει μέγεθος O(N/X), διαφορετικά το G_i έχει ένα επικαλύπτον δέντρο ύψους O(N/Y).



Εικόνα 5.4: Επίπεδα-αφέτες και επίπεδα-κόφτες στο T. Τα γραφήματα μεταξύ δύο συνεχόμενων επιπέδων-κοφτών είτε έχουν μέγεθος μικρότερο από N/X είτε έχουν επικαλύπτον δέντρο ύψους μικρότερου από N/Y.

Για να υπολογίσουμε τον διαχωρισμό πολλαπλών τρόπων του γραφήματος G διαιρούμε τα γραφήματα με όριο ύψους χρησιμοποιούμε το ΛΗΜΜΑ 5.1 της προηγούμενης ενότητας : Ένα επίπεδο γράφημα G με ένα επικαλύπτον δέντρο ύψους H μπορεί να διαμοιραστεί σε Θ(N/R) υπογραφήματα μεγέθους O(R) χρησιμοποιώντας O((N/R)H) κορυφές διαχωριστές σε O(sort(N))I/Os . Έπειτα επαναληπτικά διαιρούμε και τα γραφήματα μεγέθους O(N/X). Για να το κάνουμε αυτό χρειαζόμαστε ένα BFS δέντρο για κάθε υπογράφημα G_i .Το τμήμα του T που αναφέρεται σε κάθε υπογράφημα δεν είναι BFS δέντρο γιατί δεν είναι συνεκτικό. Ωστόσο μπορούμε εύκολα να παράγουμε ένα BFS δέντρο για κάθε υπογράφημα εισάγοντας μία ψεύτικη ρίζα v_i και συνδέοντας την με ψεύτικες ακμές με όλες τις κορυφές που βρίσκονται κάτω από το πάνω επίπεδο-κόφτη που ορίζει το συγκεκριμένο υπογράφημα G_i .Εάν το T είναι δοσμένο

επίπεδο-επίπεδο τότε τα BFS δέντρα για όλα τα G_i μπορούν εύκολα να υπολογιστούν σε O(N/B) I/Os. Καθώς η v_i αντικαθιστά τουλάχιστον μία κορυφή στο επίπεδο-κόφτη το συνολικό μέγεθος των υπογραφημάτων οποιουδήποτε επιπέδου της επανάληψης θα παραμένει O(N). Οι ψεύτικες ακμές και κορυφές σημειώνονται και αφαιρούνται από το τελικό διαχωρισμένο γράφημα. Αυτό γίνεται εύκολα σε O(N/B)I/Os.

Επιλέγοντας $Y=N/\sqrt{R}$ κάθε υπογράφημα G_i μεγέθους N_i με όριο ύψους έχει ύψος $\sqrt{R}$ (ύψος= N_i/Y) και έτσι μπορεί να διαχωριστεί χρησιμοποιώντας το ΛΗΜΜΑ 5.1 σε $\Theta(N_i/R)$ υπογραφήματα μεγέθους O(R) χρησιμοποιώντας $O((N_i/R)H)=O(N_i/R)$ κορυφές διαχωριστές. Εκτός από τις O(N/R) κορυφές-διαχωριστές που χρησιμοποιούνται για να διαχωρίσουν καθένα από τα ,το πολύ X , υπογραφήματα με όριο ύψους ,τα επίπεδα κόφτες, που είναι το πολύ X, προσθέτουν ακόμα $O(X\ Y) = O(X\ N/\sqrt{R})$ κορυφές-διαχωριστές. Έτσι ο συνολικός αριθμός κορυφών διαχωριστών δίνεται από :$S(N) \leq O(X\ N/\sqrt{R}) + O(N/R)+X\ S(N/X)$, Εάν επιλέξουμε $X=(M/B^2)^{1/4}$ και υπό την υπόθεση ότι $M>B^{2+\varepsilon}$, για κάποιο $\varepsilon>0$, μπορεί να αποδειχτεί ότι $X\ N/\sqrt{R} = O(N/B)$ και ότι $\log_x N/R=O(\log_{M/B} N/B)$, και έτσι S(N)=O(sort(N)).

Το ότι ο αλγόριθμός μας χρησιμοποιεί O(sort(N))I/Os μπορούμε να δούμε παρακάτω:

Το βήμα προεπεξεργασίας της αναπαράστασης του T ανά επίπεδο και έτσι ο υπολογισμός του BFS επιπέδου της κάθε κορυφής μπορούν εύκολα να εκτελεστούν σε O(sort(N))I/Os χρησιμοποιώντας τεχνικές όπως η ταξινόμηση λίστας και ο κύκλος του Euler. Εάν δεν συνυπολογίσουμε τις I/O πράξεις που χρησιμοποιούνται για τον διαχωρισμό των υπογραφημάτων με όριο ύψους ,κάθε βήμα επανάληψης μπορεί νε εκτελεστεί σε O(N/B)I/Os ,και η επανάληψη για τον αριθμό των I/Os γίνεται $T(N)\leq N/B+X\ T(N/X)=O(sort(N))$. Επειδή δεν επιστρέφουμε στα γραφήματα G_i με όριο ύψους αλλά τα υποδιαιρούμε απ' ευθείας χρησιμοποιώντας $O(sort(G_i))$ I/Os το συνολικό κόστος για να διαχωρίσουμε όλα τα υπογραφήματα όλων των επιπέδων της επανάληψης είναι O(sort(N)) I/Os.

Έχουμε όμως μόνο μελετήσει την περίπτωση όπου $R>B\sqrt{M}$.Εάν θέλαμε να διαμοιράσουμε το γράφημα G σε υπογραφήματα μεγέθους $R\leq B\sqrt{M}<M$ τότε μπορούμε πρώτα να χρησιμοποιήσουμε τον παραπάνω αλγόριθμο για να διαμοιράσουμε το G σε υπογραφήματα μεγέθους O(M) και έπειτα να φορτώσουμε κάθε τέτοιο υπογράφημα στην εσωτερική μνήμη διαδοχικά και να εφαρμόσουμε τον αλγόριθμο των Lipton και Tarzan επαναληπτικά μέχρις ότου όλα τα υπογραφήματα να έχουν μέγεθος O(R). Αυτό εισάγει $O(N/\sqrt{R})$ κορυφές-διαχωριστές.

ΘΕΩΡΗΜΑ 5.1:

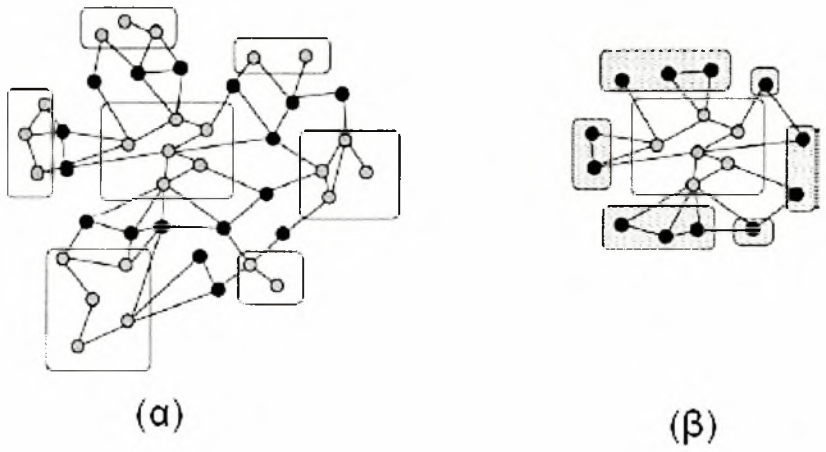
Ορίζουμε ένα επίπεδο γράφημα G=(V,E) και T το BFS δέντρο του. Για κάθε τιμή του R το γράφημα μπορεί να διαμοιραστεί σε O(N/R) υπογραφήματα G_i μεγέθους O(R) χρησιμοποιώντας ένα σύνολο S = O(sort(N)+ $N/\sqrt{R}$) κορυφών-διαχωριστών μέσα σε O(sort(N))I/Os.

Σε κάποιες εφαρμογές διαχωρισμού γραφημάτων είναι απαραίτητο να έχουμε όριο όχι μόνο στον αριθμό των κορυφών-διαχωριστών αλλά και στον αριθμό των κορυφών-διαχωριστών που συνορεύουν με κάθε υπογράφημα, αυτές τις κορυφές τις ονομάζουμε κορυφές ορίου του G_i ή για συντομία ∂G_i του G_i (η ένωση του G_i και του ορίου ∂G_i ορισμένες φορές ονομάζεται περιοχή). Ο Fredrickson ανέπτυξε έναν αλγόριθμο για να μετατρέψει τον διαχωρισμό ένας επίπεδου γραφήματος G με όριο βαθμού με ένα σύνολο S=O($N/\sqrt{R}$) κορυφές-διαχωριστές σε O(N/R) υπογραφήματα G_i μεγέθους O(R),έτσι ώστε καθένα από τα υπογραφήματα να έχει O(S/(N/R)) =O($\sqrt{R}$) κορυφές ορίου. Ο αλγόριθμος προχωρά με τον υπολογισμό του διαχωρισμού πολλαπλών τρόπων σε κάθε υπογράφημα $\partial G_i \cup G_i$. Χρησιμοποιώντας το θεώρημα ΘΕΩΡΗΜΑ 5.1 και επιλέγοντας το R τέτοιο ώστε sort(R)= O($N/\sqrt{R}$) πετυχαίνουμε έναν διαχωρισμό με S=O($N/\sqrt{R}$) κορυφές-διαχωριστές .Έτσι σε αυτή την περίπτωση έχουμε R=O($B^2/(\log^2_{M/B}N/B)$)=O(B^2)=O(M) και επειδή το ∂G_i έχει επίσης O(M) κορυφές εξαιτίας του ορίου βαθμού μπορούμε άμεσα να εφαρμόσουμε τον αλγόριθμο του Fredrickson ,δηλαδή να φορτώσουμε κάθε υπογράφημα G_i με το ∂G_i του στην κύρια μνήμη διαδοχικά και να εφαρμόσουμε έναν αλγόριθμο διαχωρισμού για να πετύχουμε έναν διαχωρισμό όπου το κάθε υπογράφημα θα έχει O($\sqrt{R}$) κορυφές ορίου. Καθώς αυτό χρειάζεται O((N/R)(R/B))=O(N/B) I/Os παίρνουμε το ακόλουθο:

ΛΗΜΜΑ 5.2:

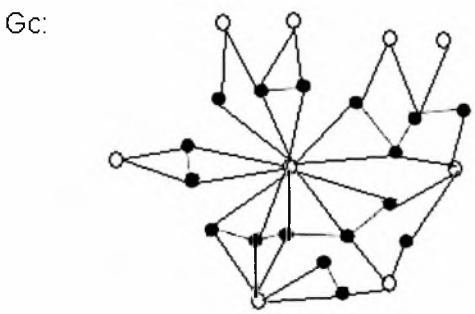
Εάν το G=(V,E) είναι ένα επίπεδο γράφημα με όριο βαθμού και T είναι ένα BFS δέντρο του τότε για R=O($B^2/(\log^2_{M/B}N/B)$) το G μπορεί να διαχωριστεί μέσα σε O(sort(N))I/Os σε O(N/R) υπογραφήματα G_i μεγέθους O(R), χρησιμοποιώντας ένα σύνολο S= O($N/\sqrt{R}$) κορυφές-διαχωριστές ,καθένα από τα οποία να έχει O($\sqrt{R}$) κορυφές ορίου.

Ένα σύνολο ορίου σε έναν διαχωρισμό πολλαπλών τρόπων είναι το μέγιστο υποσύνολο κορυφών διαχωριστών τέτοιες ώστε όλες οι κορυφές στο υποσύνολο να είναι γειτονικές στα ακριβώς ίδια υπογραφήματα.



Εικόνα 5.5: (α)Ο διαχωρισμός ενός γραφήματος σε υπογραφήματα (μέσα σε κουτάκια) και οι κορυφές-διαχωριστές (οι σκούρες) . (β) ένα υπογράφημα του διαχωρισμού με τα σύνολα ορίου (μέσα στο γκρίζα κουτάκια) των κορυφών ορίων του.

Ο Fredrickson ανέπτυξε έναν αλγόριθμο για να μετατρέψει τον διαχωρισμό ένας επίπεδου γραφήματος G με όριο βαθμού με ένα σύνολο $S=O(N/\sqrt{R})$ κορυφές-διαχωριστές σε O(N/R) υπογραφήματα G_i μεγέθους O(R),έτσι ώστε ο αριθμός των συνόλων ορίου να είναι O(N/R). Το G^- είναι ένα γράφημα που το αποκτούμε απομακρύνοντας τις κορυφές S από το G. Ο αλγόριθμος μελετά τις συνεκτικές συνιστώσες του G^- ομαδοποιώντας τες κατάλληλα. Χρησιμοποιεί μόνο την πληροφορία γειτνίασης των συνεκτικών συνιστωσών και αυτό γιατί δουλεύει σε ένα γράφημα G_c που το αποκτούμε από το G συνενώνοντας σε μία κορυφή τις κορυφές της κάθε συνεκτικής συνιστώσας του G^- (και αφαιρώντας τις διπλότυπες ακμές).



Εικόνα 5.6: Το γράφημα G_c.

Καθώς το G_c είναι συνεκτικό και με όριο βαθμού έχει O(S) κορυφές και ακμές. Χρησιμοποιώντας το θεώρημα ΘΕΩΡΗΜΑ 5.1 και επιλέγοντας το R τέτοιο ώστε $N/\sqrt{R}$ = O(sort(N)), αυτό είναι $R=\Omega(B^2/(\log^2_{M/B}N/B))$,πετυχαίνουμε έναν διαχωρισμό με ένα σύνολο S=O(sort(N)) κορυφών-διαχωριστών. Χρησιμοποιώντας έναν O(sort(N)) αλγόριθμο συνεκτικών συνιστωσών και κάποια βήματα σάρωσης και ταξινόμησης μπορούμε εύκολα να υπολογίσουμε το G_c σε O(sort(N)) I/Os .Επειδή το G_c έχει S=O(sort(N)) κορυφές και ακμές μπορούμε άμεσα να εφαρμόσουμε τον αλγόριθμο του Fredrickson και με O(sort(N))I/Os να πετύχουμε διαχωρισμό με O(N/R) σύνολα ορίου.

ΛΗΜΜΑ 5.3:

Εάν το G=(V,E) είναι ένα επίπεδο γράφημα με όριο βαθμού και T είναι ένα BFS δέντρο του τότε για $R=O(B^2/(\log^2_{M/B}N/B))$ το G μπορεί να διαχωριστεί μέσα σε O(sort(N))I/Os σε O(N/R) υπογραφήματα G_i μεγέθους O(R), χρησιμοποιώντας ένα σύνολο S= $O(N/\sqrt{R})$ κορυφές-διαχωριστές ,έτσι ώστε ο αριθμός των συνόλων ορίου να είναι O(N/R).

Συνενώνοντας τα δύο τελευταία λήμματα έχουμε το εξής θεώρημα:

ΘΕΩΡΗΜΑ 5.2:

Εάν το G=(V,E) είναι ένα επίπεδο γράφημα με όριο βαθμού και T είναι ένα BFS δέντρο του τότε για $R=O(B^2/(\log^2_{M/B}N/B))$ το G μπορεί να διαχωριστεί μέσα σε O(sort(N))I/Os σε $O((N/B^2)(\log^2_{M/B}N/B))$υπογραφήματα G_i μεγέθους $O(B^2/(\log^2_{M/B}N/B))$, χρησιμοποιώντας ένα σύνολο S= O(sort(N)) κορυφές-διαχωριστές ,έτσι ώστε:

1.Ο αριθμός των κορυφών ορίου σε κάθε υπογράφημα G_i να είναι $O(B/(\log_{M/B}N/B))$.

2.Ο αριθμός των συνόλων ορίου να είναι $O((N/B^2)(\log^2_{M/B}N/B))$.

5.3 Συντομότερα μονοπάτια μοναδικής πηγής(SSSP)

Ο αλγόριθμος Dijkstra είναι πιθανόν ο πιο γνωστός αλγόριθμος εύρεσης συντομότερων μονοπατιών μοναδικής πηγής. Επιλύει το πρόβλημα SSSP μόνο για βεβαρημένα γραφήματα θετικού κόστους. Ο αλγόριθμος διατηρεί μία διαμέριση κορυφών σε δύο σύνολα : στο πρώτο S ανήκουν οι κορυφές για τις οποίες έχει βρεθεί το συντομότερο μονοπάτι από την πηγή ,ενώ στο δεύτερο V-S ανήκουν οι κορυφές για τις οποίες είναι γνωστή μία προσέγγιση της τελικής λύσεως. Οι προσεγγίσεις αυτές βελτιώνονται, μέσω διαδοχικών χαλαρώσεων ακμών . Ο αλγόριθμος διατηρεί μία ουρά προτεραιότητας για τις κορυφές που δεν έχουν ακόμη συμπεριληφθεί στο δέντρο. Αξίζει πάντως να σημειωθεί η ομοιότητα του αλγορίθμου με τον αλγόριθμο υπολογισμού ελάχιστου επικαλύπτοντος δέντρου Prim. Και οι δύο διατηρούν μία διαμέριση του συνόλου των κορυφών σε δύο διαζευγμένα σύνολα ,ενώ η επιλογή για την επέκταση του ενός συνόλου ,με αφαίρεση μίας κορυφής από το άλλο, στηρίζεται σε μία ουρά προτεραιότητας. Όπως και στην περίπτωση του Prim μία άμεση υλοποίηση του αλγορίθμου Dijkstra αποδοτική με πράξεις I/O.

Σε αυτή την ενότητα θα δείξουμε πώς να χρησιμοποιήσουμε τα αποτελέσματα του Fredrickson που δείξαμε στην προηγούμενη ενότητα για να πετύχουμε μία τροποποιημένη και αποδοτική σε πράξεις I/O εκδοχή του αλγορίθμου Dijkstra σε επίπεδα γραφήματα με όριο βαθμού. Η κύρια ιδέα του αλγορίθμου είναι να χρησιμοποιήσουμε τον διαχωρισμό πολλαπλών τρόπων για να ελαττώσουμε ένα πρόβλημα εύρεσης συντομότερων μονοπατιών μοναδικής πηγής σε ένα γράφημα G που δεν είναι επίπεδο με O(N) κορυφές και ακμές σε ένα ίδιο πρόβλημα σε ένα γράφημα με O(sort(N)) κορυφές και O(N) ακμές και να χρησιμοποιήσουμε το ότι κάθε υπογράφημα είναι γειτονικό με έναν μικρό αριθμό κορυφών-διαχωριστών για να επεξεργαστούμε τις O(N) ακμές αποδοτικά .

Αναθέτουμε στα $G_i = \{V_i,E_i\}$ να είναι τα O(N/R) υπογραφήματα μεγέθους O(R) που αποκτήσαμε από τον διαμοιρασμό του G χρησιμοποιώντας τον αλγόριθμο Fredrickson. Θεωρούμε ένα μονοπάτι P μεταξύ της κορυφής πηγής s και μίας κορυφής t στο G και αναθέτουμε στις $\{s_0,s_1,\ldots\}$ να είναι το σύνολο των κορυφών-διαχωριστών στο P ταξινομημένες με τη σειρά που εμφανίζονται στο P. Το τμήμα του P μεταξύ των s_i και s_{i+1} ανήκει τελείως σε ένα υπογράφημα G_i και θα πρέπει να είναι το συντομότερο μονοπάτι μεταξύ των s_i και s_{i+1} μέσα στο G_i. Έτσι μπορούμε να βρούμε τα συντομότερα μονοπάτια από την s σε όλες τις κορυφές-

διαχωριστές λύνοντας το SSSP πρόβλημα σε ένα γράφημα G^R που το αποκτούμε αντικαθιστώντας κάθε υπογράφημα G_i με ένα πλήρες γράφημα πάνω στις κορυφές ορίου του ,όπου το βάρος μίας ακμής (u,υ) είναι ίσο με το βάρος του συντομότερου μονοπατιού μεταξύ της u και της υ στο G_i. Εάν η πηγή s δεν είναι μία κορυφή-διαχωριστής συμπεριλαμβάνεται επίσης στο G^R συνδεδεμένη με τις κορυφές ορίου του υπογραφήματος στο οποίο ανήκει .Το γράφημα G^R έχει O(sort(N)) κορυφές και καθώς το διαμοιρασμένο G αποτελείται από $O((N/B^2)(\log^2_{M/B}N/B))$υπογραφήματα με $O(B/(\log_{M/B}N/B))$ κορυφές ορίου το καθένα έχει $O((N/B^2)(\log^2_{M/B}N/B)(B^2/(\log^2_{M/B}N/B)) = O(N)$ ακμές.

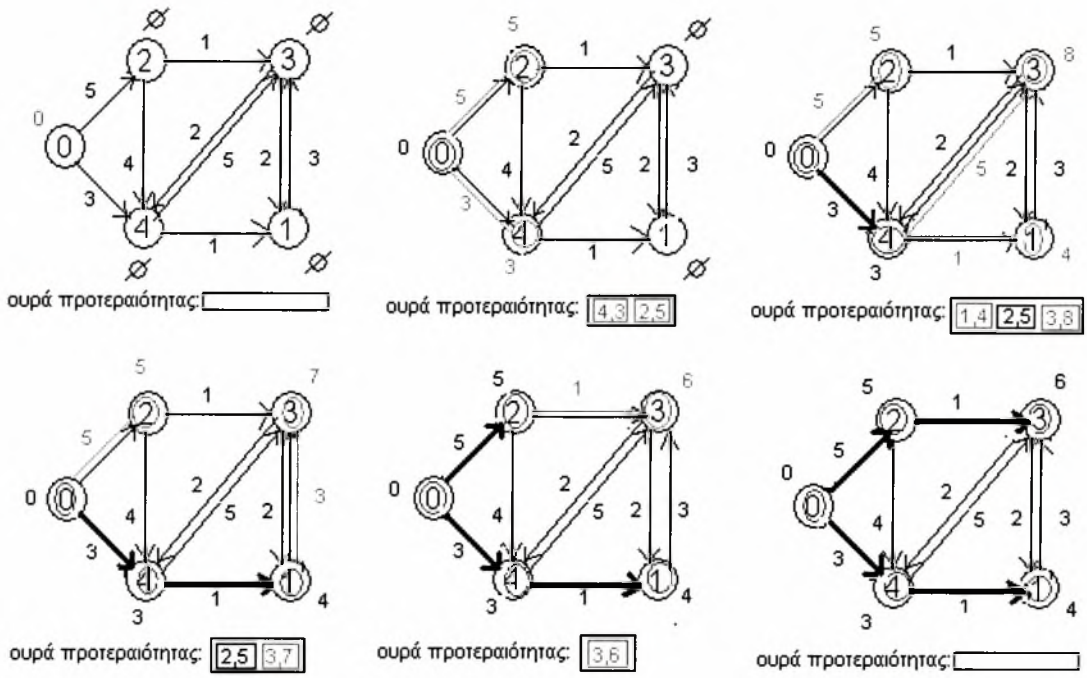
Αφού υπολογίσουμε τον διαχωρισμό του G χρησιμοποιώντας O(sort(N))I/Os ,το G^R μπορεί να υπολογιστεί φορτώνοντας κάθε υπογράφημα G_i με τις κορυφές ορίου του διαδοχικά στην κύρια μνήμη ,χρησιμοποιούμε έναν εσωτερικό αλγόριθμο υπολογισμού συντομότερων μονοπατιών όλων των ζευγών για να υπολογίσουμε τα βάρη των νέων ακμών που αναφέρονται στο εκάστοτε G_i, και γράφουμε αυτές τις ακμές πίσω στον δίσκο. Καθώς καθεμία από τις S κορυφές-διαχωριστές είναι μία κορυφή ορίου για το πολύ O(1) υπογραφήματα (εξ' αιτίας του ορίου βαθμού) χρησιμοποιούμε O(N/B+ S) = O(sort(N)) I/Os για να φορτώσουμε όλα τα υπογραφήματα και τις κορυφές ορίου τους. Επίσης χρησιμοποιούμε O(N/B) I/Os για να γράψουμε τις καινούριες ακμές ,και έτσι το G^R μπορεί να υπολογιστεί σε O(sort(N))I/Os συνολικά. Παρομοίως με τον τρόπο που υπολογίστηκε το G^R από το G σε O(sort(N))I/Os , τα μήκη των συντομότερων μονοπατιών από την s σε όλες τις κορυφές του G μπορούν να υπολογιστούν σε O(sort(N))I/Os άπαξ τα μήκη των συντομότερων μονοπατιών του G^R υπολογιστούν . Απλά φορτώνουμε κάθε υπογράφημα G_i και τις κορυφές ορίου του(τώρα με σημειωμένα τα μήκη των συντομότερων μονοπατιών) στην κύρια μνήμη διαδοχικά, και χρησιμοποιούμε έναν εσωτερικό αλγόριθμο για να υπολογίσουμε το συντομότερο μονοπάτι δ(s,t) από την s σε κάθε κορυφή t που ανήκει στο V_i χρησιμοποιώντας τον τύπο:

$\delta(s,t)=\min_{\upsilon}\{\delta(s,\upsilon)+\delta_{Gi}(\upsilon,t)\}$ όπου το υ κυμαίνεται σε όλες τις κορυφές ορίου του G_i.

Για να επιλύσουμε το πρόβλημα εύρεσης συντομότερων μονοπατιών μοναδικής πηγής στο γράφημα G^R χρησιμοποιώντας O(sort(N))I/Os χρησιμοποιούμε μία τροποποιημένη εκδοχή του αλγορίθμου Dijkstra .Η ιδέα του αλγορίθμου Dijkstra είναι να αναπτύσσει σταδιακά ένα SSSP δέντρο T_G ενώ διατηρεί μία ουρά προτεραιότητας για τις κορυφές που δεν έχουν συμπεριληφθεί ακόμα στο T_G. Η προτεραιότητα μίας κορυφής υ είναι το βάρος του συντομότερου μονοπατιού από την πηγή s στην υ στο οποίο όλες οι ακμές εκτός της τελευταίας ανήκουν στο T_G. Ο αλγόριθμος εξάγει από την ουρά προτεραιότητας την κορυφή υ που έχει την χαμηλότερη προτεραιότητα (και την αντίστοιχη προσκείμενη ακμή της) και την προσθέτει στο T_G. Στη συνέχεια ανανεώνει την προτεραιότητα κάθε κορυφής u που είναι γειτονική με την υ. Ειδικότερα εάν με w_e συμβολίζουμε το βάρος της ακμής e=(υ,u), το βάρος του ,$\delta(s,\upsilon)+w_e$, μονοπατιού από την s στην u μέσω της υ συγκρίνεται με την τρέχουσα προτεραιότητα της u (δηλ. το βάρος του τρέχοντος συντομότερου μονοπατιού για την u) και εκτελείται ανανέωση αν το καινούριο βάρος είναι μικρότερο. Ακόμη και αν το G^R έχει μόνο O(sort(N))κορυφές ,μία άμεση υλοποίηση του αλγορίθμου Dijkstra δεν θα οδηγούσε σε αποδοτικό αλγόριθμο I/O

πράξεων, κυρίως γιατί η τρέχουσα προτεραιότητα της κάθε κορυφής δεν μπορεί να επιτευχθεί χωρίς να εκτελέσουμε μία πράξη I/O. Έτσι η επεξεργασία των O(N) ακμών οδηγεί σε έναν Ω(N) αλγόριθμο.

Παράδειγμα αλγορίθμου Dijkstra :



Για να αποκτήσουμε αποδοτικά την προτεραιότητα μίας κορυφής και έτσι να μπορούμε να εκτελέσουμε O(N) ανανεώσεις/ελαττώσεις προτεραιοτήτων σε O(sort(N)) I/Os χρησιμοποιώντας μία πράξη διαγραφής και μία ένθεσης στις εξωτερικές ουρές προτεραιοτήτων ,εκμεταλλευόμαστε την ομαδοποίηση των κορυφών ορίου σε σύνολα ορίου. Τα σύνολα ορίου μας επιτρέπουν να υλοποιήσουμε τον αλγόριθμο του Dijkstra αποδοτικά με πράξεις I/Os έτσι:

Εκτός από την ουρά προτεραιότητας ,PQ ,των κορυφών διατηρούμε και μία λίστα L με τις τρέχουσες προτεραιότητες των κορυφών ,έτσι διατηρούμε την πληροφορία προτεραιότητας και στην PQ και στην L. Σώζουμε σε συνεχόμενες θέσεις στην L τις κορυφές που ανήκουν στο ίδιο σύνολο ορίου .Ο αλγόριθμος τώρα επανειλημμένα εξάγει την κορυφή υ με την χαμηλότερη

προτεραιότητα από την PQ και φορτώνει στην κύρια μνήμη τις $O(B/(\log_{M/B}N/B))=O(B)$ προσκείμενες ακμές της υ. Έπειτα οι $O(B/(\log_{M/B}N/B))$ κορυφές ορίου που είναι γειτονικές της υ ανακτώνται από την L και καθορίζεται δίχως επιπλέον I/Os ποίες από αυτές τις κορυφές χρειάζεται να ανανεώσουν τις προτεραιότητές τους στην PQ και στην L. Τέλος εκτελούνται οι σχετικές ανανεώσεις στην PQ (χρησιμοποιώντας μία διαγραφή και μία ένθεση σε κάθε ανανέωση) και οι κορυφές ορίου με τις ανανεωμένες προτεραιότητες γράφονται πίσω στην L.

Για κάθε μία από τις O(sort(N)) κορυφές υ του G^R ο αλγόριθμός μας εκτελεί O(1) I/Os για να φορτώσει τις προσκείμενες ακμές της υ. Ο αριθμός των I/Os που χρειάζονται για να φορτώσουμε και να γράψουμε τις $O(B/(\log_{M/B}N/B))$ κορυφές ορίου που είναι γειτονικές τις υ από (στην) L αναλύεται παρακάτω: Καθώς κάθε κορυφή είναι γειτονική σε $O(B/(\log_{M/B}N/B))=O(B)$ κορυφές, κάθε σύνολο ορίου επίσης περιέχει O(B) κορυφές. Καθώς αυτές σώζονται συνεχόμενα στην L ,μπορούμε να φορτώσουμε ένα σύνολο ορίου σε O(1) I/Os. Κατά την διάρκεια όλου του αλγορίθμου ,κάθε σύνολο ορίου προσπελαύνεται $O(B/(\log_{M/B}N/B))$ φορές (μία φορά για καθεμία από τις γειτονικές του κορυφές) ,και έτσι χρησιμοποιούμε $O((B/(\log_{M/B}N/B)) (N/B^2) (\log^2_{M/B}N/B)) = O(\text{sort}(N))$ I/Os συνολικά για να προσπέλαση των $O((N/ B^2) (\log^2_{M/B}N/B))$ συνόλων ορίου στην L. (Σημειώνουμε ότι εάν τα σύνολα ορίου δεν τα σώζαμε στην L συνεχόμενα τότε θα χρησιμοποιούσαμε $O(B/(\log_{M/B}N/B))$ I/Os για να φορτώσουμε τις γειτονικές κορυφές την υ και για τις O(sort(N)) κορυφές θα είχαμε συνολικά $O((B/(\log_{M/B}N/B))\ \text{sort}(N))=O(N)$ I/Os.) Τελικά ο αλγόριθμός μας εκτελεί O(N) πράξεις στην PQ χρησιμοποιώντας O(sort(N))I/Os συνολικά.

ΘΕΩΡΗΜΑ 5.3:

Αν G είναι ένα επίπεδο γράφημα με όριο βαθμού και T είναι ένα BFS δέντρο του τότε τα βάρη των συντομότερων μονοπατιών από μία δοθείσα κορυφή s προς όλες τις υπόλοιπες κορυφές του G μπορούν να υπολογιστούν σε O(sort(N)) I/Os.

Στον παραπάνω αλγόριθμο επικεντρώσαμε την προσοχή μας στον υπολογισμό των βαρών των συντομότερων μονοπατιών στο G. Εάν ενδιαφερόμασταν για τα καθαυτά μονοπάτια δηλαδή για το δέντρο συντομότερων μονοπατιών T_G τότε συνήθεις τεχνικές μπορούν να χρησιμοποιηθούν για να αναπτύξουμε εύκολα έναν αλγόριθμο που θα έχει ως έξοδό του τις ακμές του T_G. Δοθέντος του T_G ο Hutchinson έδειξε πώς να το σώζουμε έτσι ώστε για κάθε κορυφή t να μας επιστρέφει το συντομότερο μονοπάτι μεταξύ της κορυφής πηγής s και της t ,σε P/B I/Os, όπου P είναι ο αριθμός κορυφών του μονοπατιού.

ΠΟΡΙΣΜΑ 5.1:

Αν G είναι ένα επίπεδο γράφημα με όριο βαθμού και T είναι ένα BFS δέντρο του τότε μία δομή δεδομένων μπορεί να κατασκευαστεί σε O(sort(N)) I/Os έτσι ώστε τα συντομότερα μονοπάτια από μία δοθείσα κορυφή πηγή s και μία κορυφή t να μπορούν να βρεθούν σε P/B I/Os, όπου P είναι ο αριθμός κορυφών του μονοπατιού.

Κεφάλαιο 6

APSP

6.1 Εισαγωγή:

[14] Για ένα γράφημα G=(V,E) το πρόβλημα εύρεσης συντομότερων μονοπατιών όλων των ζευγών υπολογίζει τα συντομότερα μονοπάτια μεταξύ όλων των ζευγών των κορυφών του γραφήματος G. Η κλασσική μέθοδος για να επιλύσουμε το APSP πρόβλημα και η πιο απλή που μπορούμε να σκεφτούμε είναι να εφαρμόσουμε διαδοχικά έναν SSSP αλγόριθμο για κάθε κορυφή του γραφήματος .Αυτή η λύση απαιτεί O(V ElogV) I/Os ,καθώς η πολυπλοκότητα του πιο διαδεδομένου αλγόριθμο SSSP είναι O(ElogV) και αυτός θα εφαρμοστεί V φορές (ο αλγόριθμος Dijkstra προϋποθέτει ότι δεν υπάρχουν στο γράφημα αρνητικά βάρη, αλλά αν υπήρχαν τότε θα αναγκαζόμασταν να κατασκευάσουμε ένα βοηθητικό γράφημα για την εξάλειψή τους .Παρόλη αυτή την διαδικασία όμως η πολυπλοκότητα του APSP αλγορίθμου θα παρέμενε η ίδια γιατί οι πράξεις αυτές κοστίζουν γραμμικό χρόνο) .Ακόμα και αν έχουν αναπτυχθεί βελτιωμένοι αλγόριθμοι για το πρόβλημα APSP στα πυκνά γραφήματα (η πιο πρόσφατη καλύτερη λύση στηρίζεται στον πολλαπλασιασμό πινάκων και απαιτεί $O(V^{2.575})$χρόνο) για τα αραιά η κλασσική μέθοδος των διαδοχικών SSSP παραμένει η γρηγορότερη λύση.

Στα μη βεβαρημένα γραφήματα το πρόβλημα SSSP αναφέρεται ως BFS ενώ το APSP ως AP-BFS.Μία άμεση μετατροπή που μπορεί να γίνει στην κλασσική προσέγγιση της λύσης του APSP στην εξωτερική μνήμη απαιτεί έναν SSSP αλγόριθμο για αραιά γραφήματα που να είναι αποδοτικός σε πράξεις I/O. Σε αυτό το κεφάλαιο θα δείξουμε πως το APSP πρόβλημα και το πρόβλημα της εύρεσης διαμέτρου ενός γραφήματος(δηλαδή το πρόβλημα εύρεσης της μεγαλύτερης απόστασης(μήκος συντομότερου μονοπατιού) μεταξύ δύο οποιονδήποτε κορυφών του G) μπορούν να επιλυθούν αποδοτικά ακόμη και αν δεν υπάρχουν αποδοτικοί σε πράξεις I/O BFS/SSSP αλγόριθμοι. Στην ενότητα 1 αυτού του κεφαλαίου θα δώσουμε τον πρώτο αποδοτικό αλγόριθμο εξωτερικής μνήμης για μη κατευθυνόμενα με θετικά βάρη γραφήματα. Υπό την ρεαλιστική συνθήκη $E/V \leq B/\log B$, ο αλγόριθμος μας χρειάζεται $O(V(\sqrt{(VE\log V)/B} + sort(E)))$ I/Os το οποίο είναι $O(V^2\sqrt{(\log V)/B})$ I/Os για αραιά γραφήματα. Εάν συγκρίνουμε αυτό το αποτέλεσμα με το αποτέλεσμα της προηγούμενης προσέγγισης (της λύσης δηλαδή των διαδοχικών SSSP) θα δούμε πως έχει βελτιωθεί κατά έναν παράγοντα μεγαλύτερο από $\Theta(\sqrt{B/V\log V})$. Επιπλέον θα δείξουμε στην ενότητα 2 αυτού του

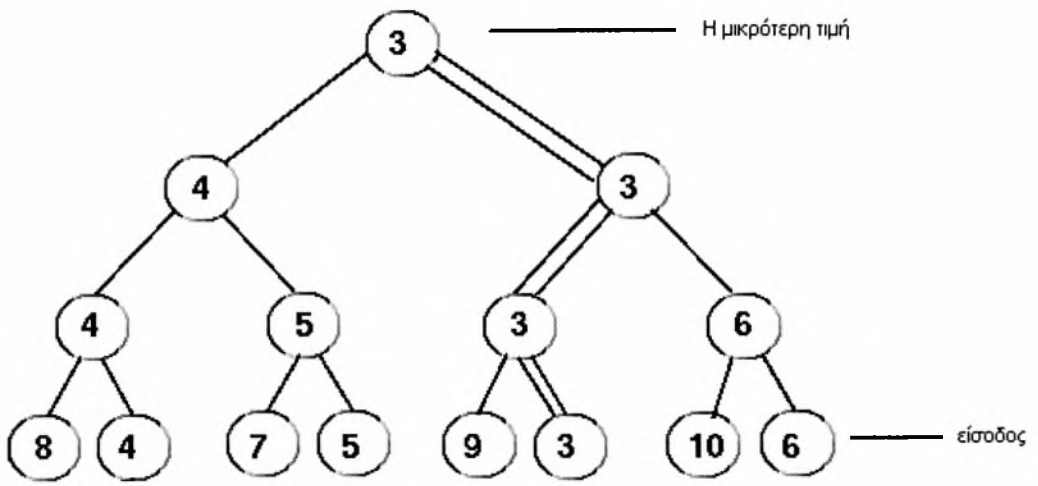
κεφαλαίου ότι ο AP-BFS αλγόριθμος μπορεί να εκτελεστεί με μόνο O(V sort(E)) I/Os.Για αραιά γραφήματα αυτό είναι μία βελτίωση κατά έναν παράγοντα κοντά στο $\sqrt{B}$. Τέλος στην ενότητα 3 αυτού του κεφαλαίου θα μελετήσουμε τα επίπεδα κατευθυνόμενα γραφήματα(διαγράμματα).

6.2 Εξωτερικός αποδοτικός APSP αλγόριθμος σε γενικά αραιά γραφήματα

Σε αυτή την ενότητα θα δώσουμε έναν APSP αλγόριθμο για μη κατευθυνόμενα γραφήματα με μη αρνητικά βάρη ακμών και θα δείξουμε ότι χρειάζεται O(V $(\sqrt{(VElogV)/B} + sort(E))$) I/Os υποθέτοντας πως E/V ≤ B/logB. Προτού περιγράψουμε τον αλγόριθμό μας θα εισάγουμε την έννοια του tournament δέντρου. Ένα tournament δέντρο χρησιμοποιείται για την υλοποίηση της ουράς προτεραιότητας στον SSSP αλγόριθμο. Για την υλοποίηση του APSP θα χρειαστούμε πολλαπλά tournament δέντρα και παράλληλους SSSP υπολογισμούς.

6.2.1 Tournament tree

[16] Ένα tournament δέντρο σώζει όλους τους κόμβους εισόδου στο κατώτατο επίπεδο ενός ολοκληρωμένου δυαδικού δέντρου. Εάν ο αριθμός των κόμβων n δεν είναι δύναμη του 2 τότε συμπληρώνει κόμβους στα δεξιά που έχουν ως κλειδί τους το άπειρο. Εάν το n είναι δύναμη του 2 τότε οι εσωτερικοί κόμβοι είναι n-1, οι συνολικοί κόμβοι είναι 2n-1, και οι ακμές είναι 2(n-1).Η τιμή του κλειδιού ενός εσωτερικού κόμβου είναι η μικρότερη τιμή των δύο παιδιών του. Με αυτόν τον τρόπο η μικρότερη τιμή των κόμβων εισόδου βρίσκεται στην κορυφή .Για αυτό το λόγο είναι χρήσιμο για την υλοποίηση μίας ουρά προτεραιότητας ,όπου προτεραιότητα έχει το στοιχείο με την χαμηλότερη τιμή.



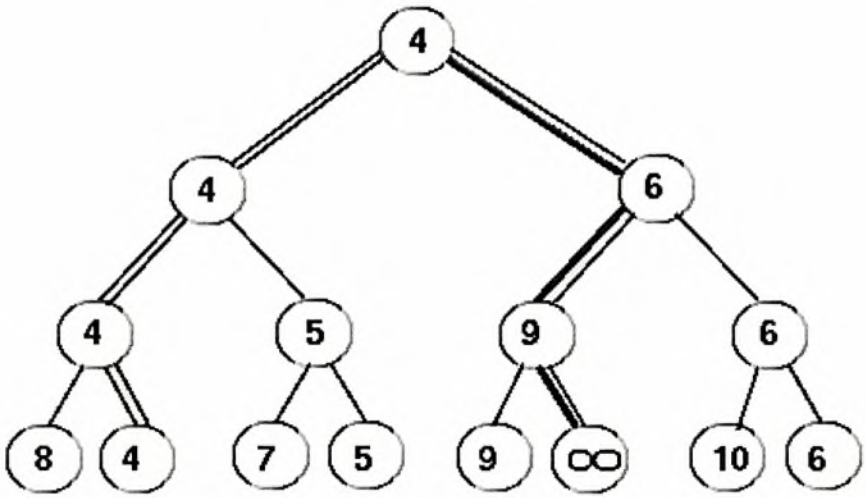
Εικόνα 6.1: Παράδειγμα ενός tournament δέντρου

Το tournament δέντρο στον SSSP υποστηρίζει τρείς πράξεις:

1.deletemin :εξάγει το στοιχείο <x,k> με το μικρότερο κλειδί k ,δηλαδή την κορυφή και διατρέχει όλο το μονοπάτι μέχρι το φύλλο του και το αντικαθιστά με ένα νέο στοιχείο <x,∞> (ώστε να αποκλειστεί μετά την επιλογή του και να μπορέσει να βρεθεί η αμέσως επόμενη κορυφή με την χαμηλότερη προτεραιότητα).

2.delete(x) :αντικαθιστά το <x,oldkey> με <x,∞>.

3.update(x,newkey) :αντικαθιστά το <x,oldkey> με <x,newkey> εάν το newkey<oldkey.



Εικόνα 6.2: Το παράδειγμα του παραπάνω tournament δέντρου μετά από την πράξη deletemin.

Στην περίπτωση του εξωτερικού SSSP αλγορίθμου θα αναθέσουμε σε μία μεταβλητή M' να είναι M' = cM όπου c είναι μία σταθερά c<1 και για τις V κορυφές του γραφήματος στο I/O tournament δέντρο ,I/O TT για συντομία, δεν θα υπάρχουν V φύλλα αλλά $\lceil V/M' \rceil$ φύλλα και για αυτό O(log(V/M)) επίπεδα. Δηλαδή σε κάθε φύλλο θα υπάρχουν περισσότερα από ένα στοιχεία. Τα στοιχεία με δείκτες στην περιοχή {iM',….,(i+1)M' -1} αποτυπώνονται στο i-οστό φύλλο. Η έκταση των δεικτών των εσωτερικών κόμβων του I/O TT δίνεται από την ένωση των εκτάσεων των δεικτών των παιδιών τους. Οι εσωτερικοί κόμβοι κρατούν μία λίστα από το λιγότερο M'/2 έως το περισσότερο M' στοιχεία ταξινομημένες βάσει των προτεραιοτήτων των στοιχείων .Εάν η λίστα ενός κόμβου υ του δέντρου περιέχει z στοιχεία τότε αυτά θα είναι τα μικρότερα z από όλα τα στοιχεία του δέντρου που αποτυπώνονται στα φύλλα που είναι απόγονοι του υ. Επιπλέον

κάθε εσωτερικός κόμβος κρατά και μία εσωτερική μνήμη σήματος μεγέθους M'. Αρχικά το I/O TT σώζει τα στοιχεία < 0,+∞>,<1, +∞>,…,<,V-1,+∞> (μιας και αρχικά οι αποστάσεις όλων των κορυφών είναι +∞) από τα οποία οι λίστες των εσωτερικών κόμβων κρατούν τουλάχιστον M'/2 η καθεμία. Οι πράξεις 1-3 του I/O TT παράγουν σήματα που βοηθούν στη διάδοση της πληροφορίας στο δέντρο προς τα κάτω. Τα σήματα εισάγονται στο κόμβο-ρίζα, ο οποίος πάντα κρατείται στην εσωτερική κύρια μνήμη .Όταν ένα σήμα φτάνει σε έναν κόμβο μπορεί δημιουργεί, να διαγράφει ή να τροποποιεί ένα στοιχείο που κρατείται σε αυτόν τον κόμβο. Το σήμα από μόνο του μπορεί να απορρίπτεται, να μετατρέπεται ή να παραμένει αμετάβλητο. Τα μη-απορριφθέντα σήματα σώζονται στον κόμβο μέχρις ότου η χωρητικότητα της εσωτερικής μνήμης σήματος του συγκεκριμένου κόμβου υπερβληθεί. Έπειτα στέλνονται στο δέντρο προς τα κάτω προς την κατεύθυνση του μοναδικού φύλλου που αποτυπώνει το στοιχείο στο οποίο αναφέρεται το σήμα. Έχει αποδειχτεί το παρακάτω όριο:

ΛΗΜΜΑ 6.1:

Κάθε ακολουθία από z deletemin ,delete ,update πράξεις σε ένα αποδοτικό σε πράξεις I/O tournament δέντρο V στοιχείων απαιτεί το πολύ O(z/B log(V/M)) = O(z/B log(V/B)) I/Os.

6.2.2 Πολλαπλά tournament δέντρα και παράλληλοι SSSP υπολογισμοί

Πρώτα μελετούμε πώς να ομαδοποιήσουμε τα I/O-TTs με σκοπό να πετύχουμε έναν αποδοτικό εξωτερικό APSP αλγόριθμο. Έπειτα παρουσιάζουμε τη νέα μας προσέγγιση ,τον γρήγορο-APSP αλγόριθμο. Σημειώνουμε ότι το όριο του παραπάνω λήμματος το πετύχαμε έχοντας επιλέξει M'=Θ(B). Θέτοντας M'<<B θα πάρουμε ως αποτέλεσμα ένα χειρότερο όριο ίσο με O(z/M' log(V/M')) I/Os το οποίο δεν είναι επιθυμητό. Ωστόσο μία τέτοια επιλογή μας επιτρέπει να ομαδοποιήσουμε τους σχετικούς κόμβους ενός αριθμού I/O-TTs σε ένα μπλοκ του δίσκου. Ειδικότερα θεωρούμε I/O αποδοτικά πολλαπλά TTs ή για συντομία I/OMTTs. Ένα I/O-MTT αποτελείται από L (όπου L≥2,μία παράμετρος που θα καθοριστεί αργότερα) ανεξάρτητα I/O-TTs ,$T_0,\ldots,T_{L-1}$ με παράμετρο M'=Θ(B/L). Συγκεκριμένα ,αυτό σημαίνει ότι οι κόμβοι ρίζες όλων των L ομαδοποιημένων I/O-TTs κρατούνται σε ένα μπλοκ της εσωτερικής κύριας μνήμης ενώ εκτελούνται οι πράξεις στα διαφορετικά I/O-TTs.

Η καινούρια μας προσέγγιση, ο γρήγορος-APSP, λύνει το APSP πρόβλημα δουλεύοντας σε όλα τα βασικά SSSP προβλήματα παράλληλα. Αυτό απαιτεί V ζευγάρια (Q_i,Q_i^*) ουρών προτεραιοτήτων,$0 \leq i \leq V$, όπου οι καταχωρήσεις στο (Q_i,Q_i^*) ζευγάρι ανήκουν στο i-οστό SSSP πρόβλημα. Χρησιμοποιούμε ζευγάρια ουρών προτεραιοτήτων για να θυμόμαστε

αποδοτικά τις κορυφές που έχουν ήδη τακτοποιηθεί. Για να εκτελέσουμε μία πράξη ανανέωσης για κάθε γείτονα μίας κορυφής που έχουμε τακτοποιήσει δεν χρειάζεται να θυμόμαστε και τις κορυφές που έχουμε τακτοποιήσει προηγουμένως .Χρησιμοποιώντας όμως μία ουρά προτεραιότητας Q μπορεί κορυφές που έχουν ήδη τακτοποιηθεί να ξανά-τοποθετηθούν στην Q. Ενώ χρησιμοποιώντας και μία δεύτερη ουρά προτεραιότητας Q^* οι κορυφές που ξανά-εισήχθησαν απομακρύνονται από την Q προτού τις επεξεργαστούμε για δεύτερη φορά. Το σύνολο των ουρών προτεραιοτήτων υλοποιείται με I/O-MTTs. Αυτό απαιτεί $O(V^2)$ χώρο.

Ο γρήγορος-APSP αλγόριθμος διαρκεί O(V) γύρους. Σε κάθε γύρο φορτώνει τις ρίζες όλων των I/O-MMs και εξάγει έναν τακτοποιημένο κόμβο του γραφήματος με την μικρότερη απόσταση από καθένα από τα L/2 ζευγάρια προτεραιοτήτων (Q_i,Q_i^*) του τρέχοντος I/O-MTT. Σημειώνουμε πως για κάθε ζευγάρι (Q_i,Q_i^*) αυτό μπορεί να απαιτεί κάποιες αρχικές πράξεις deletemin στην Q_i^* συνδυασμένες με κάποιες πράξεις delete στην Q_i προτού μπορέσει να εξαχθεί ένας τακτοποιημένος κόμβος από την Q_i. Αντί να προσπελαύνομε τις απαιτούμενες λίστες γειτνίασης των τακτοποιημένων κόμβων για κάθε SSSP πρόβλημα ξεχωριστά ,ο γρήγορος-APSP δημιουργεί μία ακολουθία κόμβων K για τις τακτοποιημένες κορυφές που εξήχθησαν από αυτόν τον γύρο, ταξινομημένες ανάλογα με τους δείκτες των κόμβων τους. Έπειτα εφαρμόζει μία παράλληλη σάρωση στην αναπαράσταση του γραφήματος για να ανακτήσει τις λίστες γειτνίασης των κορυφών που ανήκουν στην K. Τέλος ένα ακόμη βήμα ταξινόμησης και παράλληλης σάρωσης εκτελείται για να μετακινηθούν αυτές οι λίστες γειτνίασης πίσω στα ζευγάρια ουρών προτεραιοτήτων του SSSP προβλήματος από το οποίο προήλθαν ώστε να εκτελεστούν οι απαραίτητες ανανεώσεις προτεραιοτήτων εκεί. Αυτό απαιτεί ακόμα ένα πέρασμα από όλα τα I/O-MTTs τα οποία μπορεί να συμπλέκονται με το ξεκίνημα του επόμενου γύρου. Κατά τη διάρκεια του κάθε γύρου το συνολικό μέγεθος των δεδομένων που σαρώνονται και ταξινομούνται οριοθετείται από $O(V^2)$, και αθροίζοντας όλους τους γύρους οριοθετείται από O(VE). Όλες οι υπολογισμένες τιμές των αποστάσεων απλά να προσαρτηθούν σε μία λίστα εξόδου μεγέθους $O(V^2)$ που είναι ταξινομημένη με σκοπό να παράγει τον τελικό πίνακα αποστάσεων .

Ο γρήγορος-APSP αντικαθιστά τις $\Omega(V^2)$ πράξεις I/O για πρόσβαση στις ξεχωριστές λίστες γειτνίασης με O(sort(VE)) πράξεις I/O για συλλογικές προσβάσεις. Η I/O πολυπλοκότητα για πράξεις στην ουρά προτεραιότητας είναι: Για τους O(V)γύρους ,το πέρασμα από τις ρίζες των O(V/L) I/O-MTTs χρειάζεται $O(V^2/L)$ I/Os. Σε καθένα από τα 2V I/O-TTs (2 γιατί έχουμε 2 ουρές προτεραιότητας που υλοποιούνται με I/O-TT για καθένα από τα V SSSP προβλήματα) εκτελούμε O(E) πράξεις ουράς προτεραιότητας ,ή O(VE logV/B) I/Os συνολικά. Η ταξινόμηση της λίστας εξόδου απαιτεί $O(\text{sort}(V^2))$ I/Os. Το άθροισμα των παραπάνω βελτιστοποιείται επιλέγοντας $L = 2\sqrt{B\left(\frac{V}{E\log V}\right)} \geq 2$.

ΘΕΩΡΗΜΑ 6.1:

Το APSP πρόβλημα στα μη κατευθυνόμενα αραιά γραφήματα με μη αρνητικά βάρη ακμών μπορεί να επιλυθεί χρησιμοποιώντας $O(V\ (\sqrt{\frac{V E \log V}{B}} + \text{sort}(E))\ )$ I/Os και $O(V^2)$ χώρο κάθε φορά που ισχύει $E/V \leq B/\log V$.

6.3 AP-BFS αλγόριθμος σε γενικά γραφήματα

Σε αυτή την ενότητα θα παρουσιάσουμε έναν βελτιωμένο αλγόριθμο για την επίλυση του προβλήματος εύρεσης συντομότερων μονοπατιών όλων των ζευγών σε γενικά μη κατευθυνόμενα μη βεβαρημένα γραφήματα (δηλαδή του προβλήματος AP-BFS). Πρώτα παρουσιάζουμε μία O(V sort(E))I/O λύση ,ωστόσο στην χειρότερη περίπτωση αυτή η προσέγγιση απαιτεί $\Theta(V^2)$ χώρο ακόμα και αν παραλείψουμε να παρουσιάσουμε τον πίνακα εξόδου. Για αυτό προτείνουμε μια εναλλακτική προσέγγιση που χρησιμεύει ως βάση για τις I/O ανταλλαγές χώρου. Σε αυτή την ενότητα θα υποθέσουμε E=O(VB) καθώς για τα πυκνά γραφήματα V φορές ο BFS απαιτεί O(V sort(E))I/Os.

6.3.1 Γρήγορος BFS

Ο AP-BFS αλγόριθμός μας, ο γρήγορος-AP-BFS αλγόριθμος στηρίζεται στον γρήγορο-BFS αλγόριθμο ο οποίος λειτουργεί σε δύο φάσεις .Η μ είναι μία παράμετρος μ Є [0,1] και θα καθοριστεί αργότερα. Στην πρώτη φάση διαμοιράζουμε το G σε O(μV) υπογραφήματα S_i,χρησιμοποιώντας την μέθοδο ενός επικαλύπτοντος δέντρου ή του κύκλου του Euler,τέτοια ώστε η απόσταση μεταξύ οποιωνδήποτε δύο κορυφών του S_i να είναι το πολύ 1/μ στο G. Κάθε S_i αποτελείται από το πολύ 1/μ κορυφές. Για κάθε S_i δημιουργεί μία λίστα F_i η οποία περιέχει τις λίστες γειτνίασης όλων των κορυφών που ανήκουν στο S_i. Κατά τη δεύτερη φάση του γρήγορου-BFS αλγόριθμου διατηρείται μία hot pool H λιστών γειτνίασης. Η hot pool H εμποδίζει την εκτέλεση V τυχαίων προσβάσεων στις λίστες γειτνίασης. Αν το τρέχον επίπεδο L(t) του BFS δέντρου περιέχει μια κορυφή από το υπογράφημα S_i, τότε στην hot pool θα περιέχεται η αντίστοιχη λίστα F_i. Δηλαδή για την πρώτη κορυφή του κάθε υπογραφήματος που επισκεπτόμαστε, στην Η εισάγονται οι λίστες γειτνίασης όλων των κορυφών που ανήκουν στο

ίδιο υπογράφημα, έτσι ο αριθμός των I/Os μειώνεται σε σχέση με τα I/Os που θα χρειαζόμασταν για την εκτέλεση V τυχαίων προσβάσεων στις λίστες γειτνίασης. Η hot pool είναι ταξινομημένη με βάση τους δείκτες των κορυφών. Για να κατασκευάσουμε το επόμενο επίπεδο L(t+1) σαρώνουμε την H και το L(t) παράλληλα. Δουλεύουμε στην H με τις λίστες γειτνίασης που αναφέρονται στις κορυφές που ανήκουν στο L(t) επίπεδο. Σε αυτές διαγράφουμε τις διπλότυπες κορυφές και όσες έχουμε ήδη επισκεφτεί σε προηγούμενα επίπεδα. Οι κορυφές που απομένουν απαρτίζουν το L(t+1) επίπεδο και οι αντίστοιχες F_i τους(που είτε υπάρχουν ήδη στην hot pool είτε εισάγονται τώρα) αποτελούν την νέα H.

Η κύρια ιδέα της ανάλυσης των I/O πράξεων έχει ως εξής: Η σάρωση και η ταξινόμηση όλων των BFS επιπέδων και η εισαγωγή της κάθε F_i στην H χρειάζεται συνολικά O(sort(E)) I/Os. Κάθε F_i αντιγράφεται μέσα στην hot pool ακριβώς μία φορά και επειδή υπάρχουν O(μV) υπογραφήματα αυτό απαιτεί O((μV)+scan(E))I/Os συνολικά. Μόλις η F_i εισαχθεί στην hot pool η λίστα γειτνίασης μίας κορυφής του S_i παραμένει μέσα στην H μέχρις ότου το BFS δέντρο να παραδώσει την κορυφή και τότε διαγράφει την λίστα γειτνίασης της από την H. Επειδή το συντομότερο μονοπάτι μεταξύ δύο οποιωνδήποτε κορυφών του S_i στο G είναι O(1/μ) η λίστα γειτνίασης μίας κορυφής μπορεί να παραμείνει στην H για O(1/μ) επίπεδα το πολύ .Έτσι κάθε λίστα γειτνίασης σαρώνεται O(1/μ) φορές. Για την σάρωση των λιστών γειτνίασης όλων των κορυφών του γραφήματος καθ' όλη την διάρκεια του αλγορίθμου χρειαζόμαστε O(1/μ E/B) I/Os. Ο συνολικός αριθμός των I/O πράξεων για τις δύο φάσεις ελαχιστοποιείται επιλέγοντας $\mu=\sqrt{E/VB}$. Το τελικό συνολικό όριο του γρήγορου-BFS είναι O($\sqrt{V\ E/B}$ +sort(E) loglog(V B/E))I/Os.

6.3.2 Ο γρήγορος-AP-BFS αλγόριθμος:

Ο ακριβής τρόπος για να εκτελέσουμε τον γρήγορο-AP-BFS αλγόριθμο είναι να χρησιμοποιήσουμε τον γρήγορο-BFS για κάθε κορυφή του γραφήματος χρησιμοποιώντας O($V\sqrt{V\ E/B}$ +sort(E) loglog(V B/E))I/Os συνολικά. Αυτό το όριο μπορεί να βελτιωθεί αν εκτελέσουμε τον γρήγορο-BFS παράλληλα για όλες τις πηγές κορυφές που ανήκουν στο ίδιο υπογράφημα και μελετώντας το κάθε υπογράφημα μετά το άλλο. Έτσι ο γρήγορος-AP-BFS αλγόριθμος αρχικά υπολογίζει τον διαμοιρασμό το G σε O(μV) υπογραφήματα S_i όπως γίνεται και στην πρώτη φάση του γρήγορου-BFS. Έπειτα επαναλαμβάνει για κάθε υπογράφημα S_i τα εξής: Ορίζοντας το S ως υπογράφημα-πηγή εκτελεί τον γρήγορο-BFS(s) για όλες τις κορυφές-πηγές s που ανήκουν στο S παράλληλα και διατηρεί μία κοινή hot pool H. Η μετάβαση σε επόμενο επίπεδο L(t) για όλα τα BFS-δέντρα καλείται γύρος. Όπως και στον γρήγορο-BFS κάθε F_i αντιγράφεται μέσα στην hot poll ακριβώς μία φορά για κάθε υπογράφημα-πηγή. Ωστόσο όταν

εισάγεται στην hot pool οι λίστες γειτνίασης κάθε κορυφής υ που ανήκει σε διαφορετικό υπογράφημα από το υπογράφημα-πηγή πρέπει να παραμείνει στην H μέχρις ότου όλα τα BFS-δέντρα του τρέχοντος υπογραφήματος-πηγή S να συμπεριλάβουν την υ. Αυτό χρειάζεται το πολύ O(1/μ) γύρους (αρκούν απλές μέθοδοι καταμέτρησης για να απομακρυνθούν οι αντίστοιχες λίστες μετά από αυτούς τους γύρους από την H). Ας ορίσουμε η εξερεύνηση της κορυφής υ Є S_i με επίπεδο k στο BFS-δέντρο της s Є S να έχει προκαλέσει την ένωση της F_i στην H. Τώρα είναι εύκολο να δούμε ότι το BFS-επίπεδο κάθε άλλης κορυφής υ' Є S_i που αναφέρεται σε οποιοδήποτε άλλο BFS-δέντρο με πηγή s' Є S είναι το πολύ k+3/μ. Αυτό είναι άμεσο συμπέρασμα του διαμοιρασμού του γραφήματος σε S_i υπογραφήματα, που εξασφαλίζει ότι τα συντομότερα μονοπάτια δ(s,s') και δ(υ,υ') στο G είναι το πολύ 1/μ το καθένα.

6.3.3 Πολυπλοκότητα I/O πράξεων του γρήγορου-AP-BFS αλγορίθμου:

Για καθένα από τα O(μV) υπογραφήματα-πηγές έχουμε τα εξής: Η σάρωση και η ταξινόμηση όλων των BFS επιπέδων και η ένωση όλων των F_i στην H χρειάζεται O(sort(E)) I/Os για κάθε κορυφή-πηγή ή O((1/μ)sort(E))I/Os για κάθε υπογράφημα-πηγή. Επίσης ο αριθμός των I/O πράξεων για να φορτώσουμε στην H όλες τις λίστες γειτνίασης είναι O(μV+scan(E)). Τέλος η κάθε λίστα γειτνίασης παραμένει στην H για το πολύ 2/μ γύρους το οποίο μετριέται ως O(scan(E/μ))I/Os. Αθροίζοντας για όλα τα υπογραφήματα ,προσθέτοντας την αρχική φάση προετοιμασίας και ανακαλώντας πως $\mu=\sqrt{E/VB}$ βλέπουμε πως ο γρήγορος-AP-BFS αλγόριθμος απαιτεί O(μV (μV+1/μ sort(E))+sort(E) loglog(V B/E))I/Os = O(V sort(E))I/Os.

ΘΕΩΡΗΜΑ 6.2:

Ο μη κατευθυνόμενος AP-BFS μπορεί να υπολογιστεί χρησιμοποιώντας O(V sort(E))I/Os και $O(V^2)$ χώρο.

Ο γρήγορος-AP-BFS αλγόριθμος μπορεί να χρησιμοποιηθεί για να υπολογίσει την (μη βεβαρημένη) διάμετρο με το ίδιο όριο I/O πράξεων και με O((V+E)/μ)=O($\sqrt{V\,E\,B}$)χώρο. Για αραιά γραφήματα αυτός ο χώρος γίνεται O($V\sqrt{B}$) .

6.4 APSP για επίπεδα κατευθυνόμενα γραφήματα

Σε αυτή την ενότητα θα δείξουμε πώς να υπολογίζουμε τα συντομότερα μονοπάτια όλων των ζευγών ενός επίπεδου διαγράμματος G χρησιμοποιώντας O(scan(N^2)) I/O πράξεις. Στην αρχή θα δώσουμε κάποιες προεισαγωγικές σημειώσεις και έπειτα θα παρουσιάσουμε τον καινούριο μας APSP αλγόριθμο.

6.4.1 Εισαγωγή:

Εάν G=(V,E) ένα επίπεδο γράφημα ένας f(N)-διαχωριστής του G είναι ένα υποσύνολο S των κορυφών του G μεγέθους f(N) του οποίου η αφαίρεση χωρίζει το G σε δύο υπογραφήματα μεγέθους το πολύ 2N/3 το καθένα. Ο Lipton και ο Tarzan έδειξαν πως κάθε επίπεδο γράφημα έχει έναν $O(\sqrt{N})$-διαχωριστή και χρησιμοποιώντας αυτό το αποτέλεσμα περιοδικά ο Fredrickson έδειξε ότι το G μπορεί να διαχωριστεί σε Θ(N/R) υπογραφήματα G_i μεγέθους O(R), καθένα από τα οποία να έχει $O(\sqrt{R})$ κορυφές ορίου, χρησιμοποιώντας συνολικά ένα σύνολο $S= O(N/\sqrt{R})$ κορυφές-διαχωριστές και έχοντας συνολικά μόνο O(N/R) σύνολα ορίων ,για μία παράμετρο $R \in [1,N]$. Η ένωση του G_i και του ορίου ∂G_i ονομάζεται G_i-περιοχή. Αυτή τη διαδικασία την ονομάζουμε διαδικασία R-διαχωρισμού. Για επίπεδα κατευθυνόμενα γραφήματα ο R-διαχωρισμός ορίζεται και υπολογίζεται με τον ίδιο ακριβώς τρόπο αγνοώντας την κατεύθυνση των ακμών. Κύρια ιδέα του SSSP αλγορίθμου για επίπεδα γραφήματα είναι να υπολογίσουμε έναν B^2-διαχωρισμό του G και να μειώσουμε το SSSP πρόβλημα του G σε SSSP πρόβλημα ενός γραφήματος G^R που έχει ως κορυφές του όλες τις O(N/B) κορυφές-διαχωριστές και τις ακμές μεταξύ αυτών που υπάρχουν στο G και για κάθε υπογράφημα G_i για κάθε ζεύγος κορυφών διαχωριστών α,β στο ∂G_i περιέχει μία ακμή (α,β) με βάρος ίσο με το βάρος του συντομότερου μονοπατιού $\delta_{Gi\text{-περιοχή}}(\alpha,\beta)$. Μπορεί να αποδειχτεί ότι για κάθε $\alpha,\beta \in G^R$ το συντομότερο μονοπάτι $\delta_{G^R}(\alpha,\beta) = \delta(\alpha,\beta)$. Έτσι το G^R διατηρεί συντομότερα μονοπάτια του G.

6.4.2 Αποδοτικός APSP αλγόριθμος για επίπεδα γραφήματα:

Για να υπολογίσουμε το συντομότερα μονοπάτια όλων των ζευγών χρησιμοποιούμε την ίδια τεχνική όπως και στον SSSP αλγόριθμο για επίπεδα γραφήματα. Υπολογίζουμε το γράφημα G^R και υπολογίζουμε τα συντομότερα μονοπάτια σε αυτό το γράφημα G^R. Το επιπλέον στοιχείο είναι ότι πρέπει να υπολογίσουμε τα συντομότερα μονοπάτια μεταξύ όλων των κορυφών ενός υπογραφήματος και των κορυφών ολόκληρου του γραφήματος όταν αυτό το υπογράφημα βρίσκεται στην κύρια μνήμη. Τα βασικά βήματα είναι τα εξής:

1. Υπολογίζουμε έναν B^2-διαχωρισμό του αρχικού γραφήματος G με O(N/B) κορυφές-διαχωριστές και $O(N/B^2)$ υπογραφήματα μεγέθους $O(B^2)$ και O(B)σύνολα ορίου το καθένα.
2. Υπολογίζουμε το γράφημα G^R.
3. Για κάθε υπογράφημα G_i :
   (α) Για κάθε κορυφή α που ανήκει στο ∂G_i υπολογίζουμε τα συντομότερα μονοπάτια από την α σε όλες τις υπόλοιπες κορυφές του G^R , δηλαδή το SSSP(α) στο G^R χρησιμοποιώντας έναν SSSP αλγόριθμο για επίπεδα γραφήματα.
   (β) Για κάθε κορυφή u που ανήκει στην G_i-περιοχή υπολογίζουμε το SSSP(u) στο G ως εξής:
   Φορτώνουμε την G_i-περιοχή στην μνήμη. Για κάθε υπογράφημα G_j φορτώνουμε στη μνήμη την G_j-περιοχή και τις αποστάσεις από το ∂G_i στο ∂G_j ,και για οποιεσδήποτε κορυφές u Є G_i-περιοχή και υ Є G_j-περιοχή υπολογίζουμε το συντομότερο μονοπάτι από την u στην υ σαν $d(u,\upsilon)=\min_{\alpha,\beta}\{\delta_{Gi\text{-περιοχή}}(u,\alpha)+ \delta_{G^R}(\alpha,\beta)+ \delta_{Gj\text{-περιοχή}}(\beta,\upsilon)\}$,όπου α Є ∂G_i και β Є ∂G_j. Εάν οι u και η υ βρίσκονται στο ίδιο υπογράφημα τότε το d(u,υ) είναι το μικρότερο από τα d(u,υ) που υπολογίστηκαν με τον παραπάνω τρόπο και της δ(u,υ).

Μπορεί να αποδειχτεί ότι το d(u,υ) είναι πράγματι το συντομότερο μονοπάτι από την u στην υ για το γράφημα G. Τώρα θα συζητήσουμε με περισσότερες λεπτομέρειες τα παραπάνω βασικά βήματα και θα αναλύσουμε την πολυπλοκότητά τους σε I/O πράξεις. Ο υπολογισμός ενός B^2-διαχωρισμού του αρχικού γραφήματος G γίνεται σε O(sort(N)) I/Os. Το γράφημα G^R μπορεί να υπολογιστεί σε O(scan(N)) I/Os. Ορίζουμε ως V_σ μία λίστα κορυφών του G με την παρακάτω σειρά: Όλες οι κορυφές-διαχωριστές βρίσκονται στην αρχή της V_σ ομαδοποιημένες βάσει του συνόλου ορίου τους και μέσα στο ίδιο σύνολο ορίου ταξινομημένες βάσει των δεικτών τους. Έπειτα ακολουθούν οι κορυφές των υπογραφημάτων G_i ,ομαδοποιημένες σύμφωνα με το υπογράφημα στο οποίο ανήκουν και μέσα στο ίδιο υπογράφημα ,ομαδοποιημένες με βάση τους

δείκτες τους .Δεδομένου ότι εμείς γνωρίζουμε για κάθε κορυφή-διαχωριστή το σύνολο ορίου που την εμπεριέχει και για κάθε κορυφή που δεν είναι κορυφή-διαχωριστής το υπογράφημα που την εμπεριέχει ,μπορούμε να κατασκευάσουμε την V_σ χρησιμοποιώντας O(sort(N)) I/Os. Επιπλέον σε χρόνο ταξινόμησης μπορούμε να συσχετίσουμε σε κάθε κορυφή υ την θέση της σ(υ) στην V_σ.

Για κάθε υπογράφημα G_i ο υπολογισμός του SSSP στο G^R από όλες τις κορυφές του ορίου του G_i χρειάζεται O(B sort(N)) I/Os και O(B N/B) = O(N) χώρο. Ορίζουμε ως L_i την παραγόμενη λίστα των αποστάσεων $L_i=\{\delta(\alpha,\beta)|\ \alpha \in \partial G_i, \beta \in G^R\}$. Στην επόμενη φάση του αλγορίθμου θα προσπελάσουμε την L_i προκειμένου να ανακτήσουμε τις αποστάσεις από το ∂G_i στο ∂G_j για κάθε υπογράφημα G_j. Για να γίνουν αυτές οι προσπελάσεις πιο αποδοτικές σώζουμε την L_i έτσι ώστε οι αποστάσεις για τις κορυφές του ίδιου συνόλου ορίου να είναι γειτονικές. Μία τέτοια αναπαράσταση της L_i μπορούμε να την πετύχουμε ταξινομώντας τις αποστάσεις δ(α,β) στην L_i αρχικά βάσει των σ(α) και δευτερευόντως βάσει των σ(β).

Για κάθε υπογράφημα G_i η φόρτωση ενός υπογραφήματος G_j στην μνήμη χρειάζεται $O(|G_i|/B)I/Os = O(B)I/Os$. Η φόρτωση του ορίου του G_j, ∂G_j, χρειάζεται O(1) I/Os για κάθε σύνολο ορίου του ∂G_j (καθώς κάθε σύνολο ορίου έχει μέγεθος O(B)). Επειδή το γράφημα G έχει όριο βαθμού ,κάθε σύνολο ορίου μπορεί να είναι όριο σε O(1) υπογραφήματα. Έτσι για κάθε υπογράφημα G_i η φόρτωση όλων των υπογραφημάτων G_j και των ορίων τους χρειάζεται O(N/B)I/Os. Η ανάκτηση των αποστάσεων για μία κορυφή στο ∂G_i προς όλες τις κορυφές του ∂G_j από την λίστα L_i χρειάζεται O(1) I/Os για κάθε σύνολο ορίου του ∂G_j καθώς όλες οι αποστάσεις για το ίδιο σύνολο ορίου είναι αποθηκευμένες συνεχόμενα στην L_i. Για κάθε κορυφή στο ∂G_i ,αθροίζοντας για όλα τα υπογραφήματα G_j ,κάθε σύνολο ορίου προσπελαύνεται O(1) φορές ,άρα συνολικά $O(N/B^2)$ I/Os.Έτσι για κάθε υπογράφημα G_i ,η ανάκτηση των αποστάσεων από το ∂G_i στο ∂G_j από την L_i αθροισμένη για όλα τα υπογραφήματα G_i χρειάζεται $O(B\ N/B^2) = O(N/B)$ I/Os. Μόλις τα υπογραφήματα G_i-περιοχή και G_j-περιοχή και οι $|\partial G_i||\partial G_j|=O(B^2)$ αποστάσεις από το ∂G_i στο ∂G_j φορτωθούν στην κύρια μνήμη μπορούμε να υπολογίσουμε τα συντομότερα μονοπάτια $d(u,\upsilon)=\min_{\alpha,\beta}\{\delta_{Gi\text{-περιοχή}}(u,\alpha)+\delta_G{}^R(\alpha.\beta)+\ \delta_{Gj\text{-περιοχή}}(\beta,\upsilon)\}$,όπου $\alpha \in \partial G_i$ και $\beta \in \partial G_j$,από τις κορυφές $u \in G_i$-περιοχή προς τις κορυφές $\upsilon \in G_j$-περιοχή ,χωρίς επιπλέον I/O πράξεις, χρησιμοποιώντας έναν συνήθη APSP αλγόριθμο εσωτερικής μνήμης. Υπάρχουν $|G_i\text{-περιοχή}||G_j\text{-περιοχή}|=O(B^4)$ υπολογισμένες αποστάσεις ένα υπογράφημα G_i-περιοχή σε υπογράφημα G_j-περιοχή. Για κάθε υπογράφημα G_i ,αθροίζοντας για όλα τα υπογραφήματα G_j υπάρχουν $O(B^4\ N/B^2)$ υπολογισμένες αποστάσεις συνολικά. Υποθέτουμε πως καθώς υπολογίζουμε τις αποστάσεις εξόδου ,από υπογράφημα G_i-περιοχή σε υπογράφημα G_j-περιοχή ,τις γράφουμε σε μία λίστα L_{ij} με $O(scan(B^4))$I/Os. Για κάθε υπογράφημα G_i αυτό χρειάζεται $O(scan(B^2N))$I/Os συνολικά.

Έτσι για κάθε υπογράφημα G_i, ο υπολογισμός των συντομότερων μονοπατιών από τις κορυφές του υπογραφήματος G_i-περιοχή προς όλες τις κορυφές του G χρειάζεται O(B sort(N)+N/B) = O(B sort(N)) I/Os και η καταγραφή των αποστάσεων εξόδου σε λίστες L_{ij} χρειάζεται

$O(scan(B^2N))$I/Os. Αθροίζοντας για όλα τα υπογραφήματα G_i ο υπολογισμός των συντομότερων μονοπατιών όλων των ζευγών χρειάζεται $O(N/B^2\ (B\ sort(N)+scan(B^2N))\ ) = O(sort(N^2)/B\ + scan(N^2)) = O(scan(N^2))$ I/Os.

ΘΕΩΡΗΜΑ 6.3:

Ο υπολογισμός των συντομότερων μονοπατιών όλων των ζευγών σε ένα επίπεδο κατευθυνόμενο γράφημα μπορεί να γίνει χρησιμοποιώντας $O(scan(N^2))$ I/O πράξεις και $O(N^2)$ χώρο.

Ο APSP αλγόριθμος για επίπεδα διαγράμματα μπορεί να χρησιμοποιηθεί για να υπολογίσουμε την διάμετρο του γραφήματος. Η διαφορά είναι ότι για να υπολογίσουμε την διάμετρο δεν είναι απαραίτητο να παράγουμε ως έξοδο όλα τα συντομότερα μονοπάτια ,αλλά μόνο να κρατάμε την μεγαλύτερη απόσταση που έχουμε συναντήσει μέχρι εκείνη την στιγμή. Έτσι ,ο αλγόριθμος υπολογισμού της διαμέτρου του γραφήματος δεν επιφέρει τα κόστη των $O(scan(N^2))$ I/O πράξεων και $O(N^2)$ χώρου για την καταγραφή όλων των μονοπατιών.

ΘΕΩΡΗΜΑ 6.4:

Η διάμετρος ενός επίπεδου διαγράμματος μπορεί να υπολογιστεί χρησιμοποιώντας O(N) χώρο και $O(sort(N^2)/B)$ I/O πράξεις.

Βιβλιογραφία

[1] Andreas Crauser. LEDA-SM: External Memory Algorithms and Data Structures in Theory and Practice, 2001.

[2] Andreas Crauser and Kurt Mehlhorn .LEDA-SM. A Platform for Secondary Memory Computation , 1999.

[3] Stanslav Drangajov, Vassil Vassilev, Nikolay Dimitrov, Mariana Djelatova .Graph and Network Optimization Software Package, 2005.

[4] Lie-Quan Lee ,Jeremy G. Siek ,Andrew Lumsdaine .The Generic Graph Component Library, 1999.

[5] Emden R. Gansner and Stephen C. North .An open graph visualization system and its applications to software engineering, 1999.

[6] Maurizio Talamo and Paola Vocca. Compact Implicit Representation of Graphs (Extended Abstract), 1998.

[7] Kurt Mehlhorn and Ulrich Meyer. External Memory Breadth-First Search with Sublinear I/O, 2002.

[8] Ulrich Meyer. On Dynamic Breadth-First Search in External Memory, 1998.

[9] Lars Arge ,Ulrich Meyer ,Laura Toma, and Norbert Zeh. On External Memory Planar Depth First Search, 1999.

[10] Adam L. Buchsbaum ,Michael Goldwasser ,Suresh Venkatasubramanian and Jeffery R. Westbrook. On external memory graph traversal.

[11] Roman Dementiev, Peter Sanders and Dominik Schultes. Engineering an external memory minimum spanning tree algorithm.

[12] Lars Arge, Gerth Stoliting Brodal and Laura Toma. On external memory MST, SSSP and multiway planar graph separation, 2002.

[13]] Lars Arge and Laura Toma. External Data Structures for Shortest Path Queries on Planar Digraphs, 2005.

[14] Lars Arge ,Ulrich Meyer and Laura Toma. External Memory for Diameter and All-Pairs Shortest-Paths on Sparse Graphs.

[15] Yi-Jen Chiang, Michael T.Goodrich, Edward F.Grove, Robert Tamassia, Darren Erik Vengroff and Jeffrey Scott Vitter. External Memory Graph Algorithms.

[16] Irit Katriel and Ulrich Meyer. Elementary Graph Algorithms in External Memory, 2003.

[17] Παναγιώτης Μποζάνης. Αλγόριθμοι: σχεδιασμός και ανάλυση, 2003