Δρομολόγηση Και Πολύ-χρωματισμός Μονοπατιών Γραφημάτων

ΚΑΡΑΓΕΩΡΓΟΣ ΤΙΜΟΘΕΟΣ
Α.Μ 1026

Εισαγωγή.

- Το πρόβλημα με το οποίο θα ασχοληθούμε εδώ είναι γνωστό σαν:
  - Δρομολόγηση και Πολύ-χρωματισμός Διαδρομών (*Routing and Path Multi-Coloring Problem - RPMC)*: δοθέντος ενός μη-κατευθυνόμενου γραφήματος G(E, V), μιας λίστας ζευγαριών κόμβων από το V και ενός αριθμού διαθέσιμων χρωμάτων w, καθορίστε μονοπάτια που συνδέουν τους κόμβους των δοσμένων ζευγαριών και χρωμάτων (ανάμεσα στο 1 και στο w), έτσι ώστε το κόστος:
    - $\sum_{e \in E} \max_{1 \le c = w} \mu(e,c)$
  - να είναι ελάχιστο, όπου μ(e,c) είναι η πολλαπλότητα του χρώματος c στην ακμή e (δηλαδή ο αριθμός των μονοπατιών που χρησιμοποιούν την ακμή e και χρωματίστηκαν με το χρώμα c).

Συνέχεια

- Εδώ παρουσιάζουμε αλγόριθμους που λύνουν το RPMC για αλυσίδες βέλτιστα και για αστέρες και δακτυλίους με παράγοντα δύο. Παρόλο που οι άπληστοι αλγόριθμοι δεν φαίνονται ικανοποιητικοί, οι τεχνικές ξεπερνούν το πρόβλημα με ένα αρκετά κομψό τρόπο: με τη χρήση μετασχηματισμών σε χρωματισμό ακμών διμερών γραφημάτων.

Πολύ-χρωματισμός σε Αλυσίδες.

Αλυσίδες

- **Βήμα 1ο:** Για να κάνουμε όλα τα βάρη των ακμών ακέραια πολλαπλάσια του w προσθέτουμε μερικά μονοπάτια μήκους ένα (όπου χρειάζεται). Ο αριθμός των παραπάνω μονοπατιών για μια ακμή e είναι w - (L(e,P) mod w).
- Το βήμα αυτό υλοποιείται σε χρόνο O(m+n*w).

Υλοποίηση.

- Για τα μονοπάτια χρησιμοποιούμε την εξής δομή:
  - struct path{
  - int begin;
  - int end;
  - int color;
  - }P[MAX];
- Για να βρω σε ποια μονοπάτια θα προσθέσω ακμές πρέπει να βρω το βάρος των ακμών στην αλυσίδα:
  - Load[i] = Load[i-1] + beg[i] - end[i];

Αλυσίδες Συνέχεια

- **Βήμα 2ο:** Όσο υπάρχει μονοπάτι p_1 που τελειώνει και μονοπάτι p_2 που αρχίζει στον ίδιο κόμβο, τα ενώνουμε για να κατασκευάσουμε ένα μεγαλύτερο μονοπάτι. Συμβολίζουμε με P΄ τη λίστα με τα μονοπάτια που προκύπτουν.
- Το βήμα αυτό υλοποιείται σε χρόνο O(m+n*w).

Υλοποίηση.

- Χρειαζόμαστε δύο βοηθητικούς πίνακες λιστών. Στη μία λίστα βάζουμε τα μονοπάτια που αρχίζουν σε ένα κόμβο και στην άλλη αυτά που τελειώνουν, πράγμα που το κάνουμε για όλους τους κόμβους. Χρησιμοποιώ την δομή:

```
typedef struct liststr{
int pathid;
    struct liststr *next;
}*list;

list B[MAX], E[MAX];
```

Υλοποίηση Συνέχεια.

- Επίσης χρειαζόμαστε ένα βοηθητικό πίνακα LongPath[] ο οποίος θα αντιστοιχίζει κάθε μονοπάτι του P με το αντίστοιχο μεγάλο μονοπάτι.
- Διατρέχω τις λίστες και, αν δεν τελειώνει κάποιο μονοπάτι στον κόμβο που είμαστε, δίνουμε μια τιμή στον LongPath[] κατά αύξουσα σειρά. Αν όμως τελειώνει, τότε η τιμή που θα πάρει ο πίνακας είναι η τιμή που έχει ο LongPath[] για το μονοπάτι που τελειώνει.

Αλυσίδες Συνέχεια

- **Βήμα 3ο:** Για κάθε κόμβο υ που είναι τελικό σημείο μονοπατιού, το P΄ περιέχει είτε μονοπάτια που αρχίζουν στον υ είτε μονοπάτια που τελειώνουν στον υ. Και στις δύο περιπτώσεις ο αριθμός αυτών των μονοπατιών είναι ακέραια πολλαπλάσια του w. Τα χωρίζουμε σε συλλογές από w στοιχεία με αυθαίρετο τρόπο. Συμβολίζουμε το σύνολο με όλες τις συλλογές μονοπατιών που αρχίζουν (τελειώνουν) με Β (Ε αντίστοιχα).
- Το βήμα αυτό υλοποιείται σε χρόνο O(m'+n).

Υλοποίηση.

- Χρησιμοποιώ τους προηγούμενους βοηθητικούς πίνακες λιστών στα καινούρια «μεγάλα» μονοπάτια. Για κάθε στοιχείο του πίνακα B (E) ελέγχω τη λίστα και, αν η λίστα έχει w στοιχεία, τότε αυτά ανήκουν σε μία συλλογή, αν έχει k*w, τότε έχουμε κ συλλογές.
- Τα αποτελέσματα για τα μονοπάτια που αρχίζουν (τελειώνουν) τα βάζω στον πίνακα GroupB (GroupE).

Αλυσίδες Συνέχεια

- **Βήμα 4ο:** Κατασκευάζουμε ένα διμερές γράφημα H = (B,E,A). Κάθε μονοπάτι στο P ανήκει σε ακριβώς δύο συλλογές, σε μια που "αρχίζει" και σε μια που "τελειώνει". Για κάθε μονοπάτι στο P υπάρχει μία ακμή στο A που ενώνει την συλλογή που αρχίζει και την συλλογή που τελειώνει. Άρα, το H είναι ένα w-κανονικό διμερές γράφημα.
- Το βήμα αυτό υλοποιείται σε χρόνο O(m'+n).

Υλοποίηση.

- Για το διμερές γράφημα χρησιμοποιούμε έναν πίνακα από διπλά συνδεδεμένες λίστες. Χρησιμοποιούμε την εξής δομή:

```
typedef struct setstr{/*Struct for the bipartite graph.*/
    int JoinNode;
    int EdgeLoad;
    int PositionNumber;
    int B_Color[MAX];
    struct setstr *VtoV;
    struct setstr *VtoH1;
    struct setstr *VtoH2;
    struct setstr *next;
    struct setstr *prev;
}*set;
```

Υλοποίηση Συνέχεια.

- Πριν κάνουμε την εισαγωγή, πρέπει να ταξινομήσουμε τα GroupB και GroupE, πρώτα ως προς GroupE και μετά ως προς GroupB. Αυτό το κάνουμε για να έχουμε σε γειτονικές θέσεις τους ίδιους αριθμούς στο GroupE.
- Η ταξινόμηση γίνεται με χρήση του Counting sort σε γραμμικό χρόνο.

Υλοποίηση Συνέχεια.

- Αφού βάλουμε τα στοιχεία στον πίνακα λιστών, ο οποίος αποτελεί το σύνολο V1 διμερούς γραφήματος, πρέπει να φτιάξουμε και το σύνολο V2. Αυτό το σύνολο είναι μια επανάληψη πληροφορίας, αλλά μας είναι χρήσιμο παρακάτω. Για κάθε κόμβο του V1 έχουμε και έναν δείκτη που δείχνει στον αντίστοιχο κόμβο του V2.

Αλυσίδες Συνέχεια

- **Βήμα 5ο:** Χρωματίζουμε τις ακμές του Η με w χρώματα χρησιμοποιώντας έναν κατάλληλο αλγόριθμο. Σε κάθε μονοπάτι του Ρ΄ αναθέτουμε το χρώμα της αντίστοιχης ακμής του Η.
- Με χρήση του αλγορίθμου των Cole και Hopcroft το βήμα απαιτεί χρόνο O(ElogV).

Αλυσίδες Συνέχεια

- **Βήμα 6ο:** Σε κάθε μονοπάτι του P ανατίθεται το χρώμα του αντίστοιχου (μεγάλου) μονοπατιού του P΄.
- Το βήμα αυτό υλοποιείται σε χρόνο O(m+n*w).

Χρωματισμός ακμών σε διμερή γραφήματα.

Χρωματισμός.

- Αν ο μέγιστος βαθμός του γραφήματος είναι περιττός, βρίσκουμε ένα ταίριασμα που καλύπτει τους κόμβους με το μέγιστο βαθμό, χρωματίζουμε με ένα χρώμα τις ακμές του ταιριάσματος και στη συνέχεια να σβήνουμε τις ακμές αυτές από το γράφημα. Μειώνουμε τον αριθμό των χρωμάτων.
- Στη συνέχεια κάνουμε ένα Euler Split.
- Κάνουμε το ίδιο για τα γραφήματα που παίρνουμε από το Euler Split κάθε φορά πρώτα στο μικρότερο γράφημα .
- Ο αλγόριθμος απαιτεί O(ElogV) χρόνο.

Ταίριασμα.

- Ο Αλγόριθμος τρέχει σε O(ELogV).
- Για να βρούμε ένα ταίριασμα που καλύπτει τους κόμβους με μέγιστο βαθμό δουλεύουμε ως εξής:
  - Αν ο μέγιστος βαθμός του γραφήματος είναι ένα τότε το γράφημα είναι το ζητούμενο ταίριασμα.
  - Αλλιώς πραγματοποιείται ένα Partition του γραφήματος. Οι μεγιστοβάθμιοι κόμβοι καθενός από τα δυο γραφήματα που προκύπτουν περιέχουν τους μεγιστοβάθμιους του αρχικού)
  - Στη συνέχεια ο αλγόριθμος εκτελείται πάλι για το γράφημα με τις λιγότερες ακμές.

Partition

- Αν ο μέγιστος Βαθμός του γραφήματος είναι άρτιος τότε κάνουμε ένα Euler split και το αποτέλεσμα είναι το ζητούμενο Partition.

Partition Συνέχεια.

- Αν όμως ο μέγιστος βαθμός είναι περιττός τότε: κρατάμε ένα σύνολο Μ με τους μεγιστοβάθμιους κόμβους. Κάνουμε ένα Euler Split και παίρνουμε δύο νέα γραφήματα. Τώρα κάποιο από αυτά τουλάχιστο τους μισούς μεγιστοβάθμιους του Μ που έχουν περιττό βαθμό. Αυτούς τους κόμβους τους βάζουμε στο σύνολο Μ1 και στο σύνολο Μ2 βάζουμε το Μ-Μ1. Τώρα, μέχρι το Μ2 να γίνει κενό εμείς πρέπει να κάνουμε Euler split στο γράφημα με το μη περιττό μεγιστοβάθμιο κόμβο. Μετά από το Euler split χρησιμοποιούμε την συνάρτηση Union για να ενώσουμε τα προηγούμενα γραφήματα με τα καινούρια.

Euler Split

- Euler Split ενός διμερούς γραφήματος είναι ένα ζευγάρι διμερών γραφημάτων των οποίων οι ακμές προέρχονται από μία διαμέριση των ακμών του αρχικού γραφήματος σε ανοιχτά και κλειστά μονοπάτια, έτσι ώστε ο κάθε κόμβος με περιττό βαθμό είναι στο τέλος ακριβώς ενός ανοιχτού μονοπατιού και κάθε ζυγός δεν είναι στο τέλος κανενός ανοιχτού μονοπατιού.
- Εμείς κατασκευάσαμε έναν αλγόριθμο για πολυγραφήματα ο οποίος απαιτεί χρόνο O(E).

Υλοποίηση.

- Κάνουμε αντιγραφή του V1 σε δύο πίνακες λιστών τους H1, H2. Βάζοντας πολλαπλότητα 0 σε κάθε κόμβο. Στην δομή που εξετάσαμε προηγουμένως υπάρχουν ακόμη δύο δείκτες, οι VtoH1 και VtoH2. Κάθε κόμβος του V1 πρέπει να συνδεθεί με τον αντίστοιχο κόμβο στα H1 και H2. Με αυτό τον τρόπο όταν βρίσκω την κατάλληλη πολλαπλότητα που πρέπει να μπει σε κάθε κόμβο πηγαίνω κατευθείαν και βάζω την αντίστοιχη πολλαπλότητα.
- Όταν τελειώσει το Euler Split πρέπει να ελέγξω αν στα δύο γραφήματα που φτιάξαμε υπάρχουν κόμβοι με πολλαπλότητα 0 αν ναι πρέπει να τους διαγράψουμε.

Υλοποίηση Συνέχεια.

- Αρχίζουμε από έναν κόμβο με περιττό βαθμό:
  - Αν η ακμή που θα διασχίσουμε έχει πολλαπλότητα 2κ+1, έστω από τη μεριά του V1 βάζω στο H1 πλευρά πολλαπλότητας κ+1 και στο H2 πλευρά πολλαπλότητας κ και αλλάζω μεριά, δηλαδή από το V1 πάω στο V2.
  - Αν η ακμή που θα διασχίσουμε έχει πολλαπλότητα 2κ, τότε ενεργούμε όπως στην παραπάνω περίπτωση, αλλά δεν αλλάζουμε μεριά και συνεχίζουμε από το V1.
- Και στις δύο περιπτώσεις διαγράφουμε την ακμή και την συμμετρική της σε O(1). Στη συνέχεια ψάχνουμε για άλλον κόμβο με περιττό βαθμό και ακολουθούμε τα παραπάνω βήματα. Αν δεν υπάρχει κόμβος με περιττό βαθμό, τότε επιλέγουμε έναν κόμβο με άρτιο βαθμό.

Δεύτερος αλγόριθμος ταιριάσματος.

- Η ιδέα του αλγορίθμου είναι να σβήσουμε ορισμένες ακμές από το γράφημα και να αυξήσουμε την πολλαπλότητα σε άλλες κρατώντας όμως σε όλους τους κόμβους τον ίδιο βαθμό που είχαν πριν.
- Αυτό το πετυχαίνουμε βρίσκοντας κύκλους στο αρχικό γράφημα. Για να βρούμε κύκλους κάνουμε μια DFS διέλευση. Όταν βρούμε έναν κύκλο αυξάνουμε και μειώνουμε εναλλάξ κατά την ίδια ποσότητα τις πολλαπλότητες των ακμών του κύκλου. Η DFS διέλευση συνεχίζεται από την κορυφή του κύκλου.
- Αφού τελειώσουμε στη συνέχεια εφαρμόζουμε τον πρώτο αλγόριθμο στο νέο γράφημα.

Δεύτερος αλγόριθμος ταιριάσματος. Συνέχεια.

- Όπως αλλάζουμε τις πολλαπλότητες στο γράφημα από την DFS διέλευση, μπορεί να δημιουργηθούν νέοι κύκλοι με ακμές μεγαλύτερης πολλαπλότητας οπότε πρέπει πάλι να κάνουμε DFS κοκ.
- Εμείς έχουμε γραφήματα με πολλαπλότητα μεγαλύτερη του ένα. Οπότε αναλύουμε την πολλαπλότητα σε άθροισμα δυνάμεων του δύο (κάθε δύναμη του δύο είναι ένα επίπεδο). Πρέπει να κάνουμε DFS για κάθε δύναμη του δύο που είναι μικρότερη ή ίση του μεγίστου βαθμού του γραφήματος.

Δεύτερος αλγόριθμος ταιριάσματος. Συνέχεια.

- Ο σκοπός που γίνεται η παραπάνω DFS διέλευση είναι ώστε το γράφημα που θα προκύψει να έχει λίγες ακμές με μεγάλη πολλαπλότητα. Έτσι ώστε όταν καλέσουμε μετά τον πρώτο αλγόριθμο για το ταίριασμα αυτό να γίνει πιο γρήγορα μιας και το Euler split που χρησιμοποιείται θα κατατάξει πιο γρήγορα τις ακμές.
- Ο χρόνος που απαιτεί ο αλγόριθμος είναι $O(E+VlogV(logD)^2)$ που είναι πάντα τουλάχιστο τόσο καλό όσο το $O(ElogV)$ πρώτου αλγορίθμου.

Δρομολόγηση και Πολύ-χρωματισμός σε Δακτυλίους.

Δακτύλιοι PMC.

- Ορίζουμε το clockwise span (μέτρημα με την φορά του ρολογιού) d(υ) του κόμβου υ να είναι η μέγιστη απόσταση (με κατεύθυνση σύμφωνα με τη φορά του ρολογιού) από τον υ προς τελευταίο κόμβο κάθε μονοπατιού στο Pυ. Αν το Pυ είναι κενό τότε d(υ) = 0.
- Πρώτα ο αλγόριθμος υπολογίζει το clockwise span για όλους τους κόμβους και μετονομάζει τους κόμβους έτσι ώστε ο νέος κόμβος 0 να έχει το ελάχιστο clockwise span. Αντί για d(0) γράφουμε d για συντομία. Μετά ο αλγόριθμος μετατρέπει το δοσμένο στιγμιότυπο δακτυλίου σε στιγμιότυπο αλυσίδας (G',P',w). Το γράφημα αλυσίδα G' περιέχει n+d+1 κόμβους αριθμημένους από το 0 έως το n+d.

Δακτύλιοι Συνέχεια

- Για κάθε μονοπάτι <i, j>εP, αν i<j, τότε και το P’ περιέχει το <i, j>, αλλιώς το P’ περιέχει το <i,j+n>
- Η πολυπλοκότητα του αλγορίθμου είναι ίδια με αυτή του αλγορίθμου για αλυσίδες, μιας και ο υπολογισμός του clockwise span και οι μετασχηματισμοί μπορούν να επιτευχθούν σε χρόνο O(m+n).

Δακτυλιοι RPMC

Ο αλγόριθμος για RPMC κάνει ένα routing ελαχίστων μονοπατιών, δηλαδή επιλέγει το μικρότερο από δύο εναλλακτικά μονοπάτια. Αν και τα δύο είναι ίσα, τότε επιλέγουμε ένα τυχαία. Στη συνέχεια χρησιμοποιεί τον αλγόριθμο για PMC για να χρωματίσει το αποτέλεσμα των μικρότερων μονοπατιών.

Η επιλογή των μικρότερων μονοπατιών απαιτεί O(m) χρόνο.

Πολύ-χρωματισμός σε Αστέρες

Αστέρες

- **Βήμα 1ο:** Αναθέτουμε μια αυθαίρετη διεύθυνση σε κάθε μονοπάτι στο P.

Αστέρες Συνέχεια

- **Βήμα 2ο:** Για κάθε κόμβο υ:
  - ·τα εξερχόμενα μονοπάτια out_{υ} (σύμφωνα με την διεύθυνση που δόθηκε στο βήμα 1), διαιρούνται σε $[out_{\upsilon}/w]$ συλλογές με το πολύ w στοιχεία με αυθαίρετο τρόπο.
  - ·ομοίως, τα εισερχόμενα μονοπάτια in_{υ}, χωρίζονται σε $[in_{\upsilon}/w]$ συλλογές.

Αστέρες Συνέχεια

- **Βήμα 3ο:** ένα διμερές γράφημα $H=(V_{out}, V_{in}, A)$ κατασκευάζεται. Για κάθε μονοπάτι στο P υπάρχει μία ακμή στο A που συνδέει την συλλογή που αρχίζει και την συλλογή που τελειώνει. Άρα το H είναι ένα διμερές γράφημα βαθμού το πολύ w.

Αστέρες Συνέχεια

- **Βήμα 4ο:**Πετυχαίνεται χρωματισμός ακμών στο Η. Σε κάθε μονοπάτι ανατίθεται το αντίστοιχο χρώμα της ακμής στο Η.

Αστέρες Συνέχεια

- Όλα τα βήματα εκτός του 4ου απαιτούν O(m) χρόνο.
- Το 4ο βήμα όπως ήδη αναφέρθηκε με χρήση του αλγορίθμου των Cole και Hopcroft το βήμα απαιτεί χρόνο O(ElogV).