Αλγόριθμοι και Πολυπλοκότητα
2η σειρά γραπτών ασκήσεων

Νικόλαος Παγώνας, el18175

Άσκηση 1: Δίσκοι και Σημεία

Για κάθε σημείο p βρίσκουμε τα δύο σημεία p_1, p_2 που βρίσκονται πάνω στην l και απέχουν απόσταση r από το p. Έτσι, το κριτήριο κάλυψης του p από δίσκο είναι αν υπάρχει δίσκος με κέντρο ανάμεσα στα p_1, p_2.

Κάθε ζεύγος p_1, p_2 ορίζει ένα διάστημα πάνω στην ευθεία l, οπότε έχουμε n διαστήματα.

Το πρόβλημα πλέον ανάγεται στο να υπολογίσουμε τον ελάχιστο αριθμό σημείων πάνω στην ευθεία l ώστε κάθε διάστημα να περιέχει τουλάχιστον ένα σημείο.

Ο αλγόριθμος έχει ως εξής:

- Ταξινομούμε τα διαστήματα με βάση το τέλος τους. Έστω ότι μετά την ταξινόμηση έχουμε $s_1, ..., s_n$ τις αρχές των διαστημάτων και $f_1, ..., f_n$ τα τέλη τους.
- Δημιουργούμε μία (αρχικά κενή) λίστα η οποία θα περιέχει τα ζητούμενα σημεία. Έστω p_{top} το πιο πρόσφατο στοιχείο που έχουμε προσθέσει στη λίστα κάθε φορά.
- Για $i = 1, 2, ..., n$:
  - Αν $s_i \leq p_{\text{top}}$, τότε το i-οστό διάστημα περιέχει το p_{top} (επειδή τα διαστήματα είναι ταξινομημένα με βάση το τέλος τους), οπότε δεν κάνουμε τίποτα.
  - Αν $s_i > p_{\text{top}}$, τότε βάζουμε το f_i στην λίστα των σημείων.

Η πολυπλοκότητα του αλγορίθμου είναι $O(n \log n)$ λόγω της ταξινόμησης.

Απόδειξη ορθότητας:
Έστω ότι τα σημεία που δίνει ο greedy αλγόριθμος είναι τα $g_1, ..., g_m$ και αυτά που δίνει ο optimal/best αλγόριθμος είναι τα $b_1, ..., b_k$. Και στις δύο περιπτώσεις τα σημεία είναι ταξινομημένα από αριστερά προς τα δεξιά.

Αποδεικνύουμε (μέσω επαγωγής) ότι για κάθε $l \leq k$ ισχύει $g_l \geq b_l$. Η βάση της επαγωγής είναι $l = 1$. Και έχουμε $g_l \geq b_l$ για να καλύψουμε το πρώτο διάστημα.

Επαγωγική υπόθεση: $g_{l-1} \geq b_{l-1}$

Τώρα θα δείξουμε ότι $g_l \geq b_l$. Έστω a_i το αμέσως επόμενο διάστημα που έχει $s_i > g_{l-1}$. Ο greedy αλγόριθμος δημιουργεί ένα καινούργιο σημείο $g_l = f_i$ για να καλύψει το διάστημα αυτό. Όμως αν λάβουμε υπόψιν μας ότι $b_{l-1} \leq g_{l-1} \leq s_i$, αυτό σημαίνει ότι το b_{l-1} δεν καλύπτει το s_i. Δηλαδή το b_l οφείλει να καλύψει το a_i ($s_i \leq b_l \leq f_i$). Άρα τελικά $b_l \leq g_l$.

Παρομοίως μπορούμε να δείξουμε και ότι $m = k$. Ας υποθέσουμε ότι $m > k$. Αν ο greedy αλγόριθμος χρειάζεται να βάλει κάποιο extra σημείο μετά το g_k, αυτό θα σημαίνει ότι υπάρχει ένα διάστημα (έστω a_i) που έχει $s_i > g_k$. Όμως αν λάβουμε υπόψιν μας ότι $b_k \leq g_k$, τότε $b_k < s_i$. Επομένως το b_k δεν καλύπτει το a_i. Όμως το b_k είναι το τελευταίο σημείο της optimal/best λύσης, άτοπο.

Άσκηση 3: Τοποθέτηση Στεγάστρων (και Κυρτό Κάλυμμα)

(α)

Θα χρησιμοποιήσουμε Δυναμικό Προγραμματισμό. Η αναδρομική σχέση που περιγράφει το πρόβλημα είναι η εξής:

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

όπου $c(i)$ είναι το βέλτιστο κόστος για να στεγάσουμε τα σημεία από x_0 έως και x_i. Ορίζουμε $c(-1) = 0$ για να καλύψουμε την ειδική περίπτωση όπου έχουμε ένα ενιαίο στέγαστρο από το x_0 μέχρι και το x_i.
Σε κάθε βήμα θα πρέπει να βρίσκουμε το ελάχιστο μεταξύ i στοιχείων, επομένως η πολυπλοκότητα του αλγορίθμου θα είναι:

$$1 + 2 + 3 + ... = O(n^2).$$

(β)

Στην ουσία αναζητούμε μία κατάλληλη διάταξη των ευθειών $[l_1, l_2, ..., l_m]$ τέτοια ώστε η ευθεία l_1 να έχει ελάχιστη τιμή από το $-\infty$ μέχρι το σημείο τομής της με την l_2, η l_2 να έχει ελάχιστη τιμή από το σημείο τομής της με την l_1 έως το σημείο τομής της με την l_3 και ούτω καθεξής. Λόγω της ταξινόμησης των κλίσεων, κάθε νέα ευθεία θα μπαίνει στην διάταξη αυτή (και ενδεχομένως μπορεί να διαγράφει κάποιες από τις προηγούμενες ευθείες). Τελικά όμως κάθε ευθεία θα εισαχθεί και θα διαγραφεί το πολύ μία φορά, άρα η διάταξη αυτή φτιάχνεται σε $\Theta(n)$.

Αφού φτιάξουμε αυτή την διάταξη, παίρνουμε τα σημεία με τη σειρά, και για κάθε σημείο ξεκινάμε από την πρώτη ευθεία της διάταξης, βρίσκουμε ποια ευθεία καλύπτει το σημείο αυτό, και επιστρέφουμε την ευθεία αυτή. Λόγω της ταξινόμησης των σημείων όμως, είναι σίγουρο ότι η ευθεία που ελαχιστοποιεί το σημείο x_{i+1} θα είναι πιο μετά από την ευθεία που ελαχιστοποιεί το x_i.

Έτσι, θα χρειαστεί μία διάσχιση συνολικά, και έχουμε τελική πολυπλοκότητα $\Theta(n+k)$.

Για να εντάξουμε την παραπάνω λύση στο προηγούμενο πρόβλημα, πρέπει να αναδιατάξουμε την αναδρομική σχέση ως εξής:

$$\begin{aligned} c(i) &= \min_{0 \le j \le i} \left\{ c(j-1) + x_i^2 - 2x_i x_j + x_j^2 + C \right\} \\ &= x_i^2 + C + \min_{0 \le j \le i} \left\{ a_j x_i + b_j \right\} \end{aligned}$$

όπου $a_j = -2x_j$ και $b_j = c(j-1) + x_j^2$. Τώρα φαίνεται πως η αναδρομική σχέση μπορεί να λυθεί με άμεση εφαρμογή του παραπάνω αλγορίθμου σε γραμμικό χρόνο (εφόσον τόσο τα σημεία όσο και οι κλίσεις είναι ταξινομημένα).

Άσκηση 5: Το Σύνολο των Συνδετικών Δέντρων

(α)

- Έστω $T_1 \neq T_2$ συνδετικά δέντρα και έστω ακμή $e \in T_1 \setminus T_2 \neq \emptyset$.
- Αυτό σημαίνει ότι η προσθήκη της e στο T_2 θα δημιουργήσει κύκλο. Επίσης, αναγκαστικά θα υπάρχει τουλάχιστον μία ακμή e' του κύκλου που δεν ανήκει στο T_1, γιατί αν όλες οι ακμές του κύκλου ανήκαν στο T_1, τότε το T_1 θα περιείχε κύκλο.
- Αφαιρούμε την e' από το $\mathrm{T}_2 \cup \{e\}$ και προκύπτει πάλι συνεκτικό δέντρο, αφού χαλάσαμε τον κύκλο και επιπλέον έχουμε $n-1$ ακμές.
- Αυτό σημαίνει ότι υπάρχουν ακμές μεταξύ των δέντρων που είναι "ανταλλάξιμες" όσον αφορά την συνεκτικότητα.
- Το ζητούμενο προκύπτει ως το συμμετρικό του παραπάνω συμπεράσματος.

Έστω τώρα $e' = (u, v)$. Ο αλγόριθμος που βρίσκει την ακμή e' είναι ο εξής:

- Με αρχή το u κάνουμε DFS μέχρι να βρούμε μονοπάτι M που να συνδέει τα u, v στο T_1.
- Για κάθε ακμή e του μονοπατιού M ελέγχουμε αν ανήκει στο T_2 και όταν βρούμε $e \in M$ τέτοια ώστε $e \notin T_2$, την αφαιρούμε.

Το DFS κοστίζει $O(|V|)$ και ο έλεγχος $e \in T_2$ κοστίζει $O(1)$, άρα συνολικά έχουμε κόστος $O(|V|)$.

(β)

Έστω T_1, T_2 δύο συνεκτικά δέντρα που ανήκουν στο H και $d(T_1, T_2)$ η απόσταση των T_1, T_2 στο H. Θα χρησιμοποιήσουμε επαγωγή για να δείξουμε ότι $|\mathrm{T}_1 \setminus \mathrm{T}_2| = k$ αν $d(T_1, T_2) = k$.

Βάση: Εξ' ορισμού γνωρίζουμε ότι όταν $d(T_1, T_2) = 1$ τότε $|T_1 \setminus T_2| = 1$. Αυτή είναι και η βάση μας $(k = 1)$.

Επαγωγική υπόθεση: Έστω ότι $|T_1 \setminus T_2| = k$ αν $d(T_1, T_2) = k$.

Επαγωγικό βήμα: Θα δείξουμε το ζητούμενο για $k + 1$.

- Αφού $d(T_1, T_2) = k+1$ τότε θα υπάρχει $T' \in H$ τέτοιο ώστε $d(T_1, T') = 1$ και $d(T', T_2) = k$.
- Από την επαγωγική υπόθεση γνωρίζουμε ότι $|T' \setminus T_2| = k$ και από την βάση γνωρίζουμε ότι $|\mathrm{T}_1 \setminus T'| = 1$. Άρα θα ισχύει είτε $|T_1 \setminus T_2| = k - 1$ είτε $|T_1 \setminus T_2| = k + 1$.
- Αν $|\mathrm{T}_1 \setminus T_2| = k-1$ τότε από επαγωγική υπόθεση προκύπτει ότι $d(T_1, T_2) = k-1$ το οποίο είναι άτοπο αφού έχουμε υποθέσει ότι $d(T_1, T_2) = k + 1$.
- Επομένως, αναγκαστικά προκύπτει ότι $|T_1 \setminus T_2| = k + 1$.

Ο αλγόριθμος για τον υπολογισμό ενός συντομότερου μονοπατιού είναι ο εξής:

- Υπολογίζουμε το σύνολο ακμών του $T_2 \setminus T_1$.
- Για κάθε ακμή αυτού του συνόλου προσθέτουμε την ακμή e στο υπάρχον δέντρο.

Ο αλγόριθμος είναι ορθός διότι αν $T_2 \setminus T_1 = k$ τότε χρειαζόμαστε k βήματα για να μετατρέψουμε το T_1 στο T_2. Επομένως έχουμε βρει ένα μονοπάτι μήκους k στο H, δηλαδή το συντομότερο μονοπάτι.

Όσον αφορά το κόστος του αλγορίθμου, έχουμε τα εξής:

- Σε $O(|V|)$ αποθηκεύουμε το T_1 ως ένα δέντρο όπου κάθε κόμβος δείχνει στον πατέρα του.
- Για κάθε ακμή $e \in T_2$ ελέγχουμε σε σταθερό χρόνο αν $e \in T_1$
- Έτσι κάνουμε k updates στις ακμές, και κάθε update χρειάζεται $O(|V|)$.

Επομένως, η συνολική πολυπλοκότητα του αλγορίθμου είναι $O(k \cdot |V|)$.

(γ)

Για να λύσουμε αυτό το πρόβλημα, αρχικά θα φτιάξουμε το MST (μία φορά) και για κάθε ακμή e που δεν ανήκει σ' αυτό, θα βρίσκουμε το MST που την περιέχει, βάση του MST που φτιάξαμε αρχικά. Αυτό γίνεται αν προσθέσουμε την e στο MST και μετά αφαιρέσουμε την βαρύτερη ακμή στον κύκλο που δημιουργείται.

Υπολογίζουμε λοιπόν το MST (έστω T) σε $O(|E|log|E|)$ με Kruskal. Για την εύρεση του κύκλου που περιέχει την e, μπορούμε να κάνουμε ένα DFS (σε $O(|V|)$) από κάθε κορυφή u και κρατάγαμε το βάρος της βαρύτερης ακμής στο μονοπάτι $u - v$ πάνω στο T, για κάθε άλλη κορυφή $v \neq u$. Συνολικά λοιπόν θα είχαμε χρόνο $O(|V|^2)$.