Αλγόριθμοι και Πολυπλοκότητα – 2η Σειρά Ασκήσεων
Κυριακόπουλος Γεώργιος – el18153

Άσκηση 1:

Αρχικά, θα δείξουμε ότι το πρόβλημα αυτό είναι ισοδύναμο με ένα πιο απλό. Για κάθε σημείο n με κάθετο πάνω στην ευθεία l στο σημείο x, μπορούμε να βρούμε σε σταθερό χρόνο τα δύο σημεία της ευθείας l που απέχουν r από το σημείο n, έστω y και z (πχ με χρήση πυθαγόρειου θεωρήματος, έχοντας r υποτείνουσα και το πολύ r κάθετο). Λόγω της γεωμετρίας του κύκλου ακτίνας r και αφού κάθε σημείο n απέχει το πολύ r από την ευθεία l και ακριβώς r από τα σημεία y και z, αυτό θα καλύπτεται από ένα δίσκο, αν και μόνο αν το κέντρο αυτού του δίσκου βρίσκεται μεταξύ των σημείων y και z.

Με αυτή τη λογική, το πρόβλημα μας απλοποιείται σε ένα μονοδιάστατο αντίστοιχο, όπου για κάθε σημείο n έχουμε ένα διάστημα [y, z] πάνω σε μία ευθεία, και πρέπει να υπολογίσουμε τον ελάχιστον αριθμό σημείων στην ευθεία (τα κέντρα των κύκλων), ώστε κάθε διάστημα [y, z] να περιλαμβάνει τουλάχιστον ένα τέτοιο σημείο. Θα ακολουθήσουμε έναν άπληστο αλγόριθμο:

- Αρχικά, ταξινομούμε τα διαστήματα αυτά με βάση το δεξί τους άκρο, καθώς θέλουμε να ελέγχουμε με βάση αυτό τα διαστήματα και την ύπαρξη κύκλου που καλύπτει το κάθε σημείο.
- Χρησιμοποιούμε ένα, αρχικά άδειο, σύνολο για τα σημεία (τα κέντρα των κύκλων) και αρχικοποιούμε και μία μεταβλητή last σε μία τιμή μικρότερη από όλα τα αριστερά άκρα των διαστημάτων (πχ -Inf ή INT_MIN), όπου κρατάμε το τελευταίο σημείο που προστέθηκε στο σύνολο.
- Διατρέχουμε τα ταξινομημένα διαστήματα από τα αριστερά προς τα δεξιά, ελέγχοντας κάθε φορά εάν το αριστερό άκρο του τρέχοντος διαστήματος είναι μεγαλύτερο από το last σημείο. Εάν ισχύει αυτό, τότε το τρέχον διάστημα δεν καλύπτεται από κάποιο σημείο μέχρι τώρα και επομένως προσθέτουμε το δεξί του άκρο στο σύνολο και ανανεώνουμε και την τιμή της μεταβλητής last.
- Στο τέλος, θα έχουμε ένα σύνολο με τα σημεία που αντιστοιχούν στα κέντρα των κύκλων και το μήκος του θα είναι ο ελάχιστος αριθμός των σημείων που ψάχνουμε.

Η υπολογιστική πολυπλοκότητα του αλγορίθμου είναι O(nlogn), όπου n ο αριθμός των σημείων. Ο υπολογισμός των διαστημάτων γίνεται με σταθερού χρόνου πράξεις για n σημεία, επομένως O(n), η ταξινόμηση των διαστημάτων με βάση το δεξί άκρο τους γίνεται σε O(nlogn) και το πέρασμα των διαστημάτων είναι O(n) αφού έχει σταθερού χρόνου πράξεις για κάθε

στοιχείο (έλεγχος, add σε σύνολο και ανανέωση μιας μεταβλητής). Τέλος ο υπολογισμός του μήκος του set είναι επίσης σταθερού χρόνου. Συνολικά, έχουμε O(n + nlogn + n) = O(nlogn).

Για την ορθότητα του άπληστου αυτού αλγορίθμου θα χρησιμοποιήσουμε επαγωγή. Έστω g_1, g_2, …, g_k τα σημεία των κέντρων των κύκλων που υπολογίζει ο άπληστος αλγόριθμος και έστω o_1, o_2, …, o_m τα σημεία των κέντρων των κύκλων που υπολογίζει ένας βέλτιστος αλγόριθμος. Έστω, επίσης, a_1, a_2, …, a_n τα ταξινομημένα με βάση το δεξί τους άκρο διαστήματα, ενώ s_1, s_2, …, s_n είναι τα αριστερά και f_1, f_2, …, f_n τα δεξιά τους άκρα.

Πρώτα, θα δείξουμε ότι για κάθε $r \leq m$, ισχύει $g_r \geq o_r$ με επαγωγή:

- Για r = 1, ισχύει, αφού αρκεί ένα σημείο.
- Για να καλύψουμε το διάστημα a_1, έχουμε την άπληστη επιλογή $g_r \geq o_r$.
- Έχουμε, επίσης, $g_{r-1} \geq o_{r-1}$, από επαγωγική υπόθεση.
- Έστω a_i το επόμενο διάστημα, όπου $s_i > g_{r-1}$. Ο άπληστος αλγόριθμος θα προσθέσει ένα νέο σημείο στο δεξί άκρο, δηλαδή $g_r = f_i$ για να καλύψει το διάστημα a_i. Επειδή $o_{r-1} \leq g_{r-1} < s_i$, το διάστημα a_i δεν καλύπτεται από το o_{r-1}. Επομένως, ο βέλτιστος αλγόριθμος χρειάζεται να προσθέσει ένα νέο σημείο όπου $s_i \leq o_r \leq f_i \rightarrow o_r \leq g_r$.
- Ομοίως, αποδεικνύεται ότι k = m. Έστω k > m, τότε εάν ο άπληστος αλγόριθμος πρέπει να προσθέσει κάποιο ακόμη σημείο μετά το g_m, αυτό σημαίνει ότι υπάρχει ένα διάστημα a_j με $s_j > g_m$. Όμως, ισχύει ότι $o_m \leq g_m \rightarrow o_m < s_j$, δηλαδή ότι το διάστημα a_j δεν καλύπτεται από το o_m, το οποίο αποτελεί το πιο δεξιό σημείο της βέλτιστης λύσης, κάτι που σημαίνει η λύση o_1, o_2, …, o_n δεν είναι πραγματοποιήσιμη.

Άσκηση 2:

(α) Στην περίπτωση που έχουμε 1 υπάλληλο, αυτός πρέπει να εξυπηρετήσει όλους τους πελάτες. Θα χρησιμοποιήσουμε έναν άπληστο αλγόριθμο με σκοπό να ελαχιστοποιήσουμε το βεβαρυμμένο χρόνο εξυπηρέτησης. Συγκεκριμένα:

- Υπολογίζουμε τα $x_i = \frac{w_i}{p_i}$.
- Ταξινομούμε τα x_i σε αύξουσα σειρά.
- Ξεκινάμε από το μεγαλύτερο, έστω x_n να υπολογίσουμε το βεβαρυμμένο χρόνο (ο συμβολισμός που ακολουθεί αφορά τα ταξινομημένα x_1 έως x_n και τα αντίστοιχα τους w_i και p_i):

$$t = w_n \cdot p_n + w_{n-1}(p_{n-1} + p_n) + w_{n-2}(p_{n-2} + p_{n-1} + p_n) + \cdots + w_1(p_1 + p_2 + \cdots + p_n)$$

Ουσιαστικά πρόκειται για μία επανάληψη από n μέχρι 1, όπου κρατώντας μία μεταβλητή p_{sum}, αρχικοποιημένη στο 0, και το βεβαρυμμένο χρόνο t, επίσης αρχικοποιημένο στο 0, προσθέτουμε στο t το $w_i \cdot (p_{sum} + p_i)$ και αυξάνουμε το p_{sum} κατά p_i.

Σχετικά με τη χρονική πολυπλοκότητα έχουμε $O(nlogn)$, αφού έχουμε έναν υπολογισμό των x_i σε $O(n)$, μία ταξινόμηση αυτών σε $O(nlogn)$ και τέλος έναν υπολογισμό του χρόνου σε $O(n)$, άρα σύνολο $O(nlogn)$.

Γενικά η λύση βασίζεται στο γεγονός ότι θέλουμε να εξυπηρετήσουμε νωρίτερα τους πελάτες με μεγαλύτερο βάρος w_i, αφού έχουν μεγαλύτερη σημασία για την εταιρία μας, όπως και τους πελάτες με το μικρότερο p_i, αφού έτσι θα πετύχουμε τον ελάχιστο βεβαρυμμένο χρόνο, σύμφωνα και με τον αλγόριθμο του Ελάχιστου Ατομικού Χρόνου Εξυπηρέτησης. Για να πετύχουμε και τους δύο στόχους αυτούς χρησιμοποιούμε τα $x_i = \frac{w_i}{p_i}$, ως το κριτήριο με το οποίο θα ταξινομήσουμε τους πελάτες για να διαλέξουμε τη σειρά εξυπηρέτησης, αφού αυτό αυξάνεται όσο αυξάνεται το βάρος w_i και αυξάνεται όσο μειώνεται ο χρόνος p_i.

Σχετικά με την ορθότητα αυτής της επιλογής (δηλαδή του να διαλέγουμε με τη σειρά να εξυπηρετούνται αυτοί με τα μεγαλύτερα x_i, δηλαδή σειρά εξυπηρέτησης $\frac{w_n}{p_n} > \frac{w_{n-1}}{p_{n-1}} > \cdots > \frac{w_1}{p_1}$) μπορούμε να πούμε τα εξής. Έστω ότι σε έναν άλλο βέλτιστο αλγόριθμο για 2 συνεχόμενους πελάτες στη σειρά εξυπηρέτησης ισχύει ότι $\frac{w_{πρώτου}}{p_{πρώτου}} \leq \frac{w_{δεύτερου}}{p_{δεύτερου}}$. Η συνεισφορά του βέλτιστου αλγορίθμου με αυτή τη σειρά, στον βεβαρυμμένο χρόνο είναι:

$$t_{opt} = w_1(p_1 + x) + w_2(p_2 + p_1 + x),$$

όπου με w_1 συμβολίζω το w του πρώτου στη σειρά και αντίστοιχα για το w_2, p_1, p_2, ενώ με x είναι το άθροισμα των προηγούμενων χρόνων p_i των πελατών που εξυπηρετήθηκαν ήδη.

Έστω, ότι σύμφωνα με τον άπληστο μας αλγόριθμο ανταλλάσσουμε σειρά σε αυτούς τους δύο ώστε να εξυπηρετηθεί πρώτα ο δεύτερος που έχει μεγαλύτερο x_i. Η αντίστοιχη συνεισφορά στο βεβαρυμμένο χρόνο είναι τώρα:

$$t_{gr} = w_2(p_2 + x) + w_1(p_1 + p_2 + x)$$

Για να αποδείξουμε ότι ο άπληστος αλγόριθμος δίνει την καλύτερη λύση θέλουμε η βέλτιστη λύση να προσθέτει μεγαλύτερο χρόνο στο βεβαρυμμένο, ώστε ο άπληστος να τον ελαχιστοποιεί. Με άλλα λόγια θέλουμε, δηλαδή:

$$t_{opt} \geq t_{gr} \Rightarrow w_1(p_1 + x) + w_2(p_2 + p_1 + x) \geq w_2(p_2 + x) + w_1(p_1 + p_2 + x) \Rightarrow$$

$$w_1p_1 + w_2p_2 + w_2p_1 \geq w_2p_2 + w_1p_1 + w_1p_2 \Rightarrow w_2p_1 \geq w_1p_2 \Rightarrow \frac{w_2}{p_2} \geq \frac{w_1}{p_1},$$

το οποίο ισχύει από υπόθεση. Άρα, ο άπληστος αλγόριθμος που σχεδιάσαμε δίνει πάντα καλύτερη λύση εάν ακολουθείται η σειρά εξυπηρέτησης με βάση τα μεγαλύτερα x_i που υπολογίσαμε πρώτα.

(β) Θα χρησιμοποιήσουμε Δυναμικό Προγραμματισμό και θα ελέγχουμε ουσιαστικά εάν μας συμφέρει να επιλέξουμε κάθε πελάτης να εξυπηρετηθεί από τον υπάλληλο 1 ή τον υπάλληλο 2 με βάση το ποιος έχει μέχρι τώρα το μικρότερο βεβαρυμμένο χρόνο. Θα χρησιμοποιήσουμε και πάλι τη λογική του ερωτήματος (α) με έναν υπάλληλο, επομένως θα ξεκινήσουμε τον έλεγχο από τα μεγαλύτερα x_i προς τα μικρότερα. Η αναδρομική σχέση είναι η εξής:

$$time(i) = time(i-1) + w_i \cdot \{\min(A, B) + p_i\}$$

Ουσιαστικά, ξεκινάμε κρατώντας δύο μεταβλητές με τους χρόνους εξυπηρέτησης των πελατών που στέλνουμε στον υπάλληλο 1 (A) και στον υπάλληλο 2 (B), δηλαδή το άθροισμα των p_j για τους j πελάτες κάθε υπαλλήλου. Αρχικά αυτές είναι 0 και αναλόγως το ποια απάντηση δίνει το $\min(A, B)$, η αντίστοιχη μεταβλητή αυξάνεται κατά p_i, αφού θα στείλουμε τον πελάτη i στον αντίστοιχο πελάτη που έχει μέχρι τώρα το μικρότερο βεβαρυμμένο χρόνο.

Η χρονική πολυπλοκότητα της λύσης είναι $O(nlogn)$, αφού πάλι χρειαζόμαστε τον υπολογισμό των x_i σε $O(n)$, την ταξινόμηση τους σε $O(nlogn)$ και μετά έναν υπολογισμό n τιμών για το Δυναμικό Προγραμματισμό, όπου κάθε υπολογισμός είναι σταθερού κόστους (αφού έχουμε min 2 τιμών μόνο), άρα πάλι $O(n)$ και συνολικά $O(nlogn)$.

Για $m \geq 3$, μπορούμε να ακολουθήσουμε μία παρόμοια λογική όπου κρατάμε μία μεταβλητή (ή καλύτερα έναν πίνακα m μεγέθους) για το άθροισμα των χρόνων εξυπηρέτησης των πελατών που έχει αναλάβει κάθε υπάλληλος και αντίστοιχα διαλέγουμε το min αυτών. Η πολυπλοκότητα αυτού του προβλήματος ανεβαίνει σε $O(nlogn + nm)$, αφού πλέον πέρα από τον υπολογισμό των x_i και την ταξινόμηση τους σε $O(nlogn)$, έχουμε κάθε φορά στους υπολογισμούς n τιμών για το Δυναμικό Προγραμματισμό, έναν υπολογισμό min μεταξύ m τιμών των αθροισμάτων χρόνων εξυπηρέτησης των πελατών κάθε υπαλλήλου, δηλαδή $O(nm)$.

Άσκηση 3:

(α) Για τη λύση του προβλήματος θα χρησιμοποιήσουμε Δυναμικό Προγραμματισμό. Έχουμε στην είσοδο τα σημεία x_i με $0 \leq i \leq f$ και $x_0 = 0, x_f = L$, με τα n σημεία που χρειάζονται στέγαση να είναι αυτά για $1 \leq i \leq n$. Επίσης, παρατηρούμε ότι κάθε στέγη που θα χρησιμοποιήσουμε για να καλύψουμε τα σημεία του πεζόδρομου θα ξεκινάει από κάποιο x_i και θα τελειώνει σε κάποιο x_j με j όχι απαραίτητα διαφορετικό του i, ειδάλλως θα καλύπτουμε ενδιάμεσα σημεία που δεν απαιτείται και επομένως δεν θα έχουμε βέλτιστη λύση. Έχουμε, επίσης, ότι η πρώτη στέγη θα ξεκινάει από το σημείο x_1 και η τελευταία στέγη θα τελειώνει στο σημείο x_n.

Θα χρησιμοποιήσουμε, επομένως, Δυναμικό Προγραμματισμό, ξεκινώντας από την τελευταία στέγη (η οποία τελειώνει στο x_n) και ψάχνοντας ποιο σημείο x_i είναι το βέλτιστο για να ξεκινήσει η στέγη αυτή, δηλαδή να δίνει το ελάχιστο κόστος. Παίρνουμε, δηλαδή, μια σχέση της παρακάτω μορφής:

$$cost(i) = \min_{1 \leq j \leq i} \left\{ cost(j-1) + \left(x_i - x_j\right)^2 + C \right\},$$

όπου cost(i) το ελάχιστος κόστος για να καλύψουμε τα σημεία x_1 έως x_i και $cost(0) = 0$. Η λύση μας είναι η τιμή του $cost(n)$, δηλαδή το ελάχιστο κόστος για να καλύψουμε τα σημεία x_1 έως και x_n.

Για την πολυπλοκότητα του αλγορίθμου έχουμε ότι είναι ίση την πολυπλοκότητα των υποπροβλημάτων επί τον αριθμό τους. Έχουμε, δηλαδή, O(n^2) πολυπλοκότητα, καθώς υπολογίζουμε την τιμή $cost(i)$ για n σημεία, δηλαδή O(n), ενώ για κάθε σημείο έχουμε έναν υπολογισμό ελάχιστης τιμής μεταξύ το πολύ n τιμών, δηλαδή O(n) πάλι, άρα σύνολο O(n^2).

(β) Έχουμε στην είσοδο τις ευθείες ταξινομημένες σε φθίνουσα σειρά τις κλίσεις a_j των ευθειών και με αύξουσα σειρά τα σημεία x_i, επομένως μπορούμε να εφαρμόσουμε το Convex Hull Trick και να υπολογίσουμε για κάθε σημείο ποια ευθεία δίνει την ελάχιστη τιμή. Ουσιαστικά, θα διατηρούμε το κυρτό περίβλημα των ευθειών που μας δίνονται για να λύσουμε το πρόβλημα. Δηλαδή θα κρατάμε μια διάταξη των ευθειών $y = a_j x + b_j$ για την οποία η y_1 θα δίνει την ελάχιστη τιμή από το $-\infty$ μέχρι το σημείο τομής με την y_2, στη συνέχεια η y_2 μέχρι το σημείο τομής της με την y_3 κλπ.

Ο τρόπος με τον οποίο θα εισάγουμε τις ευθείες βασίζεται στο ότι οι ευθείες δίνονται ταξινομημένες σε φθίνουσα σειρά με βάση την κλίση τους. Συγκεκριμένα, αφού η κάθε νέα ευθεία που θέλουμε να βάλουμε έχει αρνητικότερη κλίση, θα δίνει μικρότερες τιμές μετά από κάποιο x. Για να βρούμε σε ποιο σημείο συμβαίνει αυτό, ξεκινάμε από την τελευταία ευθεία που ανήκει στο κυρτό περίβλημα και ελέγχουμε αν το σημείο τομής αυτής και της νέα ευθείας που εξετάζουμε βρίσκεται πιο κάτω στον άξονα y σε σχέση με το σημείο τομής της προτελευταίας ευθείας του κυρτού περιβλήματος και της νέας. Εάν ισχύει αυτό, τότε η νέα ευθεία προστίθεται στο τέλος του περιβλήματος, αλλιώς αφαιρείται η τελευταία ευθεία αυτού

και συνεχίζουμε τον έλεγχο με την προηγούμενη. Για το σύνολο των εισαγωγών των ευθειών έχουμε O(n) χρόνο, αφού κάθε ευθεία εισάγεται και αφαιρείται από το περίβλημα το πολύ μία φορά, άρα συνολικά θα έχουμε 2n πράξεις για n ευθείες, επομένως amortized O(1) για κάθε ευθεία ή συνολικά amortized O(n).

Στη συνέχεια, για κάθε σημείο x_i θα πρέπει να υπολογίσουμε ποια ευθεία δίνει ελάχιστη τιμή. Για αυτό το σκοπό θα διασχίσουμε το κυρτό περίβλημα, ξεκινώντας από την πρώτη ευθεία για το πρώτο σημείο και συνεχίζοντας για το επόμενο σημείο από την ευθεία που δίνει ελάχιστη τιμή για το προηγούμενο σημείο, βασισμένοι στην αύξουσα σειρά με την οποία δίνονται τα σημεία x_i. Γνωρίζουμε, δηλαδή, ότι η ευθεία που ελαχιστοποιεί το σημείο x_{i+1} δεν βρίσκεται σίγουρα πριν την ευθεία που ελαχιστοποιεί το σημείο x_i. Επομένως, για να βρούμε για όλα τα k σημεία την κατάλληλη ευθεία (δηλαδή τους k δείκτες που υποδεικνύουν την ευθεία) θα διασχίσουμε συνολικά μία φορά το περίβλημα, δηλαδή O(k) πολυπλοκότητα. Άρα συνολικά έχουμε την επιθυμητή O(n+k) πολυπλοκότητα.

Για να χρησιμοποιήσουμε τον παραπάνω αλγόριθμο στο αρχικό μας πρόβλημα πρέπει πρώτα να μετατρέψουμε τη σχέση του Δυναμικού Προγραμματισμού ως εξής:

$$cost(i) = \min_{1 \le j \le i} \{cost(j-1) + x_i^2 - 2x_i x_j + x_j^2 + C\} \Rightarrow$$

$$cost(i) = x_i^2 + C + \min_{1 \le j \le i} \{cost(j-1) + x_j^2 - 2x_i x_j\},$$

αφού παρατηρούμε πως το $x_i^2 + C$ δεν εξαρτάται από το δείκτη j στον υπολογισμό του ελαχίστου και προστίθεται όπως μια σταθερά σε κάθε ένα. Επίσης, παρατηρούμε ότι το εσωτερικό του υπολογισμού του ελαχίστου μπορεί να πάρει τη μορφή ευθείας ως εξής:

$$cost(i) = x_i^2 + C + \min_{1 \le j \le i} \{a[j]x_i + b[j]\},$$

$$με\ a[j] = -2x_j\ και\ b[j] = cost(j-1) + x_j^2$$

Επομένως, έτσι μπορούμε με τη χρήση του Convex Hull Trick να μειώσουμε τον γραμμικό χρόνο υπολογισμού O(n) κάθε ελάχιστου σε amortized σταθερό O(1), αφού μπορούμε για όλα τα σημεία x_i να υπολογίσουμε όλα τα ελάχιστα σε συνολικό χρόνο O(n). Άρα, τελικά, θα βελτιώσουμε την πολυπλοκότητα από O(n^2) σε O(n) για το αρχικό μας πρόβλημα, καθώς θα έχουμε υπολογισμό amortized O(1) για συνολικά n σημεία.

Άσκηση 4:

(α) Μπορούμε να λύσουμε το πρόβλημα χρησιμοποιώντας Δυναμικό Προγραμματισμό. Συγκεκριμένα, ξεκινώντας από το τέλος μπορούμε να ψάξουμε να βρούμε μέχρι ποιον φοιτητή πρέπει να βάλουμε στο τελευταίο λεωφορείο και να συνεχίσουμε προς τα πίσω. Ορίζουμε ως $sens(i,j)$ τον ελάχιστο δείκτη ευαισθησίας για να τοποθετηθούν οι φοιτητές 1 έως j σε i λεωφορεία. Φυσικά, ισχύει ότι $sens(i,j) = 0$, για $i = j$, αφού μπορούμε να χωρίσουμε τους φοιτητές έναν ένα σε ισάριθμο πλήθος λεωφορείων, ενώ για το $sens(1,j)$ υπολογίζουμε μόνο ένα διπλό άθροισμα για όλους τους j φοιτητές και τα ζευγάρια A_{ij} που μπορούν να σχηματίσουν μεταξύ τους (δηλαδή $\sum_{a=1}^{j}\sum_{b=a+1}^{j} A_{ab}$). Έχουμε, επομένως, για τις υπόλοιπες περιπτώσεις, την παρακάτω αναδρομική σχέση:

$$sens(i,j) = \min_{1\le l<j}\{sens(i-1,l) + \sum_{a=l+1}^{j}\sum_{b=a+1}^{j} A_{ab}\}$$

Δηλαδή, για κάθε υπολογισμό ενός δείκτη ευαισθησίας για i λεωφορεία και 1 έως j φοιτητές, επιλέγουμε το ελάχιστο άθροισμα δείκτη ευαισθησίας υπολογίζοντας αναδρομικά με ποιον τρόπο μπορούμε να χωρίσουμε το πολύ $j-1$ φοιτητές σε $i-1$ λεωφορεία και να βάλουμε τους υπόλοιπους στο i-οστό λεωφορείο. Η λύση του προβλήματος θα είναι η τιμή του $sens(n,k)$.

Με μία πρώτη ματιά, φαίνεται να έχουμε $O(kn^4)$ πολυπλοκότητα λόγω των kn υποπροβλημάτων, όπου κάθε υποπρόβλημα πρέπει να υπολογίσει το ελάχιστο μεταξύ το πολύ n ποσοτήτων, με κάθε ποσότητα να χρειάζεται $O(n^2)$ υπολογισμούς για το διπλό άθροισμα, άρα σύνολο $kn \cdot n \cdot n^2$, δηλαδή $O(kn^4)$.

Ωστόσο, μπορούμε με έναν προϋπολογισμό για τα αθροίσματα να πετύχουμε τελική πολυπλοκότητα $O(kn^2)$, αφού θα βελτιωθεί ο $O(n^2)$ υπολογισμός του διπλού αθροίσματος σε σταθερό $O(1)$ χρόνο (ουσιαστικά θα έχουμε προϋπολογισμό κόστους $O(n^2)$ που δεν αυξάνει την πολυπλοκότητα και σταθερού χρόνου υπολογισμό διπλού αθροίσματος). Ο τρόπος με τον οποίο θα γίνει αυτό είναι μέσω του υπολογισμού των ποσοτήτων $S(x,y)$, το οποίο αποτελεί το άθροισμα των στοιχείων του A με δείκτη γραμμής έως και x και δείκτη στήλης έως και y (2-D Array Prefix Sum), με βάση την ακόλουθη αναδρομική σχέση (όπου βασιζόμενοι στης προηγούμενης γραμμής και σειράς τα αθροίσματα, αφαιρούμε τα στοιχεία που αθροίζουμε δύο φορές, λόγω της δισδιάστατης μορφής και προσθέτουμε και την νέα τιμή):

$$S(x,y) = S(x-1,y) + S(x,y-1) - S(x-1,y-1) + A_{xy},$$

βάση της οποίας θα μπορέσουμε να υπολογίσουμε τις τιμές $S(x,y)$ για όλα τα $x,y \in [1,n]$ σε χρόνο $O(n^2)$. Έτσι, θα έχουμε έτοιμο σε σταθερό χρόνο $O(1)$ κάθε διπλό άθροισμα μέσω της παρακάτω σχέσης (όπου παίρνουμε το άθροισμα όλων των στοιχείων μέχρι τα όρια των αθροισμάτων, δηλαδή το άθροισμα των $j \cdot j$ στοιχείων, ή $S(j,j)$ και αφαιρούμε αυτά που δεν

θέλουμε, λόγω του ότι συνήθως το κάτω άκρο του αθροίσματος δεν είναι ίσο με 1, ενώ τέλος προσθέτουμε μία φορά αυτά που αφαιρέσαμε δύο φορές, λόγω της δισδιάστατης μορφής):

$$\sum_{a=l+1}^{j}\sum_{b=a+1}^{j} A_{ab} = \frac{1}{2}\{S(j,j) - S(j,l) - S(l,j) + S(l,l)\}$$

Στη διαφορετική περίπτωση που αυτό που μας ενδιαφέρει είναι μία ανάθεση $(q'_1, \dots, q'_k)$ των n φοιτητών στα k λεωφορεία που ελαχιστοποιεί τον μέγιστο δείκτη ευαισθησίας των λεωφορείων, τότε η αναδρομική σχέση που θα χρησιμοποιούσα είναι η παρακάτω:

$$sens(i,j) = \min_{l<j}\{\max(sens(i-1,l), \sum_{a=l+1}^{j}\sum_{b=a+1}^{j} A_{ab})\}$$

Δηλαδή, αυτή τη φορά δεν μας νοιάζει να ελαχιστοποιήσουμε τον αθροιστικό δείκτη ευαισθησίας όλων των λεωφορείων, αλλά να ελαχιστοποιήσουμε τον δείκτη ευαισθησίας κάθε λεωφορείου και για αυτό κρατάμε τον ελάχιστο μέγιστη δείκτη των επιμέρους υπολογισμών για κάθε λεωφορείο, αφού συγκρίνουμε τον παλιότερο max δείκτη του προηγούμενου λεωφορείου με τον υπολογιζόμενο δείκτη του νέου λεωφορείου. Ο υπόλοιπος αλγόριθμος δεν αλλάζει (π.χ. ο προϋπολογισμός των αθροισμάτων με το 2-D Prefix Sum), όπως και η πολυπλοκότητα, η οποία είναι και πάλι $O(kn^2)$, αφού έχουμε πάλι kn υποπροβλήματα με n σταθερού κόστους υπολογισμούς για το ελάχιστο το καθένα.

Και για τα δύο προβλήματα, η τελική ανάθεση που ψάχνουμε, δηλαδή το πλήθος των φοιτητών σε κάθε λεωφορείο, βρίσκεται εύκολα εάν κατά την επίλυση με τη χρήση Δυναμικού Προγραμματισμού κρατάμε σε ένα πίνακα μεγέθους k (όσα και τα λεωφορεία) τον αντίστοιχο αριθμό φοιτητών που ελαχιστοποιεί το συνολικό δείκτη ευαισθησίας των λεωφορείων. Αυτός ο αριθμός είναι ίσος, για το i-οστό λεωφορείο, με τη διαφορά $j - l$ της παράστασης που δίνει min τιμή στο $sens(i,j)$, εάν $i > 1$, αλλιώς είναι ίσος με j για $i = 1$.

Άσκηση 5:

(α) Έστω T_1 και T_2 δύο διαφορετικά συνδετικά δέντρα του G και μία ακμή $e \in T_1 \backslash T_2$ με $T_1 \backslash T_2 \neq \emptyset$. Εάν προσθέσουμε την ακμή e στο δέντρο T_2, τότε αυτό θα περιέχει έναν κύκλο, έστω C και επειδή το T_1 δεν περιέχει τον C (αφού είναι συνδετικό δέντρο), υπάρχει μία ακμή έστω e' που ανήκει στον κύκλο C και δεν ανήκει στο T_1, δηλαδή $e \in C$ και $e' \notin T_1$. Άμα αφαιρεθεί, δηλαδή, η ακμή e από το T_1 και προστεθεί η ακμή e' (και οι δύο ανήκουν στον κύκλο C και ενώνουν τα υποδέντρα που δημιουργούνται κατά την αφαίρεση της e), αυτό θα διατηρήσει τη συνεκτικότητα του, θα σπάσει τον κύκλο και το δέντρο θα έχει $n-1$ ακμές, άρα το $T_1 \backslash \{e\} \cup \{e'\}$ είναι ένα συνδετικό δέντρο.

Για να υπολογίσουμε μία ακμή e' με δεδομένα εισόδου τα T_1, T_2 και e = {u, v} θα ακολουθήσουμε τον παρακάτω αλγόριθμο:

- Αρχικά, θα τρέξουμε διάσχιση κατά βάθος από το u στο v στο δέντρο T_2 και θα βρούμε το μονοπάτι P που συνδέει τους δύο κόμβους σε χρόνο O(|V|).
- Για κάθε ακμή e που ανήκει στο P ($e \in P$) (το πολύ $V-1$ ακμές) κάνουμε έλεγχο σε O(1) για το εάν ανήκει και στο T_1 μέχρι να βρούμε μια ακμή e' $\in P$ και $e' \notin T_1$, η οποία είναι η απάντηση μας.
- Συνολικά, επομένως, έχουμε O(|V|) πολυπλοκότητα από την διάσχιση κατά βάθος και O(|V|*1) = O(|V|) από τους ελέγχους για κάθε ακμή του μονοπατιού P, άρα τελικά O(|V|) πολυπλοκότητα.

(β) Θα χρησιμοποιήσουμε την επαγωγή στο μήκος του μονοπατιού:

- Επαγωγική βάση: από ορισμό του H, έχουμε ότι $|T_1 \backslash T_2| = 1$, ανν $d_H(T_1, T_2) = 1$.
- Επαγωγική υπόθεση: $|T_1 \backslash T_2| = k$, ανν $d_H(T_1, T_2) = k$.
- Επαγωγικό βήμα: $|T_1 \backslash T_2| = k+1$, ανν $d_H(T_1, T_2) = k+1$.

Για την απόδειξη του επαγωγικού βήματος έχουμε:

- Έστω $e \in T_1 \backslash T_2$. Από (α), υπάρχει $e' \in T_2 \backslash T_1$, τέτοια ώστε $T_1' = (T_1 \backslash \{e\}) \cup \{e'\} \in H$.
- Αλλά, $|T_1' \backslash T_2| = k \overset{E.Y.}{\Longrightarrow} d_H(T_1, T_2) \leq k+1$.
- Από Ε.Υ. αν $d_H(T_1, T_2) = k$, τότε $|T_1 \backslash T_2| = k$, οπότε, αναγκαστικά, $|T_1 \backslash T_2| = k+1$.
- Άρα, $d_H(T_1, T_2) \geq k+1 \Rightarrow d_H(T_1, T_2) = k+1$.
- Εφόσον $d_H(T_1, T_2) = k+1$, υπάρχει $T_1' \in H$, τέτοιο ώστε $d_H(T_1, T_1') = 1$ και $d_H(T_1', T_2) = k$. Τότε, από Ε.Υ. $|T_1' \backslash T_2| = k$.
- Άρα, $|T_1 \backslash T_2| = k-1$ ή $|T_1 \backslash T_2| = k+1$.
- Αν $|T_1 \backslash T_2| = k-1 \Rightarrow d_H(T_1, T_2) = k-1$, άτοπο, αφού $|T_1 \backslash T_2| = k+1$.

Για να υπολογίσουμε το συντομότερο μονοπάτι μεταξύ των T_1 και T_2 θα υπολογίσουμε αρχικά το σύνολο των ακμών που ανήκουν στο T_2 και δεν ανήκουν στο T_1, δηλαδή $e \in T_2 \backslash T_1$. Κάθε μία από αυτές την προσθέτουμε στο υπάρχον δέντρο. Αν $T_2 \backslash T_1 = k$, τότε σε k βήματα έχουμε

μετατρέψει το T_1 σε T_2, δηλαδή έχουμε βρει ένα μονοπάτι στο H μήκους k, το οποίο είναι και το συντομότερο μονοπάτι.

Σχετικά με την χρονική πολυπλοκότητα, σε O(|V|) αποθηκεύουμε το T_1 ως rooted δέντρο, δηλαδή κάθε κόμβος να δείχνει στον πατέρα του. Στη συνέχεια, για κάθε ακμή $e \in T_2$ ελέγχουμε αν $e \in T_1$ σε σταθερό χρόνο O(1), άρα O(|V|). Επίσης, κάνουμε k ανανεώσεις ακμών σε O(|V|) βήματα. Συνολικά, επομένως, έχουμε χρονική πολυπλοκότητα O(k|V|).

(γ) Θα κατασκευάσουμε το ελάχιστο συνδετικό δέντρο μία φορά και για κάθε ακμή που δεν ανήκει σε αυτό θα βρίσκουμε το ελάχιστο συνδετικό δέντρο που περιέχει και αυτήν βάση του πρώτου. Συγκεκριμένα, για μία ακμή $e \in E$, που δεν ανήκει στο ΕΣΔ, αν την προσθέσουμε σε αυτό και αφαιρέσουμε τη βαρύτερη ακμή στον κύκλο που δημιουργεί η προσθήκη της e (και την περιλαμβάνει), θα έχουμε το ζητούμενο ΕΣΔ που να περιέχει και την e (φαίνεται και σκεπτόμενοι τη συμπεριφορά του αλγορίθμου Kruskal σε περίπτωση που η e, για παράδειγμα, είχε βάρος οριακά μεγαλύτερο από αυτό της δεύτερης μεγαλύτερης ακμής του κύκλου).

Επομένως, αρκεί να βρούμε ένα ΕΣΔ, το βάρος του ($W_{ΕΣΔ}$) και έχουμε απάντηση για όλες τις ακμές που περιέχει αυτό. Για τις υπόλοιπες, πρέπει να βρούμε έναν τρόπο να εντοπίζουμε τη μέγιστη ακμή στον κύκλο που δημιουργούν με την προσθήκη τους και να προσθέτουμε τη διαφορά των βαρών τους στο $W_{ΕΣΔ}$ για να πάρουμε το τελικό βάρος του συνδετικού δέντρου που τις περιέχει.

Για τον υπολογισμό του ΕΣΔ, έστω T*, χρησιμοποιούμε τον αλγόριθμο του Kruskal με χρονική πολυπλοκότητα $O(mlogm)$. Για τον υπολογισμό της μέγιστης ακμής του κύκλου όταν προσθέτουμε μία ακμή $e = \{u, v\}$ που δεν περιέχει το ΕΣΔ (δηλαδή τη βαρύτερη ακμή στο μονοπάτι που συνδέει τις κορυφές u και v στο T*) τρέχουμε ένα DFS για κάθε ακμή e από τον κόμβο v μέχρι να βρούμε το μονοπάτι που μας πάει από την v στην u στο T* (και επομένως και το βάρος της βαρύτερης ακμής σε αυτό). Αυτό θα είχε πολυπλοκότητα $O(mn)$ αφού για όλες τις m ακμές τρέχουμε ένα DFS κόστους $O(n)$, όπου n οι κορυφές. Αυτή η πολυπλοκότητα μπορεί να βελτιωθεί σε $O(n^2)$ αν με ένα ακόμα 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.