Αλγόριθμοι και Πολυπλοκότητα 7ου εξαμήνου

3η Σειρά Γραπτών Ασκήσεων

Όνομα: Κουτσογιαννακόπουλος Ιωάννης
Α.Μ.: 03109061

Άσκηση 1

Μπορούμε να βρούμε τις γέφυρες και τα σημεία κοπής σε γραμμικό χρόνο αν χρησιμοποιήσουμε τον αλγόριθμο DFS. Το τελικό κόστος θα είναι $O(V + E)$ καθώς χρησιμοποιούμε DFS με κάποιους έξτρα πίνακες.

Για την εύρεση των σημείων κοπής, θα εκτελέσουμε DFS και θα δημιουργήσουμε ένα δέντρο Τ. Μια κορυφή *u* είναι σημείο κοπής, αν ισχύει μία από τις παρακαω δύο συνθήκες:

1. Η *u* είναι ρίζα του Τ και έχει τουλάχιστον δύο παιδιά.
   (Αν αφαιρέσουμε τη *u* το γράφημα θα σταματήσει να είναι συνεκτικό, καθώς δεν θα υπάρχει κορυφή που να ενώνει τα υποδέντρα-απόδειξη με άτοπο)
2. Η *u* δεν είναι ρίζα και κάποιο υποδέντρο με ρίζα παιδί της *u* δεν έχει πίσω ακμή προς κάποια κορυφή-πρόγονο του *u*.
   (Αν αφαιρέσουμε τη *u* τότε καποιο από τα παιδιά της, δεν θα μπορεί να συνδεθεί με το υπόλοιπο δέντρο, επομένως θα σταματήσει να είναι συνεκτικό-απόδειξη με άτοπο)

Για τον έλεγχο της πρώτης συνθήκης καθώς εκτελούμε το DFS έχουμε έναν πίνακα parent όπου αποθηκεύουμε στο *parent[u]* τον πατέρα της εκάστοτε κορυφής. Στην περίπτωση που ο κόμβος *u* είναι ρίζα τότε αποθηκεύουμε την τιμή NULL και αν έχουμε περισσότερα από 2 παιδιά ξέρουμε πως είναι σημείο κοπής.

Για την δεύτερη συνθήκη διατηρούμε δύο πίνακες. Στον πρώτο, έστω descendant, αποθηκεύουμε την χρονική στιγμή που επισκεφτήκαμε τον κόμβο *u* από έναν κόμβο v, που είναι παιδί του u. Στο δεύτερο αποθηκεύουμε το $min\{descendant[u], descendant[w]\}$, όπου w πρόγονος του *u* και υπάρχει πίσω δρόμος από κάποιο απόγονο του *u* στο w.

Για να βρούμε τις γέφυρες θα ακολουθήσουμε παρόμοια πρακτική με την παραπάνω. Μία ακμή είναι γέφυρα αν και μόνο αν δεν υπάρχει άλλη διαδρομή που να ενώνει τη *u* ή ένα πρόγονο της με κάποιο υποδέντρο που έχει ρίζα κάποιο παιδί της, v. Χρησιμοποιούμε και πάλι την τιμή *low[v],* που δείχνει την μικρότερη χρονική στιγμή που φτάνουμε στον κόμβο από κάποιο υποδέντρο με ρίζα το v. Η συνθήκη για να είναι γέφυρα είναι

$low[v] > descendant[u]$

Έστω γέφυρα *u-v.* Τότε υπάρχει πίσω ακμή στο υποδέντρο διαφορετική της *u-v,* που συνδέει το *v* με το w(πρόγονος της u). Άτοπο. Γιατί θα υπάρχει κύκλος

και επομένως αν αφαιρεθεί η ακμή $u - v$ τότε το γράφημα συνεχίζει να είναι συνεκτικό και επομένως $u - v$ δεν μπορεί να είναι γέφυρα. Επομένως ισχύει ο αλγόριθμος που περιγράψαμε παραπάνω.

Άσκηση 2

α) Στην περίπτωση που το G είναι ένα Κατευθυνόμενο Ακυκλικό Γράφημα (DAG) ξέρουμε πως υπάρχει ένας τουλάχιστον κόμβος που δεν έχει εισαρχόμενες ακμές. Επομένως, ξεκινώντας από αυτές τις κορυφές με DFS μπορούμε να φτάσουμε σε όλους τους κόμβους σε γραμμικό χρόνο ίσο με τον αριθμό των ακμών και των κόμβων (η αναζήτηση κατά DFS δεν μπορεί να εξασφαλίσει την προσπέλαση κάθε κόμβου ακριβώς μία φορά ή την προσπέλαση κάθε ακμής. Γνωρίζουμε πως ο κόμβος έχει υπολογιστεί με επιτυχία όταν όλες οι ακμές που εξέρχονται από αυτόν έχουν ήδη προσπελαστεί. Επομένως αν φτάσουμε σε ένα τέτοιο κόμβο δεν χρειάζεται να επαναλάβουμε το DFS για τον κόμβο αυτό, αλλά μπορούμε απ'ευθείας να χρησιμοποιήσουμε την τιμή του σε σταθερό χρόνο). Τελικά θα έχουμε κόστος $O(V + E)$.

Για κάθε κόμβο u που προσπελαύνεται, αν θεωρήσουμε v_i τα παιδιά του (δηλαδή τους κόμβους που είναι άμεσα προσβάσιμοι από τον u), τότε ως κόστος ορίζεται το $min\{p(u),c(v_i)yi\}$. Επομένως κάθε κόμβος χωρίς εξαρχόμενες τιμές θα έχει την κόστος $c(u) = p(u)$. Κάθε άλλος κόμβος θα παίρνει τιμή το min μεταξύ της τρέχουσας τιμής (αρχική $c(u) = p(u)$) και του εκάστοτε παιδιού v_i που ήδη έχει υπολογιστεί.

β) Αν βρούμε τις ισχυρά συνεκτικές συνιστώσες μπορούμε να να συμπτίξουμε τις ακμές κάθε μίας, κρατώντας ως εκπρόσωπο κάθε μίας την πιο φθηνό κόμβο. Αυτό μπορεί να γίνει σε γραμμικό χρόνο. Έτσι δημιουργούμε ένα DAG και στη συνέχεια εφαρμόζουμε τον αλγόριθμο που περογράφηκε στο (α).

Άσκηση 3

Θα δημιουργήσουμε το παρακάτω κατευθυνόμενο γράφημα G. Αρχικά δημιουργούμε ένα πίνακα με δείκτες σε συνδεδεμένες λίστες (μία για κάθε υπολογιστη C_n, αρχικοποιημένες σε null). Διασχίζουμε τις m τριάδες δεδομένων και για κάθε τριάδα δημιουργούμε τους κόμβους (C_i,t) και (Cj, t) τους οποίους ενώνουμε με ακμή διπλής κατεύθυνσης. Προσθετουμε επιπλέον αυτούς τους κόμβους στη λίστα για τα Ci και Cj αντίστοιχα. Αν δεν είναι η πρώτη τριάδα που εμφανίζεται ο Ci συμπεριλαμβάνουμε και μία ακμή που συνδέει τον κόμβο (C_i,t) με τον (C_i, t_k) (με την κατεύθυνση της ακμής από $t \wedge t_k$), όπου t είναι ο χρόνος που αντιστοιχεί στον προηγούμενο κόμβο σχετικό με το C_i. Ακολουθούμε την αντίστοιχη διαδικασία και για το Cj. Επειδή διατηρούνται οι λίστες αυτές για κάθε κόμβο, η δημιουργία των νέων κόμβων και ακμών που χρειάζονται για κάθε τριάδα, γίνεται σε σταθερό χρόνο.

Δεδομένου αυτών των τριάδων, θέλουμε να υπολογίσουμε τη χρονική στιγμή tj κατά την οποία κάθε υπολογιστής Cj μολύνθηκε. Ξεκινώντας από τον Ci, ο

οποίος γνωρίζουμε πως είναι ο πρώτος υπολογιστής που μολύνθηκε τη χρονική στιγμή $t_1 = 0$, εκτε- λούμε BFS στο γράφημα. Με αυτόν τον τρόπο βλέπουμε για κάθε υπολογιστή *Cj* αν και πότε μολύνθηκαν (αν δεν φτάσουμε ποτέ σε κόμβο σχετικό με το *Cj* σημαίνει πως αυτός δεν έχει μολυνθεί).

Ο αλγόριθμος αυτός είναι γραμμικού χρόνου. Αυτό συμβαίνει διότι για κάθε τριάδα προκαλεί προσθήκη σταθερού αριθμού κόμβων και ακμων, καταλήγοντας στη δημιουργία ενώ γραφήματος με *O(m)* κόμβους και ακμές. Η BFS αναζήτηση εκτελείται σε χρόνο γραμμικό ίσο με το μέγεθος του γραφήματος, επομένως εκτελείται σε χρόνο *O(m).* Καταλήγουμε έτσι σε αλγόριθμο με γραμμικό χρόνο εκτέλεσης *O(m*).

Άσκηση 4

α) Έστω T_1, T_2 συνεκτικά δέντρα του γραφήματος G. Έστω επίσης ακμή *e* ανήκει T\\T₂. Τότε δεν υπάρχει ακμή *e' €* T₂\T*ι :* T\\{e} *U {e'}* να είναι συνδετικό δέντρο.

Γνωρίζουμε ότι αν αφαιρέσουμε μια ακμή από ένα συνδετικό δέντρο τότε παίρνουμε ακριβώς δύο δέντρα S_1, S_2 που δεν συνδέονται μεταξύ τους. Επιπλέον σε κάθε συνδετικό δέντρο υπάρχει μία ακμή που διασχίζει την τομή του γράφου. Άρα αν αφαιρέσουμε την ακμή *e,* αφού αυτή δεν υπάρχει στο T_2 τότε το T_2 δεν μπορεί να είναι συνδετικό δέντρο, πράγμα άτοπο από υπόθεση.

Σύμφωνα με την απόδειξη αρκεί να βρούμε την τομή στα δύο δέντρα S_1, S_2 και τις ακμές που την διασχίζουν. Μπορούμε να βρούμε την τομή του T_1 σε γραμμικό χρόνο (πχ κατά DFS) και στη συνέχεια με τον ίδιο τρόπο να αναζητήσουμε στο T_2 την ακμή *e'* που διασχίζει την τομή.

β) Το γράφημα *H* που θα προκύπτει θα είναι πάντα συνεκτικό. Για να το αποδείξουμε αρκεί να αποδείξουμε ότι αν T_1, T_2 δύο συνεκτικά δέντρα του G, με |T₂\T*i*| = |T*1* \T₂| = k, πάντα θα υπάρχει μονοπάτι στο *H* που ενώνει το T_1 με το T_2 με μήκος k.

Αν k=1 η παραπάνω πρόταση είναι αληθής από τον ορισμό του γραφήματος *H*. Για *k > 1* αν |T*ι*\T₂| = k, διαλέγουμε *e' € T₂\T₁.* Το δέντρο T_1 *U'}* περιέχει έναν κύκλο που περιέχει μία ακμή *e / T₂.* Το δέντρο T_1 *U {e} — {e'}* = T₃. Το δέντρο αυτό θα έχει την ιδιότητα |T₃\T₂| = |T₂\T₃| = *k — 1*. Επαναλαμβάνοντας αυτή τη διαδικασία καταλήγουμε στο συμπέρασμα ότι με k βήματα μπορούμε να φτάσουμε από το δέντρο T_1 στο T₂, δηλαδή με το συντομότερο δυνατό μονοπάτι.

Για την υλοποίηση του αλγόριθμου που περιγράφηκε, δηλαδή την εύρεση του συντομότερου μονοπατιού, απαιτείται η εύρεση κάθε φορά της ακμής *e'.* Αυτό υλοποιείται με τη χρήση του αλγόριθμου που περιγράφεται στο α).

γ) Θέλουμε να βρούμε ένα συνδετικό δέντρο με ακριβώς k ακμές στο E_1. Βρίσκουμε το mst, το οποίο θα έχει a ακμές στο E_1 και ένα συνδετικό δέντρο που θα έχει το μέγιστο δυνατό αριθμό ακμών στο E_2, έστω b. Συμπεραίνουμε ότι το ζητούμενο δέντρο θα υπάρχει μόνο αν $a < k < b$. Αν βρισκόμαστε στην περίπτωση κάποιας ισότητας τότε έχουμε ήδη βρει το ζητούμενο δέντρο. Αλλιώς χρησιμοποιούμε τον αλγόριθμο που περιγράφεται στο (β) για να κατασκευάσουμε το μονοπάτι που ενώνει τα δύο δέντρα που έχουμε δημιουργήσει και γνωρίζουμε ότι θα συναντήσουμε το ζητούμενο δέντρο σε ακριβώς $k - a$ βήματα από το mst.

Άσκηση 5

α)

Έστω mst του *G.* και έστω *e = {u,v} £ T : d(u,v) < w(e).* Αν πάρουμε μια τομή του T-e, τότε:

- αν η τομή δεν περιέχει άλλη ακμή, άτοπο και πρέπει να ισχύει w(e)=d(u,v)
- αν η τομή περιέχει κι άλλη ακμή, έστω e^1, δημιουργείται κύκλος, άρα δεν μπορεί να περιέχονται και οι δύο ακμές στο mst. Ωστόσο ο μόνος τρόπος να χρησιμοποιηθεί μία άλλη ακμή *e!* είναι να ισχύει *w(e')* < w(e). Αλλά για να είναι το mst πρέπει να ισχύει *w(e') = w(e).*

Σε κάθε περίπτωση προκύπτει ότι η *e* αποτελεί συντομότερο μονοπατι μεταξύ των κορυφών *u,v.*

β)

Γνωρίζουμε πως για κάθε διαμέριση σε S_1,S_2 η απόσταση μεταξύ τους θα είναι η ελάχιστη, επομένως η ακμή αυτή θα ανήκει στο mst. Άρα αναζητούμε την μέγιστη ακμή του mst. Αυτό μπορεί να γίνει με τον υπολογισμό του mst και αφαίρεση της μέγιστης ακμής. Επομένως μπορούμε να χρησιμοποιήσουμε τον αλγόριθμο του Krnskal και να σταματήσουμε όταν μείνουν ακριβώς δύο δέντρα, που περιλαμβάνουν όλες τις κορυφές. Αυτό λειτουργεί καθώς ο αλγόριθμος του Krnskal προσθέτει κάθε φορά την μικρότερη ακμή που δεν έχει χρησιμοποιηθεί και δεν δημιουργεί κύκλο.