ΤΣΩΛΑΣ ΛΕΩΝΙΔΑΣ
03112422
ΗΜΜΥ ΕΜΠ

Αλγόριθμοι και Πολυπλοκότητα

3η γραπτή σειρά ασκήσεων

Άσκηση 1

α.

Μπορούμε πράγματι να αποφανθούμε για τη ζητούμενη σχέση μεταξύ δύο κόμβων του δέντρου σε γραμμικό O(n) χρόνο,
Συγκεκριμένα, για δύο κόμβους x και y ενός δέντρου T, ισχύει πως το x είναι πρόγονος του y αν και μόνο αν το x συναντάται πριν το y στην προδιατεταγμένη διάσχιση preorder traversal του δέντρου και μετά το y στην μεταδιαταγμένη διάσχιση (*postorder traversal*) .
Η ορθότητα της πρότασης επιβεβαιώνεται από τον ορισμό των δυο τρόπων διάσχισης δέντρου που αναφέρθηκαν.

Υπενθυμίζεται ότι οι δύο αλγόριθμοι λειτουργούν ως εξής:

Προδιαταγμένη διάσχιση (*preorder traversal*)

1. επίσκεψη της ρίζας
2. επίσκεψη του αριστερού υποδένδρου
3. επίσκεψη του δεξιού υποδένδρου

Μεταδιαταγμένη διάσχιση (*postorder traversal*)

1. επίσκεψη του αριστερού υποδένδρου
2. επίσκεψη του δεξιού υποδένδρου
3. επίσκεψη της ρίζας

Οι δύο διασχίσεις είναι δυνατό να πραγματοποιηθούν σε γραμμικό χρόνο έκαστη.

Συνεπώς το άθροισμα των πολυπλοκοτήτων παραμένει γραμμικό και η συνθήκη που ελέγχει εάν το x είναι πρόγονος του y είναι:

```
.order(x) < order(y) <= order(x) + size(x)
```

β.

Έστω G ένα ισχυρά συνδετικό κατευθυνόμενο γράφημα. Τρέχουμε BFS από ένα αρχικό κόμβο s. Ο BFS δημιουργεί ένα ισορροπημένο δέντρο, όπου το επίπεδο του κόμβου v είναι η κατευθυνόμενη απόσταση από τον αρχικό κόμβο s.
Εάν κατά τη διάρκεια εκτέλεσης του BFS δεν παρατηρηθεί κάποια ακμή (u,v) που να βρίσκεται μεταξύ ακμών του ίδιου επιπέδου τότε το G είναι διμερές γράφημα. Εφόσον κάθε κόμβος μπορεί να προσεγγισθεί από τον αρχικό κόμβο s , τότε ο BFS θα εξερευνήσει όλες τις ακμές με αποτέλεσμα όλες οι ακμές να διατρέχονται μεταξύ των δύο block της διαμέρισης.
Διαφορετικά, εάν βρεθεί μία ακμή (u,v) που περιγράφηκε προηγουμένως με το u και επίπεδο του v να είναι περιττός αριθμός, τότε τρέχουμε BFS από το v για να λάβουμε ένα μονοπάτι στον αρχικό κόμβο s. Αν το μονοπάτι είναι άρτιου μήκους, τότε λαμβάνουμε ένα κύκλο περιττού μήκους $s \to u \to v \to s$. Ανν το μονοπάτι είναι περιττού μήκους, λαμβάνουμε περιττό κύκλο $s \to v \to s$.

Όμοια συνεχίζουμε εάν το επίπεδο του u είναι περιττός αριθμός και το v είναι επίσης περιττός.
Ο αλγόριθμος τρέχει δύο φορές, άρα συνολικά τρέχει σε γραμμικό χρόνο, αφού και ο BFS τρέχει σε γραμμικό χρόνο.

<u>Άσκηση 2</u>

α.
Έστω μια κορυφή u.
Τότε όλες οι κορυφές που είναι προσπελάσιμες από τη u είναι :

- η κορυφή u
- οι κορυφές με τις οποίες συνδέεται με ακμή $u \rightarrow \quad v_i \ : \ v_1, \ldots, v_d$
- οι κορυφές που είναι προσπελάσιμες από τις $\quad v_{1,} \ldots, v_d$

Επομένως, ισχύει
$c(u) = min \ \{ \ p_u \ , \ min_{v:(u,v)\in E} \ \{ \ c(v) \ \} \}$

Πρέπει λοιπόν να υπολογίσουμε πρώτα τις τιμές $\quad c(v) \quad , \quad \forall v:(u,v)\in e$ και να κρατήσουμε την μικρότερη τιμή min{ $\quad p_u , \quad min_{v:(u,v)\in E} \ \{ \ c(v) \ \}$ }

Επειδή όμως ο αλγόριθμος είναι DAG θα γίνει χρήση τοπολογικής διάταξης και εξέταση των κορυφών σε αντίστροφη σειρά.

Ο αλγόριθμος πραγματοποιεί :

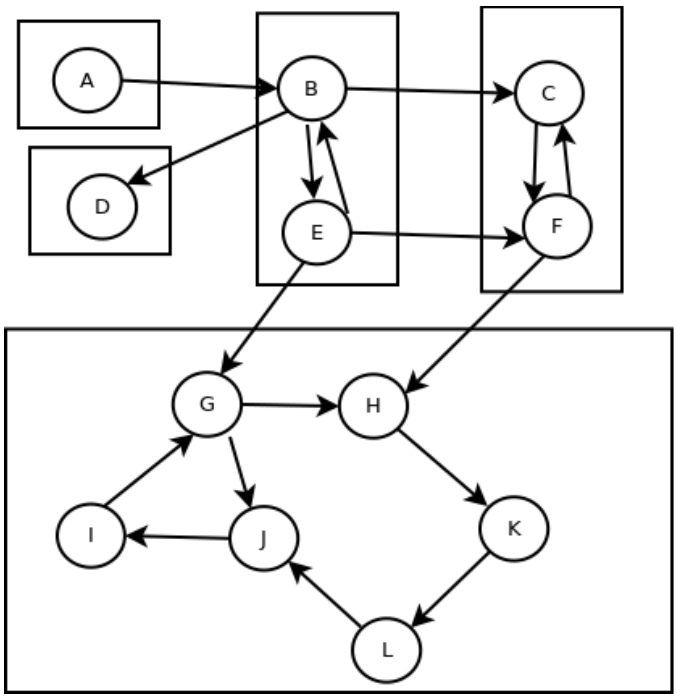
- Τοπολογική διάταξη των κορυφών στο G
- Εξετάζοντας τις κορυφές σε αντίστροφη σειρά από την τοπολογική διάταξη, για κάθε $u \in V$ γίνεται υπολογισμός της τιμής $\quad c(u) = min \ \{ \ p_u \ , \ min_{v:(u,v)\in E} \ \{ \ c(v) \ \} \}$

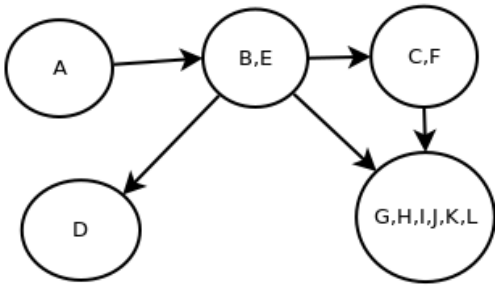
Ο αλγόριθμος είναι γραμμικής πολυπλοκότητας αφού τα δύο βήματα απαιτούν χρόνο O(|V|+|E|).

β.
Ισχυρά συνεκτική συνιστώσα ΙΣΣ ενός κατευθυνόμενου γραφήματος G(V,E) είναι ένα μέγιστο σύνολο κορυφών $\quad U \subseteq V \quad$ ώστε $\quad u, v \in V \quad$ να υπάρχει διαδρομή $\quad u \Rightarrow v \quad$ και $\quad v \Rightarrow u \quad .$

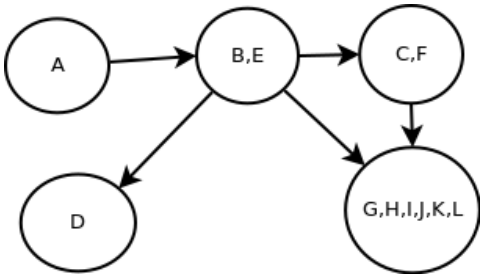
Ομώς ισχύει πως οι ισχυρά συνεκτικές συνιστώσες ενός κατευθυνόμενου γραφήματος υπολογίζονται σε γραμμικό χρόνο.

Έστω το ακόλουθο κατευθυνόμενο γράφημα G:

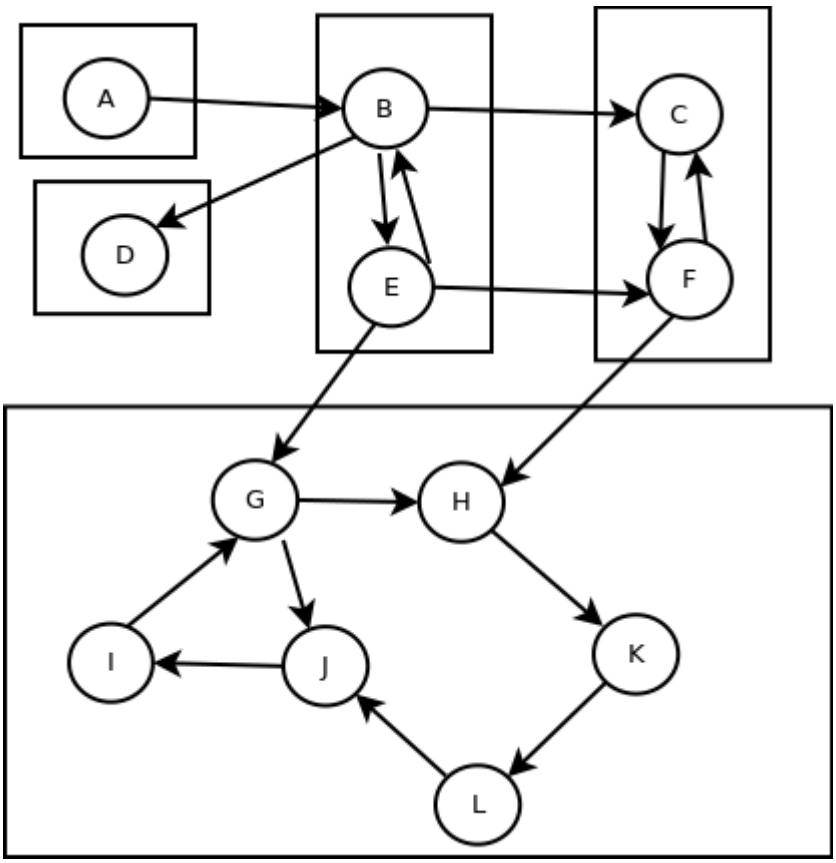




Το γράφημα G’ που προκύπτει αν συρρικνώσουμε κάθε ΙΣΣ σε μία κορυφή είναι DAG.



- Αν βρούμε την τοπολογική διάταξη του μεταγραφήματος αυτού, ουσιαστικά βάζουμε τις ΙΣΣ σε μια φθίνουσα σειρά χρόνων αναχώρησης του DFS.
- Οι {G,H,I,J,K,L} , {D} είναι τερματικές ισχυρά συνεκτικές συνιστώσες.
- Στην τοπολογική διάταξη του μεταγραφήματος, μια τερματική ΙΣΣ είναι η μετακορυφή με το χαμηλότερο χρόνο αναχώρησης, δηλαδή η τελευταία στη διάταξη.



Αν ξεκινήσουμε τον DFS από έναν κόμβο που ανήκει σε μια τερματική ΙΣΣ, τότε θα εξερευνήσουμε όλη την ΙΣΣ αυτή και θα σταματήσουμε.

Ο αλγόριθμος είναι ο ακόλουθος:

- Ξεκινάμε από έναν κόμβο που ανήκει σε μια τερματική ΙΣΣ.
- Βρίσκουμε μεσω DFS όλους τους κόμβους σε αυτή την ΙΣΣ.
- Αφαιρούμε την ΙΣΣ αυτή και επαναλμβάνουμε.

Το ερώτημα που τίθεται είναι πώς βρίσκουμε μια τερματική ΙΣΣ. Στο αντίστροφο γράφημα G^R μια τερματική ΙΣΣ γίνεται μια αρχική ΙΣΣ, δηλαδδή μια ΙΣΣ χωρίς εισερχόμενες ακμές.

Συνεπώς ο αλγόριθμος για ισχυρά συνεκτικές συνιστώσες είναι ο ακόλουθος:

- $DFS(G)$ για τον υπολογισμό των χρόνων αναχώρησης f[u] για κάθε $u \in V$
- υπολογισμός αντίστροφου γραφήματος G^R δηλαδή με ανεστραμμένες τις ακμές
- $DFS(G^R)$ → στον κύριο βρόχο for της DFS_init εξετάζουμε τις κορυφές σε φθίνουσα σειρά των χρόνων f[u] από το πρώτο βήμα
- Οι κορυφές κάθε δέντρου από το τρίτο βήμα αποτελούν μια συνεκτική συνιστώσα.

Πολυπλοκότητα : O(|V|+|E|)

Μένει να δικαιολογηθεί πώς ο αλγόριθμος για ισχυρά συνεκτικές συνιστώσες μας βοηθά στον υπολογισμό του $c(u)$.

- Σε μια ισχυρά συνεκτική συνιστώσα $U \subseteq V$ για κάθε $u, v \in U$ υπάρχει διαδρομή $u \Rightarrow v$ και $v \Rightarrow u$. Άρα $c(u) = c(v), \forall u, v \in U$

- Συνεπώς αρκεί να υπολογίσουμε τη μικρότερη τιμή $p(u)$ κάθε ΙΣΣ και να τρέξουμε τον αλγόριθμο του πρώτου ερωτήματος στο μεταγράφημα των ΙΣΣ του G.

<u>Άσκηση 3</u>

Η λύση βασίζεται στην κατασκευή του εξής κατευθυνόμενου γράφου:

- Διαβάζουμε τις τριάδες σε αύξουσα σειρά χρόνου.
- Για κάθε τριάδα (C_i, C_j, t) δημιουργούμε έναν κόμβο $[C_i/t]$ και έναν $[C_j/t]$
- Προσθέτουμε μία κατευθυνόμενη ακμή $[C_i/t] \rightarrow [C_j/t]$ και μια κατευθυνόμενη ακμή $[C_i/t] \leftarrow [C_j/t]$.
- Ελέγχουμε εάν είναι η πρώτη φορά που συναντάμε στις τριάδες τον υπολογιστή C_i
- Εάν όχι, βρίσκουμε την τελευταία στιγμή $t' \leqslant t$ που συναντάμε τον C_i σε κάποια τριάδα. Έτσι, προσθέτουμε μια κατευθυνόμενη ακμή $[C_i/t'] \rightarrow [C_i/t]$
- Όμοια για τον C_j

Η υλοποίιηση αυτού του βήματος γίνεται με πίνακα n θέσεων. Στη θέση i για τον C_i αποθηκεύουμε την πιο πρόσφατη επικοινωνία που είχε.

Έχοντας κατασκευάσει αυτό το γράφο, το αρχικό πρόβλημα μεταφράζεται στο πρόβλημα εύρεσης όλων των κόμβων που ανήκουν στην ίδια συνεκτική συνιστώσα με τον κόμβο $[C_1/0]$.

Δημιουργούμε έναν ακόμα πίνακα res[1,…,n] θέσεων, όπου στην i-οστή θέση θα αποθηκεύουμε τη χρονική στιγμή που προσβλήθηκε ο C_i .

- Αρχικοποιούμε τις θέσεις του πίνακα res[1,…,n] στο NULL (υπολογιστές που δεν έχουν μολυνθεί).
- Ξεκινώντας από τον κόμβο $[C_1/0]$ εκτελούμε τον BFS στον κατευθυνόμενο γράφο που κατασκευάσαμε.
- Κάθε φορά που συναντάμε ένα νέο κόμβο $[C_i, t]$ και ο C_i δεν είναι μολυσμένος, θέτουμε $res[i]=t$.
- Διαφορετικά θέτουμε $res[i]=min(t, res[i])$.

Βασίζουμε την υλοποίηση του αλγόριθμου στην εξής πρόταση:

Υπάρχει ένα κατευθυνόμενο μονοπάτι από τον κόμβο $[C_1/0]$ στον `[C*/t],` αν και μόνο αν ο C* έχει μολυνθεί ως τη χρονική στιγμή t.

- Από την κατασκευή του γράφου, ακμή υπάρχει μόνο ανάμεσα σε δυο υπολογιστές που επικοινωνούν την ίδια στιγμή t΄ ή ανάμεσα σε δύο κόμβους $[C_1/0] \rightarrow$ [C*/t΄].

Χρονική πολυπλοκότητα $O(m)$.

- Μέγεθος κατευθυνόμενου γράφου |V| = O(m) και |E|=O(m)
- Για κάθε τριάδα προσθέτω το πολύ 2 κόμβους και το πολύ 4 ακμές.
- Κατασκευή γράφου και πίνακα ελέγχου $O(m)$.
- Εκτέλεση BFS σε γραμμικό χρόνο ως προς το γράφο $O(m)$.

Άσκηση 4

α.

Για την επίλυση του προβλήματος στηριζόμαστε σε πρώτη φάση στις εξής ιδιότητες:

- Έστω T_1, T_2 και δύο διαφορετικά συνδετικά δέντρα και $e \in T_2 / T_1 \neq \emptyset$
- $T_1 \cup [e]$ περιέχει κύκλο C. Υπάρχει ακμή $e' \in C, e' \notin T_2$, αλλιώς το T_2 περιέχει το C.
- Η ένωση των T_1 \ {e} και {e΄} είναι άκυκλη με n-1 ακμές → Συνδετικό δέντρο.

Θα βρούμε τώρα το e΄:

Έστω T_1, T_2 και $e = [u, v]$.

- Σε O(|V|) κάνουμε BFS από το u στο v στο δέντρο T_1 και βρίσκουμε μονοπάτι P τους συνδέει.
- Για κάθε $e \in P$ ελέγχουμε σε O(1) αν ανήκει στο T_2 και μόλις βρούμε $e \in P, e \notin T_2$ την αφαιρούμε. Συνολικά O(|V|).

β.

Ιδιότητα: Έστω $T_1, T_2 \in H$ και $d_h(T_1, T_2)$ το μήκος του συντομότερου μεταξύ T_1, T_2 στο H. Τότε

| T_1 \ T_2 | = k αν και μόνο αν $d_H(T_1, T_2) = k$.

Απόδειξη της ιδιότητας:

Θα χρησιμοποιήσουμε επαγωγή στο μήκος του μονοπατιού:

- Επαγωγική βάση : Από ορισμό του H, | T_1 \ T_2 | = 1 αν και μονο αν $d_H(T_1, T_2) = 1$
- Επαγωγική υπόθεση: | T_1 \ T_2 | = k αν και μόνο αν $d_H(T_1, T_2) = k$.
- Επαγωγικό βήμα: Θα δείξουμε ότι | T_1 \ T_2 | = k+1 αν και μόνο αν $d_H(T_1, T_2) = k+1$

Απόδειξη επαγωγικού βήματος:

Έστω $e \in T_1 / T_2$. Υπάρχει $e' \in T_2 / T_1$ τέτοιο ώστε $T_1' = (T_1 / e) \cup e' \in H$. Αλλά | T_1 \ T_2 | = k → (από επαγωγική υπόθεση) $d_H(T_{1,}, T_2) \leq k+1$.

Όμως από επαγωγική υπόθεση προκύπτει $d_H(T_1, T_2) = k$. Άτοπο, λόγω της τελευταίας σχέσης.

Αν τώρα $d_H(T_1, T_2) = k+1$ υπάρχει $T_1' \in H$ τέτοιο ώστε $d_H(T_1, T_1') = 1, d_H(T_1', T_2) = k$
Από επαγωγικό βήμα προκύπτει $[T_1 / T_2] = k-1$ ή $[T_1 / T_2] = k+1$.
Όμως αν $[T_1 / T_2] = k-1 = ¿ d_H(T_1, T_2) = k-1$, άρα άτοπο.

Υπολογίζουμε λοιπόν το σύνολο ακμών T_2 \ T_1 . Για κάθε $e \in T_2$ \ T_1 προσθέτουμε την e στο υπάρχον δέντρο.

Αν T_2 \ $T_1 = k$ τότε σε k βήματα έχουμε μετατρέψει το T_1 σε T_2 . Συνεπώς έχει βρεθεί μονοπάτι μήκους k που είναι δηλαδή το συντομότερο μονοπάτι.

Σε O(|V|) αποθηκεύουμε το T_1 ως rooted δέντρο . Για κάθε ακμή $e \in T_2$ ελέγχουμε σε O(1) αν $e \in T_1$. Άρα O(|V|). Επίσης κάνουμε k ανανεώσεις ακμών σε O(|V|) βήματα.
Συνολικά έχουμε O(k * |V|).

γ.
Ορισμός: Έστω T_1 , T_2 τα δυνδετικά δέντρα με το μέγιστο και ελάχιστο αντίστοιχα αριθμό ακμών από το σύνολο E_1 .

- Αν το T_1 έχει λιγότερες από k ακμές από το σύνολο E_1 τότε δεν υπάρχει συνδετικό δέντρο με ακριβώς k ακμές από το σύνολο E_1 .
- Αν το T_1 έχει περισσότερες από k ακμές από το σύνολο E_1 τότε δεν υπάρχει συνδετικό δέντρο με ακριβώς k ακμές από το σύνολο E_1 .

Ο αλγόριθμος λοιπόν είναι ο εξής:

Έστω A είναι η τομή των συνόλων $T_1 \setminus T_2$ και E_1 .

- Αν $T_1 \cap E_1 < k$ ή $T_2 \cap E_1 > k$ η απάντηση είναι αρνητική.
- Διαφορετικά για κάθε $e \in A$: Προσθέτουμε e στο T_2 αφαιρώντας την κατάλληλη ακμή $e' \notin T_1$.
- Αν $T_2 \cap E_1 = k$ σταματάμε.

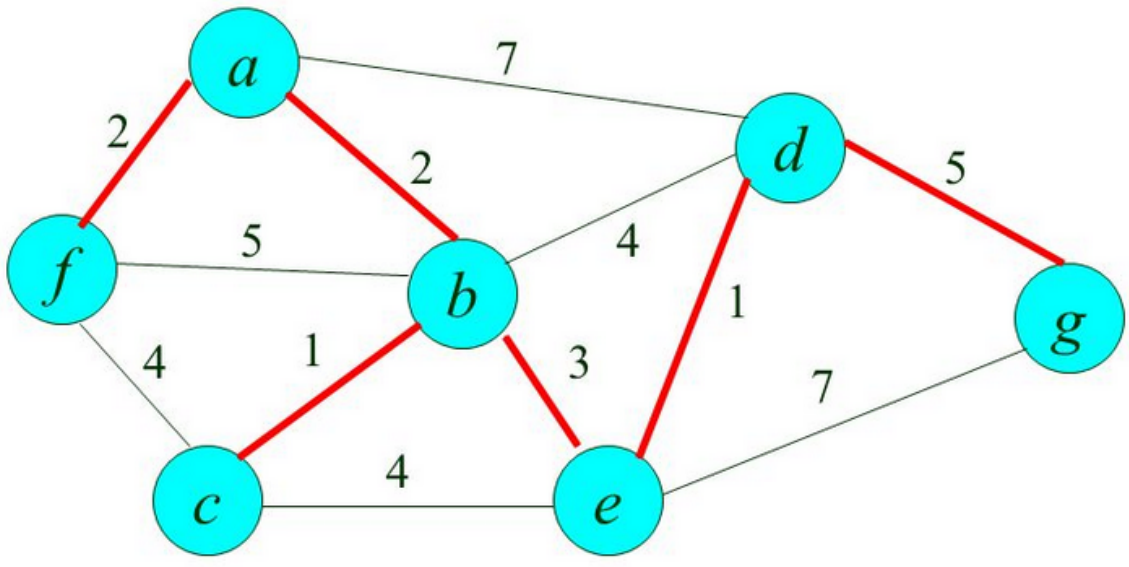
Η ορθότητα του αλγορίθμου αποδεικνύεται ως εξής:
Έστω ότι προσθέτουμε όλες τις ακμές του A . Τότε το T_2 θα περιέχει περισσότερες από k ακμές στο E_1 . Αρχικά, το T_2 περιείχε λιγότερες από k ακμές στο E_1 και εφόσον σε κάθε βήμα οι E_1 ακμές αυξάνουν το πολύ κατά ένα, είναι σίγουρο πως κάποια στιγμή θα έχει ακριβώς k.

Υπολογισμός T_1 . Τοποθετούμε τις ακμές στο E_1 πριν από τις ακμές του E_2 και εφαρμόζουμε τον αλγόριθμο Kruskal. Αντίστοιχα γίνεται ο υπολογισμός του T_2 .

Η πολυπλοκότητα του αλγόριθμου είναι O(|E|) για Kruskal χωρίς ταξινόμηση και O(k*|V|) για την προσθήκη των ακμών. Άρα συνολικά O(|E| + k*|V|).

Άσκηση 5

α.
Ακολουθεί κατευθυνόμενο γράφημα G στο οποίο φαίνεται το ελάχιστο συνδετικό δέντρο με κόκκινη διαγράμμιση και το οποίο παρουσιάζει δύο ακμές ίσου βάρους (2) , ενώ το MST είναι μοναδικό.



β.

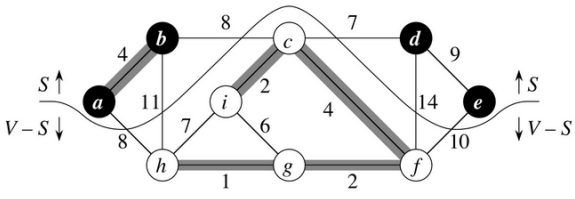
Έστω T_1 και T_2 είναι διαφορετικά ελάχιστα συνδετικά δέντρα.

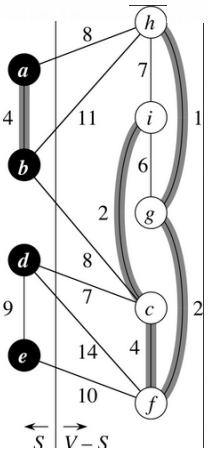
Εξετάζουμε την ακμή που περιγράφεται, δηλαδή την μοναδική ακμή ελαχίστου βάρους. Θεωρούμε, προφανώς χωρίς βλάβη της γενικότητας ότι η ακμή αυτή βρίσκεται στο T_1 και την ονομάζουμε e_1 .

Τότε η ένωση $T_2 \cup \{ e_1 \}$ περιέχει κύκλο. Επομένως υπάρχει μια ακμή , έστω e_2 που δεν είναι στο T_1 . Από την αρχική μας υπόθεση προκύπτει και $w(e_1)<w(e_2)$ και το T , όπου $T=T2 \cup \{e1\} \setminus \{e2\}$ είναι συνδετικό δέντρο.

Το συνολικό δέντρο του T είναι μικρότερο από το συνολικό βάρος του T_2 . Άτοπο αφού υποθέσαμε ότι το T_2 είναι ένα ελάχιστο συνδετικό δέντρο. Συνεπώς υπάρχει μοναδικό συνδετικό δέντρο.

Το αντιπαράδειγμα που ζητείται φαίνεται στα ακόλουθα γραφήματα:





γ.

Έστω γράφημα $G=(V,E)$ και $M=(V,F)$ ένα ελάχιστο συνδετικό δέντρο του G.

Πρόταση: Αν υπάρχει ακμή $e=\{v,w\} \in E \setminus F$ με βάρος $w(e)=m$ τέτοια ώστε αν προσθέσουμε την ακμή e στο M, να λάβουμε κύκλο C και m να είναι η ακμή με το μικρότερο βάρος στο $F \cap C$, τότε μπορούμε να δημιουργήσουμε ένα δεύτερο ελάχιστο συνδετικό δέντρο εναλάσσοντας μία ακμή του $F \cap C$ βάρους m με την ακμή e. Στην περίπτωση αυτή δεν υπάρχει μοναδικότητα.

Το ευθύ – ότι δηλαδή είναι ικανή συνθήκη είναι προφανές και προκύπτει από την περιγραφή. Αρκεί λοιπόν να δειχθεί η αναγκαιότητα της συνθήκης.

Αντίστροφο: Έστω ότι ένα γράφημα $G=(V,E)$ διαθέτει μοναδικό ελάχιστο συνδετικό δέντρο $M=(V,F)$.Τότε δεν πρέπει να ισχύει η συνθήκη που περιγράφεται στην "Πρόταση" , δηλαδή δεν πρέπει να υπάρχει η ακμή που περιγράφεται. Αυτό όμως συνεπάγεται ότι δεν μπορεί να υπάρχει ακμή e βάρους m η οποία να δημιουργεί κύκλο εάν προστεθεί στο M το οποίο λόγω γενικότητας είναι άτοπο (γενικεύοντας καταλήγουμε ότι δεν μπορεί να υπάρχει το ελάχιστο συνδετικό δέντρο που υποθέσαμε στην αρχή).Επομένως η πρόταση αποτελεί ικανή και αναγκαία συνθήκη.

δ.

Ο αλγόριθμος ουσιαστικα βασίζεται στην πρόταση που περιγράφεται στο γ:

Θεωρώ ως δεδομένο τον αλγόριθμο Dijkstra-Jarık-Prim (DJP) που βρίσκει το ελάχιστο συνδετικό δέντρο M με πολυπλοκότητα O(Elog|E|).

Στη συνέχεια, για κάθε ακμή του γραφήματος $E \setminus F$ ελέγχουμε εάν μπορεί να δημιουργηθεί κύκλος από εναλλαγή με κάθε ακμή του γραφήματος M που έχει προφανώς ίσο βάρος με την ακμή που εξετάζουμε σε κάθε επανάληψη. Για να βρούμε τις ακμές που έχουν ίσο βάρος μπορούμε να πραγματοποιήσουμε δυαδική αναζήτηση σε λογαριθμικό χρόνο .

Πολυπλοκότητα:

- Αλγόριθμος Dijkstra-Jarık-Prim (DJP) : O(Elog|E|)
- Έλεγχος πρότασης για E ακμές με πολυπλοκότητα log|E| έκαστη: Elog|E|.

Συνολικό κόστος αλγορίθμου O(2* Elog|E|) = O(Elog|E|) .