Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή ΗΜ&ΜΥ
Αλγόριθμοι και Πολυπλοκότητα
7ο εξάμηνο, Ροή Λ
Ακαδημαϊκή περίοδος: 2011-2012

3η Σειρά Γραπτών Ασκήσεων

Γερακάρης Βασίλης
<vgerak@gmail.com>
Α.Μ.: 03108092

1 Φεβρουαρίου 2012

1 Προβολή ταινιών

Θεωρούμε ότι οι k ταινίες αποτελούν τους κόμβους και τα n ζεύγη (k_i, k_j) συμβολίζουν τις ακμές ενός μη-κατευθυνόμενου γραφήματος G. Τότε, το ερώτημα του προβλήματος (να χωριστούν οι ταινίες σε 2 ημέρες, και να εμφανίζεται ακριβώς 1 φορά έκαστη) ανάγεται σε έλεγχο διμερότητας του γραφήματος.

Επιλέγουμε (χωρίς βλάβη της γενικότητας) ένα τυχαίο κόμβο s και τον τοποθετούμε στη λίστα του Σαββάτου. Εφαρμόζοντας BFS με αρχικό κόμβο τον s (επίπεδο i=0), τοποθετούμε τους κόμβους που βρίσκονται σε άρτια επίπεδα ($i \mod 2 = 0$) στη λίστα του Σαββάτου και τους κόμβους των περιττών επιπέδων στη λίστα της Κυριακής.

Αν το γράφημα είναι συνεκτικό προχωράμε στο τελικό βήμα, ειδάλλως πρέπει να επαναλάβουμε το BFS με αρχή έναν από τους ανεξερεύνητους κόμβους, εώς ότου καλυφθούν όλες οι συνεκτικές συνιστώσες του γραφήματος.

Τέλος, εξετάζουμε τις n ακμές του γραφήματος:

- Αν δεν υπάρχει ακμή με τα 2 άκρα της στην ίδια λίστα, το γράφημα είναι διμερές και η διαμέριση δίνεται από τις δημιουργημένες λίστες
- Αν υπάρχει τέτοια ακμή, το γράφημα περιέχει κύκλο περιττού μήκους και δε γίνεται να πραγματοποιηθεί η επιθυμητή διαμέρηση χωρίς να προβληθεί τουλάχιστον μία ταινία 2 φορές.

Η υπολογιστική πολυπλοκότητα της μεθόδου μας είναι $\Theta(k+n+n) = O(k+n)$, αφού τόσο είναι το κόστος του BFS και ενός ελέγχου πάνω στις ακμές.

2 Μέτρηση Συντομότερων Μονοπατιών

Για την επίλυση του ερωτήματος θα εφαρμόσουμε μια μικρή μετατροπή στον αλγόριθμο του BFS: Κάθε κόμβος θα περιέχει ένα πεδίο στο οποίο θα αποθηκεύεται το πλήθος των διαφορετικών τρόπων με τον οποίο μπορεί κανείς να φτάσει σε αυτόν. Το πλήθος αυτό προκύπτει αθροίζοντας τις τιμές πλήθους από τους κόμβους που βρίσκονται στο από πάνω επίπεδο και έχουν ακμή προς τον υπό εξέταση κόμβο.

Algorithm 1 Shortest path counting

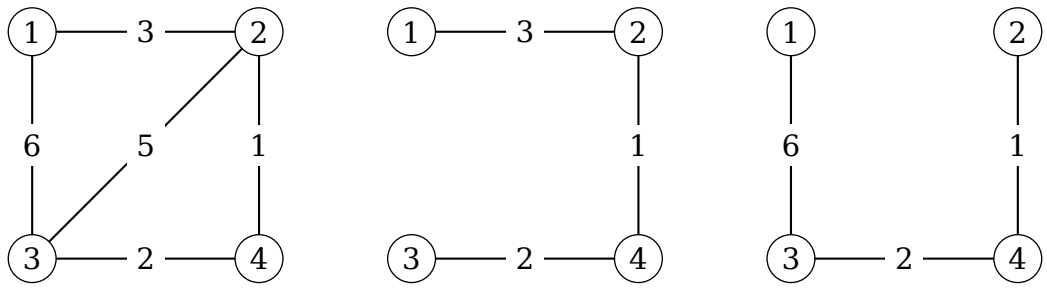
```
1:  procedure cBFS(G(V, e))
2:      L[0] ← s
3:      C[s] ← 1
4:      i ← 0
5:      while L[i] ≠ ∅ do
6:          L[i + 1] = ∅
7:          for each u ∈ L[i] do
8:              if u == target then
9:                  return C[u]
10:             for each e ∈ G.incidentEdges(u) do
11:                 w ← G.opposite(u, e)
12:                 if Discovered[w] == False then
13:                     Discovered[w] ← True
14:                     L[i + 1].append(w)
15:                 if w ∉ L[i] then
16:                     C[w] ← C[w] + C[u]
17:         i ← i + 1
```

Η πολυπλοκότητα του αλγορίθμου είναι $\Theta(m+n)$, αφού υλοποιεί ένα BFS.

Η ορθότητα προκύπτει λογικά, αφού το πλήθος των διαφορετικών μονοπατιών προς ένα κόμβο ισούται με το άθροισμα των διαφορετικών μονοπατιών των κόμβων που οδηγούν σε αυτό. Στην αλγοριθμική υλοποίηση του BFS που παρουσιάζεται παραπάνω, η συνθήκη if (γραμμή 15) μας εξασφαλίζει ότι καταμετρούνται μόνο οι ακμές από το παραπάνω επίπεδο και όχι οριζόντιες ακμές.

3 Ελάχιστο Συνδετικό Δέντρο Υπό Περιορισμούς (DVP 5.24)

3.1 Παράδειγμα



(α') Αρχικό γράφημα (β') MST (γ') L = {2}

3.2 Αλγόριθμος

Μπορούμε να επιλυσουμε το πρόβλημα με 2 τρόπους:

i) Γνωρίζουμε ότι οι κορυφές που βρίσκονται στο L είναι φύλλα, επομένως είμαστε βέβαιοι ότι δε λειτουργούν συνδετικά ούτε δημιουργούν κύκλο. Μπορούμε οπότε να εφαρμόσουμε τον αλγόριθμο του Prim για να υπολογίσουμε το MST του γράφου G\L σε χρόνο $\Theta(m + n \log n)$ και στη συνέχεια για κάθε κόμβο $u \in L$ επιλέγουμε την ελαχίστου κόστους ακμή που τον ενώνει με το MST που βρήκαμε (αποθηκεύοντας την προσωρινά, χωρίς να την προσθέσουμε ακόμα στο MST) σε χρόνο O(n).

Όταν γίνει η επιλογή για όλους τους κόμβους του L, προσθέτουμε τις αποθηκευμένες ακμές στο MST και προκύπτει το επιθυμητό αποτέλεσμα σε χρόνο:
$O(m + n \log n + n) = O(m + n \log n)$.

*(Η προσωρινή αποθήκευση γίνεται για την περίπτωση όπου ένας κόμβος $u_1 \in L$ προστεθεί στο MST και έπειτα η ελάχιστου κόστους ακμή ενός κόμβου $u_2 \in L$ είναι η (u_1, u_2) και λανθασμένα επιλεγεί.)

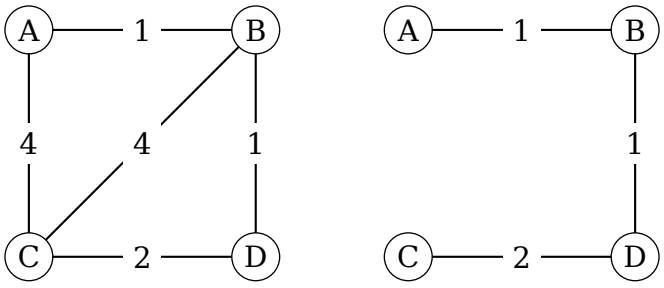
ii) Δημιουργούμε το MST ακολουθώντας τον αλγόριθμο του Kruskal. Κάθε φορά, καθώς ψάχνουμε για κύκλο κρατάμε την τιμή της μέγιστου βάρους ακμής που βρίσκουμε. Αν βρεθεί κύκλος, κρατάμε την τιμή της διαφοράς (βάρος υπο εξέταση ακμής - βάρος μέγιστης ακμής) σε ένα πίνακα.

Όταν τελειώσει η διαδικασία αυτή ο πίνακας θα περιέχει τα κόστη αναπλήρωσης για κάθε ακμή $e \notin MST$. Απο αυτά επιλέγουμε το μικρότερο και τοποθετούμε την ακμή στο MST, αφαιρώντας τη δεύτερη βαρύτερη στον κύκλο που θα σχηματιστεί. Η υλοποίηση αυτή έχει κόστος $\Theta(m \log m)$, αφού είναι παραλλαγή του αλγορίθμου του Kruskal.

4 Μοναδικότητα Ελάχιστου Συνδετικού Δέντρου

4.1

Παρακάτω δίνεται παράδειγμα γράφου με ίδιο βάρος σε κάποιες ακμές που έχει όμως μοναδικό MST. Επομένως το αντίστροφο δεν ισχύει.



(α′) Αρχικό γράφημα (β′) MST

4.2

Θα το αποδείξουμε με εις άτοπον απαγωγή. Έστω ότι το MST T, δεν είναι μοναδικό, δηλαδή υπάρχει ένα δέντρο T' με ίδιο βάρος ακμών. Αφού είναι διαφορετικά, θα υπάρχει τουλάχιστον μία τομή του γραφήματος, όπου η ακμή e του T και η e' του T' θα είναι διαφορετικού βάρους (χωρίς βλάβη της γενικότητας, με $e < e'$). Προσθετοντας την e στο T' δημιουργείται κύκλος που περιέχει τις e, e' . Το δέντρο $T' \cup \{e\} \backslash \{e'\}$ παραμένει συνδετικό και έχει βάρος $|T'| - e' + e < |T|$. Επομένως μπορούμε να κατασκευάσουμε ένα καλύτερο MST, πράγμα άτοπο, αφού υποθέσαμε ότι το B ήταν MST (άρα ελάχιστο εξ'ορισμού).

Το αντίστροφο δεν ισχύει, όπως φαίνεται και από το παράδειγμα του 1ου ερωτήματος. Η τομή που χαρακτηρίζεται από τις ακμές (Α,Β),(Β,Γ),(Β,Δ) έχει ακμές ίδιου βάρους αλλά το MST είναι μοναδικό.

4.3 Συνθήκη

Η ικανή και αναγκαία συνθήκη για να υπάρχει μοναδικό MST ενός γράφου G έχει 2 συνιστώσες:

- Για κάθε διαμέριση των κόμβων του G σε 2 συνεκτικές συνιστώσες, η ελαχίστου βάρους ακμή με άκρα σε αμφότερα τα υποσύνολα είναι μοναδική.
- Η μεγίστου βάρους ακμή σε κάθε κύκλο του G είναι μοναδική.

Το $1^ο$ μέρος της απόδειξης έγινε προηγουμένως, ότι δηλαδή καμία από τις ακμές που χωρίζουν το γράφημα σε 2 συνεκτικές συνιστώσες, δε γίνεται να αντικατασταθεί από κάποια άλλη.

Το αντίστροφο (δηλαδή το ότι αν υπάρχει μοναδικό MST, τότε δε γίνεται να αντικατασταθεί καμία από τις ακμές του με άλλη) προκύπτει από τη λειτουργία του αλγορίθμου του Prim: Στην περίπτωση που 2 συνεκτικές συνιστώσες του γραφήματος συνδέονταν με 2 ακμές ίδιου βάρους, ο αλγόριθμος του Prim θα επέλεγε τυχαία κάποια από τις 2, οπότε δε θα είχαμε μοναδικό MST.

4.4 Αλγόριθμος

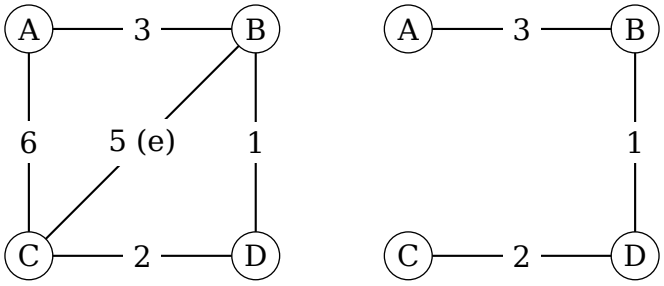
Ένας αλγόριθμος με πολυπλοκότητα $O(|V|^2)$ είναι ο εξής:

Εφαρμόζουμε BFS στο MST και αποθηκεύουμε το βάρος της βαρύτερης τιμής για κάθε μονοπάτι (u_1, u_2) σε ένα πίνακα μεγέθους V^2.
Για κάθε τομή του G που τον χωρίζει σε 2 συνεκτικές συνιστώσες, συγκρίνουμε τις τιμές των ακμών που τη διασχίζουν με την τιμή που βρίσκεται στον πίνακα.

Αν προκύψει ισότητα, τότε το MST δεν είναι μοναδικό, ειδάλλως, αν ολοκληρωθεί η διαδικασία τότε το MST του γραφήματος είναι μοναδικό.

5 Υπολογισμός Ελάχιστου Συνδετικού Δέντρου με Διαγραφή Ακμών

i) Στο παρακάτω παράδειγμα φαίνεται πως η ακμή e μέγιστου βάρους (5) που βρίσκεται στον κύκλο B->C->D->B δεν περιέχεται στο MST.



(α′) Αρχικό γράφημα (β′) MST

ii) (1) Αρχικά, ο παραπάνω αλγόριθμος είμαστε βέβαιοι ότι θα υπολογίσει ένα spanning tree, γιατί αφαιρεί μόνο τις ακμές που δημιουργούν κύκλους στο γράφημα. Το τελικό αποτέλεσμα θα είναι ένα δέντρο (αφού δεν περιέχει κύκλους) το οποίο θα είναι και συνδετικό, αφού η διαδικασία διαγραφής ακμών δεν επιτρέπει διαγραφή ακμών που θα καταστήσουν το γράφημα μη-συνεκτικό.

(2) Επιπλέον το συνδετικό δέντρο T που προκύπτει θα είναι και ελάχιστο.
Έστω ότι δεν ήταν ελάχιστο, αυτό σημαίνει ότι θα μπορούσε κάποιος να προσθέσει κάποια ακμή e_1 που θα δημιουργούσε κύκλο και αφαιρώντας μία άλλη, e_2 μεγαλύτερου κόστους θα κατέληγε σε MST.
Αυτό είναι άτοπο, αφού ο αλγόριθμος που περιγράφηκε θα επέλεγε για διαγραφή την e_2, όταν έφτανε στον κύκλο που περιείχε τις e_1 και e_2.

iii) Μία αποδοτική υλοποίηση θα ήταν ο αλγόριθμος του Kruskal με φθίνουσα ταξινόμηση στο βάρος των ακμών. Ακολουθώντας τη λογική του αλγορίθμου που περιγράφηκε προκύπτει η παρακάτω μορφή του αλγορίθμου:

Algorithm 2 Άσκηση 5

```
1: procedure Deletion_MSP(G(V,e,w))
2:     Sort edges by decending weight
3:     T ← G
4:     for each e_i ∈ G do
5:         if {e_i} belongs in cycle then
6:             T ← T − {e_i}
```

Η υλοποίηση του αλγορίθμου του Kruskal με χρήση BFS για τον έλεγχο κύκλων απαιτεί χρόνο $O(m(m+n)) = O(m^2)$.