Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Αλγόριθμοι και Πολυπλοκότητα

Διδάσκοντες: Δημήτρης Φωτάκης, Δώρα Σούλιου, Θανάσης Λιανέας

3η Σειρά Γραπτών Ασκήσεων - Σχέδιο Λύσεων

Άσκηση 1: Rendez-vous μετά το Lockdown

Είσοδος: Κατευθυνόμενο γράφημα $G = (V, E)$ του οποίου οι ακμές αντιστρέφονται κάθε λεπτό, μια αρχική κορυφή $s \in V$ και μια τελική κορυφή $t \in V$.

Έξοδος: Αν κάθε λεπτό μπορούμε να διανύσουμε μια ακμή η να μην κουνηθούμε καθόλου, ποιός είναι ο ελάχιστος χρόνος που χρειάζεται για να φτάσουμε στην κορυφή t, ξεκινώντας από την s.

Ιδέα: Αφού, όσον αφορά στον χρόνο, μας ενδιαφέρει το αν είμαστε σε περιττή ή άρτια χρονική στιγμή σε έναν κόμβο, θα κατασκευάσουμε έναν εκτεταμένο γράφο G' του G στον οποίο θα υπάρχουν δυο κόμβοι v_o και v_e για κάθε κόμβο $v \in V$ του G.

Κατασκευή:

- Ο v_o θα αντιστοιχεί στο ότι είμαστε στον κόμβο v σε περιττή χρονική στιγμή και ο v_e σε άρτια.
- Αν στον αρχικό γράφο υπάρχει μια ακμή (v, u), αυτό σημαίνει ότι τις περιττές στιγμές μπορούμε να πάμε από τον v στον u και τις άρτιες από τον u στον v. Άρα, στον G', βάζουμε μια μη κατευθυνόμενη ακμή από τον v_o στον u_e.
- Τέλος, συνδέουμε με μια μη κατευθυνόμενη ακμή κάθε δυο κόμβους v_o, v_e με $v \in V$, καθώς σε κάθε χρονική στιγμή μπορούμε να παραμείνουμε στον ίδιο κόμβο (και να πάμε από άρτια σε περιττή χρονική στιγμή η αντίστροφα).

Λύση: Όπως έχουμε κατασκευάσει τον G', το ελάχιστο μονοπάτι από τον s_o στον t_o μας δίνει τον ελάχιστο χρόνο που χρειαζόμαστε για να φτάσουμε στον προορισμό μας σε περιττή χρονική στιγμή ενώ το ελάχιστο μονοπάτι από τον s_o στον t_e σε ακέραιη (ξεκινάμε από τον s_o καθώς έχουμε θεωρήσει ότι η πρώτη χρονική στιγμή είναι περιττή). Αφού ο (μη κατευθυνόμενος) γράφος μας δεν έχει βάρη, μπορούμε να βρούμε αυτές τις αποστάσεις με ένα BFS, ξεκινώντας από τον t_o. Κρατάμε την ελάχιστη από τις δυο ως τη λύση του προβλήματος.

Πολυπλοκότητα: $O(|V| + |E|)$

Άσκηση 2: Προγραμματίζοντας την Αντίδραση

Είσοδος: Γράφος $G = (V, E)$ και αρχικοί κόμβοι $v_1, ..., v_k \in V$ των σωματιδίων.

Έξοδος: Κόμβος t του γράφου με την ελάχιστη μέγιστη απόσταση από τους $v_1, ..., v_k$ καθώς και η απόσταση και το ελάχιστο μονοπάτι από κάθε $v_1, ..., v_k$ στον t.

Ιδέα: Αν είχαμε την ελάχιστη απόσταση από κάθε κόμβο στο $\{v_1, ..., v_k\}$ σε κάθε κόμβο $v \in V$, θα είχαμε τη λύση. Αφού έχουμε γράφο χωρίς βάρη, μπορούμε για κάθε έναν από τους αρχικούς κόμβους να το κάνουμε με ένα BFS. Καθώς οι κόμβοι κάθε επιπέδου του BFS είναι πιο μακριά από τους κόμβους του προηγούμενου επιπέδου (από την ρίζα του BFS) αρκεί να βρούμε τον κόμβο στον οποίο θα φτάνανε πρώτα όλα τα BFS που ξεκινάνε από τους $\{v_1, ..., v_k\}$.

Λύση:

- Διατρέχουμε τους κόμβους του πρώτου επιπέδου του BFS του κόμβου v_1 μετά του v_2 κ.ο.κ.
- Σε κάθε έναν κόμβο που διατρέχουμε προσαυξάνουμε τον μετρητή των BFS που έχουν φτάσει σε αυτόν και προσθέτουμε σε μια λίστα την ρίζα του BFS απο την οποία φτάσαμε σε αυτόν.
- Συνεχίζουμε επαναληπτικά με τα υπόλοιπα επίπεδα των BFS των κόμβων, μέχρι να βρεθεί κόμβος στον οποίο να έχουν φτάσει k BFS.
- Ορίζουμε αυτόν τον κόμβο ως τον t. Ο χρόνος του να να γίνει η σύγκρουση σε αυτόν είναι όσο το επίπεδο που είχαμε φτάσει, του τελευταίου BFS που έφτασε εκεί. Μπορούμε για κάθε έναν από τους αρχικούς κόμβους να βρούμε την συνολική διαδρομή, χρησιμποιώντας τις λίστες που κρατάμε στους κόμβους.

Πολυπλοκότητα: Στην χειρότερη περίπτωση θα τρέξουμε k BFS σε χρόνο $O(k(|V| + |E|))$.

Άσκηση 3: Ένας Παράξενος Περίπατος

Είσοδος Κατευθυνόμενος γράφος $G = (V, E, c)$ με βάρη στις κορυφές $c(v)$, $\forall v \in V$.

Έξοδος Ένας περίπατος που να ελαχιστοποιεί τον μέγιστο κοινό διαιρέτη των βαρών των κορυφών του.

Ιδέα Θα βασιστούμε στην εξής παρατήρηση: Υπάρχει ένας βέλτιστος περίπατος, ο στον οποίο για κάθε ισχυρά συνεκτική συνηστώσα του γράφου, είτε ανήκουν όλες οι κορυφές της σε αυτόν είτε καμία.

Βασική απόδειξη:

- Έστω κορυφές v_i, v_j που ανήκουν στην ίδια συνεκτική συνυστώσα, αλλά μόνο η v_i ανήκει στον βέλτιστο περίπατο.
- Αφού v_i, v_j στην ίδια συνεκτική συνηστώσα, υπάρχει περίπατος που ξεκινάει από την v_i, περνάει από την v_j και καταλήγει στην v_i.
- Αν αντικαταστίσουμε την εμφάνηση της v_i με τον παραπάνω περίπατο στον βέλτιστο περίπατο, ο gcd δεν μπορεί να αυξηθεί και η λύση παραμένει εφικτή.

Έτσι, μπορούμε να συμπτύξουμε τον αρχικό γράφο, σε έναν γράφο $G' = (V', E', c')$ στον οποίο κάθε συνεκτική συνηστώσα του G απεικονίζεται ως ένας κόμβος με βάρος τον gcd των βαρών των κόμβων του.

Λύση

- Τώρα παρατηρούμε ότι ο G' είναι ένας DAG (αφού έχουμε συμπτήξει όλες τις συνεκτικές συνηστώσες του.
- Άρα μπορούμε να βρούμε την τοπολογική του διάταξη.

- Ξεκινώντας από τον τελευταίο κόμβο στην διάταξη και πηγαίνωντας προς τον πρώτο, βρίσκουμε τα πιθανά κόστη των μονοπατιών που ξεκινάνε από αυτόν. Με δυναμικό προγραμματισμό:

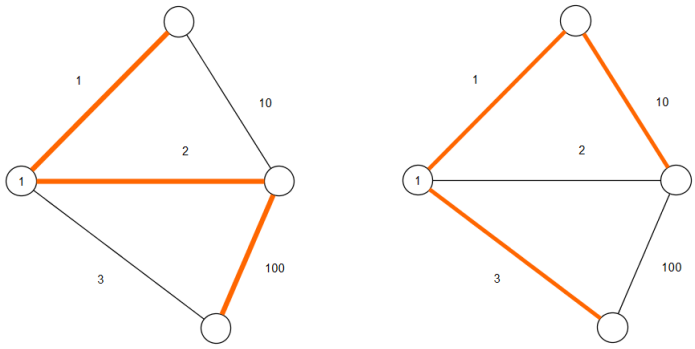
Έστω $\text{dp}[v_i]$ οι μκδ όλων των πιθανών μονοπατιών που ξεκινάνε από τον κόμβο v_i. Τότε:

$$\text{dp}[v_i] = \text{dp}[v_i].\text{append}\left(\bigcup_{(v_i,v_j)\in E', d\in \text{dp}[j]} \gcd(d, c(v_i))\right)$$

Πολυπλοκότητα Αφού ο μκδ του κάθε κόμβου v_i στο dp δεν μπορεί να είναι μεγαλύτερος από v_i έχουμε: $O(|V| + |E| \max_{v_i \in V}(c(v_i)) \log |V|)$. (αφού μπορούμε να βρούμε όλες τις ισχυρά συνεκτικές συνιστώσες σε $O(|V| + |E|)$.)

Άσκηση 4: Ελάχιστο Συνδεκτικό Δέντρο με Περιορισμούς

Ερώτημα Α: Το άπληστο κριτήριο δε λειτουργεί. Για παράδειγμα, σε αυτό το γράφημα με απαιτούμενο βαθμό 2 στον κόμβο 1 θα παίρναμε το αριστερό με βάρος 103, ενώ το ελάχιστο έχει βάρος 14.



Ερώτημα Β: Γενική Ιδέα:

- Προσθέτουμε μια σταθερά Κ σε κάθε ακμή που προσπίπτει στο σχετικό κόμβο.
- Όσο μεγαλώνει το Κ, τόσο λιγότερες ακμές περιέχει το ΕΣΔ του επαγόμενου γραφήματος του προσπίπτουν στο σχετικό κόμβο.
- Αρκεί να κάνουμε δυαδική αναζήτηση για να βρούμε τη κατάλληλη τιμή του Κ.

Πρόταση 1: Για κάθε απαιτούμενο βαθμό το απαιτούμενο ΕΣΔ με περιορισμούς υλοποιείται ως το ΕΣΔ του δέντρου επαυξημένου κατά Κ.

Πρόταση 2: Τα μοναδικά Κ στα οποία αλλάζει το ο βαθμός του απαιτούμενου κόμβου στο ΕΣΔ είναι αυτά για τα οποία μια ακμή προσπίπτουσα στον κόμβο γίνεται ίση με μια άλλη ακμή του γράφου όταν αυξηθεί κατά Κ. Δηλαδή, αρκεί να εξετάσουμε τα $K \in \{e - a : e \in E \wedge a \in A\}$ όπου E το σύνολο των διαφορετικών βαρών ακμών όλου του γραφήματος και A το σύνολο των διαφορετικών βαρών ακμών που προσπίπτουν στον απαιτούμενο κόμβο.

Απόδειξη 1: Για αρκετά αρνητικό Κ το ΕΣΔ παίρνει όλες τις σχετικές ακμές, ενώ για αρκετά θετικό παίρνει τις ελάχιστες δυνατές. Συνολικά, όλες οι τιμές ανάμεσα στην ελάχιστη και μέγιστη δυνατή επιτυγχάνονται.

Απόδειξη 2: Αφού για να βγεί από το ΕΣΔ μια ακμή πρέπει να αντικατασταθεί από μια άλλη, σημαίνει πως για κάποιο Κ γίνεται ίση μια προσπίπτουσα στον απαιτούμενο κόμβο με κάποια άλλη του γραφήματος. Αυτό μπορεί να συμβεί μόνο όταν το Κ είναι ίσο με τη διαφορά μια προσπίπτουσας ακμής και μιας άλλης ακμής του γραφήματος.

Σημείωση: Σε κάποια βήματα ενδέχεται να βγαίνουν πάνω από μια ακμές από το ΕΣΔ για το ίδιο Κ.

Αλγόριθμος: Άρχικά, ταξινομούμε το σύνολο των Κ που εξετάζουμε, δηλαδή $K \in \{e-a : e \in E \wedge a \in A\}$, το οποίο παίρνει χρόνο το πολύ $O(|E|\,|A|\log(|E|\,|A|))$.

Έπειτα, κάνουμε δυαδική αναζήτηση πάνω σε αυτό το σύνολο. Αυτό γίνεται το πολύ $O(log(|E| * |A|)) \leq O(\log(|E|\,|E|)) = O(2\log(|E|))$ φορές. Σε κάθε επανάληψη, προσθέτουμε το συγκεκριμένο Κ στις ακμές που προσπίπτουν στον απαιτούμενο κόμβο και εφαρμόζουμε Kruskal, που παίρνει $O(E\log(|E|))$ κάθε φορά. Αν ο απαιτούμενος βαθμός δεν είναι αρκετός τότε μειώνουμε το Κ και αντίστροφα.

Συνολικά, απαιτείται $O(|E|\,|A|\log(|E|) + |E|\log^2(|E|))$

Σημείωση: Φυσικά, και χωρίς να περιορίζαμε το σύνολο των Κ και απλά κάναμε δυαδική αναζήτηση στο $[-E_{max}, E_{max}]$, πάλι θα είχαμε πολυωνυμικό φράγμα στον χρόνο.

Άσκηση 5: Αλγόριθμος Boruvka

Είσοδος: Μη κατευθυνόμενο γράφημα $G(V, E, w)$, με n κορυφές, m ακμές και βάρος $w(e)$ σε κάθε ακμή $e \in \mathbb{E}$, με όλα τα βάρη διαφορετικά μεταξύ τους.

Ερώτημα Α:

Ζητούμενα: Για τον αλγόριθμο Boruvka νδο

1. Ο αλγόριθμος καταλήγει πάντα σε συνδετικό δέντρο.
2. Το συνδετικό δέντρο είναι ελαχίστου βάρους.

Ιδέα: Στον αλγόριθμο σε κάθε βήμα ενώνουμε τα συνδεδεμένα κομμάτια ενός δάσους. Για κάθε κομμάτι βρίσκουμε την προσπίπτουσα ακμή ελάχιστου κόστους. Προσθέτουμε τις ακμές στο δάσος και επαναλαμβάνουμε τη διαδικασία.

1. Έστω ότι το γράφημα που προκύπτει περιέχει κύκλο. Έστω ότι στην τρέχουσα επανάληψη έχουμε εντοπίσει τις k προσπίπτουσες ακμές ελαχίστου κόστους $e_1, e_2, \ldots, e_k$ που σχηματίζουν κύκλο και e_1 η ελάχιστη ακμή. Επομένως πρέπει να έχουμε $w(e_1) < w(e_2) < \cdots < w(e_k) < w(e_1)$ προκειμένου να διαλέξει ο αλγόριθμος και την ακμή e_k. **Άτοπο**

   Σε κάθε επανάληψη, για κάθε ακμή που επιλέγεται ενώνονται συνεκτικές συνιστώσες.

   Έτσι καταλήγουμε πάντα σε ένα συνεκτικό και ακυκλικό γράφο, δηλαδή σε ένα συνδετικό δέντρο.

2. Έστω ότι το γράφημα που προκύπτει δεν είναι είναι ελαχίστου βάρους. Αυτό σημαίνει ότι υπάρχει ένα δέντρο T (MST) που δεν χρησιμοποιεί τις ελάχιστες προσπίπτουσες ακμές σε κάθε κόμβο. Έστω λοιπόν ότι σε κάποια τομή $(S, \bar{S})$ το MST δεν περιέχει την ακμή ελαχίστου βάρους $e*$. Αν προσθέσουμε την $e*$ στο T τότε δημιουργούμε ένα κύκλο C. Υποχρεωτικά πρέπει να υπάρχει τουλάχιστον μία ακμή που διασχίζει την τομή και ανήκει στον κύκλο C. Για όλες τις ακμές $e \in C \cap (S, \bar{S}), e \neq e*$ ισχύει $w(e) > w(e*)$ καθώς όλα τα βάρη είναι μοναδικά.
Άρα αν στο δέντρο αντικαταστήσουμε την e με την $e*$ καταλήγουμε σε ένα συνδετικό δέντρο με αυστηρά μικρότερο βάρος. **Άτοπο**
Έτσι καταλήγουμε πάντα σε MST.

Ερώτημα Β:

Αλγόριθμος:

1. Ξεκινάμε με ένα δάσος από απομονωμένους κόμβους
2. Όσο υπάρχουν περισσότερα του ενός δέντρων στο δάσος, διαλέγουμε για κάθε ένα από αυτά την ακτίνα με το μικρότερο βάρος το ενώνει με άλλο δέντρο και την προσθέτουμε στο MST.
3. Η έξοδος είναι το MST.

Πολυπλοκότητα: Το βήμα 2 γίνεται σε $O(m)$, χρησιμοποιώντας δομή union-find. Η δομή θα περιέχει όλους τους κόμβους (αρχικά disjointed) και κάθε σύνολο της είναι ένα δέντρο που έχουμε φτιάξει μέχρι αυτήν την επανάληψη. Έτσι, κάνουμε για κάθε ακμή ένα find στη δομή για να βρούμε για κάθε δέντρο, την ελάχιστη που το συνδέει με άλλο δέντρο. Έπειτα, για κάθε ελάχιστη ακμή που έχουμε βρει, εκτελούμε ένα union, για να συνδέσουμε τα δυο δέντρα που συνδέει.

Τέλος, σε κάθε εκτέλεση του βήματος 3 το πλήθος των δέντρων υποδιπλασιάζεται (ενώ έχουμε ξεκινήσει αρικά με n δέντρα). Έτσι, ο αλγόριθμος θα κάνει το πολύ $\log n$ επαναλήψεις για να καταλήξει στο συνολικό spanning tree. Επομένως, συνολικά απαιτείται χρόνος $O(m \log n)$.

Ερώτημα Γ:

Ιδέα: Να τρέξουμε το βήμα 3 του αλγορίθμου για να μειώσουμε τον αριθμό των κόμβων και στη συνέχεια να τρέξουμε τον αλγόριθμο του Prim.

Έστω ότι το τρέχουμε r φορές το βήμα 3, με κόστος (mr). Μετά από αυτό ο αριθμός των κόμβων είναι το πολύ $\frac{n}{2^r}$. Άρα ο αλγόριθμος του Prim θέλει $O((n/2^r)\log(\frac{n}{2^r}))$. Συνολικά έχουμε $O(mr + (n/2^r)\log(n/2^r))$. Διαλέγουμε το $r = \log\log n$ και καταλήγουμε στο $O(m \log\log n)$

Ερώτημα Δ: Μπορούμε να εφαρμόσουμε τον αλγόριθμο Boruvka, με την εξής παραλλαγή:

Παραλλαγή Boruvka: Σε κάθε επανάληψη του αλγορίθμου κάνουμε contract τις ακμές που έχουμε επιλέξει (ως ελάχιστες που συνδέουν δυο disjoint δέντρα).

- Φροντίζουμε στην "μετατροπή του γράφου σε κανονικό" μετά από κάθε contraction, να κρατάμε τις ακμές με το ελάχιστο βάρος (όταν έχουμε πολλαπλές ακμές ανάμεσα σε δυο κόμβους).
- Η ορθότητα του αλγορίθμου διατηρείται με προφανή τρόπο, καθώς τώρα σε κάθε επανάληψη, κάθε κόμβος αντιστοιχεί σε ένα δέντρο. (Θυμόμαστε τις ακμές που έχουμε κάνει contract και τις επιστρέφουμε στο τέλος ως MST).

Πολυπλοκότητα:

- Είναι εύκολο να υλοποιήσουμε το contraction των επιλεγμένων ακμών σε $O(m)$ (χρεάζεται ένα πέρασμα από όλες τις ακμές για να διαλέξουμε τις ελάχιστες σε κάθε multi-edge).
- Αν αρχικά έχουμε m ακμές ξέρουμε ότι στο βήμα i θα υπάρχουν το πολύ $m/2^{2i}$ ακμές.
- Λόγω της κλειστότητας όμως, γνωρίζουμε ότι για κάθε γράφο που προκύπτει σε κάθε βήμα i, με πλήθος κόμβων n_i ισχύει $m_i = O(n_i)$.
- Συνεπώς, η πολυπλοκότητα προκύπτει από την αναδρομική σχέση:

$$T(V) = T\big(n/2\big) + O(n) \Rightarrow T(n) = O(n)$$

- Προσέξτε ότι στη γενική περίπτωση αυτή η παραλλαγή χρειάζεται χρόνο $O(n^2)$, καθώς το m θα μπορούσε να φθάνει μέχρι και n^2.

1 Άσκηση 6: Το Σύνολο των Συνδετικών Δέντρων

Ερώτημα Α:

- Έστω T_1, T_2 δύο διαφορετικά συνδετικά δετρα και $e \in T_2 \setminus T1$ ($T_2 \setminus T1 \neq \emptyset$)
- Άρα το $T_1 \cup \{e\}$ περιέχει κύκλο C, και υπάρχει ακμή $e' \in C$, $e' \notin T_2$ (Αλλιώς το T_2 θα έπρεπε να περιέχει τον C)
- Άρα το $(T_1 \setminus \{e\}) \cup \{e'\}$ είναι άκυκλο με $n-1$ ακμές $\Rightarrow$ Συνδετικό δέντρο.

Εύρεση e': Εστω T_1, T_2 και $e = \{u, v\}$.

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

Ερώτημα Β:

Ιδιότητα: Έστω $T_1, T_2 \in H$ και $d_H(T_1, T_2)$ το μήκος του συντομότερο μονοπατιού μεταξύ T_1, T_2 στο H. Τότε $|T_1 \setminus T_2| = k$ ανν $d_H(T_1, T_2) = k$.

Απόδειξη: Θα χρησιμοποιήσουμε επαγωγή στο μήκος του μονοπατιού:

- Επαγωγική Βάση: Από ορισμό του H, $|T_1 \setminus T_2| = 1$ ανν $d_H(T_1, T_2) = 1$.
- Επαγωγική Υπόθεση: $|T_1 \setminus T_2| = k$ ανν $d_H(T_1, T_2) = k$.
- Επαγωγική Βήμα: Θ.δ.ο. $|T_1 \setminus T_2| = k+1$ ανν $d_H(T_1, T_2) = k+1$

Απόδειξη Επαγωγικού Βήματος:

- $\Rightarrow$ Εστω $e \in T_1 \setminus T_2$.
- Λόγω (α), υπαρχει $e' \in T_2 \setminus T_1$ τ.ω. $T_1' = (T_1 \setminus \{e\}) \cup \{e'\} \in H$.

- Αλλά, $|T_1' \setminus T_2| = k \Rightarrow_{\text{E.Y.}} d_H(T_1, T_2) \leq k + 1$.
- Από E.Y. αν $d_H(T_1, T_2) = k$ τότε $d_H(T_1, T_2) = k$. Οπότε, αναγκαστικά $|T_1 \setminus T_2| = k + 1$.
- Αρα, $d_H(T_1, T_2) \geq k + 1 \Rightarrow d_H(T_1, T_2) = k + 1$.
- $\Leftarrow$ Εφόσον $d_H(T_1, T_2) = k + 1$ υπάρχει $T_1' \in H$ τ.ω. $d_H(T_1, T_1') = 1, d_H(T_1', T_2) = k$.
- Από E.Y. $|T_1' \setminus T_2| = k$.
- Άρα, $|T_1 \setminus T_2| = k - 1$ ή $|T_1 \setminus T_2| = k + 1$.
- Αν $|T1 \setminus T_2| = k - 1 \Rightarrow_{\text{E.Ψ.}} d_H(T_1, T_2) = k - 1$, άτοπο

Υπολογισμός Συντομότερου Μονοπατιού:

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

Ορθότητα: Αν $T_2 \setminus T_1 = k$ τότε σε k βήματα έχουμε μετατρέψει το T_1 σε T_2. Αρα, έχουμε βρει μονοπάτι στο H μήκους $k \Rightarrow$ Το συντομότερο μονοπάτι.

Πολυπλοκότητα: Σε $O(|V|)$ αποθηκεύουμε το T_1 ως $rooted$ δέντρο (κάθε κόμβος δείχνει στον πατέρα του). Για κάθε ακμή $e \in T_2$ έλεγχουμε σε $O(1)$ αν $e \in T_1$. Αρα, $O(|V|)$. Κάνουμε k ανανεώσεις ακμών σε $O(|V|)$ βήματα. Σύνολο, $O(k \cdot |V|)$.

Ερώτημα Γ:

Ιδέα: Θα κατασκευάσουμε μια φορά το MST και για κάθε ακμή που δεν ανήκει σε αυτό, θα βρίσκουμε το MST που την περιέχει βάση αυτού. Συγκεκριμένα, είναι εύκολο να δούμε ότι για μια ακμή $e \in E$ που δεν ανήκει στο MST, αν την προσθέσουμε στο MST και αφαιρέσουμε τη βαρύτερη ακμή στον κύκλο που θα δημιουργηθεί (που περιλαμβάνει την e) έχουμε το ζητούμενο MST που περιέχει την e (αυτό είναι εύκολο να το δουμε, αν φανταστούμε πχ την συμπεριφορά του Kruskal σε περίπτωση που η e π.χ. είχε βάρος οριακά μεγαλύτερο από αυτό της δεύτερης μεγαλύτερης ακμής του κύκλου) .

Έτσι, αρκεί να βρούμε ένα MST το βάρους (w_{mst}), που είναι το ζητουμενο για τις ακμές που περιέχει, και για τις υπόλοιπες ακμές να έχουμε έναν τρόπο να βρούμε τη μέγιστη ακμή στον κύκλο που δημιουργούν και να προσθέσουμε τη διαφορά των βαρών τους στο w_{mst} για να πάρουμε το ζητούμενο βάρος του συνδετικού τους δέντρου.

Λύση: Αφού μπορούμε να υπολογίσουμε το MST T^* σε $O(m \log m)$, χρησιμοποιώντας τον Kruskal, μένει να βρούμε έναν γρήγορο τρόπο για να βρίσκουμε τον κύκλο που δημιουργείται για κάθε ακμή $e = \{u, v\}$ που προσθέτουμε (για την ακρίβεια την βαρύτερη ακμή στο μονοπάτι που συνδέει τις κορυφές u και v στο T^*).

Η προφανής λύση είναι για κάθε ακμή $e = \{u, v\}$, να τρέχουμε ένα DFS από πχ την v, μέχρι να βρούμε το μονοπάτι από την v στην u στο T^* (και άρα και το βάρος της βαρύτερης ακμής σε αυτό το μονοπάτι). Αν θέλαμε το κάνουμε αυτό για κάθε ακμή, θα είχαμε συνολική πολυπλοκότητα $\Theta(mn)$. Η πολυπλοκότητα αυτή μπορεί να βελτιωθεί σε $\Theta(n^2)$, αν με ένα DFS από κάθε κορυφή $v \in V$, υπολογίζαμε (και αποθηκεύαμε) το βάρος της βαρύτερης ακμής στο μονοπάτι $v - u$ στο T^*, για κάθε άλλη κορυφή u. Για περισσότερες λεπτομέρειες, δείτε την 6η Άσκηση, στην 3η σειρά προτεινόμενων ασκήσεων http://old.corelab.ntua.gr/courses/algorithms/problemsets/set03.pdf

Bonus: Για να το κάνουμε καλύτερα παρατηρούμε, ότι για (οποιαδήποτε) ρίζα (v_s) του MST T^*, αν v_a ο ελαχίστου ύψους κοινός πρόγονος των v και u (LCA - στο MST, με ρίζα το v_s) ο κύκλος αποτελείται (εκτός από την $e = \{u, v\}$) από τις ακμές που ανήκουν στα μονοπάτια από την v στην v_s και από τον v_s στην u (προφανώς ο LCA ανήκει στο μοναδικό $v - u$ μονοπάτι στο T^*).

Έτσι, θα ακολουθήσουμε μια λογική παραλλαγής του LCA, ώστε να βρίσκουμε γρήγορα την ακμή μέγιστου βάρους στα μονοπάτια από την v στην v_a και από την u στην v_a για κάθε $v, u \in V$ με $\text{LCA}(v, u) = v_a$. Πιο συγκεκριμένα, στην αντίστοιχη sparse table δομή που θα χρησιμοποιούσαμε για τον LCA, θα προ-αποθηκεύσουμε, στο πεδίο ancest$[v, k]$, για κάθε κορυφή v και για κάθε $k = 0, 1, \ldots, \lceil \log n \rceil$, την "προ-γονική" κορυφή της κορυφής v που βρίσκεται 2^k επίπεδα ψηλότερα στο δέντρο.

Μπορούμε να το κάνουμε αυτό με ένα DFS στο T^* (από αυθαίρετη αρχική κορυφή v_s, για το οποίο κρατάμε τους χρόνους πρώτης και τελευταίας επίσκεψης, οι οποίοι μας επιτρέπουν να απαντάμε ερωτήματα "είναι η κορυφή v πρόγονος της κορυφής u στο T^* με ρίζα v_s;" σε σταθερό χρόνο. Έχοντας αυτή τη δυνατότητα, μπορούμε να αρχικοποιήσουμε το sparse table ancest σε χρόνο $\Theta(n \log n)$ με βάση την παρακάτω αναδρομική σχέση:

$$\text{ancest}[v, k+1] = \text{ancest}\big[\text{ancest}[v][k], k\big]$$

Επιπλέον, κρατάμε, στο maxe$[v, k]$, για κάθε ζεύγος $(v, \text{ancest}[v, k])$ το βάρος της μεγίστου βάρους ακμής στο μονοπάτι από την v στην ancest$[v, k]$. Με δεδομένο το sparse table ancest, μπορούμε να αρχικοποιήσουμε το sparse table maxe σε χρόνο $\Theta(n \log n)$ ως εξής:

$$\text{maxe}[v, k+1] = \max\Big\{\text{maxe}[v, k], \text{maxe}\big[\text{ancest}[v, k], k\big]\Big\}$$

Στην πραγματικότητα οι δύο πίνακες ancest και maxe μπορούν να υπολογιστούν ταυτόχρονα.

Αυτό που μένει τώρα, είναι για κάθε ακμή $e = \{v, u\}$, να βρίσκουμε τον $\text{LCA}(v, u) = v_a$ γρήγορα και μετά να πέρνουμε τη μεγαλύτερη μεγίστου βάρους ακμή ανάμεσα στις δυο στα μονοπάτια από την v στην v_a και από την u στην v_a. Προσέξτε ότι κάθε μήκους μονοπάτι "σπάει" σε μονοπάτια με μήκη δυνάμεις του δυο, συνεπώς αν έχουμε τους πίνακες ancest και maxe, μπορούμε να υπολογίσουμε το βάρος της βαρύτερης ακμής στο $v - u$ μονοπάτι στο MST T^* σε χρόνο $\Theta(\log n)$, με *binary lifting* – ουσιαστικά δυαδική αναζήτηση στο μήκος των μονοπατιών $v - v_a$ και $u - v_a$.

Γενικώς, για δοσμένες v και u, κοιτάμε αν η v είναι πρόγονος της u ή αντίστροφα (σε σταθερό χρόνο, έχοτας διατηρήσει το DFS traversal). Αν ναι, αυτή που είναι πρόγονος είναι και ο LCA τους. Αλλιώς "ανεβαίνουμε" τους προγόνους της v μέχρι να βρούμε το μεγαλύτερο k για το οποίο ο $v_{a-1} =$ ancest$[v, k]$ δεν είναι πρόγονος της u, αλλά ο ancest$[v, k+1] =$ ancest$[v_{a-1}, k]$ είναι. Αντίστοιχα κάνουμε από την "πλευρά" της u. Και συνεχίζουμε αναδρομικά, μειώνοντας κάθε φορά το k.

Πολυπλοκότητα: $O(m \log m)$ για τον Kruskal, $O(n)$ για το DFS, $(n \log n)$ για την αρχικοποίηση των πινάκων ancest και maxe, και $O(m \log n)$ για όλα τα queries αντικατάστασης ακμών. Συνολικά $O(m \log m)$.