Αλγόριθμοι και Πολυπλοκότητα

2η Σειρά Ασκήσεων

Βασίλης Βαρσαμής-el18033

Άσκηση 1

Έστω σημείο p, τότε υπάρχει ένα διάστημα πάνω στην ευθεία l με άκρα $left$ και right αντίστοιχα που απέχουν από το p απόσταση(ευκλείδια) r, μέσα στο οποίο πρέπει να υπάρχει τουλάχιστον ένα κέντρο κύκλου.Έτσι το πρόβλημά μας ανάγεται στο να βρούμε τον ελάχιστο αριθμό σημείων στην ευθεία, έτσι ώστε κάθε ένα από τα n διαστήματα που αναφέραμε παραπάνω να περιέχει τουλάχιστον ένα σημείο.Ορίζουμε κάθε διάστημα $a_i = [s_i, f_i]$, όπου s_i η αρχή του και f_i το τέλος του.Η greedy στρατηγική εδώ είναι να ταξινομήσουμε το κάθε διάστημα σε αύξουσα σειρά f_i και κάθε φορά που παίρνουμε το τέλος ενός διαστήματος ως το βέλτιστο σημείο, να πετάμε όσα σημεία έχουν σημείο εκκίνησης s_i μικρότερο του επιλεχθέντος βέλτιστο σημείου.Η διαδικασία αυτή θα συνεχιστεί μέχρι να μην υπάρχουν άλλα διαστήματα προς εξέταση.Η πολυπλοκότητα της λύσης είναι $O(nlogn)$ που χρειάζεται για την ταξινόμηση, γιατί άπαξ και ταξινομήσαμε με ένα πέρασμα βρίσκουμε την λύση.Θα αποδείξουμε την ορθότητα του παραπάνω αλγόριθμου χρησιμοποιώντας επαγωγή. Έστω $o_1, \ldots, o_m$ τα σημεία που επιστρέφει ο OPT αλγόριθμος και $g_1, \ldots, g_k$ ο Greedy.

Για το πρώτο διάστημα ισχύει $g_1 = f_1$, οπότε για να το καλύψει και ο OPT πρέπει $o_1 \leq g_1$.

Υποθέτουμε με επαγωγή ότι για κάθε $r \leq m$ ισχύει $o_{r-1} \leq g_{r-1}$.

Τότε για το r-οστό διάστημα ο greedy θα παρει $g_r = f_r$ και ισχύει ότι $g_{r-1} < s_r$ αλλιώς θα υπήρχε επικάλυψη.Έτσι $o_{r-1} < s_r$, άρα για να καλύψει το r-οστό διάστημα ο OPT θα πρέπει $s_i \leq o_r \leq f_r \leq g_r$. Τελικά $o_r \leq g_r$.. Προφανώς k = m γιατί αν δεν ίσχυε τότε θα υπήρχε ένα διάστημα (μετά το m) το οποίο θα χρειαζόταν να καλυφθεί το οποίο θα το κάλυπτε ο greedy, αλλά όχι ο OPT πράγμα που είναι άτοπο διότι το o_m είναι το δεξιότερο σημείο της βέλτιστης λύσης.

Άσκηση 2

2α) Ταξινομούμε τους λόγους $\frac{w_i}{p_i}$ κατά φθίνουσα σειρά. Η σειρά που πετυχαίνει τον μικρότερο χρόνο ολοκλήρωσης είναι η σειρά που επιλέγει κάθε φορά το p_i με το μεγαλύτερο λόγο $\frac{w_i}{p_i}$ (προφανώς από τα p_i που δεν έχουν ακόμα επιλεγεί).Η πολυπλοκότητα του παραπάνω αλγορίθμου είναι $O(nlogn)$ γιατί από τη στιγμή που ταξινομήσουμε τα παραπάνω πηλίκα, χρειαζόμαστε ένα πέρασμα στον ταξινομημένο πίνακα για να βρούμε την σειρά με την οποία θα εισέλθει το κάθε p_i.

Υποθέτουμε χωρίς βλάβη της γενικότητας ότι η ταξινόμηση είναι ως εξής:

$$\frac{w_n}{p_n} \leq \frac{w_{n-1}}{p_{n-1}} \leq \ldots \leq \frac{w_1}{p_1} \qquad (1)$$

Επίσης ορίζουμε $W = \sum_{i=1}^{n} w_i$.

Η αρχή της βελτιστότητας (A.B) είναι η εξής:

$$T^*(p_1, p_2, \ldots, p_n, W) = \min_i \{p_i W + T^*(p_1, \ldots, p_{i-1}, p_{i+1}, \ldots, p_n, W - w_i)\}$$

Ο Greedy αλγόριθμος θα επιλέξει πρώτο το p_1, υποθέτοντας την (1).

$$T(p_1, p_2, \ldots, p_n, W) = p_1 W + T(p_2, \ldots, p_n, W - w_1).$$

Απόδειξη ορθότητας:

- Υποθέτουμε με ισχυρή επαγωγή ότι για κάθε πρόβλημα που έχει $|P| \leq n-1$, όπου P το σύνολο των πελατών ισχύει ότι ο greedy αλγόριθμος είναι optimal, δηλαδή: $T^* = T$.Προφανώς για $|P| = 1$ ισχύει τετριμμένα.
- Για $|P| = n$:

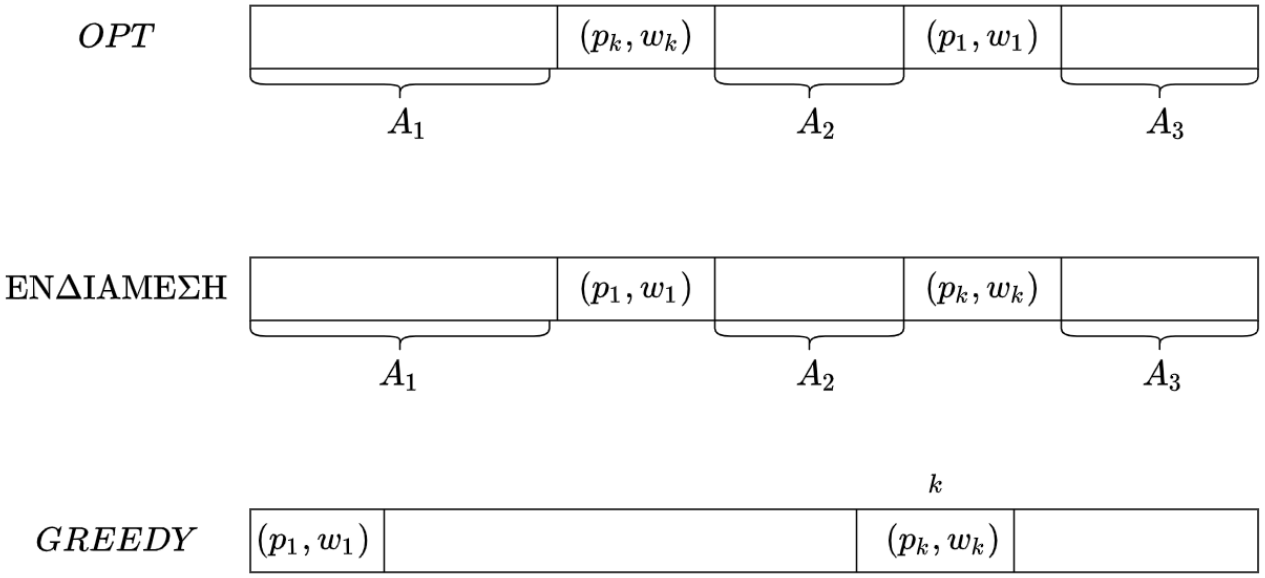
$$T(p_1, p_2, \ldots, p_n, W) = p_1 W + T(p_2, \ldots, p_n, W - w_1)$$

$$= p_1 W + T^*(p_2, \ldots, p_n, W - w_1) \qquad (E.Y)$$

$$\geq \min_i \{p_i W + T^*(p_1, \ldots, p_{i-1}, p_{i+1}, \ldots, p_n, W - w_i)\} \qquad (2)$$

$$\geq T^*(p_1, p_2, \ldots, p_n, W) \qquad (A.B)$$

- Θα δείξουμε την (2) χρησιμοποιώντας το επιχείρημα ανταλλαγής



Οι OPT και ΕΝΔΙΑΜΕΣΗ όπως φαίνεται διαφέρουν μόνο στις θέσεις του p_k και του p_i.

Τα A_1, A_2, A_3 περιέχουν τα υπόλοιπα (p_i, w_i) δηλαδή όλα εκτός (p_k, w_k) $\kappa\alpha\iota$ (p_1, w_1) και είναι ανά δύο ξένα μεταξύ τους.

Συμβολίζουμε με $c_{OPT}(A)$ τον επιμέρους χρόνο ολοκλήρωσης για το διάστημα Α της optimal λύσης (που προστίθεται στον τελικό). Όμοια ορίζουμε για την ενδιάμεση λύση $c_{EN\Delta}(A)$.

• $c_{OPT}(A_1) = c_{EN\Delta}(A_1)$, παίρνουμε ένα τυχαίο p_{i_m} : $1 \le m \le k-1$ στο A_1.

Τότε: $c_{OPT}(p_{i_m}) = p_{i_m}(W - \sum_{1}^{m-1} w_{i_m}) = c_{EN\Delta}(p_{i_m})$ διότι ο πίνακας OPT και ΕΝΔΙΑΜΕΣΟΣ ταυτίζονται σε κάθε θέση πριν την m-οστη για κάθε $1 \le m \le k-1$.

Το ζητούμενο προκύπτει διότι $c_{OPT}(A_1) = \sum_{m=1}^{k-1} c_{OPT}(p_{i_m})$ και $c_{EN\Delta}(A_1) = \sum_{m=1}^{k-1} c_{EN\Delta}(p_{i_m})$

• $c_{OPT}(A_3) = c_{EN\Delta}(A_3)$, παίρνουμε ένα τυχαίο p_{i_m} : $l+1 \le m \le n$ στο A_1.

Τ ό τ ε :

$$c_{OPT}(p_{i_m}) = p_{i_m}(W - \sum_{w_i \in A_1} w_i - w_k - \sum_{w_j \in A_2} w_j - w_1) = p_{i_m}(W - \sum_{w_i \in A_1} w_i - w_1 - \sum_{w_j \in A_2} w_j - w_k) = c_{EN\Delta}(p_{i_m})$$

Το ζητούμενο προκύπτει ομοίως με παραπάνω.

$$c_{OPT}(p_k) = p_k(W - \sum_{w_i \in A_1} w_i)$$

$$c_{EN\Delta}(p_k) = p_k(W - \sum_{w_i \in A_1} w_i - \sum_{w_j \in A_2} w_j - w_1)$$

$$c_{OPT}(p_1) = p_1(W - \sum_{w_i \in A_1} w_i - \sum_{w_j \in A_2} w_j - w_k)$$

$$c_{EN\Delta}(p_1) = p_1(W - \sum_{w_i \in A_1} w_i)$$

$$c_{OPT}(p_k) - c_{EN\Delta}(p_k) = p_k(\sum_{w_j \in A_2} w_j + w_1) =$$

$$c_{OPT}(p_1) - c_{EN\Delta}(p_1) = -p_1(\sum_{w_j \in A_2} w_j + w_k)$$

$$c_{OPT}(p_k) - c_{EN\Delta}(p_k) + c_{OPT}(p_1) - c_{EN\Delta}(p_1) = (p_k - p_1) \sum_{m=k+1}^{l-1} w_{i_m} + p_k w_1 - p_1 w_k \quad (3)$$

Παίρνουμε ένα τυχαίο p_{i_m} : $k+1 \leq m \leq l-1$ στο A_1, όπου l η θέση του p_1 στον OPT.

$$c_{OPT}(p_{i_m}) = p_{i_m}(W - \sum_{j=k+1}^{m-1} w_{i_j} - w_k - c_{OPT}(A_1))$$

$$c_{EN\Delta}(p_{i_m}) = p_{i_m}(W - \sum_{j=k+1}^{m-1} w_{i_j} - w_1 - c_{OPT}(A_1))$$

Οπότε:

$$C_{OPT}(A_2) - C_{EN\Delta}(A_2) = (w_1 - w_k) \sum_{m=k+1}^{l-1} p_{i_m} \quad (4)$$

Προσθέτοντας τις (3) και (4) έχουμε:

$$\sum_{m=k+1}^{l-1} p_k w_{i_m} - w_k p_{i_m} + \sum_{m=k+1}^{l-1} w_1 p_{i_m} - w_{i_m} p_1 + p_k w_1 - p_1 w_k$$

Αν πάρουμε το p_k να είναι το **στοιχείο με το μικρότερο** $\frac{w_i}{p_i}$ **αριστερά του** p_1 τότε

Το παραπάνω άθροισμα είναι θετικό διότι $\frac{w_1}{p_1} \geq \frac{w_{i_m}}{p_{i_m}}$ για κάθε m , $\frac{w_1}{p_1} \geq \frac{w_k}{p_k}$ και

$\frac{w_{i_m}}{p_{i_m}} \geq \frac{w_k}{p_k}$ $\forall m : k+1 \leq m \leq l-1$αφού επιλέξαμε το p_k να είναι αυτό με το μικρότερο

πηλίκο αριστερά του p_1.

$$T_{OPT} = c_{OPT}(A_1) + c_{OPT}(A_2) + c_{OPT}(A_3) + c_{OPT}(p_1) + c_{OPT}(p_k)$$
$$T_{EN\Delta} = c_{EN\Delta}(A_1) + c_{EN\Delta}(A_2) + c_{EN\Delta}(A_3) + c_{EN\Delta}(p_1) + c_{EN\Delta}(p_k)$$

Αφαιρώντας κατά μέλη και λαμβάνοντας υπόψιν τα παραπάνω έχουμε ότι:

$$T_{OPT} \geq T_{EN\Delta}$$

Αν εφαρμόσουμε την παραπάνω διαδικασία το πολύ l φορές (όπου l η θέση του p_1 στο OPT) μπορούμε να κατασκευάσουμε ένα $T'_{EN\Delta} \leq T_{EN\Delta} \leq T_{OPT}$ που θα έχει το p_1 στην πρώτη θέση.Έτσι μπορούμε για τον υπόλοιπο πίνακα μήκους n-1 να εφαρμόσουμε την Ε.Υ και να συμπεράνουμε ότι $T'_{EN\Delta} \geq T_{greedy}$.

<u>Σημείωση:</u> Επειδή το p_1 είναι το στοιχείο με τον μεγαλύτερο λόγο, σε κάθε ανταλλαγή έχει έρθει τουλάχιστον μία θέση πιο αριστερά.Αν αριστερά από το p_1 είναι τα στοιχεία σε φθίνουσα σειρά πηλίκων τότε θα χρειαστούμε ακριβώς l ανταλλαγές.

2β) Κάθε επιμέρους ουρά(που σχηματίζουν οι πελάτες στους υπάλληλους) σύμφωνα με το ερώτημα (2α) θα πρέπει να ικανοποιεί την συνθήκη ότι πελάτες με μεγαλύτερο πηλίκο θα πρέπει να εξυπηρετούνται πρώτοι στην ουρά τους. Αυτό όμως δεν μπορεί να γίνει με μοναδικό τρόπο για $m \geq 2$ και συνεπώς καταφεύγουμε σε δυναμικό προγραμματισμό.

Αν υποθέσουμε πάλι $\frac{w_n}{p_n} \leq \frac{w_{n-1}}{p_{n-1}} \leq \ldots \leq \frac{w_1}{p_1}$ (χωρίς βλάβη της γενικότητας) τότε

συμπεραίνουμε ότι σε κάθε ουρά η σειρά εισαγωγής γίνεται με αύξουσα σειρά δεικτών.

Για παράδειγμα η λύση (2,1) (3,4) σίγουρα δεν είναι καλύτερη από την βέλτιστη, διότι η λύση (1,2) (3,4) σύμφωνα με την παραπάνω διάταξη πηλίκων και το (2α) έχει μικρότερο ή ίσο συνολικό χρόνο ολοκλήρωσης.

Αν συμβολίσουμε με $C[i,p]$ τον βέλτιστο(μικρότερο) χρόνο ολοκλήρωσης για i πελάτες και p το άθροισμα των p_i στον πρώτο εξυπηρετητή, τότε έχουμε την εξής αναδρομική σχέση:

$$C[i,p] = \min\{C[i-1,p+p_i] + w_i(p+p_i), C[i-1,p] + w_i(\sum_{j=1}^{i} p_j - p)\} \quad (5)$$

Η πρώτη περίπτωση στο min είναι ο i-οστός πελάτης να μπει στον πρώτο εξυπηρετητή,ενώ η δεύτερη είναι να μπει στον δεύτερο.

Θέλουμε να υπολογίσουμε το $C[n,0]$ οπότε με DP θα πρέπει να υπολογίσουμε τιμές για έναν πίνακα $n\sum_{j=1}^{n} p_j$ θέσεων. Οπότε η πολυπλοκότητα είναι $O(n*sum)$ όπου sum το προηγούμενο άθροισμα. Για $m \geq 3$ η συνάρτηση C έχει m-1 παραμέτρους μετά το i που δηλώνουν τα αθροίσματα στους πρώτους m-1 εξυπηρετητές, και το min μέσα έχει m περιπτώσεις, μία για κάθε εξυπηρετητή. Συνεπώς με την ίδια υλοποίηση DP κατ'επέκταση ο αλγόριθμος για $m \geq 3$ έχει πολυπλοκότητα $O(n*(sum)^{m-1})$.Παρατηρούμε ότι για $m = 1$ έχουμε πολυπλοκότητα $O(n)$ που είναι ακριβώς όσο θέλουμε στον greedy με τον έναν εξυπηρετητή(αφού έχουμε ταξινομήσει τα πηλίκα).

Άσκηση 3

3α) Ορίζουμε ως $A(i)$ το ελάχιστο κόστος για να καλύψουμε από το σημείο $x_0 = 0$ μέχρι το x_i $i = 0,1,2,...,n$.Τότε το $A(i)$ ικανοποιεί την παρακάτω σχέση:

$$A(i) = \min_{0 \leq j \leq i} \{A(j-1) + (x_i - x_j)^2 + C\}$$

Με $A(-1) = 0$. Διαισθητικά η παραπάνω σχέση λέει ότι το ελάχιστο κόστος για να καλύψουμε μέχρι το x_i είναι το ελάχιστο κόστος να καλύψουμε μέχρι ένα σημείο $x_{j-1} < x_i$ συν το κόστος άλλου ένα στέγαστρου από το x_j έως το x_i.

Ο αλγόριθμος που υλοποιεί την παραπάνω αναδρομική σχέση με DP έχει χρονική πολυπλοκότητα $O(n^2)$ διότι πρέπει να γεμίσουμε έναν πίνακα μεγέθους n, και για κάθε

υπολογισμό πληρώνουμε $O(n)$ (λόγω του min που υπάρχει στην αρχή της αναδρομικής σχέσης).

3β) Πρέπει να κρατάμε πληροφορία για το κυρτό περίβλημα του συνόλου των ευθειών που μας δίνεται.Εφόσον οι ευθείες μας δίνονται κατά φθίνουσα σειρά κλίσης και τα σημεία σε αύξουσα σειρά μπορούμε να κάνουμε το εξής: Οι πρώτες δύο ευθείες (δηλαδή οι ευθείες με κλίσεις α_1 και α_2) θα μπουν σίγουρα. Για τις υπόλοιπες έχουμε: συγκρίνουμε την νέα ευθεία με την τελευταία ευθεία στο σύνολο σύμφωνα με το παρακάτω κριτήριο σύγκρισης: αν το σημείο τομής της με την τελευταία ευθεία είναι πιο αριστερά από ότι με την προτελευταία, τότε η τελευταία ευθεία "χάνει" γιατί θα είναι για κάθε x ανάμεσα στη νέα και στην προτελευταία, αλλιώς αν το σημείο τομής της νέας με την τελευταία είναι πιο δεξιά από ότι με την προτελευταία τότε δεν χάνει και μπαίνει στο τέλος.Ο αλγόριθμος αυτός για την εισαγωγή ευθειών είναι $O(n)$ amortized διότι (όπως και στο Cartesian tree) κάθε φορά που μια ευθεία "χάνει" απορρίπτεται και βγαίνει από το σύνολο οπότε δεν ξαναεξετάζεται, ενώ αν δεν χάνει τότε πληρώσαμε το κόστος μιας σύγκρισης.Με λίγα λόγια κάθε φορά που μια ευθεία "κερδίζει" έναντι κάποιας άλλης πληρώνουμε το κόστος μιας σύγκρισης, το οποίο όμως δεν θα το πληρώσουμε για την επόμενη ευθεία, γιατί όσες ευθείες "χάνουν" δεν θα ξανασυγκριθούν με επόμενες ευθείες.Παρομοίως για να βρούμε σε ποια ευθεία του κυρτού περιβλήματος ανήκει το κάθε σημείο, εφαρμόζουμε παρόμοια τακτική, δηλαδή: αφού τα σημεία είναι ταξινομημένα σε αύξουσα σειρά, αν το πρώτο σημείο δεν ελαχιστοποιείται από κάποια ευθεία του περιβλήματος, τότε αυτή η ευθεία για τα επόμενα σημεία δεν θα ξαναεξεταστεί γιατί είναι σίγουρο ότι αφού αυτά είναι μετά το σημείο δεν θα ανήκουν στην ευθεία που (ορθώς) απορρίψαμε. Τελικά έχουμε και εδώ $O(k)$ amortized.Τελικά για το συνολικό πρόβλημα έχουμε $O(n+k)$ πολυπλοκότητα.

Το παραπάνω εφαρμόζεται στο αρχικό μας πρόβλημα ως εξής:

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

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

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

Με $b_j = A(j-1) + x_j^2$ και $a_j = -2x_j$ η παραπάνω γράφεται:

$$A(i) = x_i^2 + C + \min_{0 \le j \le i} \{a_j x_i + b_j\}$$

Μετά από αυτό το πρόβλημα μπορεί να αναχθεί σε αυτό με τις ευθείες, διότι τώρα ψάχνουμε τις παραμέτρους a_i, b_i της ευθείας που ελαχιστοποιεί την τιμή της στο σημείο x_i.Συνεπώς η πολυπλοκότητα εφαρμόζοντας αυτήν την βελτιστοποίηση γίνεται $O(n)$.

Άσκηση 4

4α) Συμβολίζουμε ως $M[i,l]$ τον ελάχιστο συνολικό δείκτη ευαισθησίας για i φοιτητές και $1 \le l \le k$ λεωφορεία.Τότε για το $M[i,l]$ ισχύει η εξής αναδρομική σχέση.

$$M[i,l] = \min_{j<i}\{M[j,l-1] + cost[j+1,i]\}$$

$M[i,l] = 0$ αν $i \le l$ ή $l = 0$.

Όπου $cost[j,i]$ η προστιθέμενη ευαισθησία που προκύπτει αν έχουμε στο ίδιο λεωφορείο όλους τους φοιτητές από j έως i.Η παραπάνω σχέση μας λέει διαισθητικά ότι για την βέλτιστη λύση με i φοιτητές και l λεωφορεία πρέπει να βρούμε το ελάχιστο του να βάλουμε τους πρώτους j φοιτητές στα πρώτα l-1 λεωφορεία (αναδρομικά) και τους υπόλοιπους στο ίδιο, για όλα τα j μικρότερα του i.Ψάχνουμε το $M[n,k]$, οπότε κάνοντας το με DP πρέπει να βρούμε nk τιμές και για κάθε τιμή δαπανάμε O(n) χρόνο.Τελικά λοιπόν έχουμε πολυπλοκότητα $O(kn^2)$.

Σημειώνουμε ότι για να βρούμε την ανάθεση που πετυχαίνει το ελάχιστο, αρκεί να αρχίσουμε από το τέλος προς την αρχή και αναδρομικά (backtracking) να κοιτάμε στην προηγούμενη γραμμή του πίνακα που είμαστε (αφού είναι l-1) ποιο j έδωσε το ελάχιστο.Από το j+1 μέχρι το τέλος έχουμε όλους τους φοιτητές του τελευταίου λεωφορείου.Έτσι ,αναδρομικά, υπολογίζουμε την ανάθεση των φοιτητών στα λεωφορεία.

Για τον αλγόριθμο του να ελαχιστοποιήσουμε την μέγιστη ευαισθησία που μπορεί να έχει ένα λεωφορείο, μπορούμε να μετατρέψουμε το πρόβλημα στο εξής πρόβλημα απόφασης: Γίνεται να υπάρξει ανάθεση που η μέγιστη ευαισθησία των λεωφορείων είναι $\le$ B ;

$$B \in [0, \sum_{i \ne j} A_{ij}]$$

Τότε εκτελώντας binary search στο παραπάνω διάστημα: Αν μπορούμε να το κάνουμε με λιγότερο απο Β τότε πάμε στο αριστερό μισό, αλλιώς αν δεν μπορούμε να το κάνουμε με Β, πάμε στο δεξί μισό. Αυτό έχει πολυπλοκότητα $O(logn)$.Αυτό που μένει τώρα είναι να δώσουμε αλγόριθμο ο οποίος με δοσμένο ένα Β αποφασίζει αν υπάρχει ανάθεση που καταφέρνει μέγιστο δείκτη ευαισθησίας ανάμεσα στα λεωφορεία μικρότερο του Β.Ο αλγόριθμος αυτός είναι μια greedy προσέγγιση: βάζουμε κάθε φορά στην ουρά που θα έχει ως αποτέλεσμα συνολική ευαισθησία όσο το δυνατόν μικρότερη απο Β(με το σκεπτικό ότι αν είναι όσο το δυνατόν μικρότερη, δηλαδή ελάχιστη δημιουργούμε τον περισσότερο "χώρο" για τους επόμενους φοιτητές).Αν βάλουμε όλους τους φοιτητές χωρίς να υπερβούμε το Β επιστρέφουμε ναι, αν φτάσουμε σε σημείο που όπου και να προσθέσουμε θα υπερβαίνει το Β τότε σταματάμε και επιστρέφουμε όχι. Η πολυπλοκότητα είναι $O(n^2)$ γιατί για κάθε νέο φοιτητή που εισάγουμε πρέπει να υπολογίσουμε τις ευαισθησίες που προστίθενται λόγω των όλων των πρηγούμενων φοιτητών που έχουν ήδη εισέλθει σε κάποιο λεωφορείο.Δηλαδή για τον i-οστό φοιτητή υπολογίζουμε αθροίσματα που συμμετέχουν τα A_{ij} με $j < i$.Συνολικά έχουμε πολυπλοκότητα $O(n^2 logn)$.

Άσκηση 5

5α) Εφόσον η ακμή e ανήκει στο T_1 αλλά όχι στο T_2 τότε ανήκει σε κύκλο του G(διότι για παράδειγμα αν ήταν ακμή γέφυρα θα ανήκε αναγκαστικά και στα δύο δέντρα).Επίσης αφού δεν ανήκει στο T_2 θα πρέπει αντί για αυτή να υπάρχει μια άλλη ακμή e'(για να διατηρείται η συνεκτικότητα)που θα ανήκει στον ίδιο κύκλο του G με την e. Η e' δεν ανήκει στο T_1 γιατί αν ανήκε θα έκλεινε κύκλο στο T_1 μαζί με την e.

Εφόσον ανταλλάξαμε ακμές που ανήκουν στον ίδιο κύκλο η συνεκτικότητα δεν χάλασε οπότε το νέο δέντρο που φτιάξαμε είναι συνεκτικό. Ο αλγόριθμος έχει λοιπόν ως εξής:

Για μια δεδομένη ακμή e του T_1 την προσθέτουμε στο T_2, αυτή όπως προαναφέραμε κλείνει κύκλο οπότε με μια DFS εντοπίζουμε τον κύκλο που έκλεισε στο T_2 σε $O(n)$ αφού πρόκειται για δέντρο.Ελέγχουμε για κάθε ακμή στον κύκλο αν ανήκει στο T_1, αν δεν ανήκει την επιστρέφουμε, αλλιώς εξετάζουμε άλλη ακμή. Ο έλεγχος ύπαρξης ακμής σε δέντρο είναι $O(1)$ αν χρησιμοποιήσουμε parent tree, άρα για έναν κύκλο έχουμε $O(n)$.Οπότε τελικά η πολυπλοκότητα του αλγορίθμου είναι $O(n)$.

5β) Θέλουμε να δείξουμε ότι αν $|T_1 \setminus T_2| = k$ ανν $d(T_1, T_2) = k$ όπου d το μήκος του ελάχιστου μονοπατιού ανάμεσα στα T_1 και T_2 στο H.

Θα δείξουμε το ζητούμενο με επαγωγή:

Εξ'ορισμού έχουμε ότι $|T_1 \setminus T_2| = 1$ ανν $d(T_1, T_2) = 1$.

Υποθέτουμε με επαγωγή ότι $|T_1 \setminus T_2| = k$ ανν $d(T_1, T_2) = k$.

Ευθύ: Υποθέτουμε ότι $|T_1 \setminus T_2| = k$+1 . Έστω $e \in T_1 \setminus T_2$ τότε από (α) υπάρχει ακμή $e' \in T_2 \setminus T_1$ τέτοια ώστε το $T_1' = (T_1 \setminus \{e\}) \cup \{e'\} \in H$.

Όμως $|T_1' \setminus T_2| = k$ διότι προσθέσαμε μία ακμή που ανήκει στο T_2 και αφαιρέσαμε μία που δεν ανήκει.Το T_1' διαφέρει σε μία ακμή με το T_1 οπότε αφού ισχύει από Ε.Υ ότι $d(T_1', T_2) = k$, τότε $d(T_1, T_2) \leq k + 1$.

Αν $d(T_1, T_2) = k$ από Ε.Υ έχουμε ότι $|T_1 \setminus T_2| = k$, πράγμα που αντιτίθεται στην υπόθεση. Άρα $d(T_1, T_2) \geq k + 1$. Τελικά $d(T_1, T_2) = k + 1$.

Αντίστροφο: $d(T_1, T_2) = k + 1$ τότε υπάρχει ένα δέντρο T_1' στο ελάχιστο μονοπάτι για το οποίο ισχύει $d(T_1, T_1') = 1$ και $d(T_1', T_2) = k$. Άρα από Ε.Υ $|T_1' \setminus T_2| = k$. Άρα αφού με το T_1 διαφέρουν μόνο κατά μια ακμή έχουμε ότι $|T_1 \setminus T_2| = k - 1$ ή $|T_1 \setminus T_2| = k + 1$. Το πρώτο δεν γίνεται να ισχύει γιατί τότε από Ε.Υ θα έχουμε $d(T_1, T_2) = k - 1$.

Ο αλγόριθμος για το συντομότερο μονοπάτι έχει ως εξής:

Για κάθε ακμή $e \in (T_2 \setminus T_1)$ την προσθέτω στο δέντρο.

Αυτό εγγυάται ότι θα βρει συντομότερο μονοπάτι διότι αν $|T_2 \setminus T_1| = k$ τότε μετά από k βήματα έχουμε φτάσει στο T_2, με μονοπάτι μήκους k που όπως αποδείξαμε παραπάνω είναι το συντομότερο.Αποθηκεύουμε το T_1 ως parent tree σε $O(n)$ και για κάθε ακμή στο T_2 ελέγχουμε αν υπάρχει στο T_1 σε $O(1)$. Άρα τελικά για όλες τις ακμές έχουμε $O(n)$. Αυτή η διαδικασία θα επαναληφθεί k φορές οπότε τελικά έχουμε $O(kn)$ πολυπλοκότητα.

5γ) Η ιδέα είναι ότι με τον αλγόριθμο του Kruskal θα βρούμε το MST του T. Για κάθε ακμή που δεν ανήκει στο MST, πρέπει να βρούμε στον κύκλο που κλείνει όταν αυτή (αναγκαστικά) θα προστεθεί στο MST την βαρύτερη εκτός αυτής ακμή. Η ιδέα για να το κάνουμε σε πολυπλοκότητα $O(mlogm)$ συνολικά είναι να χρησιμοποιήσουμε την ιδέα των εκθετικών βημάτων όμως κάναμε στα RMQs και στο LCA πρόβλημα.Παρατηρούμε

ότι όλοι οι κόμβοι που συμμετέχουν στον κύκλο που κλείνει είναι σε επίπεδα χαμηλότερα από τον LCA των u και v (όπου e = (u,v)). Οπότε για να βρούμε την βαρύτερη ακμή εκτός της e στον κύκλο που δημιουργείται, θέλουμε την βαρύτερη ακμή στα μονοπάτια από το u-LCA(u,v) και v-LCA(u,v). Το max ανάμεσα στα δύο μονοπάτια είναι το max ανάμεσα στην ακμή μεγίστου βάρους για κάθε μονοπάτι. Αν αρχίσουμε από κάτω προς τα πάνω και εφαρμόσουμε την ιδέα των εκθετικών βημάτων τότε:

$w_{max}[u,2^{k+1}] = max\{w_{max}[u,2^k], w_{max}[ancest[u,2^k],2^k]\}$

Όπου $w_{max}[u,2^k]$ είναι η βαρύτερη ακμή στο μονοπάτι που αρχίζει στο u και τελειώνει 2^k επίπεδα παραπάνω. Η παραπάνω σχέση είναι αρκετά διαισθητική διότι λέει ότι η βαρύτερη ακμή σε μονοπάτι 2^{k+1} επίπεδα παραπάνω είναι η βαρύτερη ακμή ανάμεσα στο μονοπάτι 2^k επίπεδα παραπάνω και το μονοπάτι που αρχίζει από το τέλος του προηγούμενου (που στην ουσία είναι ο πρόγονος του u 2^k επίπεδα παραπάνω και συμβολίζεται με $ancest[u,2^k]$) και τελειώνει 2^k επίπεδα ακόμα παραπάνω από τον πρόγονο (σαφώς διότι $2^{k+1} = 2^k + 2^k$).Ο χρόνος υπολογισμού του παραπάνω είναι $O(nlogn)$ γιατί το κάνουμε για κάθε κόμβο του δέντρου και γιατί λόγω της ιδέας των εκθετικών βημάτων έχουμε $\lceil logn \rceil$ τιμές για το k.Για κάθε ακμή e = (u,v) αν έχουμε τον LCA(u,v) τότε η ακμή με το μέγιστο βάρος στο μονοπάτι u-v στο MST είναι η μέγιστη των δύο επιμέρους μεγίστων σε κάθε ένα από τα δύο μονοπάτια.Κάθε μήκος μονοπατιού μπορεί να αναλυθεί σε αθροίσματα δυνάμεων του 2 (προφανώς γιατί αυτό είναι η δυαδική αναπαράσταση του αριθμού του μήκους του μονοπατιού).Οπότε για να βρω το min για ένα μονοπάτι στην ουσία διασχίζω τον δυαδικό αριθμό και κάθε φορά μηδενίζω το πρώτο απο αριστερά μη μηδενικό bit. Αυτό έχει πολυπλοκότητα $O(logn)$, το μήκος της δυαδικής αναπαράστασης του n.Για δύο κόμβους u και v μπορούμε να κοιτάξουμε αν η μία είναι πρόγονος της άλλης σε σταθερό χρόνο, έχοντας την διάσχιση DFS και τους χρόνους επίσκεψης.Αν το u είναι πρόγονος του v ή ανάποδα τότε ο LCA ταυτίζεται με κάποιον από τους δύο. Αλλιώς πρέπει να βρούμε το επίπεδο που είναι πιο κοντά στη ρίζα (δηλαδή το επίπεδο 2^k με το μεγαλύτερο k) για το οποίο ο πρόγονος του ενός κόμβου σε αυτό το επίπεδο δεν είναι και του άλλου (και συνεπώς κανένας σε παρακάτω επίπεδο), ενώ στο επίπεδο 2^{k+1} είναι. Το αρχίζουμε από το μεγαλύτερο k και το μειώνουμε αναδρομικά, έχοντας πάλι πολυπλοκότητα $O(logn)$.Συνολικά έχουμε $O(mlogm)$ για Kruskal, $O(n)$ για DFS, $O(nlogn)$ για τα w_{max} και ancest και $O(mlogn)$ για το maximum query για κάθε ακμή. Τελικά $O(mlogm)$.