Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Αλγόριθμοι & Πολυπλοκότητα – 7ο Εξάμηνο 2018-2019
Αβραμίδης Κλεάνθης – ΑΜ: 03115117

3η Γραπτή Άσκηση – Απαντήσεις

➢ *Θέμα 1*

Θα προσπαθήσουμε να εκφράσουμε το πρόβλημα της εύρεσης βέλτιστου μονοπατιού με αναφορά σε ένα κόμβο συναρτήσει των βέλτιστων μονοπατιών για τους γείτονές του και στη συνέχεια θα εφαρμόσουμε Δυναμικό Προγραμματισμό προκειμένου να υπολογίσουμε αποδοτικά τις απαιτούμενες τιμές. Συγκεκριμένα, επιλέγουμε αυθαίρετα έναν κόμβο u και εκτελούμε διάσχιση DFS. Για κάθε κόμβο χρειαζόμαστε 2 τιμές:

1. Το μακρύτερο μονοπάτι στο δέντρο με άκρο το u, έστω P_{1u}
2. Το μακρύτερο μονοπάτι στο υποδέντρο που έχει το u σαν ρίζα, έστω P_{2u}

Υποθέτουμε πως έχουμε υπολογίσει τα 2 παραπάνω μονοπάτια αναδρομικά για κάθε παιδί του u, ώστε να τα χρησιμοποιήσουμε στον υπολογισμό των μονοπατιών για τον u. Θα ισχύουν τα εξής:

1. $P_{1u} = w(u) + max\{P_{1v}\}\ \forall\ παιδί\ v\ του\ u$
2. $P_{2u} = max\{max\{P_{2v}\}, P_{1u}, w(u) + max\{P_{1v_1} + P_{1v_2}, 0\}\}\ \forall\ παιδί\ v\ του\ u$

Το μακρύτερο μονοπάτι στο υποδέντρο του u είτε δε θα περιέχει καθόλου τον u (οπότε αναγόμαστε στο αντίστοιχο μακρύτερο μονοπάτι από αυτά των παιδιών του), είτε θα περιέχει μόνο αυτόν (οπότε συνυπολογίζουμε μόνο το βάρος του), είτε θα περιέχει τον u σαν άκρο (οπότε αναγόμαστε στο P_{1u}), είτε θα τον περιέχει σαν εσωτερικό κόμβο. Σε αυτή την περίπτωση θα χρειαστούμε ένα μονοπάτι να προηγείται του u και ένα να έπεται. Προφανώς αυτά τα 2 μονοπάτια θα έχουν σαν άκρα τα παιδιά του u και για να έχουμε το μέγιστο δυνατό, θα επιλέξουμε τα 2 μεγαλύτερα. Τέλος, το μονοπάτι P_1 του u υπολογίζεται απλά ως η προέκταση του αντίστοιχου μέγιστου μονοπατιού από αυτά των παιδιών του. Έτσι εξηγούνται οι παραπάνω σχέσεις, οι οποίες διασφαλίζουν όλες τις δυνατές περιπτώσεις, άρα και την ορθότητα της προσέγγισης.

Μέσω της DFS διάσχισης και ξεκινώντας τη διαδικασία από τα φύλλα, για τα οποία τα μέγιστα και μόνα μονοπάτια τους είναι οι αυτοί κόμβοι, εξασφαλίζουμε πως κάθε κόμβος θα έχει διαθέσιμα όλα τα στοιχεία που χρειάζεται (θα έχει ολοκληρωθεί ο υπολογισμός για τα παιδιά του). Με δεδομένα λοιπόν όλα τα στοιχεία, το P_{1u} υπολογίζεται σε γραμμικό χρόνο ενώ το P_{2u} θα απαιτήσει γραμμικό χρόνο για τα 2 εσωτερικά μέγιστα και σταθερό χρόνο για τον τελικό υπολογισμό του μεγίστου 3 στοιχείων. Αυτές οι διαδικασίες μπορούν να ενσωματωθούν στη γραμμική διάσχιση του δέντρου μέσω DFS. Οπότε συνολικά ο παραπάνω αλγόριθμος, με χρήση Δυναμικού Προγραμματισμού, θα έχει πολυπλοκότητα $\mathbf{O}(|\mathbf{V}|)$ και θα βρίσκει το ζητούμενο μονοπάτι P_{2u}.

➢ *Θέμα 2*

a) Υποθέτουμε πως κάθε κορυφή u έχει ακμές προς κορυφές $w_1, w_2, \dots, w_k$ στο γράφημα G. Συνεπώς οι κορυφές που είναι προσπελάσιμες από την u είναι η ίδια καθώς και όσες κορυφές είναι προσπελάσιμες από καθεμιά από τις $w_1, w_2, \dots, w_k$. Η αναδρομή αυτή, με αναφορά στο ελάχιστο κόστος των κορυφών, εκφράζεται ως εξής:

$$c(u) = \min\{\, p_u\,, \min c(w)\ \forall (u,w) \in E\,\}$$

Θα εφαρμόσουμε Δυναμικό Προγραμματισμό για να λύσουμε με αποδοτικό τρόπο το πρόβλημα, δηλαδή για να έχουμε στη διάθεσή μας όλες τις τιμές $c(w)$ πριν υπολογίσουμε την $c(u)$. Στην περίπτωση του DAG, αρκεί να εφαρμόσουμε τοπολογική ταξινόμηση και να υπολογίσουμε τα κόστη από το τέλος προς την αρχή. Κάθε μία από αυτές τις διαδικασίες απαιτεί γραμμικό πλέον χρόνο. Κάθε κόμβος προσπελαύνεται ωστόσο μοναδική φορά (έχει γνωστό κόστος), επομένως έχουμε πολυπλοκότητα συνολικά $\mathbf{O}(|\mathbf{V}|)$.

b) Στη γενική περίπτωση κατευθυνόμενου γραφήματος, το επιπλέον στοιχείο είναι η ύπαρξη κύκλων και γενικότερα ισχυρά συνεκτικών συνιστωσών (ΙΣΣ). Εύκολα διαπιστώνεται πως κάθε κορυφή μιας ΙΣΣ έχει το ίδιο σύνολο προσπελάσιμων κορυφών με κάθε άλλη κορυφή της ίδιας ΙΣΣ, άρα και το ίδιο κόστος. Μπορούμε λοιπόν να θεωρήσουμε κάθε ΙΣΣ σαν μια κορυφή, με τιμή το μικρότερο p των εσωτερικών κορυφών της, οπότε και το πρόβλημα ανάγεται στο (α). Το κόστος κάθε κορυφής μιας ΙΣΣ θα είναι προφανώς το κόστος που θα υπολογιστεί για την ενοποιημένη κορυφή/ΙΣΣ.

Ο παραπάνω αλγόριθμος είναι, όπως είπαμε, γραμμικός. Η εύρεση των ΙΣΣ του γραφήματος γίνεται επίσης σε γραμμικό χρόνο, εφαρμόζοντας τον αλγόριθμο Kosaraju (CLRS: εν. 22.5) που απαιτεί από ένα DFS στο G και στο G^T (με αντιστροφή της φοράς των ακμών του που επιτυγχάνεται σε γραμμικό επίσης χρόνο) ή και τον αλγόριθμο Tarjan[1] που απαιτεί 1 DFS. Συνολικά λοιπόν έχουμε πολυπλοκότητα $\mathbf{O}(|\mathbf{V}|)$.

➢ *Θέμα 3*

Το παιχνίδι αποτελείται γενικά από $2V^2$ καταστάσεις της μορφής $(\theta\acute{\epsilon}\sigma\eta_K, \theta\acute{\epsilon}\sigma\eta_A, K|A)$ τις οποίες αποθηκεύουμε σε κατάλληλη δομή. Κάθε μία από αυτές μπορεί να είναι αρχική κατάσταση και για κάθε αρχική κατάσταση αντιστοιχεί ένα δέντρο επόμενων καταστάσεων που καταλήγει σε φύλλα τερματισμού της παρτίδας. Οι συνθήκες τερματισμού είναι οι εξής:

- Νίκη Κ: $dist(\theta\acute{\epsilon}\sigma\eta_K, d) = 1$ και παίζει ο Κ (επιστρέφει 1)
- Νίκη Α: $dist(\theta\acute{\epsilon}\sigma\eta_K, \theta\acute{\epsilon}\sigma\eta_A) \leq 1$ και παίζει ο Α (επιστρέφει -1)
- Ισοπαλία: Η κατάσταση έχει επαναληφθεί στο τρέχον μονοπάτι / μαρκαρισμένη (επιστρέφει 0)

Με βάση αυτούς τους κανόνες θεωρούμε μια αρχική κατάσταση και κάνουμε γραμμική στο πλήθος των καταστάσεων DFS διάσχιση στο δέντρο της. Επιπλέον μαρκάρουμε όποια κατάσταση περνάμε και την ξεμαρκάρουμε όταν της αποδίδουμε τιμή. Με αυτό τον τρόπο κρατάμε κάθε φορά μαρκαρισμένο το τρέχον μονοπάτι του δέντρου στο οποίο βρισκόμαστε. Τα παιδιά κάθε κατάστασης βρίσκονται σύμφωνα με τον εξής κανόνα:

- Αν $current = (\theta\acute{\epsilon}\sigma\eta_K, \theta\acute{\epsilon}\sigma\eta_A, K)$ τότε $next = (\gamma\epsilon\acute{\iota}\tau o\nu\epsilon\varsigma(\theta\acute{\epsilon}\sigma\eta_K), \theta\acute{\epsilon}\sigma\eta_A, A)$
- Αν $current = (\theta\acute{\epsilon}\sigma\eta_K, \theta\acute{\epsilon}\sigma\eta_A, A)$ τότε $next = (\theta\acute{\epsilon}\sigma\eta_K, \gamma\epsilon\acute{\iota}\tau o\nu\epsilon\varsigma(\theta\acute{\epsilon}\sigma\eta_A), K)$

Συνεπώς μπορούμε να μεταβούμε με τις υπάρχοντες ακμές του G από κατάσταση σε κατάσταση. Σε κάθε βήμα ελέγχουμε τις συνθήκες τερματισμού. Αν το παιχνίδι λήγει (φύλλο του γραφήματος), επιστρέφουμε τις αντίστοιχες τιμές, αλλιώς συνεχίζουμε τη διάσχιση. Είναι σημαντικό να τονίσουμε πως τα φύλλα ή λαμβάνουν τιμή ως τελικές καταστάσεις ή λαμβάνουν τιμή κατά τη διάρκεια εκτέλεσης του αλγόριθμου ή πρόκειται για ισοπαλίες, οπότε και δε λαμβάνουν τιμή, καθώς δε γνωρίζουμε ακόμα αν η ισοπαλία είναι η βέλτιστη επιλογή για αυτή την κατάσταση παραπάνω στο δέντρο. Επιστρέφουν ωστόσο 0.

Κάθε κόμβος λαμβάνει τη μέγιστη τιμή από αυτές που επιστρέφουν τα παιδιά του αν είναι κλέφτης (προτιμά νίκη κλέφτη ή έστω ισοπαλία) και την ελάχιστη αν είναι αστυνόμος (προτιμά ήττα κλέφτη ή έστω ισοπαλία). Προφανώς ο υπολογισμός αρχίζει από τα φύλλα και υπολογίζει κάθε κόμβο αφού υπολογίσει όλα του τα παιδιά. Οπότε μπορούμε, αποθηκεύοντας μέγιστα, να αναθέσουμε σε σταθερό χρόνο τις απαιτούμενες τιμές σε κάθε κατάσταση. Ιδίως για τις ισοπαλίες, ο αλγόριθμος «χτυπάει» μόνο σε όσες βρίσκονται στο ίδιο μονοπάτι, καθώς, επιστρέφοντας, ο DFS θα ξεμαρκάρει τις καταστάσεις και όχι σε όλες τις εμφανίσεις της κατάστασης αυτής στο δέντρο.

Μετά το τέλος της διάσχισης θα έχουμε αναθέσει τιμή στην αρχική μας κατάσταση. Θα γνωρίζουμε τόσο το αποτέλεσμα αυτής της αρχικής κατάστασης, όσο και κάθε άλλης στο δέντρο καταστάσεων. Για να αποδώσουμε τιμή σε κάθε πιθανή κατάσταση, κάθε φορά βρίσκουμε μια κατάσταση χωρίς τιμή και επαναλαμβάνουμε τη διαδικασία. Τώρα φυσικά τα φύλλα θα είναι όσοι κόμβοι έχουν τιμή. Τα δέντρα δηλαδή θα γίνονται όλο και μικρότερα και τελικά θα ανατεθούν γραμμικά όλες οι τιμές στις καταστάσεις. Η πολυπλοκότητα που προκύπτει: $2V^2 = \boldsymbol{O}(|\boldsymbol{V}|^{\mathbf{2}})$.

- ***Θέμα 4***

a) Αφαιρώντας την ακμή e από το T_1 προκύπτουν 2 συνεκτικές συνιστώσες V_1 και V_2. Το T_2 από την άλλη, εφόσον είναι συνεκτικό, ενώνει με μια ακμή αυτές τις 2 συνιστώσες. Αυτή η ακμή e' είναι διαφορετική προφανώς της e, αφού η e δεν περιλαμβάνεται στο T_2. Επίσης, η e' δε μπορεί να ανήκει στο T_1 καθώς μαζί με την e θα σχημάτιζε κύκλο ανάμεσα στις δύο συνεκτικές συνιστώσες. Επομένως, με την προσθήκη της e' έχουμε και πάλι συνεκτικότητα και ένα νέο συνδετικό δέντρο ($T_1 - e + e$').

Ένας αλγόριθμος για την εύρεση μιας e' με δεδομένα τα T_1, T_2, e είναι ο εξής: Ξεκινώντας από μια κορυφή του T_1, στην οποία συνδεόταν η e, διασχίζουμε με DFS τη V_1 (ή τη V_2 αντίστοιχα) και μαρκάρουμε κατάλληλα τους κόμβους της. Στη συνέχεια διασχίζουμε με DFS το T_2, ξεκινώντας για ευκολία από τον τελευταίο κόμβο που μαρκάραμε. Μόλις βρούμε έναν κόμβο που δεν είναι μαρκαρισμένος, αυτός προφανώς θα ανήκει στη V_2 και η ακμή που μόλις προσπελάσαμε θα είναι η ζητούμενη e' που συνδέει τις 2 συνεκτικές συνιστώσες. Ο αλγόριθμος απαιτεί γραμμικό χρόνο για κάθε διάσχιση που κάνουμε, οπότε έχουμε πολυπλοκότητα $\mathbf{O}(|\mathbf{V}|)$.

b) Η συνεκτικότητα του H συνεπάγεται ότι από κάθε συνδετικό δέντρο του γραφήματος μπορούμε να μεταβούμε σε οποιοδήποτε άλλο. Αυτό έπεται από το (α): Έστω ότι θέλουμε να μεταβούμε από οποιοδήποτε συνδετικό δέντρο T_1 σε οποιοδήποτε άλλο T_2. Αυτό γίνεται αν αντικαταστήσουμε όλες τις ακμές του T_1 που δεν υπάρχουν στο T_2 με αντίστοιχες ακμές του T_2. Αντικαθιστώντας κάθε φορά μια τέτοια ακμή, με βάση τον παραπάνω αλγόριθμο, δημιουργούμε μια αλληλουχία από συνδετικά δέντρα μέχρι εν τέλει να καταλήξουμε στο T_2 μετά από $|T_1/T_2|$ αντικαταστάσεις, οι οποίες είναι πάντα δυνατές. Κινούμαστε δηλαδή μέσα στο δέντρο H.

Εφόσον αυτό μπορεί να γίνει για οποιαδήποτε ζευγάρια κορυφών του H, το H είναι συνεκτικό και η απόσταση μεταξύ οποιονδήποτε κορυφών του, T_1 και T_2, είναι $|T_1/T_2|$. Τέλος, το μονοπάτι το οποίο θα διασχίσουμε στο H κάνοντας την διαδικασία που περιγράψαμε στο (α) τόσες φορές όσες και οι ακμές που διαφέρουν, είναι και το συντομότερο για να μεταβούμε από το T_1 στο T_2. Διότι η ύπαρξη μικρότερου μονοπατιού θα συνεπαγόταν και απόσταση (ακμών) μεταξύ τους μικρότερη από αυτή που δείξαμε, άτοπο.

c) Θα υπολογίσουμε καταρχάς 2 συνδετικά δέντρα, ένα που θα περιέχει τον ελάχιστο αριθμό ακμών από το E_1, έστω $T(a)$ και ένα που θα περιέχει το μέγιστο αριθμό ακμών, έστω $T(b)$. Για την πρώτη περίπτωση αντιστοιχίζουμε στις ακμές που ανήκουν στο E_1 την τιμή 1 και στις υπόλοιπες την τιμή 0 και εφαρμόζουμε τον αλγόριθμο Kruskal για τον προσδιορισμό ελάχιστου συνδετικού δέντρου, εν προκειμένω του δέντρου εκείνου που περιέχει τις λιγότερες δυνατές ακμές από το E_1. Αντίστοιχα δουλεύουμε για τη δεύτερη περίπτωση, αντιστοιχώντας την τιμή 0 σε όσες ακμές ανήκουν στο E_1και την τιμή 1 στις υπόλοιπες. Κάθε φορά μετράμε τον αριθμό των ακμών από E_1, a και b και μαρκάρουμε τις ακμές του $T(a)$.

Διακρίνουμε τώρα περιπτώσεις: Αν $k < a$ ή $k > b$ τότε δεν υπάρχει το ζητούμενο συνδετικό δέντρο, οπότε ο αλγόριθμος τερματίζει ανεπιτυχώς. Αν $k = a$ ή $k = b$ έχουμε προφανώς βρει το ζητούμενο δέντρο, οπότε ο αλγόριθμος τερματίζει επιτυχώς και επιστρέφει το δέντρο αυτό. Στην τελευταία περίπτωση κατά την οποία $k \in (a, b)$ ο αλγόριθμος θα τερματίσει επιτυχώς, καθώς θα αποδείξουμε πως υπάρχει συνδετικό δέντρο με k κορυφές από το σύνολο E_1 για κάθε $k \in (a, b)$:

Ξεκινάμε από το $T = T(a)$ και για κάθε ακμή του E_1 που υπάρχει στο $T(b)$ κάνουμε τα εξής:

- Αν υπάρχει στο T (αν είναι μαρκαρισμένη) την αγνοούμε.

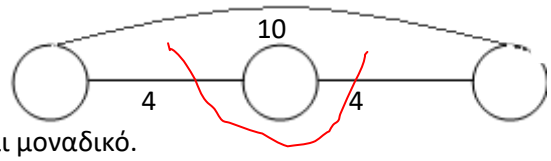
- Αλλιώς, την προσθέτουμε στο T και τη μαρκάρουμε. Πλέον το G θα έχει κύκλο οπότε αφαιρούμε μια άλλη ακμή από αυτόν, η οποία δεν περιέχεται στο $T(b)$. Αυτή η ακμή θα υπάρχει πάντα, καθώς σε αντίθετη περίπτωση το $T(b)$ δε θα ήταν προφανώς δέντρο, αφού θα περιείχε κύκλο. Θέτουμε T το νέο δέντρο.

Συνεπώς δημιουργείται ένα νέο συνδετικό δέντρο T το οποίο περιέχει το πολύ μία παραπάνω ακμή του E_1 από το προηγούμενο. Επαναλαμβάνουμε λοιπόν την παραπάνω διαδικασία μέχρις ότου καταλήξουμε στο δέντρο με ακριβώς k ακμές από το σύνολο E_1. Το επιστρέφουμε και ο αλγόριθμος τερματίζει επιτυχώς.

Σημείωση: Αξιοποιώντας το 4(β) θα μπορούσαμε να ισχυριστούμε απευθείας πως υπάρχει συνδετικό δέντρο με k κορυφές από το σύνολο E_1 για κάθε $k \in (a, b)$. Τα δύο δέντρα που υπολογίσαμε θα συνδέονται όπως αποδείξαμε στο H και θα μπορούμε να μεταβούμε από το ένα στο άλλο μέσω αλλεπάλληλων αντικαταστάσεων μίας ακμής. Συνεπώς στα ενδιάμεσα δέντρα θα υπάρχει όλο το εύρος ακμών από το E_1 στο διάστημα (a, b).

Όσον αφορά την πολυπλοκότητα, θα πρέπει να εκτελέσουμε 2 φορές τον αλγόριθμο Kruskal για την εύρεση των 2 αρχικών δέντρων, αντιστοιχίζοντας κάθε φορά τα κατάλληλα βάρη στις ακμές. Στη συνέχεια θα χρειαστεί στη χειρότερη περίπτωση να εκτελέσουμε k αντικαταστάσεις ακμών. Για την αντιστοίχιση των βαρών απαιτείται γραμμικός χρόνος ως προς το πλήθος των κορυφών. Επίσης, ο αλγόριθμος Kruskal θα τρέξει σε $O(|V| + |E|)$ λόγω του ότι έχουμε 2 μόνο διαφορετικά βάρη. Τέλος, η αντικατάσταση μιας ακμής θα χρειαστεί χρόνο $O(1)$ για να ελέγξουμε αν υπάρχει (τότε θα είναι μαρκαρισμένη) και γενικά E για να βρούμε την ακμή που θα διαγράψουμε (που δε θα είναι μαρκαρισμένη). Συνολικά προκύπτει πολυπλοκότητα $O(|V| + |E| + k|E|)$ κι επειδή για τα συνδετικά δέντρα ισχύει $E = V - 1$ η πολυπλοκότητα τελικά ανάγεται σε $\mathbf{O(k|V|)}$.

➢ *Θέμα 5*



a) Ένα απλό γράφημα – αντιπαράδειγμα είναι το εξής:
Το ΕΣΔ εδώ περιλαμβάνει τις 2 ακμές με βάρος 4 και είναι μοναδικό.

b) Θα αποδείξουμε το ζητούμενο με απαγωγή σε άτοπο. Έστω ότι η ακμή ελάχιστου βάρους που διασχίζει κάθε τομή είναι μοναδική και υπάρχουν 2 ΕΣΔ T_1 και T_2. Θεωρούμε μια ακμή e που ανήκει στο T_1 αλλά όχι στο T_2 καθώς και την ακμή ελάχιστου βάρους m της τομής που δημιουργεί η διαγραφή της e. Αν $m = e$ τότε η προσθήκη της e στο T_2 με παράλληλη διαγραφή της ακμής του T_2 που διασχίζει την συγκεκριμένη τομή θα οδηγήσει σε συνδετικό δέντρο μικρότερου βάρους, κάτι άτοπο από τη στιγμή που T_2 είναι ΕΣΔ. Αν από την άλλη $m \neq e$, η προσθήκη της m στο T_1 με παράλληλη διαγραφή της e θα οδηγήσει σε συνδετικό δέντρο μικρότερου βάρους, κάτι πάλι άτοπο από τη στιγμή που T_1 είναι ΕΣΔ. Συνεπώς η αρχική πρόταση ισχύει.

Ένα αντιπαράδειγμα του αντιστρόφου της δοθείσας πρότασης είναι το γράφημα του προηγούμενου ερωτήματος. Βλέπουμε πως για την «κόκκινη» τομή έχουμε 2 ακμές ίσου (ελάχιστου) βάρους ωστόσο το ΕΣΔ, όπως είπαμε πριν, είναι μοναδικό.

c) Συνθήκη: *Έστω G ένα μη κατευθυνόμενο συνεκτικό γράφημα και T ένα ΕΣΔ του G. Το T είναι μοναδικό ΕΣΔ ανν κάθε ακμή $e \in G - T$ είναι μέγιστου βάρους στον κύκλο $T + e$.*

Απόδειξη Ορθού: Θέλουμε να δείξουμε ότι αν T είναι μοναδικό ΕΣΔ, τότε κάθε ακμή $e \in G - T$ είναι μέγιστου βάρους στον κύκλο $T + e$. Θα αποδείξουμε ισοδύναμα ότι αν υπάρχει ακμή $e \in G - T$ που δεν είναι μέγιστου βάρους στον κύκλο $T + e$, τότε υπάρχει ΕΣΔ επιπλέον του T. Σύμφωνα λοιπόν με την υπόθεση, για την e θα υπάρχει ακμή $e' \in T$ τέτοια ώστε $w(e) \leq w(e')$. Αντικαθιστώντας την e' με την e στο T λαμβάνουμε και πάλι ένα συνδετικό δέντρο με βάρος

$$w(T + e - e') = w(T) + w(e) - w(e') \leq w(T) + w(e') - w(e') \leq w(T)$$

Ισχύει όμως και ότι $w(T + e - e') \geq w(T)$ αφού T είναι ΕΣΔ. Συνεπώς $w(T + e - e') = w(T)$ και $(T + e - e')$ είναι ΕΣΔ, διαφορετικό από το T. Απεδείχθη το ζητούμενο.

Απόδειξη Αντιστρόφου: Θα αποδείξουμε το ζητούμενο με απαγωγή σε άτοπο. Υποθέτουμε ότι κάθε ακμή $\mathrm{e} \in \mathrm{G} - \mathrm{T}$ είναι μέγιστου βάρους στον κύκλο $\mathrm{T} + \mathrm{e}$ και ότι υπάρχει δέντρο T' διαφορετικό του T που είναι επίσης ΕΣΔ. Από το 4(α) γνωρίζουμε πως αν αφαιρέσουμε ακμή $e \in T/T'$, υπάρχει ακμή $e' \in T'/T$, τέτοια ώστε το $(T - e + e')$ να είναι δέντρο. Προφανώς η e είναι μέρος του κύκλου που δημιουργείται στο $T + e'$, αφού η αφαίρεσή της κάνει το γράφημα ακυκλικό. Άρα $w(e') > w(e)$.

Όμως, η e συνδέει επίσης τις ίδιες συνεκτικές συνιστώσες με την e' και δεν υπάρχει στο T'. Μπορούμε λοιπόν αντίστοιχα να δημιουργήσουμε και το δέντρο $(T' - e' + e)$. Θα ισχύει τότε:

$$\mathrm{w(T' - e' + e) = w(T) - w(e') + w(e) < w(T') - w(e') + w(e') < w(T')}$$

Δηλαδή το T' δεν είναι ΕΣΔ. Άτοπο οπότε απεδείχθη το ζητούμενο.

d) Καταρχάς βρίσκουμε ένα ΕΣΔ για το δοθέν γράφημα με τον αλγόριθμο Kruskal και για κάθε ακμή που επιλέγουμε τη μαρκάρουμε. Έπειτα επαναλαμβάνουμε τον αλγόριθμο, αυτή τη φορά όμως όταν επιλέγουμε μεταξύ ακμών με ίσο βάρος, αποφεύγουμε να επιλέξουμε τυχόν μαρκαρισμένες. Συγκεκριμένα τις κρατάμε σε μια δομή και τις προσθέτουμε στο δέντρο μόνο αν δε βρεθεί εναλλακτική ακμή, κατά το πρότυπο της 4(γ). Αυτό γίνεται συνολικά σε γραμμικό χρόνο, άρα για κάθε περίπτωση σε amortised $O(1)$. Μόλις στο δέντρο προστεθεί ακμή μη μαρκαρισμένη, ξέρουμε πως θα καταλήξουμε σε ένα ΕΣΔ διαφορετικό από το προηγούμενο, οπότε ο αλγόριθμος τερματίζει ανεπιτυχώς. Αν ο Kruskal ολοκληρωθεί και καταλήξουμε στο ίδιο ΕΣΔ, τότε προφανώς είναι μοναδικό. Όσον αφορά τον χρόνο εκτέλεσης, έχουμε στη χειρότερη περίπτωση 2 εκτελέσεις Kruskal, δηλαδή πολυπλοκότητα συνολικά $\mathbf{O(|E|\log|E|)}$.

- ✓ *Παρατηρήσεις επί των Απαντήσεων:*

[1] https://en.wikipedia.org/wiki/Tarjan%27s_strongly_connected_components_algorithm