ΑΛΓΟΡΙΘΜΟΙ ΚΑΙ ΠΟΛΥΠΛΟΚΟΤΗΤΑ

2η Σειρά Ασκήσεων

Ακαδημαϊκό έτος 2021-2022

Κοκκινάκης Παναγιώτης – Α.Μ.: 03118115

Άσκηση 1: Δίσκοι και σημεία

Θα χρησιμοποιούμε την τεχνική του «Διαίρει και Βασίλευε» για να βρούμε τον ελάχιστο αριθμό κύκλων με τον οποίο συμπεριλαμβάνουμε όλα τα σημεία. Αρχικά χωρίζουμε την ευθεία σε δύο ημιευθείες, ώστε σε κάθε μία να αντιστοιχούν περίπου οι μισές προβολές των σημείων. Αν θεωρήσουμε ότι γνωρίζουμε τον ελάχιστο αριθμό κύκλων ακτίνας r που απαιτούνται για να συμπεριληφθούν όλα τα σημεία για κάθε ημιευθεία από τις 2, τότε αρκεί να μελετήσουμε τι συμβαίνει στην ένωση των 2, για να βγάλουμε συμπέρασμα και για ολόκληρη την ευθεία. Αυτό γίνεται ως εξής: Ελέγχουμε κάθε κύκλο με κέντρο που έχει μέγιστη οριζόντια απόσταση r δεξιά και αριστερά του σημείου διαχωρισμού. Για κάθε έναν από αυτούς του κύκλους (θεωρώντας ότι οι κύκλοι του δεξιού μέρους, αν το κέντρο τους είναι σε απόσταση < r από το σημείο διαχωρισμού, καλύπτουν και τα σημεία του αριστερού και αντιστρόφως) ελέγχουμε τα σημεία που περιέχει και αν όλα εξ αυτών περιέχονται σε δύο κύκλους αφαιρούμε τον κύκλο αυτό από τη λύση.

Για να καταλήξουμε κάθε φορά στη συγχώνευση των δύο μερών πρέπει να βρούμε την βέλτιστη λύση για κάθε ένα από αυτά τα μέρη. Για να το επιτύχουμε αυτό χωρίζουμε σε όλο και μικρότερα μέρη μέχρι να καταλήξουμε σε κάποιο κομμάτι να έχουμε έναν κύκλο, που θα περιέχει ένα σημείο το κέντρο του οποίου είναι στην προβολή του εν λόγω σημείου. Από εκεί ξεκινάμε τη συγχώνευση με την διαδικασία που περιγράψαμε παραπάνω.

Για να βρούμε την συνολική πολυπλοκότητα του αλγορίθμου θα βρούμε την πολυπλοκότητα των επί μέρους βημάτων του. Στο βήμα του χωρισμού σε διαφορετικά μέρη αν κάθε μέρος έχουμε πολυπλοκότητα O(nlogn) με δυαδική αναζήτηση στον αριθμό των προβολών, όπου n ο αριθμός των σημείων μας. Στο βήμα της συγχώνευσης απαιτείται στη χειρότερη περίπτωση ο έλεγχος n σημείων, (αν για κάθε σημείο που ελέγχουμε καταγράφουμε σε ποιους κύκλους ανήκει, ώστε να μην έχουμε επαναλήψεις ελέγχων) επομένως η πολυπλοκότητα είναι O(n).

Συνεπώς ισχύει: $T(n) \leq 2T(n/2) + O(nlogn) \Rightarrow T(n) = O(nlog^2n)$ (από Μ.Τ).

Άσκηση 2: Παραλαβή πακέτων

Α) Θα πρέπει να ταξινομήσουμε τους πελάτες ανάλογα του λόγου $\frac{βάρος}{χρόνος\ προετοιμασίας}$ σε φθίνουσα σειρά ώστε να προκύψει ταξινόμηση $\frac{w1}{p1} \geq \frac{w2}{p2} \geq \frac{w3}{p3} \geq \cdots \geq \frac{wn}{pn}$. Θεωρούμε ότι αυτή είναι η βέλτιστη ακολουθία εξυπηρέτησης πελατών $\{c_1,c_2,c_3,\ldots,c_n\}$.

Απόδειξη ορθότητας:

Έστω βέλτιστη μετάθεση των πελατών $\{v_1,v_2,v_3,\ldots,v_n\}$, διαφορετική από την μετάθεση $\{c_1,c_2,c_3,\ldots,c_n\}$ που προέκυψε από την ταξινόμηση. Αυτό σημαίνει ότι υπάρχει πελάτης, έστω στη θέση i, που εξυπηρετείται στην βέλτιστη μετάθεση μεταγενέστερα σε σχέση με τη μετάθεση μας, οπότε $c_i = v_j$, $i < j$. Θα συγκρίνουμε το βεβαρυμμένο χρόνο εξυπηρέτησης της λύσης αυτής με των μεταθέσεων που προκύπτουν αν ανταλλάξουμε τον πελάτη v_j με τους ενδιάμεσους [i, j-1], ώστε να έρθει στην μορφή της ταξινομημένης μετάθεσης μας.

Ισχύει από την υπόθεση μας για την ταξινόμηση ότι $\forall\ v_k : k \geq i \Rightarrow w_{ck}/p_{ck} \leq w_{cj}/p_{cj}$.

Με επιχείρημα ανταλλαγής για τους πελάτες v_j, v_{j-1} ο βεβαρυμμένος χρόνος εξυπηρέτησης γίνεται:

$T = p_1(w_1+\ldots+w_n)+p_2(w_2+\ldots+w_n)+\ldots p_j(w_j+w_{j-1}+\ldots+w_n)+p_{j-1}(w_{j-1}+w_{j+1}\ldots+w_n)-$
$+p_{j+1}(w_{j+1}+\ldots+w_n)+p_nw_n$

Έστω T' ο βεβαρυμμένος χρόνος εξυπηρέτησης της βέλτιστης μετάθεσης v:

$T' = p_1(w_1+\ldots+w_n)+p_2(w_2+\ldots+w_n)+\ldots p_{j-1}(w_{j-1}+w_j+\ldots+w_n)+p_j(w_j+\ldots+w_n)+p_{j+1}(w_{j+1}+\ldots+w_n)+p_nw_n$

Για τη διαφορά T-T' έχουμε ότι:

$T-T' = p_j(w_j+w_{j-1}+\ldots w_n)+p_{j-1}(w_{j-1}+w_{j+1}+\ldots+w_n) - p_{j-1}(w_{j-1}+w_j+\ldots+w_n) - p_j(w_j+\ldots+w_n) =$

$= p_jw_{j-1} - p_{j-1}w_j \leq 0$ αφού $w_{j-1}/p_{j-1} \leq w_j/p_j$ όπως είπαμε παραπάνω.

Αν συνεχίσουμε με την ίδια διαδικασία ανταλλάσσοντας τη θέση του v_j μέχρι να φτάσει στην i-οστή θέση της βέλτιστης μετάθεσης, ο χρόνος θα μειώνεται με κάθε βήμα. Επομένως η βέλτιστη μετάθεση είναι η αρχική που θεωρήσαμε για τους ταξινομημένους ,σύμφωνα με το λόγο $\frac{βάρος}{χρόνος\ προετοιμασίας}$, αφού όποια και να υποθέσουμε ότι είναι optimal εκτός αυτής, θα καταλήξουμε με τις ανταλλαγές να βρίσκουμε ότι έχει μικρότερο βεβαρυμμένο χρόνο εξυπηρέτησης.

Β) Θα χρησιμοποιήσουμε δυναμικό προγραμματισμό. Θεωρούμε όπως στο ερώτημα (Α) ότι έχουμε μετάθεση πελατών, όπου $\frac{w1}{p1} \geq \frac{w2}{p2} \geq \frac{w3}{p3} \geq \cdots \geq \frac{wn}{pn}$. Η αναδρομική σχέση που μας δίνει τον ελάχιστο βεβαρυμμένο χρόνο εξυπηρέτησης είναι η εξής:

$$WT(i,p) = min\{\begin{matrix} WT(i+1, p+p_i) + w_i(p+p_i) \\ WT(i+1,p) + (\sum_{j=0}^{i}(p_j - p))w_i \end{matrix}\}$$

όπου i ο αριθμός του πελάτη που παίρνουμε από τη μετάθεση μας, p ο συνολικός χρόνος εξυπηρέτησης πελατών για τον εξυπηρετητή 1 και WT(i,p) δηλώνει τον ελάχιστο βεβαρυμμένο χρόνο που απαιτείται για την ολοκλήρωση της διαδικασίας αν έχουμε εξυπηρετήσει μέχρι τον i-οστό πελάτη, και αν ο πρώτος εξυπηρετητής έχει συνολικό χρόνο εξυπηρέτησης p. Οι αρχικές συνθήκες είναι WT(n,_) = 0 αφού έχουμε εξυπηρετήσει όλους τους πελάτες, ενώ η ζητούμενη τιμή βεβαρυμμένου χρόνου είναι η WT(0,0), όπου δεν έχουμε ξεκινήσει την εξυπηρέτηση τόσο για τον εξυπηρετητή 1 όσο και γενικότερα.

Τα δύο σκέλη της αναδρομικής σχέσης προκύπτουν ως εξής:

- το πρώτο που ισούται με $WT(i+1,p+p_i) + w_i(p+p_i)$ είναι για την περίπτωση που ο i-οστός πελάτης εξυπηρετείται από τον εξυπηρετητή 1 οπότε προστίθεται ο όρος $w_i(p+p_i)$ (το βάρος του i-οστού πελάτη επί τον συνολικό χρόνο αναμονής του στην ουρά του εξυπηρετητή 1) στον βεβαρυμμένο χρόνο για την εξυπηρέτηση των υπόλοιπων πελατών δεδομένο ότι ο i ανήκει πλέον στο σύνολο p.
- το δεύτερο που ισούται με $WT(i+1,p) + (\sum_{j=0}^{i}(pj - p)wj)$ είναι για την περίπτωση που ο i-οστός πελάτης εξυπηρετείται από τον εξυπηρετητή 2, οπότε προστίθεται ο όρος $(\sum_{j=0}^{i}(pj - p)wj)$ (το βάρος του i-οστού πελάτη επί τον συνολικό χρόνο αναμονής του στην ουρά του εξυπηρετητή 2) στον βεβαρυμμένο χρόνο για την εξυπηρέτηση των υπόλοιπων πελατών δεδομένο ότι ο i δεν ανήκει στο σύνολο p.

Για να μην υπολογίζουμε πάνω από μία φορά τιμές της WT(_,_), τις αποθηκεύουμε σε πίνακα διαστάσεων $n*P_{max}$ (όπου $P_{max}=\sum_{j=0}^{i} p_i$).

Η συνολική πολυπλοκότητα του αλγορίθμου είναι $O(nlogn+n*P_{max})$ (nlogn για την ταξινόμηση και $n*P_{max}$ για την αναδρομή εφόσον υπολογίζουμε κάθε τιμή της WT μια φορά).

Η γενική σχέση για m εξυπηρετητές θα πρέπει να εξετάζει τις περιπτώσεις που ο i-οστός πελάτης εξυπηρετείται από κάθε έναν από τους εξυπηρετητές, επομένως παίρνουμε την παρακάτω σχέση:

$$T(i,P) = min\begin{cases} T(i+1,\ P1+p_i,\ P2,\ \ldots,\ Pm-1) + & w_i\,(P1+\ p_i) \\ T\Big(i+1,\ P1,\ P2\ +\ p_i,\ \ldots,\ Pm-1\Big) + & w_i\Big(P2+\ p_i\Big) \\ \cdots & \cdots \\ T\Big(i+1,\ P1,\ P2,\ \ldots,\ Pm-1\Big)\ + & w_i\left(\sum_{j=0}^{i} p_j - \sum_{j=0}^{m-1} P_m\right) \end{cases}$$

m-1 φορές

Στην περίπτωση των m εξυπηρετητών ο πίνακας μας θα έχει διαστάσεις $n*P_{max}*P_{max}*\ldots*P_{max}$ και συνολική πολυπλοκότητα $O(n\log n + n*P_{max}^{m-1})$ (η πολυπλοκότητα της αναδρομής αυξάνεται εκθετικά).

Άσκηση 3: Τοποθέτηση στεγάστρων (και Κυρτό Κάλυμμα)

Α) Θα χρησιμοποιήσουμε αναδρομική σχέση. Ξεκινάμε ,ελέγχοντας από το τελευταίο σημείο, και ελέγχουμε τα ελάχιστα κόστη για κάθε πιθανό στέγαστρο που καταλήγει στο σημείο αυτό, δουλεύοντας με τον ίδιο τρόπο και για το υπόλοιπο διάστημα. Ο τύπος που μας δίνει για τα i πρώτα σημεία το ελάχιστο κόστος είναι ο εξής:

$$min\_cost(i) = \min_{(0 \le j \le i)} \{min\_cost(j-1)+(x_i-x_j)^2+C\}$$

όπου min_cost(-1) = 0 (η ανέγερση στεγάστρου για κανένα σημείο), επομένως το συνολικό ελάχιστο κόστος για την ανέγερση των στεγάστρων που θα καλύπτουν όλα τα ζητούμενα σημεία θα είναι ίσο με min_cost(n).

Θα χρησιμοποιήσουμε δυναμικό προγραμματισμό, επειδή έχουμε επαναλαμβανόμενα στιγμιότυπα. Η πολυπλοκότητα του αλγορίθμου με τη χρήση dp είναι $\Theta(n^2)$.

Β) Η παραπάνω σχέση για το ελάχιστο κόστος μπορεί να γραφεί ως εξής:

$$min\_cost(i) = \min_{(0 \le j \le i)} \{min\_cost(j-1)+(x_i-x_j)^2+C\} = \min_{(0 \le j \le i)} \{min\_cost(j-1)+x_i^2+x_j^2-2x_ix_j + C\}$$

$$min\_cost(i) = x_i^2 + C + \min_{(0 \le j \le i)} \{min\_cost(j-1)+x_j^2-2x_ix_j\}$$

Συμπεραίνουμε συνεπώς ότι οι όροι $x_i^2 + C$ προστίθενται σε κάθε περίπτωση και δεν παίζουν ρόλο στην επιλογή του ελαχίστου. Εάν θεωρήσουμε ότι $b[j] = min\_cost(j-1)+x_j^2$ και $a[j] = -2x_i$ καταλήγουμε στη σχέση: $min\_cost(i) = x_i^2 + C + \min_{(0 \le j \le i)} \{a[j]x_j+b[j]\}$. Επομένως, το πρόβλημα του ερωτήματος (Α) μετατρέπεται στο πρόβλημα του ερωτήματος (Β).

Για τη λύση του προβλήματος των ευθείων, θα χρησιμοποιήσουμε το κυρτό περίβλημα του συνόλου των ευθειών, για το οποίο πρέπει να διατηρούμε την πληροφορία του από ποιες ευθείες αποτελείται.

Για την προσθήκη μιας ευθείας στο κυρτό περίβλημα εξετάζουμε συγκεκριμένες περιπτώσεις. Οι ευθείες μας δίνονται σε φθίνουσα ακολουθία, οπότε γνωρίζουμε ότι κάθε ευθεία που εξετάζουμε θα πρέπει να προστεθεί σε κάποιο σημείο στο κυρτό περίβλημα, αφού έχει πιο αρνητική κλίση από όλες τις προηγούμενες, οπότε υπάρχει x τέτοιο ώστε για κάθε x’ > x, η καινούργια ευθεία δίνει μικρότερες τιμές από όλες τις υπάρχουσες. Για να βρούμε που προσθέτουμε την νέα ευθεία, ξεκινάμε από την τελευταία ευθεία και ελέγχουμε το σημείο τομής της με την νέα. Αν το σημείο αυτό βρίσκεται σε τιμή του y χαμηλότερη από αυτή του σημείου τομής τελευταίας και προτελευταίας ευθείας του κυρτού περιβλήματος, τότε απλά προσθέτουμε την καινούργια ευθεία

στην τελευταία θέση του κυρτού περιβλήματος. Σε άλλη περίπτωση αφαιρούμε την τελευταία ευθεία και επαναλαμβάνουμε μέχρι να ισχύει η πρώτη περίπτωση.

Η πολυπλοκότητα της διαδικασίας αυτής είναι O(n), καθώς ,παρότι ο συνολικός αριθμός ελέγχων που μπορεί να χρειαστεί μια ευθεία για να προστεθεί, είναι της τάξης του n, οι επόμενες προσθήκες θα κάνουν πολύ λιγότερους ελέγχους (πολυπλοκότητας O(1)), αφού αφαιρούνται οι ευθείες που δεν ανήκουν στο περίβλημα. Επομένως η συνολική πολυπλοκότητα όλων των εισαγωγών είναι O(n) και η πολυπλοκότητα μίας προσθήκης είναι O(1) (λόγω amortization).

Παρομοίως δουλεύουμε και για τα k σημεία, ξεκινώντας αφετηριακά από την πρώτη ευθεία του κυρτού περιβλήματος, που βρήκαμε προηγουμένως, και ελέγχοντας μία-μία για να βρούμε σε ποια περιοχή-ευθεία ανήκει το συγκεκριμένο σημείο. Όπως και προηγουμένως, έτσι και τώρα επειδή τα σημεία δίνονται σε αύξουσα σειρά (όχι φθίνουσα όπως πριν), μπορούμε να ξεκινάμε την αναζήτηση από την ευθεία του προηγούμενου στοιχείου, αφού όλες οι προηγούμενες δεν θα ικανοποιούν ούτε αυτό, αφού δεν ικανοποιούσαν το πρότερο του. Έτσι η πολυπλοκότητα εύρεσης για κάθε σημείο θα είναι amortized O(1) και η συνολική πολυπλοκότητα εύρεσης για τα k σημεία θα είναι O(k).

Η συνολική πολυπλοκότητα για την δημιουργία του κυρτού περιβλήματος και την εύρεση της ευθείας για κάθε σημείο θα είναι O(n+k).

Το πρόβλημα του ερωτήματος (A) μπορεί να αναχθεί στο πρόβλημα του ερωτήματος (B), αφού τα σημεία x_i δίνονται σε αύξουσα ακολουθία, οπότε και η τιμή των a[j] είναι σε φθίνουσα ακολουθία.

Άσκηση 4: Δρομολόγια λεωφορείων στην Εποχή του Κορονοϊού

A) Θα χρησιμοποιήσουμε προσέγγιση δυναμικού προγραμματισμού για να ελαχιστοποιήσουμε το συνολικό δείκτη ευαισθησίας. Θεωρούμε αναδρομική σχέση SI(i,j), όπου SI(i,j) δηλώνει τον ελάχιστο συνολικό δείκτη ευαισθησίας αν χωρίσουμε τους πρώτους j μαθητές σε i λεωφορεία:

$$SI(i,j) = \min_{0<k<j}(SI(i-1),k) + \sum_{a=k+1}^{j}\sum_{b=a+1}^{j} A_{ab})$$

Η λύση προκύπτει από την τιμή του SI για i = k και j= n. Ξεκινάμε από το τέλος και βρίσκουμε ποιος φοιτητής πρέπει να τοποθετηθεί στο τελευταίο λεωφορείο. Οι αρχικές συνθήκες είναι SI(_,0) = 0.

Για να λυθεί η παραπάνω αναδρομική σχέση παράγονται n*k υποπροβλήματα, για το καθένα εκ των οποίων πρέπει να υπολογιστεί ο ελάχιστος δείκτης ευαισθησίας για το πολύ n φοιτητές. Ακόμη πρέπει να υπολογίσουμε το διπλό άθροισμα για ποσότητες της τάξης του O(n). Επομένως η συνολική πολυπλοκότητα του αλγορίθμου είναι $O(kn^4)$.

Μπορούμε να μειώσουμε αυτήν την πολυπλοκότητα αν προϋπολογίσουμε τα αθροίσματα και τα αποθηκεύσουμε, ώστε να μη τα υπολογίζουμε παραπάνω από μία φορά. Το διπλό άθροισμα μας δίνει τον δείκτη ευαισθησίας για ένα λεωφορείο που έχει τους φοιτητές από k έως j.

Θα χρησιμοποιήσουμε έναν βοηθητικό πίνακα S n*n διαστάσεων. Κάθε γραμμή του πίνακα αυτού είναι ίση με τον πίνακα prefix του A, όπου S(i,j) η επίδραση του j-οστού φοιτητή σε ένα λεωφορείο που έχουμε φοιτητές από τον 0 έως τον i. Για τον υπολογισμό ενός prefix array για n στοιχεία χρειαζόμαστε γραμμικό χρόνο, επομένως συνολικά ο υπολογισμός του S έχει χρονική πολυπλοκότητα $O(n^2)$. Από τον πίνακα αυτόν θα υπολογίσουμε το διπλό άθροισμα $C(k,j) = \sum_{a=k+1}^{j} \sum_{b=a+1}^{j} A_{ab}$ που μας δίνει τον δείκτη για λεωφορείο που μεταφέρει τους φοιτητές από k έως j. Ο δείκτης του λεωφορείου αυτού ισούται με τον δείκτη λεωφορείου που μεταφέρει φοιτητές από k-1 έως j (C(k-1,j)), αν αφαιρέσουμε την επίδραση του φοιτητή k-1 στο παραπάνω λεωφορείο. Συνεπώς αφαιρούμε την επίδραση του φοιτητή k-1 για λεωφορείο που περιέχει τους πρώτους j φοιτητές, (δηλαδή το S(j,k-1)) και προσθέτουμε την επίδραση του για λεωφορείο που περιέχει τους πρώτους k-1 φοιτητές (δηλαδή το S(k-2,k-1)(η επίδραση αυτή περιέχεται στο S(j,k-1) αλλά δεν συμμετέχει στον δείκτη C(k-1,j), οπότε πρέπει να αφαιρείται). Η πρώτη σειρά του πίνακα C είναι ίση με την πρώτη σειρά του πίνακα S. Από τα παραπάνω προκύπτει η παρακάτω αναδρομική σχέση:

$$C(k,j) = \sum_{a=k+1}^{j} \sum_{b=a+1}^{j} A_{ab} = \begin{cases} S(0,j)\,, i = 0 \\ C(k-1,j) - S(j,k-1) + S(k-2,k-1) \end{cases}$$

Αν ξεκινήσουμε από την πρώτη σειρά του πίνακα, υπολογίζουμε κάθε τιμή σε σταθερό χρόνο O(1). Επομένως συνολικά ο υπολογισμός και των δύο πινάκων έχει χρονική πολυπλοκότητα $O(2n^2) = O(n^2)$ για όλα τα στοιχεία τους. Γνωρίζουμε πλέον τα διπλά αθροίσματα και μπορούμε να έχουμε πρόσβαση σε αυτά σε σταθερό χρόνο, συνεπώς η συνολική πολυπλοκότητα του αλγορίθμου μειώνεται και γίνεται $O(kn^2)$.

Άσκηση 5: Το Σύνολο των Συνδετικών Δέντρων

Α) Θα χρησιμοποιήσουμε εις άτοπον απαγωγή. Υποθέτουμε αρχικά ότι για κάποια ακμή e που ανήκει στο συνεκτικό δέντρο T_1 αλλά όχι στο συνεκτικό δέντρο T_2, δεν υπάρχει καμία ακμή e' που ανήκει στο T_2 και όχι στο T_1 τέτοια ώστε να ικανοποιείται η σχέση του ζητουμένου. Θεωρούμε τότε το δέντρο $T_1 \setminus e$. Το δέντρο αυτό θα έχει δύο συνεκτικές συνιστώσες οι οποίες δε συνδέονται μεταξύ τους. Εφόσον δεν υπάρχει ακμή e' ώστε $(T_1 \setminus e) + e'$ να είναι συνεκτικό δέντρο, συμπεραίνουμε ότι η e πρέπει να είναι γέφυρα του γράφου G, αφού εάν δεν ήταν τότε θα υπήρχε κι άλλη ακμή e' που να συνδέει τις δύο συνιστώσες. Γνωρίζουμε όμως ότι σε ένα συνεκτικό δέντρο πρέπει να περιέχεται κάθε γέφυρα αλλιώς ο γράφος που προκύπτει δεν είναι πλήρως συνεκτικός. Επομένως θα έπρεπε η ακμή e να ανήκει στο δέντρο T_2 που είναι άτοπο από την υπόθεση μας.

Αλγόριθμος: για την εύρεση της ακμής e', με δεδομένα τα T_1, T_2 και e, θα χρησιμοποιήσουμε αρχικά αλγόριθμο για να βρούμε τις δύο συνεκτικές συνιστώσες που προκύπτουν από την αφαίρεση της ακμής e από το δέντρο T_1 [1]. Ο αλγόριθμος αυτός έχει πολυπλοκότητα O(V+E), όμως επειδή έχουμε γράφο που προέκυψε από την αφαίρεση μίας ακμής από συνεκτικό δέντρο θα ισχύει E = V-2, οπότε η πολυπλοκότητα είναι τελικά O(V). Στη συνέχεια ελέγχουμε σε χρόνο O(E) μία-μία τις ακμές του T_2 για να δούμε αν κάποια από αυτές ενώνει τις δύο συνιστώσες. Επομένως ο αλγόριθμος μας έχει πολυπλοκότητα O(V).

Β) Έστω T_1 και T_2 δύο αυθαίρετα ,διαφορετικά συνεκτικά δέντρα του G. Από το ερώτημα (Α) μπορούμε να συμπεράνουμε ότι αν αφαιρέσουμε μία ακμή e από το T_1, υπάρχει ακμή e', με την οποία μπορούμε να την αντικαταστήσουμε ώστε να πάρουμε νέο συνεκτικό δέντρο T'. Ισχύει $|T_1\text{-}T'| = 1$, οπότε οι κορυφές των T_1 και T' συνδέονται με ακμή στον γράφο H. Αν αφαιρέσουμε από το T' μία ακόμη ακμή, μπορούμε να την αντικαταστήσουμε και πάλι με άλλη ακμή του T_2 και να δημιουργήσουμε νέο συνεκτικό δέντρο T''. Συνεχίζουμε τη διαδικασία αυτή μέχρι που μετά από $|T_1\text{-}T_2|$ φορές θα καταλήξουμε στο δέντρο T_2. Επομένως οι κορυφές T_1 και T_2 συνδέονται, και επειδή τις επιλέξαμε αυθαίρετα, το δέντρο μας θα είναι συνεκτικό. Ακόμη κάθε φορά που αλλάζαμε μια ακμή το μονοπάτι που διανύαμε μεγάλωνε κατά 1, επομένως το συνολικό μήκος του μονοπατιού μεταξύ των κορυφών T_1 και T_2 είναι $|T_1\text{-}T_2|$.

Για την εύρεση του μονοπατιού θα ελέγξουμε αρχικά με τον αλγόριθμο του ερωτήματος (Α) και είσοδο τα δέντρα T_1 και T_2 και κάποια ακμή του $T_1\text{-}T_2$, ώστε να προκύψει νέο δέντρο T' που θα είναι κατά μία ακμή πιο κοντά στο T_2. Επαναλαμβάνουμε τη διαδικασία αυτή $|T_1\text{-}T_2|$ φορές και θα καταλήξουμε στο T_2. Αν κρατάμε σε κάθε βήμα το δέντρο που δημιουργείται από τον αλγόριθμο του (Α), στο τέλος έχουμε το μονοπάτι με μέγεθος $|T_1\text{-}T_2|$ μεταξύ των T_1 και T_2, το οποίο περνάει από τους ενδιάμεσους κόμβους που προέκυψαν.

Γ) Μία απλή λύση για το πρόβλημα αυτό είναι να τροποποιήσουμε τον αλγόριθμο του Kruskal για την εύρεση του ελάχιστου συνεκτικού δέντρου, ώστε ,με είσοδο μία ακμή e, να βρίσκει το ελάχιστο συνεκτικό δέντρο που περιέχει την ακμή αυτή [2]. Η τροποποίηση που πρέπει να κάνουμε είναι απλά αντί να ξεκινάμε προσθέτοντας στο σύνολο την ελάχιστη ακμή, να προσθέτουμε πρώτα την ακμή e και στη συνέχεια να προχωράμε κανονικά ελέγχοντας μία-μία τις ακμές από την πιο ελαφριά στη βαρύτερη. Αυτό μπορεί να επιτευχθεί και με το να αλλάζουμε το βάρος της εκάστοτε ακμής σε 0 και να χρησιμοποιούμε τον αλγόριθμο του Kruskal ως έχει. Η πολυπλοκότητα

του αλγορίθμου αυτού είναι O(|E|log|E|). Επομένως η συνολική πολυπλοκότητα για την εύρεση του T_e για κάθε ακμή e, είναι $O(|E|^2 log|E|)$. Γνωρίζουμε ότι η διαδικασία αυτή παράγει το σωστό συνεκτικό δέντρο αφού γνωρίζουμε ότι ο αλγόριθμος του Kruskal είναι ορθός, οπότε η χρήση του, αν μπορούμε να διασφαλίσουμε ότι θα συμπεριλάβει μια ακμή, θα παράξει το ελάχιστο δέντρο που περιέχει αυτή την ακμή.

Μια πιο αποδοτική λύση είναι να φτιάξουμε το ελάχιστο συνδετικό δέντρο T_0, και να δημιουργούμε τα ελάχιστα συνδετικά δέντρα που περιέχουν κάποια ακμή από αυτό. Για να γίνει αυτό προσθέτουμε την ακμή e, για την οποία ψάχνουμε το T_e, στο T_0 και κάνουμε DFS για να βρούμε τις ακμές που ανήκουν στον κύκλο που δημιουργήθηκε από την προσθήκη της e. Από τις ακμές αυτές αφαιρούμε αυτή με το μεγαλύτερο βάρος και το δέντρο που προκύπτει είναι το T_e. Η πολυπλοκότητα του αλγορίθμου αυτού είναι O(|E|log|E|) (για την εύρεση του T_0) + O(|E|(|V|+|E'|)) (για το DFS για κάθε ακμή) όπου E' ο αριθμός ακμών του MST που είναι V-1. Επομένως η συνολική πολυπλοκότητα είναι O(V*E).

Άσκηση 6: Υποπίνακας με Μέγιστο Άθροισμα Στοιχείων (bonus)

Α) Ένας πίνακας n στοιχείων έχει συνολικά n^2 υποπίνακες που θα χρειαστεί να ελέγξουμε. Ο αλγόριθμος για την εύρεση του υποπίνακα με το μέγιστο άθροισμα στοιχείων είναι ο εξής:

- Αρχικοποιούμε τις τιμές max_sum, sum, start1, start2 και end στο 0.
- Διατρέχουμε τον πίνακα και για κάθε τιμή του a_i, μεταβάλλουμε την τιμή του sum στο max(0, sum + a_i), και την τιμή του max_sum στο max(max_sum, sum). Κάθε φορά που αλλάζει η τιμή του sum και γίνεται 0, αναθέτουμε την τιμή του εκάστοτε i στο start2, ώστε να κρατάμε την αρχή του υποπίνακα, του οποίου το άθροισμα υπλογίζουμε. Όποτε αλλάζει η τιμή του max_sum, αναθέτουμε την τιμή start2 στο start1 και την τιμή του εκάστοτε i στο end, ώστε να κρατάμε την αρχή και το τέλος του υποπίνακα με το μέγιστο άθροισμα μέχρι τώρα.
- Όταν διατρέξουμε ολόκληρο τον πίνακα επιστρέφουμε τις τιμές start1 και end, δηλαδή τα όρια του ζητούμενου υποπίνακα.

Η πολυπλοκότητα του αλγορίθμου αυτού είναι O(n), αφού διατρέχεται ο πίνακας μία φορά και για κάθε στοιχείο του, κάνουμε υπολογισμούς σε σταθερό χρόνο.

Πηγές:

1. https://www.geeksforgeeks.org/connected-components-in-an-undirected-graph/
2. https://en.wikipedia.org/wiki/Kruskal%27s_algorithm