Αλγόριθμοι & Πολυπλοκότητα

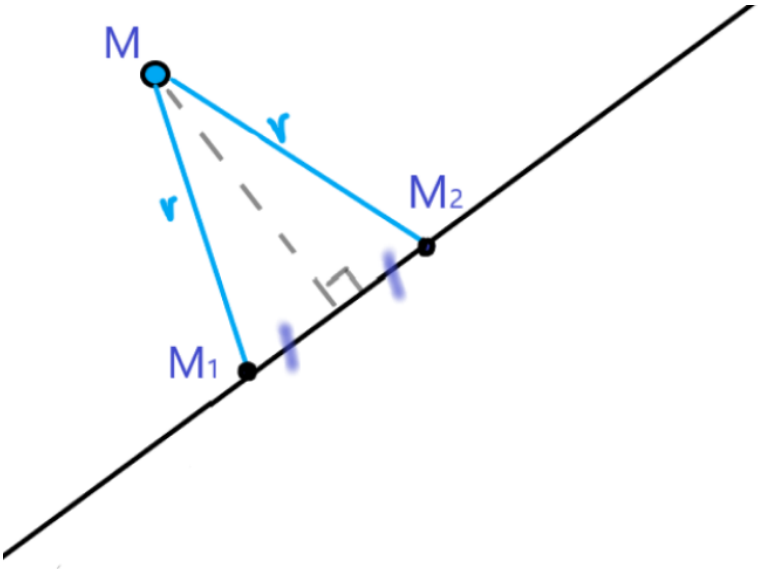
2η σειρά γραπτών ασκήσεων

---

hungrydino ΣΗΜΜΥ 7ο εξάμηνο

Άσκηση 1: Δίσκοι και Σημεία

Για κάθε σημείο $M(x, y)$ που απέχει απόσταση $d(M, l) < r$ θα υπάρχουν δύο σημεία $M_1(x_1, y_1)$ και $M_2(x_2, y_2)$ που ανήκουν στην ευθεία και απέχουν απόσταση r από το M. Αν η απόσταση $d(M, l)$ είναι ίση με r, τότε δεν έχουμε δύο σημεία άλλα ένα (το σημείο τομής της καθέτου από το M στην ευθεία). Έστω το σημείο $M(x_m, y_m)$, η ευθεία $ax + by + c = 0$ και $d(M, l) < r$. Τότε φέρνοντας την κάθετη από το σημείο M στην ευθεία και τα ευθύγραμμα τμήματα MM_1, MM_2 όπου M_1, M_2 τα αντίστοιχα σημεία με παραπάνω έχουμε το παρακάτω σχήμα:



Η γωνία που σχηματίζει η ευθεία $ax + by + c = 0$ με τον άξονα $x'x$ είναι: $\theta = arctan\left(-\frac{b}{a}\right)$. Γνωρίζουμε ότι αν θέλουμε να στρέψουμε το σύστημα συντεταγμένων αριστερόστροφα κατά μια γωνία ϕ οι νέες συντεταγμένες ενός σημείου (x, y) είναι (από εδώ και πέρα οι τονισμένες συντεταγμένες αναφέρονται στο περιστραμμένο σύστημα):

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} cos\phi & -sin\phi \\ sin\phi & cos\phi \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

Για ευκολία μπορούμε να στρέψουμε το αρχικό πρόβλημα κατά γωνία $\phi = -\theta$, ώστε η ευθεία να είναι παράλληλη στον άξονα $x'x$. Δηλαδή ο πίνακας στρέψης μας είναι:

$$\mathrm{T} = \begin{pmatrix} cos(-\theta) & -sin(-\theta) \\ sin(-\theta) & cos(-\theta) \end{pmatrix} = \begin{pmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{pmatrix}$$

Τότε το σημείο M μετά τη στροφή έχει συντεταγμένες:

$$\begin{pmatrix} x'_m \\ y'_m \end{pmatrix} = \begin{pmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{pmatrix} \begin{pmatrix} x_m \\ y_m \end{pmatrix}$$

Το μέσον M_o του ευθύγραμμου τμήματος M_1M_2 θα έχει συντεταγμένες: $(x'_o, y'_o) = (x'_m, y'_m - d)$. Το τρίγωνο $\triangle M_1MM_o$ είναι ορθογώνιο και είναι: $M_1M_o = \sqrt{r^2 - d^2}$. Άρα, τα σημεία M_1, M_2 έχουν συντεταγμένες:

$$(x'_1, y'_1) = (x'_m - \sqrt{r^2 - d^2}, y'_m - d)$$

$$(x'_2, y'_2) = (x'_m + \sqrt{r^2 - d^2}, y'_m - d)$$

Η απόσταση $d(M, l)$ υπολογίζεται εύκολα στο νέο σύστημα και είναι: $d = |y'_m - c|$. Άρα τα προηγούμενα δύο σημεία μπορούν να υπολογιστούν σε σταθερό χρόνο (αν μας δίνεται η ευθεία l και το σημείο M).

Το αρχικό μας πρόβλημα τότε μετατρέπεται στο επόμενο: δοθέντων $n - k$ ζευγαριών σημείων (ισοδύναμα: διαστημάτων) και k σημείων, ποιο είναι το ελάχιστο πλήθος σημείων που απαιτούνται ώστε κάθε διάστημα να περιέχει τουλάχιστον ένα από αυτά ή τα k δοθέντα. Τα k σημειά αντιστοιχούν στα κέντρα των κύκλων που πρέπει υποχρεωτικά να κατασκευάσουμε για τα σημεία που απέχουν απόσταση r από την ευθεία.

Ο αλγόριθμος μας είναι ο επόμενος:

↪ Κατασκευή των αντίστοιχων σημείων M_1, M_2 για όλα τα σημεία (θεωρούμε ότι για τα σημεία που απέχουν r από την ευθεία ταυτίζονται).
↪ Ταξινομούμε τα διαστήματα σε αύξουσα σειρά με βάση το δεξί άκρο του διαστήματος τους.
↪ Όσο δεν έχουμε δει όλα τα διαστήματα:

- Θεωρούμε το πρώτο στη σειρά διάστημα και το δεξί του άκρο, έστω R. Εκεί τοποθετούμε το πρώτο κέντρο.
- Ακολουθώντας την ταξινόμηση, εξετάζουμε αν το άριστερό άκρο του επόμενου διαστήματος είναι μικρότερο ή ίσο του R. Αν αυτό ισχύει τότε ο κύκλος με κέντρο το R καλύπτει αυτό το διάστημα. Αν αυτό δεν ισχύει τότε αυξάνουμε τον μετρητή των απαιτούμενων κέντρων/κύκλων κατά 1 και συνεχίζουμε με επόμενο κέντρο το δεξί άκρο αυτού του διαστήματος.

Εύκολα βλέπουμε ότι η πολυπλοκότητα του αλγορίθμου είναι $\mathrm{O}(n)$, όπου n ο αριθμός των διαστημάτων (ή των αρχικών σημείων).

Έστω ότι ο αλγόριθμος μας παράγει τα κέντρα $G_1 < G_2 < ... < G_d$ και μια βέλτιστη λύση τα $O_1 < O_2 < ... < O_k$. Αρχικά θα αποδείξουμε ότι $G_i \geq O_i$ για κάθε $1 \leq i \leq min\{k, d\}$.

- Για $i = 1$ ισχύει, αφού ο κύκλος με κέντρο O_1 πρέπει να καλύπτει το πρώτο σημείο, άρα θα πρέπει να ανήκει στο αντίστοιχο διάστημα. Το G_1 είναι το δεξί άκρο του διαστήματος, επομένως $G_1 \geq O_1$.
- Έστω ότι ισχύει για κάθε $i \leq m$.
- Για $m + 1 \leq d$ ο αλγόριθμος μας παρήγαγε το κέντρο G_{m+1} καθώς βρήκε διάστημα για το οποίο το αριστερό του άκρο ήταν δεξιότερα του G_m και έτσι το G_{m+1} θα ταυτίζεται με το δεξί άκρο αυτού του διαστήματος. Λόγω της επαγωγικής υπόθεσης ($G_m \geq O_m$) το αριστερό άκρο αυτό θα είναι δεξιότερα και του O_m. Επομένως θα πρέπει το O_{m+1} να βρίσκεται ανάμεσα σε αυτό το διάστημα και άρα θα πρέπεοι $O_{m+1} \leq G_{m+1}$.

Άρα $G_i \geq O_i$ για κάθε $1 \leq i \leq min\{k, d\}$. Με βάση αυτό τώρα θα αποδείξουμε ότι πρέπει αναγκαστικά $d = k$.

Έστω ότι $d > k$. Τότε ο αλγόριθμος μας βρήκε διάστημα για το οποίο το αριστερό άκρο του είναι δεξιότερα από το G_k. Αυτό σημαίνει όμως ότι είναι δεξιότερα και από το O_k (αφού $O_k \leq G_k$), δηλαδή η βέλτιστη λύση δεν καλύπτει όλα τα διαστήματα και άρα δεν είναι ορθή! Αυτό είναι άτοπο, άρα πρέπει $d \leq k$ και επειδή η βέλτιστη λύση δίνει k κέντρα $d = k$. Ο αλγόριθμος μας είναι ορθός.

Άσκηση 2: Παραλαβή πακέτων

α) Ο αλγόριθμος είναι πολύ απλός. Η εξυπηρέτηση των πελατών θα γίνει σύμφωνα με την φθίνουσα ακολουθία βαρών ανά μονάδα χρόνου εξυπηρέτησης, δηλαδή w_i/p_i. Αυτό μπορεί να γίνει σε χρόνο $O(n)$ με την $radixsort$.

Η πολυπλοκότητα του αλγορίθμου είναι $O(n)$ και η ορθότητα του αποδεικνύεται με το παρακάτω επιχείρημα ανταλλαγής. Αν ο βέλτιστος αλγόριθμος δεν ακολουθεί την παραπάνω τακτική/ταξινόμηση, θα υπάρχει ένα ζευγάρι πελατών με δείκτες i, j έτσι ώστε ο πελάτης i να παραλαμβάνει ακριβώς πριν από τον j, αλλά παρ'όλα αυτά να ισχύει $w_i/p_i < w_j/p_j$.
Έστω τώρα ότι ανταλλάσσουμε τις θέσεις τους, δηλαδή ο j παραλαμβάνει πριν από τον i. Τότε, οι χρόνοι εξυπηρέτησης (όπως και το κόστος) των υπολοίπων πελατών δεν αλλάζουν, μόνο του ζευγαριού αυτού. Συμβολίζουμε τον χρόνο εξυπηρέτησης του i στην αρχική κατάσταση με: $t_{oi} = t + p_i$, τότε ο χρόνος εξυπηρέτησης του j θα είναι: $t_{oj} = t + p_i + p_j$. Έτσι το συνολικό κόστος θα είναι:

$$C_o = C + w_i(t + p_i) + w_j(t + p_i + p_j)$$

Στην δεύτερη κατάσταση ο χρόνος εξυπηρέτησης του j ισούται με $t_{sj} = t + p_j$ ενώ του i με $t_i = t + p_j + p_i$. Το συνολικό κόστος στην δεύτερη κατάσταση είναι:

$$C_s = C + w_j p_j + w_i(t + p_j + p_i)$$

Τότε έχουμε:

$$C_o - C_s = w_j p_i - w_i p_j = p_i p_j \left(\frac{w_j}{p_j} - \frac{w_i}{p_i} \right) > 0$$
$$\Longrightarrow C_s < C_o$$

Πράγμα άτοπο, αφού η C_o είναι η βέλτιστη λύση, άρα αυτή πρέπει να ακολουθεί την ταξινόμηση που εξηγήσαμε.

β) Αρχικά θεωρούμε ότι ζητείται να ελαχιστοποιηθεί το άθροισμα του βεβαρυμένου κόστους που προκύπτει από τους δύο υπαλλήλους (όχι το μέγιστο από τα δύο). Αναγκαστικά θα πρέπει να χρησιμοποιήσουμε δυναμικό προγραμματισμό. Θεωρούμε ότι οι πελάτες είναι ταξινομημένοι με το κριτήριο του προηγούμενου σκέλους, αν όχι, τότε τους ταξινομούμε.

Συμβολίζουμε με $C(i,t)$ το ελάχιστο συνολικό βεβαρυμένο κόστος για τους πρώτους i πελάτες με βάση την ταξινόμηση και θεωρώντας ότι ο συνολικός χρόνος εξυπηρέτησης στην πρώτη ουρά (υπάλληλο) είναι t. Τότε αν $\mathrm{T}(i) = p_1 + p_2 + ... + p_i$ ισχύει:

$$C(i,t) = min \begin{cases} C(i-1, t-p_i) + w_i t \\ C(i-1,t) + w_i \cdot [T(i) - t] \end{cases}$$

Ο πίνακας αυτός μπορεί να γεμίσει $bottom - up$ σε χρόνο $O(n \sum_{i=1}^{n} p_i)$. Τότε η λύση θα είναι:

$$MinCost = \min_{\substack{1 \le i \le n-1 \\ t \le T(n-1)}} \{C(n,t)\}$$

Η τελική πολυπλοκότητα είναι $O(n \sum_{i=1}^{n} p_i)$.Η ορθότητα του αλγορίθμου προκύπτει από το γεγονός ότι για οποιοδήποτε σύνολο πελατών σε κάποια ουρά, το ελάχιστο κόστος σε αυτή προκύπτει ταξινομώντας τους με βάση το κριτήριο που επιλέξαμε. Ο αλγόριθμος είναι εξαντλητικός.

Αν είχαμε παραπάνω υπαλλήλους, έστω k, ο πίνακας μας θα είχε $1 + k - 1 = k$ παραμέτρους, όπου το $k-1$ προκύπτει από τους χρόνους εξυπηρέτησης των ουρών (εκτός από την τελευταία που προκύπτει από τις άλλες). Η πολυπλοκότητα σε αυτή την περίπτωση θα ήταν $O(n \left(\sum_{i=1}^{n} p_i\right)^k)$.

Άσκηση 3: Τοποθέτηση Στεγάστρων (και Κυρτό Κάλυμμα)

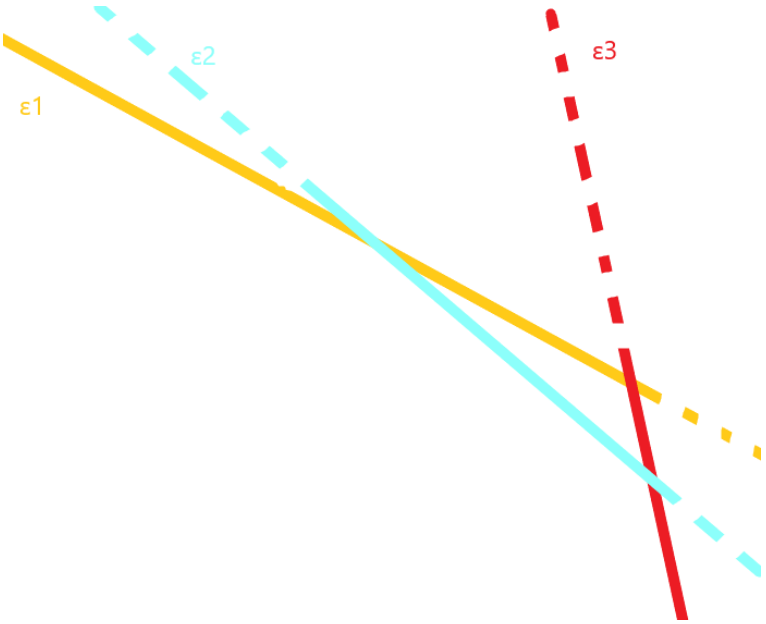
α) Η βέλτιστη λύση μας $cost(n)$ προκύπτει από την παρακάτω αναδρομική σχέση:

$$cost(k) = \begin{cases} 0 & k = 0 \\ \min\limits_{1 \le i \le k} \{cost(i-1) + (x_i - x_k)^2 + C\} & 1 \le k \le n \end{cases}$$

Δηλαδή κοιτάμε όλα τα πιθανά σημεία x_i από τα οποία μπορεί να αρχίζει στέγαστρο που να καλύπτει το τωρινό, προσθέτοντας το κόστος στο βέλτιστο κάλυμμα που έχει βρεθεί μέχρι το προηγούμενο σημείο x_{i-1}. Έτσι η ορθότητα του αλγορίθμου είναι προφανής.

Θα χρειαστούμε ένα πίνακα μεγέθους n για τά βέλτιστα κόστη μέχρι κάθε σημείο, δηλαδή έχουμε χωρική πολυπλοκότητα $\Theta(n)$. Για τον υπολογισμό του στοιχείου i του πίνακα εκτελούμε i πράξεις. Άρα έχουμε χρονική πολυπλοκότητα $\Theta(\sum_{i=1}^{n} i) = \Theta(\frac{n(n+1)}{2}) = \Theta(n^2)$.

β) Για να λύσουμε το πρόβλημα με τις ευθείες και τα σημεία θα πρέπει να αποτυπώσουμε το πρόβλημα γεωμετρικά. Ας θεωρήσουμε ότι έχουμε 3 ευθείες $\varepsilon_1, \varepsilon_2, \varepsilon_3$ με φθίνουσες κλίσεις $a_1 \ge a_2 \ge a_3$, για παράδειγμα τις παρακάτω:

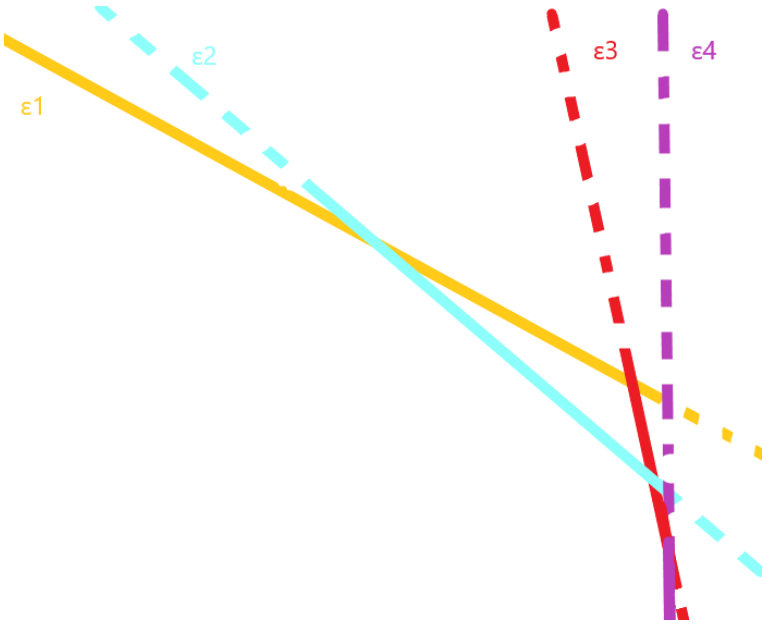


Σχήμα 1: Κυρτό Κάλυμμα με 3 ευθείες

Παρατηρούμε ότι για κάθε σημείο που βρίσκεται αριστερότερα απο το σημείο τομής M_{12} των $\varepsilon_1, \varepsilon_2$ η ε_1 ελαχιστοποιεί την τιμή του. Αντίστοιχα για κάθε σημείο δεξιότερα του M_{12}, αλλά αριστερότερα του σημείου τομής M_{23} της ε_2 με την ε_3 η ευθεία που το ελαχιστοποιεί είναι η ε_2. Τέλος, για αυτά δεξιότερα του M_{23} η αντίστοιχη ευθεία είναι η ε_3.

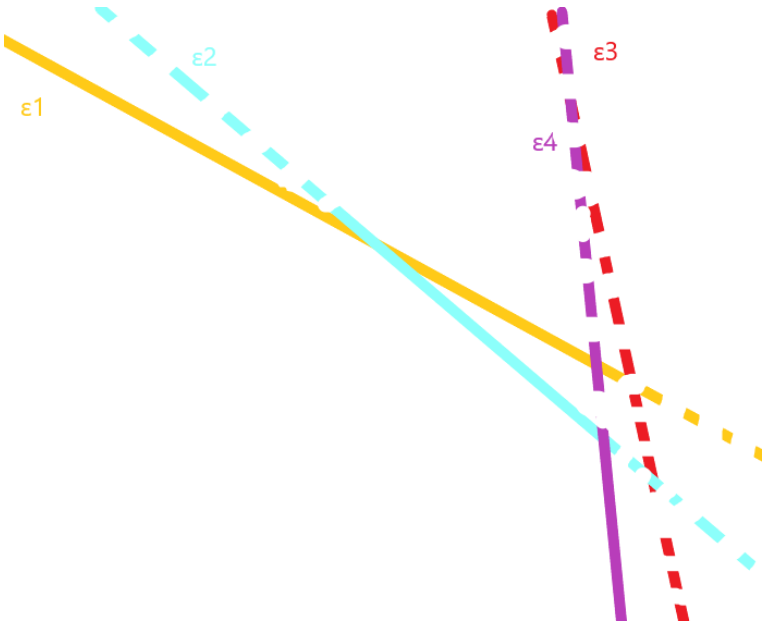
Έστω ότι στο σύνολο των ευθειών μας προστίθεται άλλη μια ε_4, με κλίση $a_4 \le a_3 \le a_2 \le a_1$.

Ένα παράδειγμα είναι το παρακάτω, η καινούρια ευθεία τέμνει το κάλυμμα στην πλευρά που ανήκει στην ε_3. Ουσιαστικά τώρα προστίθεται ένα νέο διάστημα (πλευρά στο κυρτό κάλυμμα).



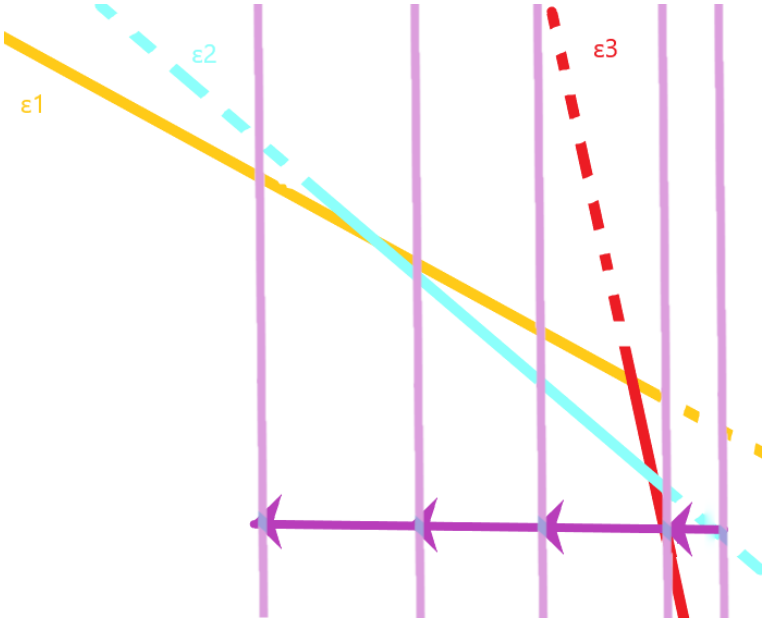
Σχήμα 2: Προστίθεται η ε_4

Σε μια άλλη περίπτωση η καινούρια ευθεία τέμνει το κάλυμμα στην πλευρά που ανήκει στην ε_2, τώρα η ε_3 δεν ανήκει στο κυρτό κάλυμμα.



Σχήμα 3: Η ε_4 έχει μικρότερο σταθερό όρο

Θα μπορούσε επίσης να εκτοπίσει και τις άλλες ευθείες με τον κατάλληλο σταθερό όρο. Αυτό που μπορούμε να σιγουρευτούμε είναι ότι θα προστεθεί στο κάλυμμα, γιατί θα υπάρχει κάποιο σημείο όπου πέρα από αυτό θα βρίσκεται "κάτω" από τις άλλες (θα δίνει την ελάχιστη τιμή). Θεωρούμε ότι έχουμε μια τέτοια ευθεία με δεδομένη κλίση, όπου το σημείο (έστω Μ) που τέμνει το ήδη υπάρχον κάλυμμα ανήκει στην τελευταία ευθεία του καλύμματος. Τότε αυτό το σημείο τομής με το κάλυμμα μπορεί να αλλάξει μόνο με έναν τρόπο: μετακινώντας την ευθεία παράλληλα με τον εαυτό της. Αν την μετακινούμε δεξιά, το σημείο Μ μεταφέρεται και αυτό δεξιά, αλλά πάντα παραμένει πάνω στην ε_3. Όσο μετακινούμε την ευθεία αριστερά όμως, θα μετακινείται το Μ αριστερά και αυτό, μέχρις η ευθεία να τέμνει πια μια άλλη πλευρά του καλύμματος, την ευθεία ε_2, τότε η ευθεία ε_3 εκτοπίζεται και παύει να είναι πλευρά του καλύμματος. Το ίδιο θα γίνει με την ε_2 αν μετακινηθούμε αρκετά αριστερά. Αυτά φαίνονται σχηματικά παρακάτω:



Σχήμα 4: Η ε_4 μετακινείται παράλληλα

Παρατηρούμε λοιπόν τα εξης:
- Για να αποθηκεύσουμε το κάλυμμα μπορούμε να κρατάμε τα διαστήματα "κυριαρχίας" κάθε ευθείας που ανήκει στο κάλυμμα, δηλαδή το διάστημα τεταγμένων που ελαχιστοποιεί, όπως και τις παραμέτρους της. Αυτό σε $\Theta(k)$ χώρο, (k οι πλευρές του καλύμματος).
- Αν έχουμε μια ακολουθία ευθειών ε_i με φθινουσες κλίσεις $a_i \leq 0$ και έχουμε κατασκευάσει το κάλυμμα για $n-1$ ευθείες, η n-στη ευθεία θα τέμνει σίγουρα το κάλυμμα και θα εκτοπίσει όλες τις πλευρές δεξιά του σημείου τομής της με αυτό (εκτός αν είναι παράλληλη με την τελευταία ευθεία όπου εξηγούμε παρακάτω).
- Αυτό το σημείο τομής μπορεί να βρεθεί ως εξης. Εξετάζουμε που τέμνονται η ευθεία που θέλουμε να προσθέσουμε με την "τελευταία" ευθεία του καλύμματος. Αν το σημείο τομής δεν ανήκει στο κάλυμμα (στην πλευρά της τελευταίας ευθείας), ελέγχουμε την προτελευταία ευθεία (και αφαιρούμε την τελευταία από το κάλυμμα). Συνεχίζουμε αναδρομικά μέχρι να βρούμε το σημείο.

- Μπορούμε να ελέγξουμε αν το σημείο τομής μιας τυχαίας ευθείας και μιας ευθείας/πλευράς του καλύμματος ανήκει στο κάλυμμα σε $O(1)$ χρόνο. Έστω $\varepsilon_n = a_n x + b_n$ η ευθεία του καλύμματος και $\varepsilon_{new} = a_{new} x + b_{new}$ η τυχαία ευθεία. Η τεταγμένη του σημείου τομής υπολογίζεται εύκολα από την σχέση $x_c = -\frac{b_{new} - b_n}{a_{new} - a_n}$. Αν αυτή η τεταγμένη ανήκει στο διάστημα που έχουμε αποθηκεύσει ότι "κυριαρχεί" η ευθεία x_n (τα σημεία του άξονα x που ελαχιστοποιεί), τότε αυτό το σημείο τομής ανήκει στο κυρτό κάλυμμα.
- Αν οι παραπάνω ευθείες είναι παράλληλες τότε αντικαθιστούμε την τελευταία ευθεία αν ο σταθερός όρος της νέας είναι μικρότερος (και επαναυπολογίζουμε το διάστημα "κυριαρχίας"). Αν ο σταθερός όρος είναι μεγαλύτερος ή ίσος την αγνοούμε.
- Παρατηρούμε ότι αυτή η διαδικασία μπορεί να πάρει $O(i)$ χρόνο για την ευθεία ε_i, αλλά κάθε φορά που κάποια ευθεία "εκτοπίζεται" δεν ανήκει πια στο κυρτό κάλυμμα, έτσι δεν χρειάζεται να την εξετάσουμε αργότερα.

■ Συμπεραίνουμε ότι μπορούμε να κατασκευάσουμε αυτό το κάλυμμα σε $O(n)$ χρόνο.

Μπορούμε τώρα να διατυπώσουμε τον παρακάτω αλγόριθμο.
• Είσοδος: n ευθείες ταξινομημένες σε φθίνουσα σειρά με βάση τις κλίσεις τους, k σημεία του άξονα $x'x$ σε αύξουσα σειρα.
• Έξοδος: k ζευγάρια σημείου-ευθείας που ελαχιστοποιεί το σημείο.

↪ Κατασκευάζουμε το κυρτό κάλυμμα των ευθειών σε $O(n)$ χρόνο. Πιο συγκεκριμένα κρατάμε τις πλευρές/ευθείες του καλύμματος και τα διαστήματα τεταγμένων που εδρεύουν. Για όλες τις τεταγμένες/σημεία ανάμεσα σε αυτά τα διαστήματα εκείνη είναι η ευθεία που ελαχιστοποιεί την τιμή τους.

↪ Αρχίζοντας από το πρώτο σημείο το x_1 βρίσκουμε σε $O(k)$ χρόνο το διάστημα από τα παραπάνω που ανήκει (ή την πλευρά που το καλύπτει), άρα και την ευθεία που το ελαχιστοποιεί, αρχίζοντας από την "αρχή" του καλύμματος. Για τα επόμενα σημεία δε χρειάζεται να ελέγξουμε τις διαστήματα/ευθείες που "απορρίφθηκαν", αφού αυτά είναι μεγαλύτερα από το προηγούμενο. Κάθε φορά που μια ευθεία απορρίπτεται δεν την ξαναελέγχουμε ποτέ. Οπότε αυτή η διαδικασία θα πάρει το πολύ $O(n+k)$ χρόνο. Μπορούμε να την επιταχύνουμε ακόμα περισσότερο (χωρίς να αλλάξουμε την συνολική πολυπλοκότητα βέβαια), αν για τους ελέγχους κάνουμε $binarysearch$.

■ Ο αλγόριθμος έχει πολυπλοκότητα $O(n+k)$.

Επιστρέφουμε στο αρχικό μας πρόβλημα με τα στέγαστρα. Η αναδρομική σχέση βέλτιστου κόστους ήταν η επόμενη:

$$cost(k) = \min_{1 \le i \le k} \{cost(i-1) + (x_i - x_k)^2 + C\} \text{ για: } 1 \le k \le n$$

Μπορούμε να την ξαναγράψουμε ως εξής:

$$cost(k) = \min_{1 \le i \le k} \{cost(i-1) + x_i^2 - 2x_i x_k + x_k^2 + C\}$$

$$= x_k^2 + C + \min_{1 \le i \le k} \{-2x_i \cdot x_k + x_i^2 + cost(i-1)\}$$

Αν εισάγουμε τους συμβολισμούς $b_i = x_i^2 + cost(i-1)$ και $a_i = -2x_i$ τότε έχουμε:

$$cost(k) = x_k^2 + C + \min_{1 \le i \le k} \{a_i x_k + b_i\}$$

Παρατηρούμε δηλαδή ότι μετατρέψαμε το αρχικό πρόβλημα σε ένα όμοιο πρόβλημα ελαχιστοποίησης τιμής σε ευθείες. Τώρα μπορούμε να γράψουμε έναν καλύτερο αλγόριθμο:

$\hookrightarrow cost[0] = 0$
$\triangleright$ Για i από 1 έως n:

- Προσθήκη της ευθείας με παραμέτρους $a[i] = -2x[i], b[i] = cost[i-1] + x[i]^2$ στο κυρτό κάλυμμα.
- Αναζητούμε στο κυρτό κάλυμμα την ευθεία που ελαχιστοποιεί το $x[i]$ και εύρεση του $cost[i]$ με αυτό τον τρόπο.

$\hookrightarrow$ Επέστρεψε $cost[n]$

Η προσθήκη στο κυρτό κάλυμμα όπως είπαμε μπορεί να επιφέρει κόστος $d = O(i)$ για την ευθεία i, αλλά αυτό θα σημαίνει ότι η ευθεία $i+1$ θα έχει κόστος $O(i+1-d)$, δηλαδή τελικά για όλες τις ευθείες τελικά θα έχουμε $O(n)$ χρόνο. Το ίδιο συμβαίνει και με την αναζήτηση της ευθείας που ελαχιστοποιεί ένα σημείο, αν αυτή έχει μεγάλο κόστος μειώνει το κόστος για τα επόμενα με τον ίδιο τρόπο. Τελικά θα έχουμε πολυπλοκότητα αλγορίθμου $O(n)$.

Άσκηση 4: Δρομολόγια Λεωφορείων στην Εποχή του Κορωνοϊού

α) Εφ'όσον δεν πρέπει να αλλαχθεί η διαδοχή των φοιτητών αν συμβολίσουμε με $total\_sens(i, m)$ το ελάχιστο συνολικό δείκτη ευαισθησίας για n φοιτητές (δηλαδή από τον πρώτο στην ουρά μέχρι τον i-οστό) σε m λεωφορεία θα έχουμε την αναδρομική σχέση:

$$total\_sens(i, m) = \min_{m-1 \le j \le n} \left\{ total\_sens(j, m-1) + \sum_{x=j+1}^{i} \sum_{y=x+1}^{i} A_{xy} \right\}$$

Η λύση του προβλήματος είναι το $total\_sens(n, k)$. Η ορθότητα επαληθεύεται από το γεγονός ότι παράγονται για όλα τα δυνατά j οι λύσεις με τους j τελευταίους φοιτητές στο τελευταίο λεωφορείο, από την βέλτιστη λύση για τους $n - j$ πρώτους φοιτητές στα προηγούμενα λεωφορεία.

Για το $sens(n, k)$ θα έχουμε $\mathrm{O}(kn)$ υποπροβλήματα/στοιχεία πίνακα. Αν δεν προσέξουμε η πολυπλοκότητα αυτής της λύσης είναι $O(kn^4)$ καθώς για κάθε υποπρόβλημα βρίσκουμε το ελάχιστο σε $O(n)$ στοιχεία υπολογίζοντας το κόστους $\mathrm{O}(n^2)$ άθροισμα.

Το μόνο που θα μπορούσαμε να βελτιώσουμε αυτή την στιγμή είναι ο υπολογισμός του αθροίσματος. Το οποίο θα μας χρειαστεί και για τις αρχικές συνθήκες $total\_sens(j, 1) = \sum_{x=1}^{j} \sum_{y=x+1}^{j} A_{xy}$.

Έχουμε τα εξής:

$$S(i, j) := \sum_{x=1}^{i} \sum_{y=1}^{i} A_{xy} = \sum_{x=1}^{i-1} \sum_{y=1}^{i} A_{xy} + \sum_{x=1}^{i} \sum_{y=1}^{i-1} A_{xy} - \sum_{x=1}^{i-1} \sum_{y=1}^{i-1} A_{xy} + A_{ij}$$

$$\iff S(i, j) = S(i-1, j) + S(i, j-1) - S(i-1, j-1) + A_{ij} \text{ (1)}$$

Άρα, εφ'όσον κάθε $S(i, j)$ εξαρτάται μόνο από "προηγούμενες" τιμές και υπολογίζεται από αυτές σε χρόνο $O(1)$ μπορούμε να υπολογίσουμε τον πίνακα $S[n][n]$ σε χρόνο $\Theta(n^2)$ (όσο τα στοιχεία του).
Αυτό δεν είναι ακριβώς αυτό που θέλουμε όμως γιατί τα αθροίσματα που μας ενδιαφέρουν δεν "αρχίζουν" πάντα από τον δείκτη 1. Παρατηρούμε το εξης όμως:

$$\sum_{x=j+1}^{i} \sum_{y=j+1}^{i} A_{xy} = \sum_{x=1}^{i} \sum_{y=1}^{i} A_{xy} - \sum_{x=1}^{i} \sum_{y=1}^{j} A_{xy} - \sum_{x=1}^{j} \sum_{y=1}^{i} A_{xy} + \sum_{x=1}^{j} \sum_{y=1}^{j} A_{xy}$$

$$\iff \sum_{x=j+1}^{i} \sum_{y=i+1}^{i} A_{xy} = S(i, i) - S(i, j) - S(j, i) - S(j, j) \text{ (2)}$$

Εμας μας ενδιαφέρει το άθροισμα $\sum_{x=j+1}^{i} \sum_{y=x+1}^{i} A_{xy}$. Ισχύει:

$$\sum_{x=j+1}^{i} \sum_{y=j+1}^{i} A_{xy} = \sum_{x=j+1}^{i} \sum_{y=x+1}^{i} A_{xy} + \sum_{y=1}^{i} \sum_{x=1}^{y} A_{xy} = 2 \sum_{x=j+1}^{i} \sum_{y=x+1}^{i} A_{xy}$$

Αφού: $A_{ii} = 0$ για κάθε i και $A_{ij} = A_{ji}$ για κάθε i, j.

Επιστρέφουμε τότε στην (2) και έχουμε:

$$2 \sum_{x=j+1}^{i} \sum_{y=x+1}^{i} A_{xy} = S(i,i) - S(i,j) - S(j,i) - S(j,j)$$

$$\iff \sum_{x=j+1}^{i} \sum_{y=x+1}^{i} A_{xy} = \frac{1}{2} \left[S(i,i) - S(i,j) - S(j,i) - S(j,j) \right]$$

Έτσι αν έχουμε προυπολογίσει τα $S(i,j)$ τότε τα αθροίσματα μπορούμε να τα υπολογίσουμε με τον παραπάνω τρόπο σε χρόνο $O(1)$. Η πολυπλοκότητα του αλγορίθμου μας τότε γίνεται $O(n^2 + kn^2) = O(kn^2)$.

Για την ελαχιστοποιήση του μέγιστου δείκτη ευαισθησίας μεταξύ των λεωφορείων έχουμε την επόμενη αναδρομική σχέση:

$$sens(i,m) = \min_{m-1 \le j \le n} \left\{ max \left\{ sens(j, m-1), \sum_{x=j+1}^{i} \sum_{y=x+1}^{i} A_{xy} \right\} \right\}$$

Η βέλτιστη λύση είναι το $sens(n,k)$ και με τον ίδιο τρόπο την βρίσκουμε σε χρόνο $O(kn^2)$.

Άσκηση 5: Το σύνολο των Συνδετικών Δέντρων

α) Αν δύο δέντρα T_1 και T_2 είναι διαφορετικά αυτό σημαίνει ότι διαφέρουν τουλάχιστον σε μια ακμή, δηλαδή το ένα έχει μια ακμή που δεν έχει το άλλο και αντίστροφα. Αλλιώς, το ένα θα ήταν υπερσύνολο του άλλου και θα έπρεπε να έχει παραπάνω ακμές. Όμως επειδή είναι συνδετικό δέντρο θα σήμαινε ότι έχει τουλάχιστον $(n-1)+1 = n$ ακμές, άρα θα περιείχε κύκλο, πράγμα άτοπο αφού είναι δέντρο. Προχωράμε στην απόδειξη.

Εφ'όσον το T_1 δεν είναι υπερσύνολο του T_2 θα υπάρχει και μια ακμή $e' \in T_2$, αλλά $e' \notin T_1$. Έστω η ακμή $e = \{u, v\}$, τότε αν την αφαιρέσουμε από το T_1 δεν είναι πια συνδετικό δέντρο. Έχουν δημιουργηθεί δύο ανεξάρτητα σύνολα. Αυτό που ανήκει το u, έστω S_u και αυτό που ανήκει το v, έστω S_v. Τότε το T_2 θα έχει σίγουρα μια ακμή που έχει άκρα μια κορυφή $u' \in S_u$ και μια κορυφή $v' \in S_v$, δηλαδή θα συνδέει δυο σημεία που ανήκουν σε αυτά τα ανεξάρτητα σύνολα. Αν δεν είχε αυτό θα σήμαινε ότι και αυτό έχει τουλάχιστον δυο ανεξάρτητα σύνολα και δε θα ήταν συνδετικό. Άρα, προσθέτοντας αυτή την ακμή (η οποία λόγω των προϋποθέσεων θα είναι σίγουρα διαφορετική από την e) στο $\mathrm{T}_1 \backslash \{e\}$, αυτό θα γίνει συνδετικό.

Αλγόριθμος για την εύρεση του e' δεδομένων των $T_1, T_2, e = \{u, v\}$:

↪ Αν τα δέντρα δε δίνονται σε μορφή πίνακα γειτνίασης (ή γενικότερα σε μορφή όπου εύρεση ακμής γίνεται σε $O(1)$) διατρέχουμε το T_1 είτε με BFS, είτε με DFS και αποθηκεύουμε όλες τις ακμές σε ένα $set/hash$ όπου η εύρεση κάποιας ακμής γίνεται σε $O(1)$.

↪ Εκτελούμε DFS στο T_2 αρχίζοντας από το u μέχρι να φτάσουμε στο v, επειδή είναι συνδετικό δέντρο το μονοπάτι είναι μοναδικό. Για κάθε ακμή στο μονοπάτι κοιτάζουμε αν υπάρχει στο T_1 σε $O(1)$ χρόνο. Αν δεν υπάρχει επιστρέφουμε αυτή την ακμή (αυτή είναι μια ζητούμενη ακμή που συνδέει το S_u με το S_v).

Πολυπλοκότητα αλγορίθμου: $2O(|V| + |E|) = O(|V|)$, αφού T_1, T_2 συνδετικά δέντρα και άρα $|E| = |V| - 1$.

β) Για να δείξουμε ότι το δέντρο είναι συνεκτικό αρκεί να δείξουμε ότι μεταξύ δύο τυχαίων κόμβων του δέντρου υπάρχει μόνοπάτι (όχι απαραίτητα μοναδικό). Αυτό είναι εύκολο να δειχθεί αφού αν έχουμε δύο συνδετικά δέντρα T_1, T_2 που διαφέρουν κατά μια ακμή (με βάση τον ορισμό της εκφώνησης), μπορούμε να εφαρμόσουμε την διαδικασία του ερωτήματος α) όσες φορές χρειαστεί στο T_1, μέχρι να ταυτίζεται με το T_2. Δηλαδή κάθε φορά αφαιρούμε μια ακμή του T_1 που δεν υπάρχει στο T_2 και προσθέτουμε μια ακμή που υπάρχει στο T_2 (αλλά όχι στο T_1) και το διατηρεί συνεκτικό. Αυτό είναι ουσιαστικά μια μετάβαση στο από το T_1 σε ένα άλλο συνδετικό δέντρο / κόμβο. του H. Εκτελόντας αυτό τον αλγόριθμο, βρίσκουμε και το συντομότερο μονοπάτι από το T_1 στο T_2, αφού κάθε φορά αφαιρούμε ακμή που δεν υπάρχει στο T_2 και προσθέτουμε μια που υπάρχει (δεν μπορούν στο H να συνδεθούν δύο κόμβοι που διαφέρουν κατά παραπάνω από μια ακμή).

Θέλουμε να αποδείξουμε ότι $d(T_1, T_2) = h \Leftrightarrow |T_2 \backslash T_1| = h$, αν με $d(T_1, T_2)$ συμβολίζουμε την απόσταση δύο συνδετικών δέντρων / κόμβων στο H.

Ανάστροφο: $|T_2 \backslash T_1| = h \Rightarrow d(T_1, T_2) = h$
- Αν $h = 1$ τότε αφού $|T_1 \backslash T_2| = |T_2 \backslash T_1| = 1$ τα T_1 και T_2 συνδέονται στο H εξ'ορισμού,
- Έστω ότι ισχύει η υπόθεση για κάποιο $h \geq 1$, δηλαδή $|T_2 \backslash T_1| = h \Rightarrow d(T_1, T_2) = h$.
- Θεωρούμε T_1, T_2 με $|T_2 \backslash T_1| = h + 1$.

Τότε από το ερώτημα α) μπορούμε να βρούμε ακμή e που υπάρχει στο T_1 αλλά όχι στο T_2 και μια e' που υπάρχει στο T_2 και όχι στο T_1. Τότε το $T_1' = T_1 \backslash \{e\} \cup \{e'\}$ διαφέρει με το T_2 σε h ακμές, δηλαδή $|T_2 \backslash T_1'| = h$. Άρα $d(T_1', T_2) = h$, λόγω επαγωγικής υπόθεσης. Ισχύει επίσης ότι $d(T_1, T_1') = 1$ εξ'ορισμού. Άρα $d(T_1, T_2) = h + 1$.

Ευθύ: $d(T_1, T_2) = h \Rightarrow |T_2 \backslash T_1| = h$
- Αν $h = 1$ τότε τα T_1 και T_2 συνδέονται στο H, άρα $|T_1 \backslash T_2| = |T_2 \backslash T_1| = 1$ εξ'ορισμού.
- Έστω ότι ισχύει η υπόθεση για κάποιο $h \geq 1$, δηλαδή $d(T_1, T_2) = h \Rightarrow |T_2 \backslash T_1| = h$.
- Θεωρούμε T_1, T_2 με $d(T_1, T_2) = h + 1$.
Τότε αρχίζοντας από το T_1 ακολουθούμε το ελάχιστο μονοπάτι που συνδέει τα T_1, T_2. Ακολουθώντας την πρώτη ακμή φτάνουμε σε ένα δέντρο T_1'. Εξ'ορισμού υπάρχει $e \in T_1 \backslash T_1'$ και $e' \in T_1' \backslash T_1$ μοναδικά ώστε: $T_1 \backslash \{e\} \cup \{e'\} = T_1'$. Το T_1 απέχει ελάχιστη απόσταση h από το T_2, δηλαδή $d(T_1', T_2) = h$. Άρα ισχύει είτε $|T_2 \backslash T_1| = h + 1$ ή $|T_2 \backslash T_1| = h - 1$. Αν ισχύει το δεύτερο, τότε λόγω του ανάστροφου που αποδείξαμε, $d(T_1, T_2) = h - 1$, άτοπο. Άρα $|T_2 \backslash T_1| = h + 1$.

Επομένως ακολουθώντας την διαδικασία που περιγράψαμε στην αρχή μπορούμε να υπολογίσουμε το συντομότερο μονοπάτι με τον παρακάτω αλγόριθμο.

- Βρίσκουμε το σύνολο $T_1 \backslash T_2$ σε $O(|V|)$ χρόνο (διατρέχουμε ένα από τα δέντρα και αποθηκεύουμε τις ακμές του σε δομή που απαιτεί χρόνο $O(1)$ για εύρεση στοιχείου).
- Με τον αλγόριθμο του (α) αφαιρούμε κάθε φορά μια $e \in T_1 \backslash T_2$ και προσθέτουμε το κατάλληλο $e' \in T_2 \backslash T_1$, ώστε να παραμείνει συνεκτικό. Επαναλαμβάνουμε μέχρι να αφαιρέσουμε όλες τις ακμές του $T_1 \backslash T_2$.

Η ορθότητα του αλγορίθμου επαληθεύεται από την παραπάνω ανάλυση μας. Η χρονική πολυπλοκότητα είναι $O(h|V|)$, όπου $h = |T_1 \backslash T_2|$.

γ) Ο αλγόριθμος έχει ως εξής:

↪ Κατασκευάζουμε το ελάχιστο συνδετικό δέντρο T του G σε χρόνο $O(|E|log|E|)$ με τον αλγόριθμο του $Kruskal$.
↪ Εκτελούμε DFS για κάθε κορυφή u του φέντρου T και αποθηκεύουμε για κάθε διαφορετική κορυφή v την μεγίστου βάρους ακμή στο μονοπάτι $u - v$. Αυτό γίνεται σε χρόνο $O\left((|E_T| + |V_T|)|V_T|\right) = O\left((|V| + |V|)|V|\right) = O(|V|^2)$.
↪ Για κάθε ακμή $e_i = \{u_i, v_i\}$ προσθέτουμε αυτή την ακμή στο ΕΣΔ T και αφαιρούμε την μέγιστη ακμή στο μονοπάτι $u_i - v_i$ στο T (που έχουμε υπολογίσει προηγουμένως). Αν η ακμή ήδη υπήρχε δεν αλλάζουμε κάτι στο T. Επιστρέφουμε το παραγόμενο T_i. Χρόνος $O(|E|)$.

Συνολική πολυπλοκότητα: $O(|V|^2 + |E|log|E|)$.

Η ορθότητα του αλγορίθμου βασίζεται στο ότι αν προσθέσουμε μια ακμή στο ΕΣΔ θα δημιουργηθεί ένας κύκλος μετατξύ των κορυφών που συνδέει. Αφαιρώντας την ακμή με το μέγιστο βάρος σε αυτό τον κύκλο (εξαιρώντας την ακμή που προσθέσαμε), έχουμε το ελάχιστο συνδετικό δέντρο που περιέχει αυτή την ακμή.