Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Εθνικό Μετσόβιο Πολυτεχνείο
Ροή Λ, 7ο εξάμηνο

Αλγόριθμοι και Πολυπλοκότητα

Τρίτη σειρά γραπτών ασκήσεων

Σπουδαστής
Παπασκαρλάτος Αλέξανδρος (Α.Μ.: 03111097)

Ημερομηνία Υποβολής: 31 Δεκεμβρίου 2018

Σε κάθε ερώτημα όπου ζητείται αλγόριθμος, η αντίστοιχη απάντηση θα έχει 4 βασικά μέρη:

Ιδέα του αλγορίθμου
Η ιδέα για τον αλγόριθμο και κάποια επεξήγηση και ανάλυση

Αλγόριθμος
Η υλοποίηση του αλγορίθμου πιο αναλυτικά

Ορθότητα
Απόδειξη ορθότητας για τον αλγόριθμο και ίσως επεξήγηση.

Πολυπλοκότητα
Υπολογισμός πολυπλοκότητας του αλγορίθμου

Ωστόσο, στην πράξη κάποια από τα μέρη μπορεί να “λείπουν”.
Η Ορθότητα θα παραλείπεται όταν εξηγείται επαρκώς στην Ιδέα του αλγορίθμου και τα μέρη Ιδέα του αλγορίθμου και Αλγόριθμος θα συμπυκνώνονται σε ένα όταν είναι πιο βολικό.

Άσκηση 1: "Μακρύτερο Μονοπάτι σε Δέντρο"

Ιδέα του αλγορίθμου

Καταρχάς, βολεύει λίγο περισσότερο για τις διατυπώσεις να ασχοληθούμε με δέντρο με ρίζα. Για να το πετύχουμε αυτό κάνουμε μια προεπεξεργασία στις ακμές (δηλαδή στη λίστα γειτνίασης) γραμμικού χρόνου.
Συγκεκριμένα, χρησιμοποιούμε ένα αλγόριθμο διάσχισης δέντρου εκκινώντας από τυχαία κορυφή.
Κάθε φορά που εξετάζουμε νέα κορυφή, σβήνουμε από τη λίστα γειτνίασης τις ακμές που πηγαίνουν προς κορυφές που είναι ήδη υπό εξέταση ή εξετασμένοι.
Έτσι, φτιάχνουμε σε γραμμικό χρόνο δέντρο με ρίζα (γονείς, παιδιά) από "γενικό" δέντρο.

Κάθε μονοπάτι p που ξεκινά από κορυφή a μπορεί να "ανεβαίνει" (δηλαδή να πηγαίνει προς προγόνους) μέχρις ότου βρει κορυφή-πρόγονο του a, έστω b και στη συνέχεια να "κατεβαίνει" μέχρις ότου βρει κορυφή-απόγονο του b, έστω c.
Δεν μπορεί να κάνει άλλα "σκαμπανεβάσματα".
Σημειώνουμε πως τόσο η a όσο και η c μπορούν να ταυτίζονται με τη b.

Θα εξηγήσουμε τον αλγόριθμο σε 3 διακριτά βήματα:

1. **Για κάθε κορυφή u, βρίσκουμε το βέλτιστο μονοπάτι που εκκινεί από την κορυφή u και πηγαίνει προς τους απογόνους της u.**

Το μονοπάτι αυτό ενδέχεται να είναι μόνη του η κορυφή u. Έστω $p_1(u)$ αυτό το μονοπάτι. Έστω q(u) το άθροισμα όλων των βαρών των κορυφών του μονοπατιού $w(p_1)$.
Αυτό το βήμα θα γίνει αναδρομικά για να πετύχουμε καλή απόδοση (βλ. Αλγόριθμος).

2. **Για κάθε κορυφή u, βρίσκουμε το δεύτερο βέλτιστο μονοπάτι (εσωτερικά διακεκριμένο από το πρώτο) που εκκινεί από την κορυφή u και πηγαίνει προς τους απογόνους της u (αν υπάρχει).**

Έστω $p_2(u)$ αυτό το μονοπάτι. Απαιτούμε το $p_2(u)$ να έχει (ως πρώτη κορυφή μετά τη u), άλλο παιδί της u απ' ό,τι είχε το $p_1(u)$. Δηλαδή θέλουμε τα p_1 , p_2 να είναι εσωτερικά διακεκριμένα.
Κρατάμε το βέλτιστο ανάμεσα α.στο $p_1(u)$ και β.την ένωση $p_1(u) \cup p_2(u)$.
Στην πράξη, η ένωση είναι καλύτερη ανν το δεύτερο μονοπάτι (χωρίς την κορυφή u) έχει θετική τιμή.
Έστω r(u) το άθροισμα όλων των βαρών των κορυφών του μονοπατιού.

3. **Περνάμε όλες τις κορυφές και κρατάμε αυτήν με το μέγιστο r.**

Έστω b αυτή η κορυφή. Κρατάμε το αντίστοιχο μονοπάτι.

Ανακεφαλαίωση των παραπάνω βημάτων:

1. Βρίσκουμε για κάθε κορυφή u το βέλτιστο μονοπάτι που πηγαίνει από την κορυφή u προς τα "κάτω".
2. Βρίσκουμε για κάθε κορυφή u το βέλτιστο μονοπάτι που "στρίβει" στη u. Δηλαδή κάποιο μονοπάτι που έρχεται προς τα πάνω μέσω ενός παιδιού α της u και πηγαίνει προς τα κάτω μέσω ενός παιδιού β της u.
3. Από όλα τα μονοπάτια συνολικά του προηγούμενου βήματος κρατάμε το καλύτερο.

Αλγόριθμος

i. Κατασκευάζω δέντρο T με ρίζα μέσω DFS στο δοθέν δέντρο από τυχαία κορυφή.

ii. Έστω κορυφή u με σύνολο παιδιών V. Είναι $q(u) = \max_{v \in V} \{ w(u), w(u) + q(v) \}$. Σημειώνουμε και την κορυφή x(u) = argmax. Το x είναι το "βέλτιστο" παιδί.
Ειδικά, αν q(u) = w(u), τότε x(u) = u.
Εφαρμόζουμε αυτή τη σχέση στη ρίζα του δέντρου. Επειδή η σχέση είναι αναδρομική, φτάνει μέχρι και όλα τα φύλλα, και όλες οι κορυφές στο T αποκτούν μια τιμή q.

iii. Για κάθε κορυφή u με σύνολο παιδιών V, υπολογίζουμε την τιμή
$r(u) = \max_{v \in V \setminus \{x(u)\}} \{ (q(v) + q(u) , q(u) \}$, εξετάζοντας όλα τα παιδιά εκτός από το x.
(Αν η νέα προσθήκη έχει αρνητική τιμή, τότε δεν αξίζει να προσθεσουμε το σκέλος οπότε r(u) = q(u))
Σημειώνουμε και την κορυφή y(u) = argmax. Το y είναι το δεύτερο "βέλτιστο" παιδί.
Ειδικά, αν r(u) = q(u), τότε y(u) = u.

iv. Βρίσκουμε την κορυφή b με το μέγιστο r(b).

v. Από την κορυφή b πηγαίνουμε στην κορυφή v = x(b), και όμοια από την επόμενη κορυφή v, πάμε στη x(v), κοκ. Συνεχίζουμε μέχρις ότου x(v) = v. Αν μη τι άλλο, συμβαίνει σίγουρα αν φτάσουμε σε φύλλο (αλλά ενδεχομένως νωρίτερα). Έστω a η τελική κορυφή (οπότε x(a) = a). Το (a,...,b) είναι το πρώτο σκέλος του μονοπατιού μας.

vi. Από την κορυφή b πηγαίνουμε στην κορυφή v = y(b). Για κάθε επόμενη κορυφή v πηγαίνουμε στην x(v) (όχι στη y(v)) μέχρις ότου x(v) = v.
Έστω c η τελική κορυφή (οπότε x(c) = c). Το (b,...,c) είναι το δεύτερο σκέλος του μονοπατιού μας.
Το βέλτιστο μονοπάτι είναι το p = (a...b...c) με τιμή w(p) = r(b).

Ορθότητα

Ο αλγόριθμος έχει επεξηγηθεί και στο σκέλος Ιδέα του αλγορίθμου.
Εδώ θα ασχοληθούμε κυρίως να αποδείξουμε τις αναδρομικές σχέσεις που χρησιμοποιούμε.

Με το βήμα (ii) βρίσκουμε για κάθε κορυφή u το βέλτιστο μονοπάτι από τη u προς τα κάτω.
Κρατάμε την τιμή αυτού καθώς και την ακριβώς επόμενη κορυφή από τη u στο μονοπάτι.
Θυμίζουμε πως από τη ρίζα δέντρου, μπορούμε να προσπελάσουμε όλες τις κορυφές.
Η αναδρομική $q(u) = \max_{v \in V} \{ w(u), w(u) + q(v) \}$ είναι έγκυρη.
Πράγματι θα το αποδείξουμε με επαγωγή στο ύψος της κορυφής.

Επαγωγική Βάση

Το u είναι φύλλο. Τότε, $q(u) = \max_{v \in V} \{ w(u), w(u) + q(v) \} = w(u)$ καθώς δεν έχει παιδιά v.
Ισχύει τετριμμένα πως το βέλτιστο μονοπάτι από το u προς τα κάτω, όντας απλά το u, έχει τιμή q(u) = w(u).

Επαγωγική Υπόθεση

Έστω η $q(u) = \max_{v \in V} \{ w(u), w(u) + q(v) \}$ ισχύει για όλες τις κορυφές u με ύψος $h \leq k$.

Επαγωγικό βήμα

Έστω κορυφή u με ύψος h = k+1. Τότε το βέλτιστο μονοπάτι προς τα κάτω που εκκινεί από τη u είναι:

α.είτε μόνη της η u, ή

β.ένα μονοπάτι p που έχει ως επόμενη κορυφή από τη u κάποιο παιδί της, $v \in V$ με V το σύνολο των παιδιών της u.

Στην περίπτωση α, είναι προφανώς q(u) = w(u).

Στην περίπτωση β, το μονοπάτι p έχει τιμή w(p) = w(u) + w(p \ {u}). Όμως p \ {u} είναι το αντίστοιχο μονοπάτι προς τα κάτω που εκκινεί από το παιδί v. Το βέλτιστο μονοπάτι προς τα κάτω από τη v είναι γνωστό από την Ε.Υ. (η v έχει ύψος h ≤ k) και έχει τιμή q(v).

Προφανώς, θέλουμε να επιλέξουμε το παιδί με το καλύτερο q(v).

Συνολικά, και για τις δύο περιπτώσεις έχουμε $q(u) = \max_{v \in V} \{ w(u), w(u) + q(v) \}$

---*Τέλος επαγωγής*---

Όμοια αποδεικνύουμε και τη δεύτερη αναδρομική $\mathbf{r(u) = \max_{v \in V \setminus \{x(u)\}} \{ (q(v) + q(u) , q(u) \}}$ όπου ψάχνουμε ανεξάρτητο σκέλος μονοπατιού.

Όπως είπαμε και στο Ιδέα του αλγορίθμου, **κάθε μονοπάτι δέντρου, μπορεί να ξεκινάει από κορυφή a με μια ανοδική πορεία μέχρι μια κορυφή b και έπειτα να έχει μια καθοδική πορεία μέχρι κορυφή c.**

Δεν είναι δυνατόν να προηγείται καθοδική πορεία μέχρι την a ή να ακολουθεί ανοδική πορεία μετά τη c.

Πολυπλοκότητα

Η πολυπλοκότητα είναι γραμμική.

Πράγματι, στο βήμα (ι) έχουμε ένα DFS.

Στο (ιι), η σχέση εξετάζει αναδρομικά μία φορά την κάθε κορυφή. Η σειρά εξέτασης των κορυφών θυμίζει έντονα διάσχιση δέντρου άλα DFS.

Στο (ιιι), εξετάζουμε τα παιδιά κάθε κορυφής u.

Στο (ιν) εξετάζουμε μία φορά κάθε κορυφή u.

Στα (ν) και (νι) εξετάζουμε εν τέλει όλες τις κορυφές του βέλτιστου μονοπατιού (οι οποίες είναι προφανώς το πολύ n σε πλήθος).

Θυμίζουμε πως σε δέντρο, είναι m = n-1, οπότε m = O(n).

Επομένως, συνολικά έχουμε πολυπλοκότητα **O(n)**.

Άσκηση 2: "Μια Συνάρτηση Κόστους σε Κατευθυνόμενα Γραφήματα"

α

Ιδέα του αλγορίθμου

Είναι $\mathbf{c(u) = \min_{(u,v) \in E} \{ p(u), c(v) \}}$

Μία κορυφή u μπορεί να προσπελάσει μόνο κορυφές που είναι μετά από αυτόν στην τοπολογική διάταξη του DAG.

Συνεπώς, για να υλοποιήσουμε ικανοποιητικά την αναδρομική, αρκεί να ξεκινήσουμε από την τελευταία κορυφή (σε τοπολογική διάταξη) του DAG, και να πηγαίνουμε με τη σειρά προς την πρώτη. Τότε, όταν βρισκόμαστε σε κορυφή u, όλες οι κορυφές v που είναι προσπελάσιμες από τη u θα έχουν ήδη γνωστή τιμή.

Ο τρόπος να βρούμε έγκυρη τοπολογική διάταξη είναι γνωστός από τις συμπληρωματικές σημειώσεις του μαθήματος.
Κάνουμε DFS και όταν ολοκληρώνεται η εξέταση μιας κορυφής τότε αυτή μπαίνει από το τέλος προς την αρχή στην τοπολογική διάταξη.

Αλγόριθμος

ι. Βρίσκουμε τοπολογική διάταξη του G μέσω DFS.

ιι. Με σειρά από την τελευταία κορυφή προς την πρώτη, για κάθε κορυφή u, εφαρμόζουμε τη σχέση $c(u) = \min_{(u,v) \in E} \{ p(u), c(v) \}$.

Ορθότητα

Εφόσον ο αλγόριθμος για εύρεση τοπολογικής διάταξης ενός DAG θεωρείται γνωστός (και σχετικά απλός), αρκεί να αποδείξουμε την εγκυρότητα της αναδρομικής.

Θα αποδείξουμε πως ισχύει η σχέση $\mathbf{c(u) = \min_{(u,v) \in E} \{ p(u), c(v) \}}$ με επαγωγή στη θέση της κορυφής u στην τοπολογική διάταξη (από το τέλος προς την αρχή).

Επαγωγική Βάση

Έστω u η τελευταία κορυφή στο DAG, τότε $c(u) = \min_{(u,v) \in E} \{ p(u), c(v) \} = p(u)$.
Ισχύει τετριμμένα καθώς η u δεν έχει εξωτερικές ακμές.

Επαγωγική Υπόθεση

Έστω πως η σχέση $c(u) = \min_{(u,v) \in E} \{ p(u), c(v) \}$ ισχύει για κάθε κορυφή u με θέση $\theta(u) \geq k$ στην τοπολογική διάταξη, δηλαδή η κορυφή u μπορεί να προσπελάσει ελαχιστοτική κορυφή w με $p(w) = c(u)$.

Επαγωγικό Βήμα

Θδο πως η σχέση ισχύει για κάθε κορυφή u με θέση $\theta(u)<k$ στην τοπολογική διάταξη.
Η u μπορεί να προσπελάσει τον εαυτό της, καθώς και όσες κορυφές μπορούν να προσπελάσουν οι κορυφές v προς τις οποίες η u έχει εξωτερική ακμή.
Όλες οι κορυφές v έχουν $\theta(v) \geq k$, οπότε ισχύει η Ε.Υ.
Συνεπώς, συνολικά για τη u ισχύει η σχέση $c(u) = \min_{(u,v) \in E} \{ p(u), c(v) \}$.

Πολυπλοκότητα

Για να βρούμε έγκυρη τοπολογική διάταξη αρκεί χρόνος O(|V| + |E|) (πολυπλοκότητα DFS).
Για μια κορυφή u, για να βρούμε το κόστος c(u) (με γνωστά όλα τα κόστη c(v) των κορυφών v που είναι προσπελάσιμες από τη u) χρειαζόμαστε χρόνο $d^+(u)+1$.
Συνεπώς, για να βρούμε τα κόστη όλων των κορυφών χρειαζόμαστε χρόνο O(|V| + |E|).
Τελικά, συνολικά χρειαζόμαστε χρόνο **O(n+m)**.

β

Ιδέα του αλγορίθμου

Θα αναγάγουμε το πρόβλημα στην περίπτωση του DAG.

Συγκεκριμένα, θα βρούμε στο G "πλήρεις" ισχυρά συνεκτικές συνιστώσες.
Ως πλήρεις εννοούμε πως μία κορυφή u ανήκει στη συνιστώσα A ανν όλες οι κορυφές της A προσπελαύνουν τη u και η u προσπελαύνει όλες τις κορυφές της A.
Σημείωση: Ίσως το "πλήρεις" να είναι πλεονασμός. Απλά, θέλω οι συνιστώσες να είναι μεγιστοτικές, δηλαδή να περιέχουν το μέγιστο αριθμό έγκυρων κορυφών. Δεν είμαι βέβαιος αν αυτό θεωρείται δεδομένο όταν μιλάω απλά για ισχυρά συνεκτικές συνιστώσες.

Θα θεωρήσουμε κάθε τέτοια συνιστώσα ως κορυφή ενός γραφήματος H.
Δύο κορυφές στο H θα συνδέονται με κατευθυνόμενη ακμή ανν οι αντίστοιχες συνιστώσες στο G συνδέονται με κατευθυνόμενη ακμή.
Παρατηρούμε πως το H είναι DAG.
Αν υπήρχε κύκλος, πχ ανάμεσα σε κορυφές u, v ϵ V(H), τότε όλες οι κορυφές του G που ανήκουν στις αντίστοιχες συνιστώσες θα αποτελούσαν μέρος της ίδιας ισχυρά συνεκτικής συνιστώσας (καθώς οι συνιστώσες είναι "πλήρεις").

Κάθε κορυφή του H θα έχει τιμή p τη μικρότερη τιμή p από τις κορυφές που ανήκουν στην αντίστοιχη συνιστώσα στο G.

Δουλεύουμε στο H όπως στο (α).
Για κάθε κορυφή u στο H με c(u) = k, θέτουμε όλα τα κόστη c(w) των κορυφών της αντίστοιχης συνιστώσας στο G στο k.

Προκειμένου να βρούμε πλήρεις ισχυρά συνεκτικές συνιστώσες, ακολουθούμε την εξής διαδικασία:

i. Κάνουμε DFS στο γράφημα G, σημειώνοντας τους χρόνους αναχώρησης για κάθε κορυφή. Έστω σ η σειρά των κορυφών από το μεγαλύτερο χρόνο αναχώρησης προς το μικρότερο.

ii. Θεωρούμε το αντίστροφο γράφημα G^T, δηλαδή γράφημα όμοιο με το G αλλά με τις ακμές ανεστραμμένες (ως προς την κατεύθυνση).

iii. Κάνουμε DFS στο G^T εκκινώντας κάθε φορά από κορυφή u, σύμφωνα με την προαναφερθείσα σειρά σ. Όσες κορυφές προσπελαύνονται από την εκάστοτε u στον DFS αποτελούν μία συνεκτική συνιστώσα.

Αλγόριθμος

Ανακεφαλαιώνοντας:

ι. Βρίσκουμε τις (πλήρεις) ισχυρά συνεκτικές συνιστώσες του G.

ιι. Για κάθε τέτοια συνιστώσα U, βρίσκουμε μια τιμή p(U), ως $\min_{u \in U} \{ p(u) \}$.

ιιι. Θεωρούμε το DAG H με κορυφές τις συνιστώσες του G. Δύο κορυφές U, V του H συνδέονται με ακμή ανν υπάρχουν κορυφές u, v στο G με $u \in U$ και $v \in V$ και ακμή (u,v).

ιν. Εφαρμόζουμε τον αλγόριθμο του (α) στο H. Για κάθε κορυφή U του H, βρίσκουμε ένα κόστος c(U).

ν. Στο G για κάθε συνιστώσα U, για κάθε $u \in U$, θέτουμε c(u) = c(U).

Ορθότητα

Βήμα (ιι): κάθε κορυφή $u \in U$ μπορεί να προσπελάσει κάθε άλλη κορυφή της ίδιας συνιστώσας, οπότε μας ενδιαφέρει μόνο η τιμή της βέλτιστης.

Βήμα (ιιι): έχουμε εξηγήσει στο Ιδέα του αλγορίθμου γιατί το H είναι DAG.

Βήμα (ιν): έχει αποδειχτεί στο (α) ερώτημα.

Βήμα (ν): οι κορυφές της ίδιας συνιστώσας μπορούν να προσπελάσουν ακριβώς τις ίδιες κορυφές.

Όσον αφορά τον αλγόριθμο για την εύρεση ισχυρά συνεκτικών συνιστωσών:

Έστω δύο συνιστώσες A, B με $A \rightarrow B$ στο H.

Τότε, με την εφαρμογή του DFS στο G, σίγουρα κάποια κορυφή του A θα έχει χρόνο αναχώρησης μετά από όλες τις κορυφές του B. Έστω $a \in A$ αυτή η κορυφή.

Αυτό συμβαίνει γιατί από την α φτάνουμε σε κάποια κορυφή β του B και από τη β φτάνουμε σε όλες τις υπόλοιπες κορυφές του B. Οπότε, για να "γυρίσουμε πίσω" στην α στον DFS πρέπει πρώτα να ολοκληρώσουμε την εξέταση όλων των κορυφών του B.

Στο G^T οι ισχυρά συνεκτικές συνιστώσες παραμένουν ισχυρά συνεκτικές συνιστώσες καθώς δεν επηρεάζονται από την αντιστροφή των ακμών.

Πάλι μια κορυφή u φτάνει σε κορυφή v και αντίστροφα αν ανήκουν στην ίδια συνιστώσα, απλά αν στο G είχαμε το μονοπάτι (v,u), στο G^T έχουμε το (u,v) και αντίστροφα.

Ωστόσο, όσον αφορά τις εξωτερικές ακμές των συνιστωσών, αν στο G έχουμε $A \rightarrow B$, στο G^T έχουμε $A \leftarrow B$.

Συνεπώς, στο DFS που εφαρμόζουμε G^T θα διαλέξουμε την α πριν διαλέξουμε οποιαδήποτε κορυφή του B (αφού είχε μεγαλύτερο χρόνο αναχώρησης).

Η α θα προσπελάσει όλες τις κορυφές του A, αλλά δε θα μπορεί να φτάσει σε καμία κορυφή του B.

Πολυπλοκότητα

Κυριολεκτικά σε κάθε βήμα χρειαζόμαστε γραμμικό χρόνο O(n+m).

Επομένως, συνολικά χρειαζόμαστε **O(n+m)**.

Άσκηση 3: "Κλέφτες και Αστυνόμοι"

Αλγόριθμος

Πρόκειται για παίγνιο δύο αντιπάλων οι οποίοι παίζουν βέλτιστα. Θα ακολουθήσουμε τη **λογική του min-max**.

Κατασκευάζουμε κατευθυνόμενο γράφημα Η με κορυφές τις δυνατές καταστάσεις του παιχνιδιού.

Κάθε κατάσταση έχει 3 μεταβλητές:

1. Θέση του Κ (n ενδεχόμενες θέσεις)
2. Θέση του Α (n-1 ενδεχόμενες θέσεις)
3. Επόμενος παίκτης (2 επιλογές, με εξαίρεση τελικές καταστάσεις)

Συνεπώς, έχουμε ~$2n(n-1) = O(n^2)$ κορυφές στο Η.

Συνδέουμε δύο κορυφές με κατευθυνόμενη ακμή ανν μπορούμε με μία έγκυρη κίνηση να μεταβούμε από τη μία κατάσταση στην άλλη.

Έχουμε O(n m) τέτοιες ακμές καθώς στη μετάβαση του παίκτη Χ από κορυφή u στη v στο αρχικό G, αντιστοιχούν n ζεύγη κορυφών στο Η (αφού ο αντίπαλος έχει n ενδεχόμενες θέσεις) και, άρα, n ακμές από κάθε κατάσταση που αντιστοιχεί στο u (και σειρά του παίκτη Χ) σε κατάσταση που αντιστοιχεί στην κορυφή v (και σειρά του αντιπάλου του Χ).

Γνωρίζουμε πως κάθε παίκτης θέλει να κερδίσει κι αν αυτό δεν είναι εφικτό, τουλάχιστον να φέρει ισοπαλία. Αν ούτε αυτό είναι εφικτό, ο παίκτης χάνει.

Ακολουθούμε την εξής διαδικασία:

ι. Στο Η εκκινούμε από κορυφή-τελική κατάσταση u όπου κερδίζει ο Κ, δηλαδή κατάσταση όπου ο Κ βρίσκεται στο καταφύγιο.
Θα δημιουργήσουμε "υποχρεωτικά" μονοπάτια που θα καταλήγουν στο u. Προφανώς, ο Κ θέλει να βρεθεί σε τέτοιο μονοπάτι, ενώ ο Α θέλει να το αποφύγει.

ιι. Λαμβάνουμε όλες τις κορυφές οι οποίες έχουν εξωτερική ακμή προς την u, έστω το σύνολο V.
Για όλες τις κορυφές $v \in V$, παίζει ο Κ. Ο Κ θέλει να φτάσει στο u, οπότε για όλες τις $v \in V$, διαγράφουμε όλες τις εξωτερικές ακμές, εκτός από την ακμή που πηγαίνει στη u.

ιιι. Έστω ένας $v \in V$. Λαμβάνουμε όλες τις κορυφές $w \in W$, οι οποίες έχουν εξωτερική ακμή προς τη v.
Για όλες τις κορυφές $w \in W$ παίζει ο Α. Ο Α δε θέλει να φτάσει στο u, οπότε για όλες τις $w \in W$, διαγράφουμε την εξωτερική ακμή που πηγαίνει στη v. Δεν επιτρέπεται να διαγράψουμε την ακμή ανν είναι η μοναδική εξωτερική ακμή της w.
Επαναλαμβάνουμε το βήμα για όλες τις $v \in V$.

ιν. Κάθε κορυφή $w \in W$ που συνδέεται ακόμα με κορυφή $v \in V$ καταλήγει σε νίκη του Κ.
"Ονομάζουμε" κάθε τέτοια κορυφή $u \in U$ και επαναλαμβάνουμε την όλη διαδικασία.
Η διαδικασία ολοκληρώνεται όταν φτάσουμε σε αδιέξοδο, δηλαδή όταν φτάσουμε σε κορυφή που δεν έχει (μη διαγραμμένες) εσωτερικές ακμές.

Επαναλαμβάνουμε όμοια τη διαδικασία για κάθε κορυφή-τελική κατάσταση που κερδίζει ο Κ.

Επαναλαμβάνουμε αντίστοιχα τη διαδικασία για κάθε κορυφή-τελική κατάσταση που κερδίζει ο Α, δηλαδή κάθε κατάσταση όπου ο Κ και ο Α βρίσκονται στην ίδια θέση.

Με αυτόν το τρόπο καταλήγουμε σε συνεκτικές συνιστώσες τριών τύπων:

1. Τύπου Α: κερδίζει ο Α.
2. Τύπου Κ: κερδίζει ο Κ.
3. Τύπου Ι : όλες οι υπόλοιπες.

Αν το παιχνίδι εκκινήσει από τύπου Α, κερδίζει ο Α, από τύπου Κ, κερδίζει ο Κ, από τύπου Ι έχουμε ισοπαλία (καθώς κάθε παίκτης θα προσπαθεί να μη χάσει και η παρτίδα θα είναι "εγκλωβισμένη" σε πεπερασμένο αριθμό καταστάσεων, οπότε κάποια στιγμή θα βρεθούμε για δεύτερη φορά στην ίδια κατάσταση).

<u>Ορθότητα</u>

Ο αλγόριθμος βασίζεται στη λογική του min-max.
Πρόκειται για παίγνιο δύο αντιπάλων οι οποίοι παίζουν βέλτιστα (και ανταγωνιστικά).

Εκκινώντας από τελικές καταστάσεις και δημιουργώντας μονοπάτια προς τις καταστάσεις αυτές, ο παίκτης που κερδίζει θέλει να βρεθεί σε τέτοια κατάσταση, ενώ ο αντίπαλος τις αποφεύγει (εκτός αν είναι απολύτως αναγκασμένος).

Αν μια κατάσταση δεν ανήκει σε κανένα υποχρεωτικό μονοπάτι τότε το παιχνίδι δε θα φτάσει ποτέ σε νίκη κάποιου παίκτη και θα καταλήξει σε ισοπαλία, καθώς μετά από κάποιο πεπερασμένο αριθμό κινήσεων θα φτάσουμε για δεύτερη φορά στην ίδια κατάσταση.

<u>Πολυπλοκότητα</u>

Αυτό που ορίζει την πολυπλοκότητα είναι τα βήματα και ο σχηματισμός του Η.
Χρειαζόμαστε **O(m n)** χρόνο για το Η, αφού $|V(H)| = O(n^2)$ και $|E(H)| = O(m\ n)$.
Θυμίζουμε πως $n-1 \leq m$ αφού το G είναι συνεκτικό, οπότε $n^2 = O(m\ n)$
Και $m < n^2 \Leftrightarrow m\ n < n^3$ (οπότε $m\ n = O(n^3)$).
Σε κάθε βήμα διαγράφεται τουλάχιστον μία ακμή του Η, το οποίο έχει αρχικά O(m n) ακμές.
Στο τέλος, θα μας μείνουν τουλάχιστον $O(n^2)$ ακμές, καθώς κάθε κορυφή (πέρα από τις τελικές καταστάσεις) οφείλει να έχει μια έγκυρη κίνηση, άρα μια εξωτερική ακμή.

Συνεπώς, ο αλγόριθμός μας, χρειάζεται O(m n) χρόνο, όπου n = |V(G)| και m = |E(G)| .
Το Η έχει $|V(H)| = O(n^2)$ και $|E(H)| = O(m\ n)$.
Καλό θα ήταν να σημειώσουμε, πως **η προεπεξεργασία του παιχνιδιού γίνεται μόνο μία φορά.**
Άπαξ και βρούμε τις συνιστώσες Α, Κ, Ι, σημειώνουμε αντίστοιχα σε κάθε κορυφή το τελικό αποτέλεσμα του παιχνιδιού.
Κρατάμε αυτήν την πληροφορία σε ένα hashtable με κλειδί την κωδικοποίηση της κατάστασης. (ή ακόμα και έναν πίνακα ταξινομημένο με βάση την κατάσταση).
Τότε, για κάθε παρτίδα, χρειαζόμαστε μόλις O(1) (ή O(logn) για πίνακα) χρόνο για να βρούμε το αποτέλεσμα, απλά βρίσκοντας την αρχική κατάσταση της παρτίδας στο hashtable.

Άσκηση 4: "Το Σύνολο των Συνδετικών Δέντρων"

α

Απόδειξη

Το T_2 είναι συνδετικό δέντρο, συνεπώς αν του προσθέσουμε μία ακμή θα κλείνει κύκλο.
Είναι $T_1 \neq T_2$, οπότε υπάρχει $e \in T_1 \backslash T_2$ με $T_2 \cup \{e\}$ να κλείνει κύκλο C.
Υπάρχει ακμή $e' \in C$, με $e' \notin T_1$. Αν το T_1 περιέχει όλες τις ακμές του C, τότε το T_1 περιέχει τον κύκλο C, άτοπο.
Τότε το $T_1' = (T_1 \setminus \{e\}) \cup \{e'\}$ έχει n-1 ακμές και δεν περιέχει κύκλο, άρα είναι συνδετικό δέντρο.

Αλγόριθμος

Έστω μια δεδομένη ακμή $e \in T_1 \backslash T_2$ που ενώνει τις κορυφές u, v.
Στο T_2 βρίσκουμε μονοπάτι p από το u στο v (με DFS ή BFS από το u).
Για κάθε ακμή στο p, εξετάζουμε αν ανήκει στο T_1 .
Όταν θα βρούμε ακμή που δεν ανήκει στο T_1, βρήκαμε τη ζητούμενη e' .

Πολυπλοκότητα

Για να βρούμε το μονοπάτι p χρειαζόμαστε χρόνο O(n), όπου n το πλήθος των κορυφών.
Θυμίζουμε πως σε δέντρο, για τις ακμές έχουμε m = n - 1 = O(n).
Για να εξετάσουμε κάθε ακμή του p, χρειαζόμαστε χρόνο O(n).
Συνολικά, χρειαζόμαστε χρόνο **O(n)**.

β

Απόδειξη

Θα αποδείξουμε με επαγωγή πως για κάθε $T_1, T_2 \in H$ με $d(T_1, T_2)$ η απόσταση των T_1, T_2 στο H, είναι $|T_1 \backslash T_2| = d(T_1, T_2)$.
Σημειώνουμε πως αν ισχύει η παραπάνω ιδιότητα, τότε προφανώς ισχύει και η συνεκτικότητα του H, καθώς για οποιαδήποτε δύο ΣΔ T_1, T_2 , το $|T_1 \backslash T_2|$ είναι πεπερασμένο.

Επαγωγική Βάση
Από υπόθεση, για $|T_1 \backslash T_2| = 1$, $d(T_1, T_2) = 1$.

Επαγωγική Υπόθεση
Έστω πως για $T_1, T_2 \in H$, είναι $|T_1 \backslash T_2| = k$, αvv $d(T_1, T_2) = k$.

Επαγωγικό Βήμα
(=>)
Έστω T_1, T_2 με $|T_1 \backslash T_2| = k+1$. Θδο $d(T_1, T_2) = k + 1$.
Από (α), μπορούμε να βρούμε $e' \in T_2 \backslash T_1$, ώστε $T_1' = (T_1 \setminus \{\epsilon\}) \cup \{e'\}$ ΣΔ, άρα $T_1 \in H$.
Επειδή $e' \in T_2$, είναι $|T_1' \setminus T_2| = |T_1 \setminus T_2| - 1 = k + 1 - 1 = k$.
Επίσης, $|T_1' \setminus T_1| = 1$.
Από (Ε.Υ.), $d(T_1', T_2) = k$ και $d(T_1', T_1) = 1$.

Συνεπώς, $d(T_1,T_2) \leq k + 1$.
Όμως, αν $d(T_1,T_2) < k + 1 \leq k$, από (Ε.Υ.) είναι $|T_1 \backslash T_2| \leq k$.
Άτοπο, καθώς από υπόθεση $|T_1 \backslash T_2| = k+1$.
Επομένως, $d(T_1,T_2) = k + 1$

(<=)
Έστω T_1,T_2 με $d(T_1,T_2) = k + 1$. Θδο $|T_1 \backslash T_2| = k+1$.
Έστω p το μονοπάτι από το T_1 στο T_2 στο Η.
Έστω T_1' η πρώτη κορυφή μετά το T_1 στο p.
Τότε $d(T_1,T_1') = 1$ και $d(T_1',T_2) = k$.
Από (Ε.Υ.) $|T_1' \backslash T_2| = k$ και $|T_1 \backslash T_1'| = 1$.
Συνεπώς, μπορούμε να φτάσουμε από το T_1 στο T_2, αλλάζοντας μία ακμή του T_1 και παίρνοντας το T_1' και, στη συνέχεια αλλάζοντας k ακμές του T_1' .
Λοιπόν, $|T_1 \backslash T_2| = k \pm 1$.
Όμως, αν $|T_1 \backslash T_2| = k - 1$, τότε από (Ε.Υ.), $d(T_1,T_2) = k - 1$.
Άτοπο, καθώς από υπόθεση, $d(T_1,T_2) = k + 1$.
Επομένως, $|T_1 \backslash T_2| = k + 1$.

Αλγόριθμος
Μετράμε τις ακμές e που ανήκουν στο T_1 και όχι στο T_2 , δηλαδή $e \in T_1 \backslash T_2$.
Έστω k το πλήθος των e.
Σε κάθε βήμα, για κάθε ακμή e, χρησιμοποιούμε τον αλγόριθμο του (α), ώστε να επιλέξουμε ακμή $e' \in T_2$ τ.ω. $T_1' = (T_1 \setminus \{e\}) \cup \{e'\}$ να είναι συνδετικό δέντρο.
Αποθηκεύουμε το T_1'.
Το T_1' είναι η επόμενη κορυφή στο μονοπάτι p από T_1 στο T_2 .
Επαναλαμβάνουμε, το βήμα με το νέο δέντρο που σχηματίστηκε μέχρις ότου φτάσουμε στο T_2.

Πολυπλοκότητα
Για το κάθε βήμα χρειαζόμαστε O(n) χρόνο (όπως δείξαμε στο (α)) για να βρούμε έγκυρη ακμή e'.
Επίσης, χρειαζόμαστε O(n) χρόνο για να αποθηκεύουμε το νέο T_1' .
Θα έχουμε συνολικά k βήματα, όπου k το πόσο "απέχει" το T_1 από το T_2.
Επομένως, συνολικά χρειαζόμαστε χρόνο $\mathbf{O(n \cdot k)}$.

γ

Αλγόριθμος

Θα αξιοποιήσουμε τον αλγόριθμο Kruskal.

i. **Θεωρούμε το γράφημα H_1** με κορυφές και ακμές όμοιες του G, αλλά με τα εξής βάρη στις ακμές του.

Όλες, οι ακμές $e \in E_1$ έχουν βάρος $w(e) = 1$ και οι ακμές του E_2 έχουν βάρος 0.

Εκτελούμε τον Kruskal στο H_1 .

Προκύπτει ΕΣΔ στο H_1 , έστω T_1 με $w(T_1) = w_1$.

Αυτό σημαίνει πως το T_1 περιέχει ακριβώς w_1 ακμές από το E_1 .

Αν $w_1 > k$, τότε δεν υπάρχει το ζητούμενο ΣΔ στο G, καθώς χρειαζόμαστε τουλάχιστον w_1 ακμές του E_1 για να φτιάξουμε ΣΔ.

Αν δεν ήταν έτσι, ο Kruskal, δε θα τις διάλεγε αφού δε "συμφέρουν" έχοντας βάρος 1 στο H_1 .

ii. **Θεωρούμε το γράφημα H_2** με κορυφές και ακμές όμοιες του G, αλλά με τα εξής βάρη στις ακμές του:

Όλες, οι ακμές $e \in E_1$ έχουν βάρος $w(e) = 0$ και οι ακμές του E_2 έχουν βάρος 1.

Εκτελούμε τον Kruskal στο H_2 .

Προκύπτει ΕΣΔ στο H_2 , έστω T_2 με $w(T_2) = w_2$.

Αυτό σημαίνει πως το T_2 περιέχει ακριβώς w_2 ακμές από το E_2 .

Αν $w_2 > n - 1 - k \Leftrightarrow k > n - 1 - w_2$, τότε δεν υπάρχει το ζητούμενο ΣΔ στο G, καθώς χρειαζόμαστε τουλάχιστον w_2 ακμές του E_2 για να φτιάξουμε ΣΔ και άρα το πολύ $n - 1 - w_2$ ακμές από το E_1.

Αν δεν ήταν έτσι, ο Kruskal, δε θα τις διάλεγε αφού δε "συμφέρουν" έχοντας βάρος 1 στο H_2 .

iii. **Τα T_1 και T_2 είναι ΣΔ στο G.**

Το T_1 περιέχει τον ελάχιστο δυνατό αριθμό από ακμές του E_1 και το T_2 τον μέγιστο.

Χρησιμοποιούμε τον αλγόριθμο του (β) ώστε να πάμε από το T_1 στο T_2 .

Σταματάμε τη διαδικασία όταν βρούμε δέντρο T_1' με k ακμές στο E_1 .

Επειδή το T_1 έχει λιγότερες (ή ίσες) από k στο E_1 και το T_2 έχει περισσότερες (ή ίσες) από k στο E_1, και ο αλγόριθμος του (β) "εναλλάσσει" ακριβώς μία ακμή σε κάθε βήμα, σίγουρα υπάρχει τέτοιο T_1'.

Πολυπλοκότητα

Για να δώσουμε βάρη στις ακμές χρειαζόμαστε O(m) χρόνο.

Για τον Kruskal χρειαζόμαστε O(m logm).

Τα παραπάνω γίνονται δύο φορές, αλλά αυτό δεν επηρεάζει την ασυμπτωτική πολυπλοκότητα.

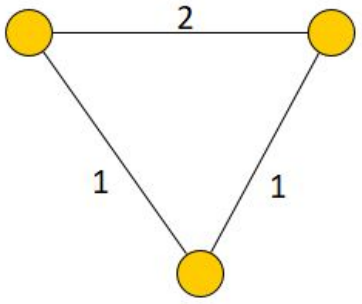
Για τον αλγόριθμο του (β) χρειαζόμαστε O(k n) χρόνο καθώς στη χείριστη περίπτωση, το T_1 δεν έχει καμία ακμή από το E_1, οπότε πρέπει να προσθέσουμε k ακμές.

Τελικά, χρειαζόμαστε O(m logm + k n) χρόνο.

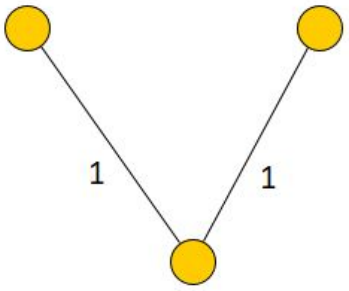
Άσκηση 5: "Μοναδικότητα Ελάχιστου Συνδετικού Δέντρου"

α

Ως αντιπαράδειγμα έχουμε το κάτωθι γράφημα G.



original graph G

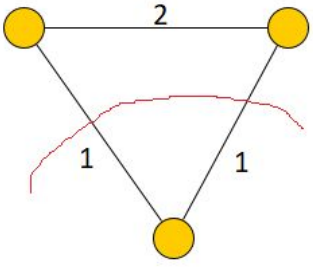


unique mst T

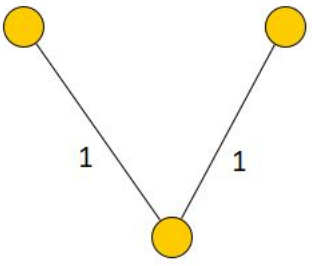
β.ι

Έστω ένα γράφημα G με T_1, T_2 δύο διαφορετικά ΕΣΔ του G.
Υπάρχει $e \in T_1 \setminus T_2$. Έστω e = (u,w) με $u,w \in V$.
Επειδή κάθε ΣΔ είναι ελαχιστοτικό, στο T_1 η e είναι γέφυρα, δηλαδή αν διαγράψουμε την e, δημιουργούνται δύο διακριτές συνεκτικές συνιστώσες έστω U και W.
Στο T_2 υπάρχει ακμή e' τ.ω. να ενώνει τις U και W.
Από υπόθεση, η ακμή ελαχίστου βάρους που διασχίζει κάθε τομή είναι μοναδική.
Ας υποθέσουμε χωρίς βλάβη της γενικότητας πως η e είναι αυτή η ακμή, η οποία διασχίζει την τομή (U, V \ U).
Προσθέτοντας την e στο T_2 κλείνει κύκλος C.
Συνεπώς, υπάρχει δεύτερο μονοπάτι από τη $u \in U$ στη $w \in W$, και άρα ακμή $e' \in C$ η οποία διασχίζει την τομή (U, V \ U).
Τότε w(e) < w(e').
Το $T_2' = (T_2 \setminus \{e'\}) \cup \{e\}$ έχει μικρότερο βάρος από το T_2 και είναι συνδετικό δέντρο.
Συνεπώς, το T_2 δεν είναι ΕΣΔ.
Άτοπο, από υπόθεση.
Επομένως, το ΕΣΔ είναι μοναδικό.

β.ιι Ως αντιπαράδειγμα για το αντίστροφο έχουμε το κάτωθι γράφημα G:



original graph G (with cut C)



unique mst T

γ

Έστω ένα γράφημα G(V,E) και ένα ΕΣΔ του, T(V,F).
Αν προσθέσουμε μία ακμή e ϵ E \ F στο T, τότε σίγουρα κλείνει κάποιος κύκλος C.
Έστω e' μια άλλη ακμή του C.
Τότε στο T μπορούμε να ανταλλάξουμε τις ακμές e, e' .
Το T' = (T \ {e'}) $\cup$ {e} είναι επίσης ΣΔ του G.
Μάλιστα, αν w(e) = w(e'), τότε το T' είναι ΕΣΔ του G.

Πρόταση

Έστω ένα γράφημα G(V,E) και ένα ΕΣΔ του, T(V,F).
Το T είναι μοναδικό ΕΣΔ του G ανν για κάθε ακμή e ϵ E \ F , ο κύκλος C που σχηματίζεται στο T $\cup$ {e} έχει μοναδική ακμή μεγίστου βάρους την e.

Απόδειξη

(=>)

Το T είναι μοναδικό ΕΣΔ του G.
Έστω ακμή e ϵ E \ F τ.ω. στον κύκλο C που σχηματίζεται στο T $\cup$ {e}, να υπάρχει διαφορετική ακμή e' με w(e) = w(e').
Τότε, το T' = (T \ {e'}) $\cup$ {e} είναι ΕΣΔ του G και διαφορετικό του T.
Άτοπο, καθώς το T είναι μοναδικό ΕΣΔ από υπόθεση.

Σημειώνουμε πως οι e, e' θεωρήθηκαν μεγίστου βάρους.
Διαφορετικά, αν είχαμε e'' στο C με w(e) < w(e''), τότε ανταλλάσσοντας την e με την e'' θα φτάναμε σε ΣΔ με μικρότερο βάρος, που είναι άτοπο καθώς το T είναι ΕΣΔ.

(<=)

Έστω T_1, T_2 δύο διαφορετικά ΕΣΔ του G.
Υπάρχει e ϵ $T_1 \backslash T_2$. Έστω e = (u,w) με u,w ϵ V.
Η e κλείνει κύκλο C' στο T_2 $\cup$ {e} από την u στη w.
Ακριβώς όπως έχουμε δείξει στο (β), υπάρχει e' ϵ $T_2 \backslash T_1$ και e' ϵ C' η οποία διασχίζει την τομή (U, V \ U).
Τότε όμως, από υπόθεση w(e) > w(e').
Μπορούμε να προσθέσουμε την e' στο T_1 . Τότε κλείνει κύκλος C από τη u στην w.
Ο κύκλος περιέχει προφανώς την e = (u,w).
Όμως, όπως προείπαμε w(e) > w(e') και οι δύο ακμές είναι ανταλλάξιμες.
Τότε το T_1' = (T_1 \ {e}) $\cup$ {e'} είναι ΣΔ με μικρότερο βάρος από το T_1 .
Άτοπο, καθώς το T_1 είναι ΕΣΔ από υπόθεση.

δ

Αλγόριθμος

Γνωρίζουμε πως ο Kruskal μπορεί να βρει όλα τα δυνατά ΕΣΔ, ανάλογα με την επιλογή που θα κάνουμε ανάμεσα σε έγκυρες ομοβαρείς ακμές.
Θα υλοποιήσουμε μια παραλλαγή του Kruskal για να ελέγξουμε τη μοναδικότητα ΕΣΔ.

Στον "κλασικό" Kruskal, εξετάζουμε μία μία τις ακμές με σειρά από το μικρότερο βάρος στο μεγαλύτερο. Σε ομοβαρείς ακμές επιλέγουμε μία "τυχαία".
Αν μια ακμή σχηματίζει κύκλο με το δάσος που έχει σχηματιστεί μέχρι εκείνη τη στιγμή, την πετάμε. Διαφορετικά, την προσθέτουμε στο δάσος μέχρι να φτάσουμε να έχουμε ΣΔ (το οποίο θα είναι ΕΣΔ). Χρησιμοποιούμε τις union-find για να το κάνουμε αποδοτικά.

Στην παραλλαγή μας, θα εξετάζουμε ταυτόχρονα όλες τις ομοβαρείς ακμές.
Συγκεκριμένα, θα λαμβάνουμε σε κάθε βήμα όλες τις ομοβαρείς ακμές.
Θα τις κάνουμε όλες ένα πέρασμα(χωρίς να κάνουμε union, απλά μέσω find) και θα πετάμε όσες τέτοιες ακμές κλείνουν κύκλο με το δάσος που έχει ήδη σχηματιστεί.
Στη συνέχεια, θα παίρνουμε τις υπόλοιπες και θα τις προσθέτουμε στο δάσος.
Αν οποιαδήποτε από αυτές κλείνει κύκλο, τότε δεν έχουμε μοναδικό ΕΣΔ.
Πράγματι, έστω ομοβαρείς ακμές e και e' με e' να κλείνει κύκλο μέσω της e (αλλά να μην κλείνει κύκλο στο προϋπάρχον δάσος).
Τότε, όπως έχουμε δει και στα προηγούμενα ερωτήματα, οι e και e' είναι ανταλλάξιμες και, επειδή w(e) = w(e'), η επιλογή οποιασδήποτε από τις δύο μπορεί να καταλήξει σε έγκυρο ΕΣΔ.

Αν φτάσουμε μέχρι το τέλος του Kruskal και δεν έχει προκύψει τέτοια περίπτωση τότε έχουμε μοναδικό ΕΣΔ.

Σημείωση:

Θα παρουσιάσουμε μια πιο επώδυνη παραλλαγή του παραπάνω αλγορίθμου στη προγραμματιστική άσκηση 3.2 .
Πρόκειται για πιο επώδυνη παραλλαγή, γιατί, εκεί, δε μας ενδιαφέρει απλά η μοναδικότητα, αλλά, επιπλέον, να μετρήσουμε πόσες είναι οι απαραίτητες ακμές και πόσες είναι ενδεχόμενες.

Πολυπλοκότητα

Η πολυπλοκότητα είναι η πολυπλοκότητα του Kruskal **O(m logm)**.
Στην πράξη, με τον τρόπο που υλοποιήσαμε τον τροποποιημένο αλγόριθμο, θα κάνουμε δύο περάσματα καθώς εξετάζουμε τις ακμές (το πρώτο για να πετάξουμε τις ακμές που κλείνουν κύκλο στο προϋπάρχον δάσος), αλλά αυτό δεν επηρεάζει την ασυμπτωτική πολυπλοκότητα.
Εξάλλου το O(m logm) οφείλεται στην ταξινόμηση των ακμών κατά βάρος.