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

Αλγόριθμοι και Πολυπλοκότητα

Διδάσκοντες: Α. Παγουρτζής, Δ. Φωτάκης, Δ. Σούλιου, Π. Γροντάς

3η Σειρά Γραπτών Ασκήσεων - Ημ/νία Παράδοσης 7/2/2023

Άσκηση 1: Υπολογισιμότητα (2 μον.)

Στο **Πρόβλημα Διοφαντικών Εξισώσεων** (γνωστό και ως '10ο πρόβλημα του Hilbert') δίνεται ως είσοδος μια πολυωνυμική εξίσωση με ακέραιους συντελεστές και πεπερασμένο αριθμό αγνώστων και ζητείται αν έχει ακέραιες λύσεις.

(α) Περιγράψτε αλγόριθμο που ημιαποφασίζει το **Πρόβλημα Διοφαντικών Εξισώσεων**.

(β) Περιγράψτε αναγωγή από το **Πρόβλημα Διοφαντικών Εξισώσεων** στο **Πρόβλημα Τερματισμού**.

Άσκηση 2: Πολυπλοκότητα - Αναγωγές (4 μον.)

Αποδείξτε τα παρακάτω:

(α) Έστω μια κλάση πολυπλοκότητας $\mathcal{C}$. Αν ένα πρόβλημα Π είναι $\mathcal{C}$-πλήρες ως προς μια αναγωγή $\leq_R$ τότε το συμπλήρωματικό πρόβλημα Π' είναι co$\mathcal{C}$-πλήρες ως προς την $\leq_R$.

(β) Το πρόβλημα **Tautology** (δίνεται τύπος του προτασιακού λογισμού και ζητείται αν είναι ταυτολογία) είναι coNP-πλήρες (ως προς την αναγωγή πολυωνυμικού χρόνου $\leq_P$).

(γ) Αν ένα πρόβλημα της κλάσης coNP είναι NP-πλήρες (ως προς την αναγωγή πολυωνυμικού χρόνου $\leq_P$) τότε NP = coNP.

(δ) Το πρόβλημα **NAE3SAT** (Not-All-Equal 3-SAT: δίνεται ένας τύπος του προτασιακού λογισμού σε μορφή 3-SAT και ζητείται αν υπάρχει ανάθεση η οποία σε κάθε clause ικανοποιεί τουλάχιστον 1 και το πολύ 2 literals) είναι NP-πλήρες.

(ε) Το πρόβλημα **Επιλογής Αντιπροσώπων** που ορίζεται παρακάτω είναι NP-πλήρες. Ορισμός προβλήματος: Δίνεται ένα σύνολο ατόμων U που διαμένουν σε μια περιοχή, και διάφορες κοινωνικές ομάδες (υποσύνολα του U) $U_1, \ldots, U_r$, όχι κατ'ανάγκη ξένες μεταξύ τους, καθώς και ένας ακέραιος αριθμός k. Κάθε ομάδα μπορεί να αντιπροσωπεύεται από οποιοδήποτε μέλος της. Στο τοπικό κοινοβούλιο μπορούν να συμμετέχουν k αντιπρόσωποι. Ζητείται εάν είναι δυνατόν να αντιπροσωπευθούν όλες οι ομάδες στο κοινοβούλιο, δηλαδή αν υπάρχει $R \subseteq U$, με $|R| \leq k$, τέτοιο ώστε $\forall i \leq r, U_i \cap R \neq \emptyset$.

Άσκηση 3: Προσεγγιστικοί Αλγόριθμοι - Vertex Cover (2 μον.)

Θεωρήστε τον παρακάτω αλγόριθμο προσέγγισης για το πρόβλημα Vertex Cover σε ένα γράφημα $G(V, E)$, όπου κάθε κορυφή $v \in V$ έχει βάρος $w(v) > 0$. Το ζητούμενο είναι να υπολογίσουμε ένα σύνολο κορυφών $C \subseteq V$, με ελάχιστο συνολικό βάρος, ώστε κάθε ακμή $e \in E$ να έχει τουλάχιστον ένα άκρο της στο C.

Να δείξετε ότι ο αλγόριθμος επιτυγχάνει λόγο προσέγγισης 2. Ειδικότερα, να δείξετε ότι για τα C και $c(e)$, $e \in E$, στο τέλος του αλγορίθμου, ισχύουν τα εξής:

1. Το C καλύπτει όλες τις ακμές.
2. Το συνολικό βάρος του C είναι μικρότερο ή ίσο του $2\sum_{e\in E} c(e)$.
3. Το συνολικό βάρος της βέλτιστης λύσης είναι μεγαλύτερο ή ίσο του $\sum_{e\in E} c(e)$.

Να βρείτε ακόμη ένα (όσο γίνεται απλούστερο) γράφημα όπου ο αλγόριθμος υπολογίζει ένα Vertex Cover με συνολικό βάρος διπλάσιο από αυτό της βέλτιστης λύσης.

WeightedVertexCover($G(V, E)$, w)
  $C \leftarrow \emptyset$;
  $\forall v \in V, t(v) \leftarrow w(v)$;
  $\forall e \in E, c(e) \leftarrow 0$;
  while C δεν είναι vertex cover **do**
    $e = \{u, v\}$ μια ακάλυπτη ακμή;
    $\delta \leftarrow \min\{t(u), t(v)\}$;
    $t(u) \leftarrow t(u) - \delta$;
    $t(v) \leftarrow t(v) - \delta$;
    $c(e) \leftarrow \delta$;
    $C \leftarrow C \cup \{z \in \{u, v\} \mid t(z) = 0\}$;
  return(C);

Άσκηση 4: Πιθανοτικοί Άλγόριθμοι - Έλεγχος Ταξινόμησης (2 μον.)

Θεωρούμε ότι ένας πίνακας ακεραίων $A[1 \dots n]$ είναι *σχεδόν ταξινομημένος* αν υπάρχουν το πολύ $n/4$ στοιχεία τα οποία αν διαγράψουμε, τα στοιχεία που απομένουν είναι ταξινομημένα. Για παράδειγμα, ο πίνακας $[1, 2, 3, 5, 4, 7, 9, 6, 10, 11, 12, 8]$ είναι σχεδόν ταξινομημένος, αφού η διαγραφή των στοιχείων 5, 6, και 8 δίνει τον πίνακα $[1, 2, 3, 4, 7, 9, 10, 11, 12]$, που είναι ταξινομημένος. Για απλότητα, υποθέτουμε στη συνέχεια ότι όλα τα στοιχεία του πίνακα A είναι διαφορετικά.

Στοχεύουμε στη διατύπωση ενός πιθανοτικού αλγόριθμου *λογαριθμικού χρόνου* που θα διακρίνει πίνακες που είναι (πλήρως) ταξινομημένοι από πίνακες που δεν είναι σχεδόν ταξινομημένοι[1]. Συγκεκριμένα, αν ο πίνακας A είναι πλήρως ταξινομημένος, ο αλγόριθμος θα αποφαίνεται πάντα (δηλ. με πιθανότητα 1) ότι ο A είναι σχεδόν ταξινομημένος. Αν ο πίνακας A δεν είναι σχεδόν ταξινομημένος, ο αλγόριθμος θα αποφαίνεται, με πιθανότητα τουλάχιστον 90%, ότι ο A δεν είναι σχεδόν ταξινομημένος.

(α) Θεωρούμε τον αλγόριθμο που επιλέγει τυχαία (και ανεξάρτητα) k θέσεις $a_1, \dots, a_k$ του πίνακα A, και αποφαίνεται ότι ο A είναι σχεδόν ταξινομημένος αν για κάθε θέση a_i που έχει επιλεγεί, $A[a_i - 1] \leq A[a_i] \leq A[a_i + 1]$. Αν υπάρχει έστω μία θέση a_i που δεν ικανοποιεί αυτό το κριτήριο, ο αλγόριθμος αποφαίνεται ότι ο A δεν είναι σχεδόν ταξινομημένος. Να δώσετε παράδειγμα πίνακα με n στοιχεία όπου ο παραπάνω αλγόριθμος χρειάζεται να ελέγξει $\Omega(n)$ θέσεις ώστε η πιθανότητα λάθους να γίνει μικρότερη του 10%.

(β) Υποθέτουμε ότι εφαρμόζουμε την παρακάτω εκδοχή της Δυαδικής Αναζήτησης σε έναν πίνακα A που μπορεί να μην είναι ταξινομημένος (με κίνδυνο φυσικά η αναζήτηση να αποτύχει, αν και το στοιχείο x υπάρχει στον A):

BINARY-SEARCH(A, x, low, up)
  if low $=$ up **then return** low;
  else mid $\leftarrow \lceil (\text{low} + \text{up})/2 \rceil$;
    if $x < A[\text{mid}]$ **then return** BINARY-SEARCH(A, x, low, mid $- 1$);
    else return BINARY-SEARCH(A, x, mid, up);

Έστω λοιπόν ότι για κάποιες τιμές x, y, η κλήση BINARY-SEARCH($A, x, 1, n$) επιστρέφει τη θέση k και η κλήση BINARY-SEARCH($A, y, 1, n$) επιστρέφει τη θέση ℓ. Να δείξετε ότι αν $k < \ell$, τότε $x < y$.

(γ) Θεωρούμε τώρα τον αλγόριθμο που επιλέγει τυχαία (και ανεξάρτητα) k θέσεις $a_1, \dots, a_k$ του πίνακα A, και αποφαίνεται ότι ο πίνακας A είναι σχεδόν ταξινομημένος αν για κάθε θέση a_i που έχει επιλεγεί, ισχύει ότι

[1] Εκ πρώτης όψης, αυτό είναι ένας φιλόδοξος στόχος. Οποιοσδήποτε ντετερμινιστικός αλγόριθμος για αυτό το πρόβλημα πρέπει να έχει τουλάχιστον γραμμικό χρόνο εκτέλεσης, αφού θα πρέπει να διαβάσει τουλάχιστον $3n/4$ από τα στοιχεία του πίνακα A.

$a_i = \textsc{Binary-Search}(A, A[a_i], 1, n)$. Αν υπάρχει έστω μία θέση a_i που δεν ικανοποιεί αυτό το κριτήριο, ο αλγόριθμος αποφαίνεται ότι ο πίνακας A δεν είναι σχεδόν ταξινομημένος. Χρησιμοποιώντας το (β), να δείξετε ότι για κάθε πίνακα A με n θέσεις, αρκεί ο αλγόριθμος να ελέγξει $O(1)$ θέσεις, ώστε η πιθανότητα λάθους να γίνει μικρότερη του 10%.