Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Αλγόριθμοι και Πολυπλοκότητα

Διδάσκοντες: Δημήτρης Φωτάκης, Δώρα Σούλιου

1η Σειρά Γραπτών Ασκήσεων - Ημ/νία Παράδοσης 10/11/2025

Άσκηση 1: Ταξινόμηση (1.5 μον.)

Πριν την επίδειξη γραπτών στο μάθημα "Προγραμματισμός Η/Υ", ταξινομούμε τα γραπτά σε αλφαβητική σειρά. Πέρυσι, κατά τη μεταφορά των γραπτών και μόλις $30'$ πριν την επίδειξή τους, η στοίβα σκορπίστηκε στο πάτωμα. Όταν μαζέψαμε τα γραπτά και προσπαθώντας να τα ταξινομήσουμε πάλι, παρατηρήσαμε ότι κάθε γραπτό βρισκόταν σε απόσταση το πολύ k θέσεων (προς τα πάνω ή προς τα κάτω) από την αρχική του θέση στην ταξινομημένη στοίβα. Έπρεπε λοιπόν να ταξινομήσουμε πολύ γρήγορα μια στοίβα με n γραπτά, που ήταν όμως k-*προταξινομημένη*, με την έννοια ότι κάθε γραπτό βρισκόταν σε απόσταση το πολύ k θέσεων από την τελική του θέση στην ταξινομημένη στοίβα.

1. Να διατυπώσετε συγκριτικό αλγόριθμο με χρόνο εκτέλεσης $O(n \log k)$ για την ταξινόμηση ενός k-προταξινομημένου πίνακα με n στοιχεία. Να αιτιολογήσετε συνοπτικά την ορθότητα και την υπολογιστική πολυπλοκότητα του αλγορίθμου σας.
2. Να δείξετε ότι κάθε συγκριτικός αλγόριθμος για την ταξινόμηση ενός k-προταξινομημένου πίνακα με n στοιχεία έχει χρόνο εκτέλεσης χειρότερης περίπτωσης $\Omega(n \log k)$.

Άσκηση 2: Υπολογισμός Κυρίαρχων Θέσεων (1.1 μον.)

Θεωρούμε πίνακα $A[1 \dots n]$ με n φυσικούς αριθμούς. Για κάθε $i = 2, \dots, n$, η θέση που *κυριαρχεί* της θέσης i στον πίνακα A είναι η πλησιέστερη θέση που προηγείται της i και η τιμή της ξεπερνά την τιμή $A[i]$. Τυπικά, θεωρώντας ότι $A[0] = \infty$, η θέση $B[i]$ που *κυριαρχεί* της θέσης i στον A είναι η μέγιστη θέση j, $0 \leq j < i$, για την οποία ισχύει ότι $A[j] > A[i]$ (βλ. ότι $B[1] = 0$). Να διατυπώσετε αποδοτικό αλγόριθμο που υπολογίζει τη θέση $B[i]$ που κυριαρχεί της θέσης i, για κάθε $i = 1, \dots, n$, στον πίνακα $A[1 \dots n]$. Να αιτιολογήσετε την ορθότητα και την υπολογιστική πολυπλοκότητα του αλγορίθμου σας.

Άσκηση 3: Πλησιέστερο Ζεύγος Σημείων κατά Προσέγγιση (3 μον.)

Θεωρούμε το πρόβλημα του πλησιέστερου ζεύγους σημείων στις d διαστάσεις, όπου δίνονται n (διαφορετικά μεταξύ τους) σημεία $\vec{x}_1, \dots, \vec{x}_n \in \mathbb{R}^d$ στον d-διάστατο χώρο, και ζητείται η ελάχιστη (Ευκλείδεια) απόσταση δ^* μεταξύ ενός ζεύγους από αυτά. Θεωρούμε τον παρακάτω αλγόριθμο που εξελίσσεται σε φάσεις:

1. Έστω S το σύνολο των διαθέσιμων σημείων στην τρέχουσα φάση. Αρχικά $S = \{\vec{x}_1, \dots, \vec{x}_n\}$.
2. Έστω $\vec{x}_k \in S$ ένα διαθέσιμο σημείο και $\delta_k = \min_{\vec{x} \in S \setminus \{\vec{x}_k\}} \{\text{dist}(\vec{x}_k, \vec{x})\}$ η ελάχιστη (Ευκλείδεια) απόσταση του $\vec{x}_k$ από τα υπόλοιπα διαθέσιμα σημεία.
3. Θεωρούμε d-διάστατο πλέγμα πλευράς $\delta_k/(3\sqrt{d})$. Συσχετίζουμε κάθε διαθέσιμο σημείο με το κελί του πλέγματος στο οποίο ανήκει και αφαιρούμε από το σύνολο S των διαθέσιμων σημείων όλα τα σημεία για τα οποία τα γειτονικά κελιά του κελιού με το οποίο συσχετίστηκαν είναι "κενά" (δηλ. τα κελιά αυτά δεν έχουν συσχετιστεί με κανένα σημείο).
4. Ενόσω $S \neq \emptyset$, προχωρούμε στην επόμενη φάση με το ενημερωμένο σύνολο S. Αν $S = \emptyset$, επιστρέφουμε ως απάντηση την απόσταση δ_k.

(α) Μπορούμε να εγγυηθούμε ότι ο παραπάνω αλγόριθμος τερματίζει; Αν ναι, να εξηγήσετε γιατί. Αν όχι, να δώσετε ένα παράδειγμα όπου ο αλγόριθμός δεν τερματίζει.

(β) Για κάθε φάση του αλγόριθμου, έχουμε ότι $\delta^* \leq \delta_k$. Δεχόμενοι ότι ο αλγόριθμος τερματίζει, να προτείνετε (και να αιτιολογήσετε κατάλληλα) ένα κάτω φράγμα στο δ^* με βάση την απόσταση δ_k στην τελευταία φάση του αλγορίθμου (από το κάτω φράγμα προκύπτει ο *λόγος προσέγγισης* στην ελάχιστη απόσταση δ^* που εγγυάται ο αλγόριθμος).

(γ) Να εξηγήσετε πως μπορούμε να υλοποιήσουμε το βήμα (3) του αλγορίθμου σε χρόνο γραμμικό στο $|S|$ και στο 3^d.

(δ) Ποιος είναι ο χρόνος εκτέλεσης του παραπάνω αλγόριθμου στη χειρότερη περίπτωση; Να προτείνετε μέθοδο επιλογής του σημείου $\vec{x}_k$ ώστε ο χρόνος εκτέλεσης να είναι γραμμικός στο $|S|$ και στο 3^d (αν χρησιμοποιήσετε τυχαιότητα, αυτό χρειάζεται να ισχύει για τον αναμενόμενο χρόνο εκτέλεσης).

(ε) Για να υπολογίσουμε το δ^*, θεωρούμε d-διάστατο πλέγμα πλευράς δ_k. Για κάθε σημείο $\vec{x}_i$ υπολογίζουμε την ελάχιστη απόσταση του $\vec{x}_i$ από τα υπόλοιπα σημεία που έχουν συσχετιστεί είτε με το ίδιο κελί είτε με κελιά γειτονικά προς το κελί του $\vec{x}_i$. Με βάση το κάτω φράγμα του (β) στο δ^*, να υπολογίσετε ένα άνω φράγμα (που δεν πρέπει να εξαρτάται από το n) στο πλήθος αυτών των σημείων. Να αιτιολογήσετε την ορθότητα και την υπολογιστική πολυπλοκότητα αυτού του τελευταίου βήματος.

Άσκηση 4: Ταξίδι με Ηλεκτρικό Αυτοκίνητο (2.2 μον.)

(α) Πρόκειται να πάμε ένα μεγάλο ταξίδι με το νέο ηλεκτρικό μας αυτοκίνητο που θα διαρκέσει $k \geq 2$ ημέρες. Έχουμε επιλέξει τους $n > k$ σταθμούς του ταξιδιού (και τη σειρά με την οποία θα τους επισκεφθούμε) και έχουμε υπολογίσει τις αποστάσεις $d_1, d_2, \ldots, d_n$ μεταξύ τους (d_1 είναι η απόσταση του πρώτου σταθμού από την αφετηρία, d_2 είναι η απόσταση του πρώτου από τον δεύτερο σταθμό, κοκ., όλες οι αποστάσεις είναι θετικοί ακέραιοι). Για να αποφύγουμε την ανάγκη στάσεων φόρτισης κατά τη διάρκεια της ημέρας, θέλουμε να προγραμματίσουμε το ταξίδι μας ώστε να ελαχιστοποιήσουμε τη μέγιστη απόσταση που θα διανύσουμε μέσα σε μία ημέρα (μπορούμε να σταματήσουμε για διανυκτέρευση και φόρτιση μόνο σε κάποιον από τους επιλεγμένους σταθμούς). Να διατυπώσετε έναν αποδοτικό αλγόριθμο για αυτό το πρόβλημα. Να αιτιολογήσετε την ορθότητα και την υπολογιστική πολυπλοκότητα του αλγορίθμου σας.

(β) Θεωρούμε δίκτυο n σταθμών φόρτισης ηλεκτρικών αυτοκινήτων κατά μήκος μιας (μεγάλης σε μήκος) οδικής αρτηρίας. Για ευκολία, θεωρούμε ότι η οδική αρτηρία είναι μονής κατεύθυνσης και τα αυτοκίνητα μετακινούνται μόνο από σταθμούς με μικρότερο δείκτη σε σταθμούς με μεγαλύτερο δείκτη. Τεχνολογικοί περιορισμοί επιτρέπουν στα αυτοκίνητα που φορτίζουν στον σταθμό i να φτάσουν το πολύ μέχρι τον σταθμό a_i, όπου $n \geq a_i \geq i+1$ (αυτό ανεξάρτητα από την ενέργεια που έχουν τα αυτοκίνητα όταν φτάνουν στον i – μάλιστα συμβαίνει συχνά να έχουμε $a_i > a_{i+1}$). Οι δείκτες $(a_1, \ldots, a_{n-1})$ του μέγιστου σταθμού a_i όπου μπορούμε να φτάσουμε φορτίζοντας σε κάθε σταθμό i είναι γνωστοί (έχουμε ακόμη τετριμμένα ότι $a_n = n$).

Για τη διαχείριση του δικτύου χρειαζόμαστε μια δομή δεδομένων που με βάση τους δείκτες $(a_1, \ldots, a_{n-1})$, υπολογίζει το ελάχιστο πλήθος φορτίσεων (και τους αντίστοιχους σταθμούς) για ένα αυτοκίνητο k με αφετηρία τον σταθμό s_k και προορισμό τον σταθμό t_k (όπου $1 \leq s_k < t_k \leq n$). Να περιγράψετε την αρχική επεξεργασία που απαιτείται στους δείκτες $(a_1, \ldots, a_{n-1})$ και τον αλγόριθμο για την απάντηση των ερωτημάτων $\text{chrg}(s_k, t_k)$ σχετικά με το πλήθος και τους σταθμούς φόρτισης για να ταξιδέψουμε από τον σταθμό s_k στον σταθμό t_k. Να αιτιολογήσετε την ορθότητα του αλγορίθμου σας και την χρονική πολυπλοκότητα για την αρχική επεξεργασία και την απάντηση κάθε ερωτήματος $\text{chrg}(s_k, t_k)$.

Άσκηση 5: Γρήγορη (Διπλή!) Επιλογή στο Πεδίο της Μάχης (2.2 μον.)

Εδώ και πολλά χρόνια, οι κάτοικοι της χώρας των Αλγορίθμων δεν μπορούν να συμφωνήσουν ως προς το πόσο σημαντικός είναι ο αλγόριθμος Quickselect! Οι κάτοικοι έχουν χωριστεί σε δύο στρατόπεδα και κατέχουν

δύο εκτοξευτές σωματιδίων τους οποίους θα χρησιμοποιήσουν σαν όπλο για να επιβάλουν τη γνώμη τους. Θεωρούμε πως το πεδίο μάχης έχει τη μορφή μιας απέραντης ευθείας. Οι δύο αντιμαχόμενες πλευρές τοποθετούν τους εκτοξευτές τους στις θέσεις $x = 0$ και $x = L$. Ο εκτοξευτής του πρώτου στρατοπέδου εκπέμπει σωματίδια τύπου a προς τα δεξιά και ο εκτοξευτής του δεύτερου στρατοπέδου εκπέμπει σωματίδια τύπου b προς τα αριστερά. Θεωρούμε πως η μάχη ξεκινάει τη χρονική στιγμή $t = 0$, και ο τρόπος με τον οποίο αλληλεπιδρούν τα σωματίδια καθορίζεται από τους παρακάτω κανόνες:

- Αν δύο σωματίδια διαφορετικού τύπου συγκρουστούν μεταξύ τους, τότε αυτά εξαϋλώνονται.
- Αν δύο σωματίδια ίδιου τύπου συγκρουστούν, τότε το ένα προσπερνάει το άλλο χωρίς να εξαϋλωθούν.

Η τεχνολογία των εκτοξευτών επιτρέπει στην ταχύτητα των εκπεμπόμενων σωματιδίων να μεταβάλλεται διαρκώς. Έτσι, τα εκπεμπόμενα σωματίδια πραγματοποιούν κίνηση με σταθερή κατεύθυνση (δηλ. τα σωματίδια τύπου a κινούνται πάντα προς τα δεξιά και τα σωματίδια τύπου b κινούνται πάντα προς τα αριστερά), αλλά όχι απαραίτητα ομαλή (δηλ. η ταχύτητα ενός σωματιδίου μπορεί να αυξομειώνεται αυθαίρετα στον χρόνο). Γνωρίζουμε μόνο ότι κάθε αντιμαχόμενη πλευρά εκτοξεύει n σωματίδια και ότι η ταχύτητα κάθε σωματιδίου δεν μπορεί ποτέ να ξεπεράσει το V_{max} και δεν μπορεί ποτέ να είναι μικρότερη από $V_{\text{min}} > 0$.

Οι στρατηγοί των δύο στρατοπέδων ενδιαφέρονται να μάθουν πότε γίνεται η πρώτη σύγκρουση σωματιδίων και ποια είναι τα σωματίδια που συμμετέχουν σε αυτή. Να διατυπώσετε έναν αποδοτικό αλγόριθμο για αυτό το πρόβλημα και να αιτιολογήσετε την ορθότητα και την υπολογιστική πολυπλοκότητα του αλγορίθμου σας, σε καθεμία από τις παρακάτω περιπτώσεις:

- Η πλευρά που εκτοξεύει τα σωματίδια a έχει στη διάθεσή της έναν υπερυπολογιστή που για κάθε σωματίδιο i (το i μπορεί να είναι τύπου a ή τύπου b) και για κάθε χρονική στιγμή $t \geq 0$ υπολογίζει σε σταθερό χρόνο (και με απόλυτη ακρίβεια) τη θέση του i τη χρονική στιγμή t (αν το i είναι το μοναδικό σωματίδιο στο πεδίο της μάχης).
- Η πλευρά που εκτοξεύει τα σωματίδια b έχει στη διάθεσή της έναν άλλο υπερυπολογιστή που για κάθε ζευγάρι σωματιδίων i (τύπου a) και j (τύπου b) υπολογίζει σε σταθερό χρόνο (και με απόλυτη ακρίβεια) τη χρονική στιγμή t που τα δύο σωματίδια συγκρούονται (αν τα i και j είναι τα μοναδικά σωματίδια στο πεδίο της μάχης).

Στην πρώτη (αντ. δεύτερη) περίπτωση, ο υπερυπολογιστής λαμβάνει ως είσοδο ένα ζεύγος (i, t) (αντ. (i, j)) και επιστρέψει τη θέση του σωματιδίου i τη χρονική στιγμή t (αντ. τη χρονική στιγμή που τα σωματίδια i και j συγκρούονται) και μόνον αυτό. Σε καθεμία από τις δύο περιπτώσεις, ο αλγόριθμός σας πρέπει να ελαχιστοποιεί το πλήθος των κλήσεων στον υπερυπολογιστή (ως συνάρτηση των n, L, V_{max} και V_{min}).

Άσκηση 6 - Bonus: Ερωτήματα Ανήκειν (1 μον.)

Βασιζόμενοι σε hashing, θέλουμε να αναπτύξουμε μια δομή δεδομένων που διατηρεί μια σύνοψη ενός συνόλου S, αποτελούμενου από n θετικούς φυσικούς αριθμούς, και υλοποιεί αποδοτικά (πρακτικά, σε σταθερό χρόνο) τις παρακάτω δύο λειτουργίες: "πρόσθεσε το x στο S" και "έλεγξε αν $x \in S$". Για αυτό τον σκοπό, διατηρούμε μια σύνοψη του S σε πίνακα A με m θέσεις, μεγέθους ενός δυαδικού ψηφίου η καθεμία, και χρησιμοποιούμε συναρτήσεις hash $h : \mathbb{N}_+ \to [m]$ τέτοιες ώστε $\mathbb{P}\text{rob}[h(x) = j] = 1/m$, για κάθε $x \in \mathbb{N}_+$ και για κάθε $j \in [m]$. Απαιτούμε, τέλος, η χρήση τυχαιότητας να μην οδηγεί σε false negatives (δηλ. κάθε αρνητική απάντηση στο ερώτημα "$x \in S$?" πρέπει να είναι σωστή), ενώ μπορεί να οδηγεί σε false positives (δηλ. μπορεί μια θετική απάντηση στο ερώτημα "$x \in S$?" να είναι λανθασμένη) με μικρή όμως πιθανότητα.

(α) Σχεδιάστε μια τέτοια δομή δεδομένων με χρήση μιας μόνο hash συνάρτησης και υπολογίστε την πιθανότητα μιας false positive απάντησης. Ποια είναι η πιθανότητα για false positive απάντηση αν $m = 8n$;

(β) Σχεδιάστε μια τέτοια δομή δεδομένων με χρήση $k \geq 1$ ανεξάρτητων hash συναρτήσεων και υπολογίστε την πιθανότητα μιας false positive απάντησης. Ποια είναι η βέλτιστη τιμή του k και ποια η αντίστοιχη πιθανότητα για false positive απάντηση αν $m = 8n$;