ΑΛΓΟΡΙΘΜΟΙ ΚΑΙ ΠΟΛΥΠΛΟΚΟΤΗΤΑ

2η ΣΕΙΡΑ ΓΡΑΠΤΩΝ ΑΣΚΗΣΕΩΝ

Αλέξανδρος Κουλάκος

03118144

ΣΗΜΜΥ, 7ο Εξάμηνο

Δεκέμβριος, 2021

Άσκηση 1

Χωρίς βλάβη της γενικότητας θεωρούμε ότι η ευθεία l είναι παράλληλη στον άξονα $x'x$, δηλαδή είναι της μορφής $y = c$ για κάποιο σταθερό αριθμό c. Η παραδοχή αυτή γίνεται καθαρά για λόγους ευκολίας στη γεωμετρική εποπτεία και δεν επηρεάζει καθόλου την ορθότητα των αλγορίθμων που θα διατυπωθούν στη συνέχεια.

Γνωρίζοντας τις συντεταγμένες (x_p, y_p) ενός σημείου p, μπορούμε πολύ εύκολα σε χρόνο $O(1)$ να υπολογίσουμε τα σημεία p_{left}, p_{right} που ανήκουν στην l και για τα οποία ισχύει ότι $d(p, p_{left}) = d(p, p_{right}) = r$ (ευκλείδειες αποστάσεις). Λόγω του γεγονότος ότι η l είναι παράλληλη στον $x'x$, τα σημεία p_{right}, p_{left} πρακτικά προσδιορίζονται από τη x συντεταγμένη τους, συνεπώς στη συνέχεια όποτε αναφερόμαστε σε σημεία στην πραγματικότητα αναφερόμαστε σε x συνιστώσες σημείων στο επίπεδο. Έχοντας προσδιορίσει τα σημεία p_{left}, p_{right}, παρατηρούμε εύκολα ότι για να ανήκει το p σε κάποιο δίσκο, πρέπει και αρκεί να υπάρχει δίσκος όπου το κέντρο του ανήκει στο διάστημα (p_{left}, p_{right}) με $p_{left} < p_{right}$.

Έτσι, βλέπουμε ότι κάθε σημείο (x_i, y_i) του επιπέδου από αυτά που μας ενδιαφέρουν μεταφράζεται σε ένα διάστημα (s_i, f_i), όπου s_i η αρχή του διαστήματος και f_i το τέλος του. Άρα, το πρόβλημα πλέον ανάγεται στην εύρεση του ελάχιστου πλήθους σημείων που καλύπτουν όλα τα διαστήματα (s_i, f_i), $i = 1,2, \dots, n$.

Άπληστος αλγόριθμος Έστω I το σύνολο που περιέχει όλα τα διαστήματα (s_i, f_i) ταξινομημένα σε αύξουσα σειρά ως προς τα f_i, δηλαδή $f_1 \leq f_2 \leq \cdots \leq f_n$. Ξεκινάμε με το διάστημα (s_1, f_1), δηλαδή αυτό που τελειώνει πρώτο. Επιλέγουμε ως σημείο το $x = f_1$ και ενημερώνουμε το πλήθος των σημείων ώστε $count = 1$. Βρίσκουμε όλα τα διαστήματα που επικαλύπτονται με το (s_1, f_1), δηλαδή όλα τα (s_i, f_i) για τα οποία $s_i \leq f_1$. Το σημείο $x = f_1$ ανήκει σε καθένα από αυτά τα διαστήματα, οπότε δεν υπάρχει λόγος να τα εξετάσουμε περαιτέρω. Στη συνέχεια βρίσκουμε το πρώτο διάστημα, έστω (s_k, f_k), που δεν επικαλύπτεται με το (s_1, f_1), δηλαδή $s_k > f_1$, επιλέγουμε ως σημείο το $x = f_k$ και ενημερώνουμε το πλήθος των σημείων αυξάνοντας τη μεταβλητή $count$ κατά 1. Η διαδικασία επαναλαμβάνεται μέχρι να έχουν

διατρεχθεί όλα τα διαστήματα που ανήκουν στο I. Τότε, ο αλγόριθμος επιστρέφει τη μεταβλητή x ως το πλήθος σημείων που χρειάζονται για να καλυφθούν όλα τα διαστήματα. Η άπληστη επιλογή του αλγορίθμου είναι ότι κάθε φορά που επιλέγεται ένα νέο διάστημα (s_i, f_i) η μεταβλητή x ενημερώνεται ώστε να αποθηκεύει την τιμή f_i, διότι το σημείο $x = f_i$ συμμετέχει σε όλες τις επικαλύψεις που έχει το (s_i, f_i) με διαστήματα από τα δεξιά του, άρα καλύπτει όσο το δυνατόν περισσότερα διαστήματα.

Απόδειξη ορθότητας

Έστω ότι ο άπληστος αλγόριθμος επιλέγει τα σημεία $G_1, G_2, \dots, G_k$ με $G_1 < G_2 < \cdots < G_k$ και ότι ο βέλτιστος αλγόριθμος επιλέγει τα σημεία $O_1, O_2, \dots, O_m$ με $O_1 < O_2 < \cdots < O_m$. Σκοπός μας είναι να δείξουμε ότι ο $GREEDY$ δε συμπεριφέρεται χειρότερα από τον OPT, ότι δηλαδή ο $GREEDY$ δεν επιλέγει περισσότερα σημεία από τον OPT. Δηλαδή, αρκεί να δείξουμε ότι $k \leq m$. Για να δείξουμε αυτό τον ισχυρισμό, πρώτα θα αποδείξουμε ότι $O_i \leq G_i$, για κάθε $i \leq m$, δηλαδή θα δείξουμε ότι ο OPT αλγόριθμος επιλέγει σημεία τα οποία δε βρίσκονται δεξιότερα από τα αντίστοιχα σημεία που επιλέγει ο $GREEDY$ αλγόριθμος. Η απόδειξη γίνεται επαγωγικά ως εξής:

• Βάση της επαγωγής: $O_1 \leq G_1$, διότι $G_1 = f_1$ και αν $O_1 > G_1 = f_1$, τότε το σημείο O_1 δεν καλύπτει το πρώτο διάστημα, οπότε δεν είναι εφικτή λύση (άτοπο, διότι ο OPT αλγόριθμος πάνω απ' όλα είναι ορθός).

• Επαγωγικό βήμα: Έστω $O_j \leq G_j$, για κάθε $j < m$, τότε θα δείξουμε ότι $O_m \leq G_m$. Έστω (s_i, f_i) το πρώτο διάστημα που δεν καλύπτεται από το G_{m-1}, δηλαδή το πρώτο διάστημα για το οποίο ισχύει $s_i > G_{m-1}$. Ο $GREEDY$ αλγόριθμος επιλέγει $G_m = f_i$ για να καλύψει το διάστημα (s_i, f_i). Όμως, $s_i > G_{m-1}$ και $G_{m-1} \geq O_{m-1}$ από επαγωγική υπόθεση, άρα $s_i > O_{m-1}$, δηλαδή το O_{m-1} δεν καλύπτει το διάστημα (s_i, f_i). Έτσι, ο OPT αλγόριθμος επιλέγει O_m τέτοιο ώστε να καλύπτει το (s_i, f_i), οπότε θα ισχύει $s_i \leq O_m \leq f_i$ και με δεδομένο ότι $G_m = f_i$ τελικά έχουμε ότι $O_m \leq G_m$ που είναι και το ζητούμενο.

Στη συνέχεια, αποδεικνύουμε ότι $k \leq m$. Έστω $k > m$, τότε G_{m+1} είναι το πρώτο επιπλέον σημείο που χρησιμοποιεί ο $GREEDY$ αλγόριθμος για να καλύψει το πρώτο διάστημα, π.χ. (s_i, f_i), από όλα όσα παραμένουν

ακάλυπτα. Για να χρειάζεται το σημείο G_{m+1} σημαίνει ότι το G_m δεν καλύπτει το (s_i, f_i), οπότε ισχύει $s_i > G_m$. Προφανώς, το (s_i, f_i) καλύπτεται από το σημείο O_m (ως το δεξιότερο) που επιλέγει ο OPT αλγόριθμος, οπότε ισχύει $s_i \leq O_m$. Συνδυάζοντας τις παραπάνω ανισότητες προκύπτει $G_m < O_m$ που είναι άτοπο, διότι παραπάνω αποδείξαμε ότι $O_i \leq G_i$, για κάθε $i \leq m$. Άρα, έχουμε ότι $k \leq m$.

Αυτό σημαίνει ότι ο $GREEDY$ αλγόριθμος δε μπορεί να έχει χειρότερη απόδοση από τον OPT αλγόριθμο και με δεδομένο ότι ο OPT είνα ο βέλτιστος, συμπεραίνουμε ότι και ο $GREEDY$ παράγει τη βέλτιστη λύση.

Η χρονική πολυπλοκότητα χειρότερης περίπτωσης του $GREEDY$ αλγορίθμου είναι $O(n + n\log n + n) = O(n\log n)$. Συγκεκριμένα, το πρώτο n οφείλεται στον υπολογισμό των n διαστημάτων, όπου κάθε υπολογισμός γίνεται σε $O(1)$. Το $n\log n$ οφείλεται στην ταξινόμηση των διαστημάτων σε αύξουσα σειρά κατά f_i (π.χ. επιλέγουμε τη $merge\ sort$ ως μέθοδο ταξινόμησης), ενώ το δεύτερο n οφείλεται στη σάρωση του συνόλου I που περιλαμβάνει όλα τα διαστήματα (s_i, f_i), έτσι ώστε να αποφανθούμε ποια επικαλύπτονται με το τρέχον διάστημα και ποιο είναι το πρώτο που δεν επικαλύπτεται ώστε να οριστεί αυτό ως τρέχον. Προφανώς, ο έλεγχος για επικάλυψη γίνεται σε $O(1)$ με τη συνθήκη $s_i \leq f_j$, όπου αυτό σημαίνει ότι το σημείο $x = f_j$ ανήκει στο (s_i, f_i).

Άσκηση 2

1.

Αναζητούμε αρχικά ένα άπληστο κριτήριο σύμφωνα με το οποίο σε κάθε βήμα του $GREEDY$ αλγορίθμου θα μπορούμε να επιλέξουμε ποιο πακέτο θα δρομολογηθεί. Το να ταξινομήσουμε τα πακέτα σε αύξουσα σειρά ως προς τα p_i είναι λάθος, διότι π.χ. για $(p_1, w_1) = (1, 1)$ και $(p_2, w_2) = (2, 5)$ αν δρομολογήσουμε πρώτα το πακέτο 1, ο συνολικός χρόνος είναι $T_1 = p_1(w_1 + w_2) + p_2 w_2 = 6 + 10 = 16$, ενώ αν δρομολογήσουμε πρώτα το πακέτο 2, ο συνολικός χρόνος είναι $T_2 = p_2(w_1 + w_2) + p_1 w_1 = 12 + 1 = 13$. Παρατηρούμε ότι $T_1 > T_2$, ενώ σύμφωνα με την ταξινόμηση θα έπρεπε $T_1 < T_2$. Με ενετελώς αντίστοιχο τρόπο μπορούμε να απορρίψουμε την ιδέα να ταξινομήσουμε τα πακέτα σε αύξουσα σειρά ως προς τα w_i. Έτσι, θεωρούμε τις ποσότητες $T_1 = p_1(w_1 + w_2) + p_2 w_2$ και $T_2 = p_2(w_1 + w_2) + p_1 w_1$ και αναρωτιόμαστε για ποιους συνδυασμούς των (p_1, w_1) και (p_2, w_2) ισχύει ότι

$$T_1 < T_2 \Rightarrow$$

$$p_1(w_1 + w_2) + p_2 w_2 < p_2(w_1 + w_2) + p_1 w_1 \Rightarrow$$

$$p_1 w_2 < p_2 w_1 \Rightarrow$$

$$\frac{w_1}{p_1} > \frac{w_2}{p_2}$$

Βρήκαμε, λοιπόν, ένα ενδεικτικό μέγεθος $\frac{w_i}{p_i}$ το οποίο θα ονομάζουμε βάρος ανά μονάδα χρόνου και που υπαγορεύει ποια είναι η καλύτερη επιλογή δρομολόγησης ανάμεσα σε δύο πακέτα. Επαγωγικά, αυτό το κριτήριο επιλογής εφαρμόζεται και για τη δρομολόγηση ανάμεσα σε n πακέτα.

Άπληστος αλγόριθμος Έστω A το σύνολο που περιέχει όλα τα ζεύγη (p_i, w_i). Ταξινομούμε το A σε φθίνουσα σειρά ως προς το πηλίκο $\frac{w_i}{p_i}$, οπότε το A έχει τη μορφή $A = \{(p_1, w_1), (p_2, w_2), \dots, (p_n, w_n)\}$ με $\frac{w_1}{p_1} > \frac{w_2}{p_2} > \cdots > \frac{w_n}{p_n}$. Η λύση που παράγει ο $GREEDY$ αλγόριθμος περιγράφεται από την παρακάτω αναδρομική σχέση:

$$GREEDY(A) = p_1 \sum_{k=1}^{n} w_k + GREEDY(A - \{(p_1, w_1)\})$$

Δηλαδή, ο $GREEDY$ αλγόριθμος επιλέγει κάθε φορά το πρώτο στοιχείο του A (το οποίο λόγω της ταξινόμησης έχει το μεγαλύτερο βάρος ανά μονάδα χρόνου).

Αντίθετα, ο OPT αλγόριθμος δοκιμάζει όλα τα (p_i, w_i) και από αυτά επιλέγει το καλύτερο. Η βέλτιστη λύση που παράγει περιγράφεται από την παρακάτω αναδρομική σχέση:

$$OPT(A) = \min_i \left\{ p_i \sum_{k=1}^{n} w_k + OPT(A - \{(p_i, w_i)\}) \right\}$$

Στη συνέχεια, θα αποδείξουμε ότι η λύση του $GREEDY$ αλγορίθμου είναι βέλτιστη. Αρκεί να δείξουμε, λοιπόν, ότι ο $GREEDY$ δε συμπεριφέρεται χειρότερα από τον OPT, δηλαδή $GREEDY(A) \leq OPT(A)$

Απόδειξη ορθότητας (με αριθμητική επαγωγή στο $n = |A|$)

• Βάση της επαγωγής: Αν $n = 1$, τότε ο OPT και $GREEDY$ συμπεριφέρονται με τον ίδιο τρόπο και συγκεκριμένα ισχύει $OPT\big((p_1, w_1)\big) = GREEDY\big((p_1, w_1)\big) = p_1 w_1$

• Επαγωγικό βήμα: Έστω $GREEDY(A') \leq OPT(A')$, για κάθε σύνολο A' με $|A'| < n$. Τότε,

$$GREEDY(A) = p_1 \sum_{k=1}^{n} w_k + GREEDY(A - \{(p_1, w_1)\}) \leq$$

$$p_1 \sum_{k=1}^{n} w_k + OPT(A - \{(p_1, w_1)\}) \leq$$

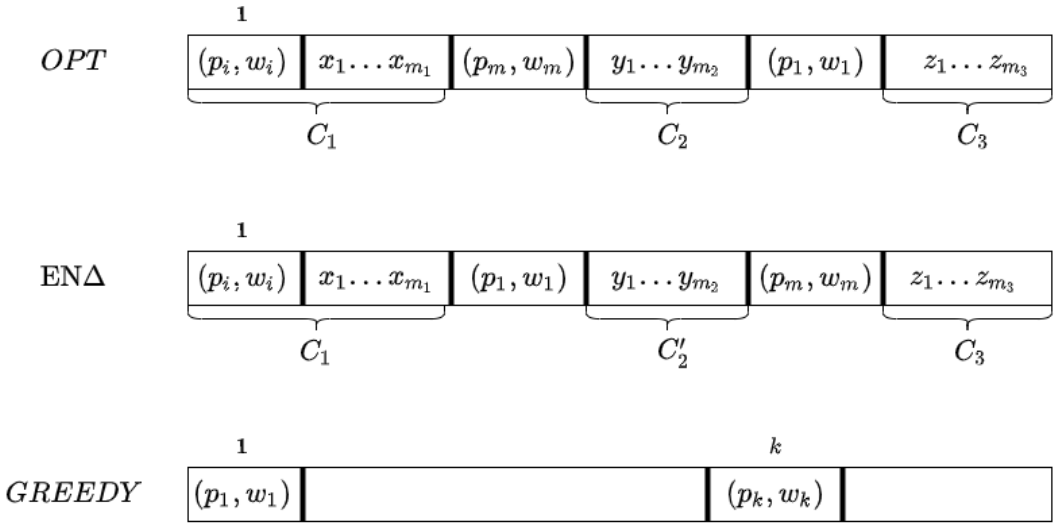
$$\min_i \left\{ p_i \sum_{k=1}^{n} w_k + OPT(A - \{(p_i, w_i)\}) \right\} =$$

$$OPT(A)$$

Στο πρώτο βήμα της απόδειξης επικαλεστήκαμε την επαγωγική υπόθεση, διότι $|A - \{(p_1, w_1)\}| = n - 1 < n$. Στο δεύτερο βήμα επικαλεστήκαμε το επιχείρημα ανταλλαγής και στο τρίτο βήμα την αρχή της βελτιστότητας.

Μένει να αποδείξουμε την ισχύ του επιχειρήματος ανταλλαγής. Προς το σκοπό αυτό, θεωρούμε έναν «ενδιάμεσο» υποθετικό αλγόριθμο τον οποίο συμβολίζουμε με $ΕΝΔ$. Στόχος του $ΕΝΔ$ είναι να «παντρέψει» τις επιλογές των $GREEDY$ και OPT. Αυτό μπορεί να γίνει αν θεωρήσουμε ότι ο $ΕΝΔ$ κάθε φορά ανταλλάσσει το ζεύγος (p_1, w_1) με το ζεύγος (p_m, w_m) που βρίσκεται τοπολογικά στα αριστερά του σύμφωνα με τη σειρά κατά την οποία επιλέγει ζευγάρια ο OPT αλγόριθμος και δίνει το ελάχιστο πηλίκο $\frac{w}{p}$. Η σειρά με την οποία κάθε αλγόριθμος επιλέγει ζεύγη μοντελοποιείται σαν ένας μονοδιάστατος πίνακας n θέσεων που διαβάζεται από αριστερά προς τα δεξιά.

Τα παραπάνω συνοψίζονται στο ακόλουθο σχήμα:



Με $i, x_1, \dots, x_{m_1}, y_1, \dots, y_{m_2}, z_1, \dots, z_{m_3}$ έχουμε συμβολίσει στο σχήμα τους δείκτες των κοινών ζευγών που επιλέγουν οι αλγόριθμοι $ΕΝΔ$ και OPT στα ίδια βήματα. Με (p_m, w_m) έχουμε συμβολίσει το ζεύγος που βρίσκεται «αριστερά» από το (p_1, w_1), δηλαδή επιλέγεται νωρίτερα από το (p_1, w_1) στον OPT αλγόριθμο, και δίνει το ελάχιστο πηλίκο $\frac{w}{p}$. Με C_1

έχουμε συμβολίσει το χρόνο καθενός από τους OPT και $EN\Delta$ αλγορίθμους για τη δρομολόγηση των πακέτων $j \in \{i, x_1, \dots, x_{m_1}\}$. Με C_3 έχουμε συμβολίσει το χρόνο καθενός από τους OPT και $EN\Delta$ αλγορίθμους για τη δρομολόγηση των πακέτων $j \in \{z_1, \dots, z_{m_3}\}$. Με C_2 έχουμε συμβολίσει το χρόνο του OPT αλγορίθμου για τη δρομολόγηση των πακέτων $j \in \{y_1, \dots, y_{m_2}\}$, ενώ με C_2' έχουμε συμβολίσει το χρόνο του $EN\Delta$ αλγορίθμου για τη δρομολόγηση των ίδιων πακέτων. Σημειώνουμε ότι $C_2' \neq C_2$, διότι $m \neq 1$, οπότε τα w_1 και w_m έχουν διαφορετική συνεισφορά σε κάθε αλγόριθμο (π.χ. στον OPT ο όρος w_m εμφανίζεται λιγότερες φορές απ' ότι στον $EN\Delta$, ενώ στον $EN\Delta$ ο όρος w_1 εμφανίζεται λιγότερες φορές απ' ότι στον OPT. Παρολαυτά, οι χρόνοι C_1, C_3 είναι κοινοί και για τους δύο αλγορίθμους, διότι κατά τη δρομολόγηση των πακέτων $j \in \{i, x_1, \dots, x_{m_1}\} \cup \{z_1, \dots, z_{m_3}\}$ τόσο ο OPT όσο και ο $EN\Delta$ έχουν δρομολογήσει τα ίδια πακέτα στα αριστερά τους. Οι χρόνοι C_2 και C_2' συσχετίζονται αν παρατηρήσουμε ότι ισχύει:

$$C_2 + w_m \sum_{j=1}^{m_2} p_{y_j} = C_2' + w_1 \sum_{j=1}^{m_2} p_{y_j} \Rightarrow C_2 - C_2' = (w_1 - w_m) \sum_{j=1}^{m_2} p_{y_j}$$

Θα αποδείξουμε ότι $T_{GREEDY} \leq T_{EN\Delta} \leq T_{OPT}$

• Αποδεικνύουμε πρώτα ότι $T_{EN\Delta} \leq T_{OPT}$. Τότε, ισχύουν οι παρακάτω ισότητες:

$$T_{OPT} = C_1 + p_m \left(w_m + w_1 + \sum_{j=1}^{m_2} w_{y_j} + \sum_{j=1}^{m_3} w_{z_j} \right) + C_2 + p_1 \left(w_1 + \sum_{j=1}^{m_3} w_{z_j} \right) + C_3$$

$$T_{EN\Delta} = C_1 + p_1 \left(w_1 + w_m + \sum_{j=1}^{m_2} w_{y_j} + \sum_{j=1}^{m_3} w_{z_j} \right) + C_2' + p_m \left(w_m + \sum_{j=1}^{m_3} w_{z_j} \right) + C_3$$

Αφαιρώντας τις παραπάνω σχέσεις κατά μέλη έχουμε:

$$T_{OPT} - T_{EN\Delta} = p_m w_1 - p_1 w_m + (p_m - p_1) \sum_{j=1}^{m_2} w_{y_j} + (w_1 - w_m) \sum_{j=1}^{m_2} p_{y_j}$$

$$T_{OPT} - T_{EN\Delta} = p_m w_1 - p_1 w_m + \sum_{j=1}^{m_2} \left(w_1 p_{y_j} - p_1 w_{y_j} \right) + \sum_{j=1}^{m_2} \left(p_m w_{y_j} - w_m p_{y_j} \right)$$

Λόγω της ταξινόμησης ισχύει $\frac{w_1}{p_1} \geq \frac{w_k}{p_k} \Rightarrow p_k w_1 - p_1 w_k \geq 0$, για κάθε $k = 1, 2, \dots, n$. Θέτοντας $k = m$ έχουμε $p_m w_1 - p_1 w_m \geq 0$ και θέτοντας $k = y_j$ για τυχαίο $j \in \{1, \dots, m_2\}$ έχουμε $w_1 p_{y_j} - p_1 w_{y_j} \geq 0$. Άρα, προκύπτει $\sum_{j=1}^{m_2} \left(w_1 p_{y_j} - p_1 w_{y_j}\right) \geq 0$. Άρα, για να δείξουμε ότι $T_{OPT} \geq T_{ENΔ}$, αρκεί να δείξουμε ότι $\sum_{j=1}^{m_2} \left(p_m w_{y_j} - w_m p_{y_j}\right) \geq 0$, δηλαδή αρκεί να δείξουμε ότι $p_m w_{y_j} - w_m p_{y_j} \geq 0$, για κάθε $j \in \{1, \dots, m_2\}$. Αυτό προφανώς ισχύει, διότι το ζεύγος (p_m, w_m) έχει το ελάχιστο πηλίκο $\frac{w}{p}$ από όλα τα ζεύγη που προηγούνται του (p_1, w_1). Σε αυτά τα ζεύγη ανήκουν και τα $\left(p_{y_j}, w_{y_j}\right)$ με $j \in \{1, \dots, m_2\}$, οπότε $\frac{w_m}{p_m} \leq \frac{w_{y_j}}{p_{y_j}} \Rightarrow$ $p_m w_{y_j} - w_m p_{y_j} \geq 0$, για κάθε $j \in \{1, \dots, m_2\}$. Συνεπώς, αποδείξαμε ότι $T_{OPT} \geq T_{ENΔ}$, ότι δηλαδή αν ανταλλάξουμε το (p_1, w_1) με το ζεύγος (p_m, w_m) που ελαχιστοποιεί το πηλίκο $\frac{w}{p}$ στα «αριστερά» του (p_1, w_1), τότε ο αλγόριθμος $ENΔ$ δεν παράγει χειρότερο κόστος.

• Στη συνέχεια αποδεικνύουμε ότι $T_{GREEDY} \leq T_{ENΔ}$. Είναι φανερό ότι αν καταφέρουμε να φέρουμε το ζεύγος (p_1, w_1) στην πρώτη θέση του $ENΔ$, τότε το ζητούμενο ισχύει, διότι ο $GREEDY$ συμφωνεί με τον $ENΔ$ στην επιλογή του πρώτου στοιχείου και για τα υπόλοιπα $n - 1 < n$ στοιχεία ο $GREEDY$ συμπεριφέρεται όπως και ο OPT (από επαγωγική υπόθεση). Όμως, ο $ENΔ$ δε μπορεί να συμπεριφέρεται καλύτερα από τον OPT, άρα και από τον $GREEDY$, οπότε θα ισχύει $T_{ENΔ} \geq T_{GREEDY}$. Άρα, αρκεί να δείξουμε ότι σε κάποιο στάδιο το (p_1, w_1) θα επιλεγεί πρώτο από τον $ENΔ$. Αυτό είναι πάλι εύκολο να δειχθεί αν θυμηθούμε ότι το πηλίκο $\frac{w_1}{p_1}$ είναι το μεγαλύτερο από όλα τα πακέτα, συνεπώς θα υπάρχει πάντα στα αριστερά του ζεύγος (p_m, w_m) με $\frac{w_m}{p_m} < \frac{w_1}{p_1}$ που δημιουργεί έγκυρη ανταλλαγή, οπότε σίγουρα μετά από κάποιες εφαρμογές του αλγορίθμου $ENΔ$ το ζεύγος (p_1, w_1) θα έρθει στην πρώτη θέση.

Αποδεικνύοντας ότι $T_{GREEDY} \leq T_{ENΔ} \leq T_{OPT}$ τελικά λαμβάνουμε $T_{GREEDY} \leq T_{OPT}$, δηλαδή ο $GREEDY$ δεν μπορεί να είναι χειρότερος από το βέλτιστο, οπότε είναι αναγκαστικά βέλτιστος και στο σημείο αυτό ολοκληρώνεται η απόδειξη της ισχύος του επιχειρήματος ανταλλαγής, άρα και η απόδειξη της ορθότητας.

Η χρονική πολυπλοκότητα χειρότερης περίπτωσης του $GREEDY$ αλγορίθμου είναι $O(n \log n + n) = O(n \log n)$. Συγκεκριμένα, το $n \log n$ οφείλεται στην ταξινόμηση των ζευγών σε φθίνουσα σειρά κατά $\frac{w_i}{p_i}$ (π.χ. επιλέγουμε τη $merge\ sort$ ως μέθοδο ταξινόμησης), ενώ το n οφείλεται στην ενημέρωση του βεβαρυμένου χρόνου εξυπηρέτησης μέσω της σάρωσης του συνόλου A που περιλαμβάνει όλα τα ζεύγη (p_i, w_i).

2.

Αρχικά, θα λύσουμε το πρόβλημα έχοντας διαθέσιμους δύο υπαλλήλους στην εξυπηρέτηση. Συγκεκριμένα, θεωρούμε πίνακα $T(i,p)$ ο οποίος εκφράζει το βέλτιστο χρόνο δρομολόγησης των πρώτων i πακέτων όταν ο πρώτος υπάλληλος έχει μέχρι στιγμής αφιερώσει χρόνο p για τη δρομολόγηση των πακέτων που έχει αναλάβει. Τότε, είναι προφανές ότι ο χρόνος p' που έχει αφιερώσει μέχρι στιγμής ο δεύτερος υπάλληλος για τη δρομολόγηση των πακέτων που έχει αναλάβει είναι $p' = \sum_{j=1}^{n} p_j - p$, δηλαδή ο δεύτερος υπάλληλος αναλαμβάνει τα πακέτα που περισσεύουν. Έτσι, μένει να αποφασίσουμε σε ποιον υπάλληλο θα αναθέσουμε το πακέτο i με βάση τη συνεισφορά του εν λόγω πακέτου στον τρέχοντα βεβαρυμένο χρόνο εξυπηρέτησης. Η αναδρομική σχέση που περιγράφει τη βέλτιστη λύση του προβλήματος φαίνεται παρακάτω:

$$T(i,p) = \min \begin{Bmatrix} T(i-1, p+p_i) + w_i(p+p_i) \\ T(i-1,p) + w_i\left(\sum_{j=1}^{n} p_j - p + p_i\right) \end{Bmatrix}$$

Ο πρώτος κλάδος αναφέρεται στα πακέτα που αναλαμβάνει ο πρώτος υπάλληλος (και για αυτό το p αυξάνεται κατά p_i), ενώ ο δεύτερος κλάδος αναφέρεται στα πακέτα που αναλαμβάνει ο δεύτερος υπάλληλος (και για αυτό το λόγο το p παραμένει ως έχει). Οι όροι $w_i(p+p_i)$ και $w_i\left(\sum_{j=1}^{n} p_j - p + p_i\right)$ αποτελούν τη συνεισφορά κάθε πακέτου στον τρέχοντα βεβαρυμένο χρόνο εξυπηρέτησης (ανάλογα με τον υπάλληλο από τον οποίο αναλαμβάνονται).

Ο πίνακας $T(i,p)$ συμπληρώνεται σύμφωνα με τη λογική του δυναμικού προγραμματισμού (το base case δίνεται για $i = 1$, όπου $T(1,p) = p_1 w_1$, για κάθε p). Η λύση του προβλήματος δίνεται από την τιμή $T(n,0)$. Η χρονική πολυπλοκότητα του αλγορίθμου εκφράζεται από το χρόνο που χρειάζεται για να συμπληρώσουμε τον πίνακα T. Ο πίνακας αυτός έχει στη χειρότερη περίπτωση $n \cdot sum$ θέσεις, όπου sum το μέγιστο άθροισμα που μπορεί να αποθηκευτεί στο p, δηλαδή $sum = \sum_{j=1}^{n} p_j$, οπότε ο αλγόριθμος χρειάζεται χρόνο $O(n \cdot sum)$

Στη συνέχεια, θα επιχειρήσουμε να γενικεύσουμε τη λύση του προβλήματος όταν έχουμε $m \geq 3$ υπαλλήλους διαθέσιμους στην

εξυπηρέτηση. Αξίζει να σημειωθεί ότι για να μπορούμε να κατασκευάσουμε τη βέλτιστη αναδρομική σχέση που λύνει το πρόβλημα πρέπει οπωσδήποτε να ξέρουμε τους χρόνους που έχουν αφιερώσει τουλάχιστον οι $m-1$ υπάλληλοι στη δρομολόγηση των πακέτων που έχουν αναλάβει μέχρι στιγμής. Έστω $p_{x_1}, \dots, p_{x_{m-1}}$ αυτοί οι χρόνοι. Τότε, ο χρόνος p_{x_m} που έχει αφιερώσει μέχρι στιγμής ο υπάλληλος m για τη δρομολόγηση των πακέτων που έχει αναλάβει είναι $p_{x_m} = \sum_{j=1}^{n} p_j - \sum_{j=1}^{m-1} p_{x_j}$, δηλαδή ο υπάλληλος m αναλαμβάνει τα πακέτα που περισσεύουν. Οπότε, η συνάρτηση – πίνακας $T(i,p)$ γενικεύεται σε $T\left(i, p_{x_1}, \dots, p_{x_{m-1}}\right)$. Ανάλογα γενικεύεται και η χρονική πολυπλοκότητα του αλγορίθμου σε $O(n \cdot sum^{m-1})$. Αξίζει να σημειώσουμε ότι για $m = 1$ υπάλληλο η πολυπλοκότητα εκφυλίζεται σε $O(n)$ και ο αλγόριθμος ταυτίζεται πλέον με τον $GREEDY$ που εφαρμόσαμε στο ερώτημα 1.

Σχόλιο Σε όλα τα παραπάνω εξακολουθεί να ισχύει η ταξινόμηση των $\frac{w_i}{p_i}$ σε φθίνουσα σειρά, ειδάλλως οι αναδρομικές σχέσεις που ορίστηκαν δεν παράγουν τη βέλτιστη λύση. Για αυτό το λόγο, τυπικά στις παραπάνω πολυπλοκότητες πρέπει να προσθέσουμε και το $O(n \log n)$ της ταξινόμησης.

Άσκηση 3

$(\boldsymbol{\alpha})$

Έστω $C(i)$ συνάρτηση που δίνει το ελάχιστο κόστος για την εγκατάσταση στεγάστρων που καλύπτουν τα σημεία $x_0, x_1, \dots, x_i$. Τότε, για να βρούμε τη βέλτιστη λύση, αρκεί να ξεκινήσουμε από το σημείο x_f και να αναζητήσουμε εκείνο το σημείο x_j το οποίο αν χρησιμοποιηθεί ως η αρχή του τελευταίου στεγάστρου ελαχιστοποιεί το συνολικό κόστος εγκατάστασης. Έτσι, η $C(i)$ δίνεται από την παρακάτω αναδρομική σχέση:

$$C(i) = \begin{cases} \min\limits_{0 \le j \le i} \left\{ C(j-1) + \left(x_i - x_j\right)^2 + C \right\}, & i \ge 0 \\ 0, & i = -1 \end{cases}$$

Επιλέγουμε $C(-1) = 0$, διότι αριστερά από το σημείο $x_0 = 0$ παύει να εκτείνεται πλέον ο πεζόδρομος, οπότε δεν απαιτούνται στέγαστρα. Η λύση του προβλήματος δίνεται από το αποτέλεσμα που επιστρέφει η συνάρτηση $C(n)$. Για να αποφεύγονται οι επαναλαμβανόμενοι υπολογισμοί ίδιων τιμών $C(i)$, αποθηκεύουμε αυτές τις τιμές σε έναν πίνακα $C'[0..n]$ με χρήση δυναμικού προγραμματισμού. Η χρονική πολυπλοκότητα του αλγορίθμου ταυτίζεται με το χρόνο που απαιτείται για το γέμισμα του πίνακα C' ο οποίος εκτιμάται με βάση το πλήθος των συνολικών συγκρίσεων. Άρα, η χρονική πολυπλοκότητα είναι $\Theta(0 + 1 + 2 + \cdots + n) = \Theta(n^2)$, διότι για τον υπολογισμό του $C(i)$ απαιτούνται i συγκρίσεις.

($\boldsymbol{\beta}$)

Αναζητούμε μια κατάλληλη ταξινόμηση των ευθειών της μορφής $\{(a_{x_1}, b_{x_1}), (a_{x_2}, b_{x_2}), \dots, (a_{x_\kappa}, b_{x_\kappa})\}$, με $k \leq n$, έτσι ώστε το σημείο τομής των ευθειών $(a_{x_1}, b_{x_1}), (a_{x_2}, b_{x_2})$ να βρίσκεται ψηλότερα (να έχει δηλαδή μεγαλύτερη τεταγμένη) από το σημείο τομής των ευθειών $(a_{x_2}, b_{x_2}), (a_{x_3}, b_{x_3})$, το σημείο τομής των ευθειών $(a_{x_2}, b_{x_2}), (a_{x_3}, b_{x_3})$ να βρίσκεται ψηλότερα από το σημείο τομής των ευθειών $(a_{x_3}, b_{x_3}), (a_{x_4}, b_{x_4})$ κοκ.

Η παραπάνω δομή ονομάζεται κυρτό περίβλημα και στη συνέχεια θα ασχοληθούμε με τη χρονική πολυπλοκότητα κατασκευής της. Έστω ότι η δομή μέχρι στιγμής έχει τη μορφή $\{(a_{x_1}, b_{x_1}), (a_{x_2}, b_{x_2}), \dots, (a_{x_{k-1}}, b_{x_{k-1}})\}$ και εμείς θέλουμε να προσθέσουμε σε αυτή την ευθεία (a_{x_k}, b_{x_k}). Τότε, βρίσκουμε το σημείο τομής των ευθειών $(a_{x_{k-1}}, b_{x_{k-1}})$, (a_{x_k}, b_{x_k}) και $(a_{x_{k-2}}, b_{x_{k-2}})$, (a_{x_k}, b_{x_k}) και ελέγχουμε αν το πρώτο βρίσκεται χαμηλότερα (δηλαδή έχει μικρότερη τεταγμένη) από το δεύτερο. Αν ισχύει αυτό, τότε η ευθεία (a_{x_k}, b_{x_k}) προστίθεται στο τέλος της δομής. Διαφορετικά, η ευθεία $(a_{x_{k-1}}, b_{x_{k-1}})$ διαγράφεται από τη δομή και επαναλαμβάνουμε την ίδια διαδικασία ελέγχου για την ενημερωμένη δομή.

Η πολυπλοκότητα κατασκευής του κυρτού περιβλήματος μετριέται με βάση τις συγκρίσεις που γίνονται για τις τεταγμένες των σημείων τομής των ευθειών. Σε κάθε σύγκριση υπάρχει μια ευθεία που «χάνει» (και είναι αυτή που εκτοπίζεται) και μια ευθεία που «κερδίζει» (και είναι αυτή που προχωρά για περαιτέρω ελέγχους). Κάθε ευθεία δικαιούται να «χάσει» το πολύ μια φορά (άπαξ και εκτοπιστεί από το κυρτό περίβλημα δεν ξαναεισάγεται σε αυτό) και με δεδομένο ότι δίνονται n τέτοιες ευθείες, θα υπάρχουν κατά μέσο όρο n συγκρίσεις που οδηγούν σε «χαμένο» και «κερδισμένο». Συνεπώς, η χρονική πολυπλοκότητα που απαιτείται για την ορθή κατασκευή του κυρτού περιβλήματος είναι $\Theta(n)$.

Έστω ότι η τελική εκδοχή του κυρτού περιβλήματος είναι η $\{(a_{x_1}, b_{x_1}), (a_{x_2}, b_{x_2}), \dots, (a_{x_\kappa}, b_{x_\kappa})\}$, με $k \leq n$. Έχοντας το κυρτό περίβλημα στη διάθεσή μας, μπορούμε να παρατηρήσουμε ότι κάθε ευθεία (a_{x_m}, b_{x_m}) που ανήκει στο κυρτό περίβλημα ορίζει ένα

«διάστημα ελαχιστοποίησης» της μορφής (x_l, x_r), όπου x_l η τετμημένη του σημείου τομής των ευθειών $\left(a_{x_{m-1}}, b_{x_{m-1}}\right)$, $\left(a_{x_m}, b_{x_m}\right)$ και x_r η τετμημένη του σημείου τομής των ευθειών $\left(a_{x_m}, b_{x_m}\right)$, $\left(a_{x_{m+1}}, b_{x_{m+1}}\right)$. Αν $m = 1$, τότε το διάστημα γίνεται $(-\infty, x_r)$, ενώ αν $m = k$ το διάστημα γίνεται $(x_l, +\infty)$. Έτσι, με βάση την αρχή κατασκευής του κυρτού περιβλήματος συμπεραίνουμε ότι η ευθεία $\left(a_{x_m}, b_{x_m}\right)$ ελαχιστοποιεί το $y_i = ax_i + b$ αν και μόνο αν $x_i \in (x_l, x_r)$. Για να βρούμε ποια ευθεία ελαχιστοποιεί την τιμή στο $x = x_i$ αρκεί να διατρέξουμε το κυρτό περίβλημα ξεκινώντας από τα δεξιά και για κάθε ευθεία $\left(a_{x_m}, b_{x_m}\right)$ που συναντάμε σε αυτό, να ελέγχουμε αν το x_i ανήκει στο «διάστημα ελαχιστοποίησης» που ορίζει αυτή η ευθεία. Αν ισχύει αυτό, τότε επιστρέφουμε την ευθεία $\left(a_{x_m}, b_{x_m}\right)$ ως την ευθεία που ελαχιστοποιεί το $y_i = ax_i + b$. Διαφορετικά, λόγω της αύξουσας ταξινόμησης των x_i, ξέρουμε ότι κανένα x_j με $j > i$ δεν πρόκειται να ανήκει σε αυτό το διάστημα ελαχιστοποίησης και προφανώς σε κανένα προηγούμενό του. Έτσι, γνωρίζοντας την ευθεία $\left(a_{x_m}, b_{x_m}\right)$ που ελαχιστοποιεί το $y_i = ax_i + b$, για να βρούμε την ευθεία που ελαχιστοποιεί το $y_{i+1} = ax_{i+1} + b$, δε χρειάζεται να διατρέξουμε το κυρτό περίβλημα από την αρχή. Αρκεί οι έλεγχοι να ξεκινήσουν από την ευθεία $\left(a_{x_m}, b_{x_m}\right)$ και να προχωρήσουν προς τα δεξιά. Λόγω αυτής της παρατήρησης βλέπουμε ότι για να καθορίσουμε ποια ευθεία ελαχιστοποιεί το $y_i = ax_i + b$ απαιτείται $amortized$ χρόνος $\Theta(1)$. Άρα, για τα k σημεία απαιτείται $\Theta(k)$. Συνολικά, λοιπόν, για την προεπεξεργασία (κατασκευή κυρτού περιβλήματος) και για την ερώτηση (καθορισμός ευθειών που ελαχιστοποιεί καθένα από τα x_i, $i = 1, \dots, k$) απαιτείται χρόνος $\Theta(n + k)$.

Στη συνέχεια, θα μετασχηματίσουμε την αναδρομική σχέση του ερωτήματος (α), ώστε να μπορούμε να χρησιμοποιήσουμε τον αλγόριθμο του ερωτήματος (β). Η αναδρομική σχέση που δίνει το $C(i)$ γράφεται:

$$C(i) = \begin{cases} \min_{0 \le j \le i}\{C(j-1) + x_i^2 + x_j^2 - 2x_i x_j + C\}, & i \ge 0 \\ 0, & i = -1 \end{cases} \quad \Rightarrow$$

$$C(i) = \begin{cases} x_i^2 + C + \min\limits_{0 \le j \le i}\{C(j-1) + x_j^2 - 2x_i x_j\}, & i \ge 0 \\ 0, & i = -1 \end{cases}$$

Στο τελευταίο βήμα εκμεταλλευτήκαμε το γεγονός ότι τα i, C είναι ανεξάρτητα του j, οπότε βγαίνουν έξω από το $\min\{\}$. Μετονομάζουμε $a[j] = -2x_j$ και $b[j] = C(j-1) + x_j^2$. Τότε, το $C(i)$ γράφεται:

$$C(i) = \begin{cases} x_i^2 + C + \min\limits_{0 \le j \le i} \{a[j]x_i + b[j]\}, & i \ge 0 \\ 0, & i = -1 \end{cases}$$

Είναι πλέον φανερό ότι το αρχικό πρόβλημα ανάγεται στο πρόβλημα του ερωτήματος (β), όπου πληρούνται όλες οι κατάλληλες προϋποθέσεις, εφόσον $0 \ge a[j] \ge a[j+1]$, για κάθε $j = 0, \dots, n-1$ και επίσης ισχύει $x_1 \le x_2 \le \cdots \le x_n$ (εδώ έχουμε $n = k$). Έτσι, μπορούμε να λύσουμε το αρχικό πρόβλημα σε γραμμικό χρόνο $\Theta(n+n) = \Theta(n)$ έναντι του $\Theta(n^2)$ που πετυχαίναμε με δυναμικό προγραμματισμό.

Σχόλιο Ο τρόπος με τον οποίο αποτιμούμε τη χρονική πολυπλοκότητα κατασκευής του κυρτού περιβλήματος είναι εντελώς ανάλογος του τρόπου με τον οποίο αποτιμούμε την πολυπλοκότητα της κατασκευής του καρτεσιανού δέντρου.

Άσκηση 4

($\boldsymbol{\alpha}$)

Θεωρούμε πίνακα $C(i,l)$ ο οποίος εκφράζει τον ελάχιστο συνολικό δείκτη ευαισθησίας για i φοιτητές οι οποίοι πρέπει να τοποθετηθούν σε l λεωφορεία. Τότε, η αναδρομική σχέση που περιγράφει τη βέλτιστη λύση είναι η ακόλουθη:

$$C(i,l) = \min_{j<i}\{C(j,l-1) + cost(j+1,i)\},$$

όπου $cost(j+1,i) = \sum_{k=j+1}^{i}\sum_{m=j+1}^{i} A_{km}$, δηλαδή το $cost(j+1,i)$ δίνει το δείκτη ευαισθησίας που προκύπτει αν οι φοιτητές $j+1$ έως i πάρουν το ίδιο λεωφορείο. Πρακτικά, για κάθε $i = 1,\dots,n$ βρίσκουμε όλα τα δυνατά $j < i$, έτσι ώστε οι πρώτοι j φοιτητές να χρησιμοποιούν τα $l-1$ λεωφορεία και οι εναπομείναντες να μπαίνουν στο ίδιο λεωφορείο. Από όλα αυτούς τους δυνατούς συνδυασμούς επιλέγουμε εκείνον που δίνει τον ελάχιστο δείκτη ευαισθησίας. Πρακτικά, η λύση του προβλήματος δίνεται από την τιμή $C(n,k)$, οπότε πρέπει αρχικά να συμπληρώσουμε έναν πίνακα $n \cdot k$ θέσεων. Ο καθορισμός της τιμής του κάθε στοιχείου γίνεται σε $O(n)$, εφόσον η μέγιστη τιμή που μπορεί να λάβει το i είναι n και σε αυτή την περίπτωση θα γίνουν $n-2$ συγκρίσεις, λόγω της αναδρομικής σχέσης. Έτσι, η συνολική πολυπλοκότητα του αλγορίθμου είναι $O(k \cdot n^2)$. Μπορούμε να παράξουμε τη ζητούμενη ανάθεση $(q_1, q_2, \dots, q_n)$ η οποία ελαχιστοποιεί το συνολικό δείκτη ευαισθησίας κάνοντας ένα backtracking στον πίνακα C κατά τον υπολογισμό της τιμής $C(n,k)$.

Άσκηση 5

$(\boldsymbol{a})$

Απόδειξη

Σημειώνουμε ότι επειδή $T_1 \neq T_2$, τότε $T_1 - T_2 \neq \emptyset$, οπότε μπορούμε να επιλέξουμε ακμή $e \in T_1 - T_2$. Όμως, επειδή το T_2 αποτελεί συνδετικό δέντρο και $e \notin T_2$, ξέρουμε ότι το γράφημα $T_2 \cup \{e\}$ έχει κύκλο. Έστω C ο κύκλος αυτός, τότε $C = \{w_1, \dots, w_k, e\}$ (δηλαδή οι ακμές $w_1, \dots, w_k, e$ κλείνουν τον κύκλο C), όπου $\{w_1, \dots, w_k\} \subseteq T_2$ (δηλαδή $w_i \in T_2$, για κάθε $i = 1, \dots, k$). Έτσι, με δεδομένο ότι $e \in T_1$, θα υπάρχει οπωσδήποτε $w_i \notin T_1$, για κάποιο $i = 1, \dots, k$, ειδάλλως το T_1 θα περιείχε τον κύκλο C και κατ' επέκταση δε θα ήταν συνδετικό δέντρο. Ονομάζοντας $e' = w_i$ έχουμε ότι θα υπάρχει $e' \in T_2 - T_1$, έτσι ώστε το $(T_1 - \{e\}) \cup e'$ να παραμένει συνδετικό δέντρο (ακυκλικό γράφημα με $|V| - 1$ ακμές).

Αλγόριθμος

Έστω ότι η δοσμένη ακμή e ενώνει τις κορυφές u, v. Αποθηκεύουμε το δέντρο T_1 σε μορφή πίνακα (κάθε κόμβος δείχνει στον πατέρα του) με μια DFS στο εν λόγω δέντρο. Αυτό μας επιτρέπει αργότερα να ελέγχουμε σε χρόνο $O(1)$ αν μια ακμή εμφανίζεται στο T_1. Η αποθήκευση του T_1 απαιτεί χρόνο $O(|V| + |V| - 1) = O(|V|)$. Εφαρμόζουμε μια DFS στο δέντρο T_2 και προσθέτουμε σε μια λίστα το μονοπάτι P (ακολουθία από ακμές) που συνδέει τις κορυφές u, v. Η διαδικασία αυτή γίνεται σε $O(|V| + |V| - 1) = O(|V|)$. Στη συνέχεια, για κάθε $p \in P$ μπορούμε να ελέγξουμε σε σταθερό χρόνο αν αυτό ανήκει στο T_1, λόγω της διαδικασίας που περιγράφηκε παραπάνω. Αν $p \notin T_1$, τότε επιστρέφουμε $e' = p$, διαφορετικά προχωράμε παρακάτω. Το κόστος για τους ελέγχους είναι $O(|V|)$, διότι $|P| \leq |V| - 1$. Άρα, η συνολική χρονική πολυπλοκότητα του αλγορίθμου είναι $O(|V| + |V| + |V|) = O(|V|)$.

($\boldsymbol{\beta}$)

Σχόλιο Το ερώτημα (α) μας διδάσκει ότι αν έχουμε δύο (διαφορετικά) συνδετικά δέντρα T_1 και T_2 που διαφωνούν τουλάχιστον στις ακμές e και e', τότε ανταλλάσσοντας στο T_1 το e με το e' και στο T_2 το e' με το e τα δύο δέντρα παραμένουν συνεκτικά και διαφέρουν σε ακριβώς μια ακμή με τους προκατόχους του (δηλαδή το T_1 με το $(T_1 - \{e\}) \cup \{e'\}$ και το T_2 με το $(T_2 - \{e'\}) \cup \{e\}$.

Για να έχει ενδιαφέρον το πρόβλημα, υποθέτουμε ότι το γράφημα G έχει τουλάχιστον δύο (διαφορετικά) συνδετικά δέντρα ή ισοδύναμα ότι το γράφημα H έχει τουλάχιστον δύο κορυφές. Για να δείξουμε ότι το H είναι συνεκτικό, αρκεί να δείξουμε ότι για τα τυχαία συνδετικά δέντρα T_i και T_j υπάρχει μονοπάτι στο H που τα ενώνει. Ξεκινάμε από το T_i και βρίσκουμε σε ποιες ακμές διαφωνεί με το T_j. Έστω ότι κάποιες από αυτές τις ακμές είναι οι $e_i \in T_i - T_j$ και $e_j \in T_j - T_i$. Με βάση το παραπάνω σχόλιο, μπορούμε στο δέντρο T_i να ανταλλάξουμε το e_i με το e_j γνωρίζοντας ότι το δέντρο $T_{i+1} = (T_i - \{e_i\}) \cup \{e_j\}$ είναι συνεκτικό και επίσης ισχύει $|T_i - T_{i+1}| = |T_{i+1} - T_i| = 1$, οπότε το T_{i+1} προστίθεται ως κορυφή στο γράφημα H. Με αυτό τον τρόπο, βλέπουμε ότι σε κάθε βήμα οι διάδοχοι του T_i «μοιάζουν» όλο και παραπάνω στο T_j. Το κριτήριο ομοιότητας είναι το μέγεθος $|T_{i+k} - T_j|$, $k = 0, 1, \ldots$ το οποίο θέλουμε σε κάθε βήμα να ελαττώνεται, γεγονός που σημαίνει ότι οδηγούμαστε στο T_j. Πράγματι, σύμφωνα με τα παραπάνω είναι εύκολο να δούμε ότι $|T_{i+1} - T_j| = |T_i - T_j| - 1 < |T_i - T_j|$, δηλαδή σε κάθε βήμα το μέγεθος $|T_{i+k} - T_j|$ ελαττώνεται ακριβώς κατά 1, οπότε προκύπτει ότι μετά από $|T_i - T_j|$ βήματα θα οδηγηθούμε στο «επιθυμητό» δέντρο T_j. Σε όρους γραφημάτων αυτό σημαίνει ότι για τις τυχαίες κορυφές T_i και T_j υπάρχει μονοπάτι μήκους $|T_i - T_j| = k$ στο H που τις ενώνει. Συνεπώς, το γράφημα H είναι συνεκτικό.

Στη συνέχεια, θα δείξουμε ότι αυτό το μονοπάτι είναι και το συντομότερο, δηλαδή αν $d_H(T_1, T_2)$ το συντομότερο μονοπάτι που ενώνει τα δέντρα T_1 και T_2, τότε $|T_1 - T_2| = k \Leftrightarrow d_H(T_1, T_2) = k$

Απόδειξη (με επαγωγή στο μήκος k του συντομότερου μονοπατιού)

• Επαγωγική βάση: Για $k = 1$ τα T_1 και T_2 διαφέρουν ακριβώς σε μια ακμή (έστω $e_1 \in T_1 - T_2$ και $e_2 \in T_2 - T_1$), οπότε σύμφωνα με τον αλγόριθμο του ερωτήματος (α), στο δέντρο T_1 ανταλλάσσουμε την e_1 με την e_2 και έχουμε $T_2 = (T_1 - \{e_1\}) \cup \{e_2\}$, οπότε με ένα βήμα, δηλαδή με μονοπάτι μήκους 1 οδηγούμαστε στο δέντρο T_2 (προφανώς, δεν μπορούμε να πετύχουμε μονοπάτι μικρότερου μήκους για δύο διαφορετικά δέντρα). Πιο απλά, $|T_1 - T_2| = 1 \Leftrightarrow d_H(T_1, T_2) = 1$ εξ' ορισμού του H.

• Επαγωγικό βήμα: Έστω $|T_1^* - T_2^*| = k \Leftrightarrow d_H(T_1^*, T_2^*) = k$, τότε θα δείξουμε ότι $|T_1 - T_2| = k + 1 \Leftrightarrow d_H(T_1, T_2) = k + 1$.

Αποδεικνύουμε πρώτα ότι $|T_1 - T_2| = k + 1 \Rightarrow d_H(T_1, T_2) = k + 1$

Αφού $|T_1 - T_2| = k + 1 \neq 0$ θα υπάρχει $e_1 \in T_1 - T_2$ και $e_2 \in T_2 - T_1$, οπότε σύμφωνα με τον αλγόριθμο του ερωτήματος (α), στο δέντρο T_1 ανταλλάσσουμε την e_1 με την e_2 και προκύπτει το δέντρο $T_1' = (T_1 - \{e_1\}) \cup \{e_2\} \in H$. Ισχύει ότι $|T_1' - T_2| = k$, άρα από επαγωγική υπόθεση $d_H(T_1', T_2) = k$. Έτσι, ισχύει $d_H(T_1, T_2) \leq k + 1$ (1) (η ισότητα ισχύει αν το T_1' ανήκει στο βέλτιστο μονοπάτι, το οποίο θα αποδείξουμε έμμεσα στη συνέχεια). Όμως, θυμίζουμε ότι $|T_1 - T_2| = k + 1$ και με δεδομένο ότι κάθε μονοπάτι στο H προκύπτει με ακριβώς μια ανταλλαγή ακμών (ώστε ο «πατέρας» και ο «γιος» να διαφέρουν σε μια ακμή), έχουμε ότι $d_H(T_1, T_2) \geq k + 1$ (2), δηλαδή για να οδηγηθούμε από το T_1 στο T_2 θα συμβούν τουλάχιστον $|T_1 - T_2|$ ανταλλαγές. Συνδυάζοντας τις σχέσεις (1) και (2) τελικά προκύπτει ότι $d_H(T_1, T_2) = k + 1$.

Στη συνέχεια αποδεικνύουμε $d_H(T_1, T_2) = k + 1 \Rightarrow |T_1 - T_2| = k + 1$

Αν $d_H(T_1, T_2) = k + 1$, τότε υπάρχει δέντρο $T_1' \in H$ έτσι ώστε $d_H(T_1, T_1') = 1$ και $d_H(T_1', T_2) = k$. Το δέντρο T_1' έχει προκύψει ανταλλάσσοντας στο δέντρο T_1 την ακμή e_1 με την e_2, δηλαδή $T_1' = (T_1 - \{e_1\}) \cup \{e_2\}$, όπου $e_1 \in T_1$ και $e_2 \notin T_1$. Αν η ακμή e_2 ανήκει στο δέντρο T_2 και επιπλέον $e_1 \notin T_2$, τότε $|T_1 - T_2| = k + 1$, με δεδομένο ότι από επαγωγική υπόθεση ισχύει $|T_1' - T_2| = k$. Όμως, αν $e_2 \in T \neq T_2$ και επιπλέον $e_1 \notin T$, τότε $|T_1 - T_2| = k - 1$. Προφανώς, δεν μπορεί να ισχύει $|T_1 - T_2| = k - 1$, διότι τότε από επαγωγική υπόθεση θα είχαμε

$d_H(T_1, T_2) = k - 1$, το οποίο είναι άτοπο, διότι υποθέσαμε ότι $d_H(T_1, T_2) = k + 1$. Άρα, $|T_1 - T_2| = k + 1$ και εδώ ολοκληρώνεται η επαγωγική απόδειξη.

Αλγόριθμος

Για την εύρεση ενός συντομότερου μονοπατιού στο γράφημα H ανάμεσα στα δέντρα T_1 και T_2, αρκεί να βρούμε όλα τα μονοπάτια $e' \in T_2 - T_1$ και στη συνέχεια να ανταλλάξουμε καθένα από αυτά στο δέντρο T_1 με κατάλληλη ακμή $e \in T_1 - T_2$. Συνεπώς, χρειαζόμαστε έναν τρόπο να ελέγχουμε σε σταθερό χρόνο αν μια ακμή εμφανίζεται σε κάποιο δέντρο. Αποθηκεύουμε το δέντρο T_1 σε μορφή πίνακα (κάθε κόμβος δείχνει στον πατέρα του) με μια DFS στο εν λόγω δέντρο. Αυτό μας επιτρέπει αργότερα να ελέγχουμε σε χρόνο $O(1)$ αν μια ακμή εμφανίζεται στο T_1. Ομοίως για το δέντρο T_2. Μέχρι στιγμής η DFS και η αποθήκευση σε μορφή πίνακα απαιτούν χρόνο $O(|V|)$. Στη συνέχεια, μπορούμε με δύο διασχίσεις των πινάκων σε χρόνο $O(|V|)$ να φτιάξουμε τα σύνολα $T_1' = \{e_1 : e_1 \in T_1 - T_2\}$ και $T_2' = \{e_2 : e_2 \in T_2 - T_1\}$, όπου $|T_1'| = |T_2'| = k$. Έτσι, έχουμε άμεση πρόσβαση στις ακμές στις οποίες διαφωνούν τα δύο δέντρα. Εφαρμόζοντας διαδοχικές ανταλλαγές ακμών στο δέντρο T_1 ανάμεσα σε στοιχεία του T_1' και του T_2' με κάθε ανταλλαγή να κοστίζει $O(|V|)$, σύμφωνα με τον αλγόριθμο του ερωτήματος (α), μετά από k βήματα έχουμε φτάσει στο δέντρο T_2. Η συνολική πολυπλοκότητα του αλγορίθμου είναι $O(k \cdot |V|)$. Η ορθότητα του αλγορίθμου επαληθεύεται από το γεγονός ότι σε κάθε βήμα ο διάδοχος του δέντρου T_1 ανανεώνεται ώστε να διαφέρει κατά μια λιγότερη ακμή με το T_2 ως προς τον προκάτοχό του. Έτσι, μετά από $|T_1 - T_2| = k$ βήματα έχουμε οδηγηθεί στο T_2, το οποίο σημαίνει ότι στο γράφημα H υπάρχει μονοπάτι μήκους k που συνδέει τα T_1 και T_2, το οποίο είναι και βέλτιστο λόγω της παραπάνω απόδειξης.

$(\boldsymbol{\gamma})$

Κατασκευάζουμε ένα MST T του γραφήματος G σε χρόνο $O(m \log m)$ χρησιμοποιώντας π.χ. τον αλγόριθμο του $Kruskal$. Τότε, είναι προφανές ότι για καθεμιά από τις ακμές που ανήκουν στο T επιστρέφουμε το $w(T)$ ως το ελάχιστο βάρος συνδετικού δέντρου που τις περιλαμβάνει (προφανώς, δεν μπορεί να κατασκευαστεί «πιο ελάχιστο» συνδετικό δέντρο από το ήδη ελάχιστο). Το ζήτημα τίθεται για τις ακμές $e \in E - T$. Δεν υπάρχει λόγος να κατασκευάσουμε από την αρχή καινούρια MST τα οποία περιλαμβάνουν τις απούσες ακμές και από αυτά να επιλέξουμε το ελαφρύτερο. Το T είναι αρκετό και για τη συνέχεια. Συγκεκριμένα, παρατηρούμε ότι αν εισάγουμε στο T κάποια ακμή $e \in E - T$ η οποία π.χ. συνδέει τις κορυφές u και v, τότε το $T \cup \{e\}$ δημιουργεί κύκλο. Για να διατηρηθεί, λοιπόν, η συνεκτικότητα πρέπει να αφαιρέσουμε από το T μια ακμή e' η οποία ανήκει στον κύκλο που κλείνει η e. Έτσι, λαμβάνουμε ένα νέο συνεκτικό δέντρο T', το οποίο βέβαια παύει να είναι ελάχιστο, με $T' = (T - \{e\}) \cup \{e'\}$. Το συνολικό βάρος αυτού του δέντρου είναι $w(T') = w(T) - w(e) + w(e')$. Παρατηρούμε ότι το T' αποκτά το μικρότερο δυνατό βάρος, όταν μεγιστοποιείται το $w(e)$. Συνεπώς, το πρόβλημα ανάγεται στη διατύπωση αποδοτικού αλγορίθμου ο οποίος εντοπίζει τη βαρύτερη ακμή στο μονοπάτι που συνδέει δύο κορυφές στο δέντρο T. Μία λύση σε αυτό το πρόβλημα είναι να βρούμε τη βαρύτερη ακμή στο μονοπάτι που συνδέει την ακμή $u \in V$ με την ακμή v, για κάθε $v \in V$ εφαρμόζοντας μια DFS στο δέντρο που έχει ως ρίζα το u. Αυτό γίνεται εύκολα σε χρόνο $O(n + n - 1) = O(n)$. Η παραπάνω διαδικασία επαναλαμβάνεται συνολικά n φορές (μία για κάθε κορυφή του γραφήματος), άρα απαιτεί μέσο χρόνο $O(n^2)$. Η συμμετρία ανάμεσα σε (u, v) και (v, u) λόγω του μη κατευθυνόμενου γραφήματος ελαττώνει σημαντικά το πλήθος των υπολογισμών. Έχοντας κάνει αυτή την προεπεξεργασία, μπορούμε σε σταθερό χρόνο για κάθε ακμή $e = (u, w) \in E - T$ να εντοπίσουμε τη βαρύτερη ακμή στο μονοπάτι που συνδέει τις κορυφές u, v στο T. Υπάρχουν $m - (n - 1)$ ακμές που δεν ανήκουν στο T, οπότε το $query\ time$ είναι $O(m)$. Συνεπώς, η συνολική χρονική πολυπλοκότητα του αλγορίθμου είναι $O(m \log m + n^2 + m) = O(m \log m + n^2)$