

Αλγόριθμοι και Πολυπλοκότητα – ενδιάμεσες εξετάσεις χειμερινού εξαμήνου 2024-2025

Τμήμα Πληροφορικής και Τηλεπικοινωνιών, Πανεπιστημίου Ιωαννίνων

5/12/2024

Διάρκεια εξέτασης: 2 ώρες

Θέμα 1 [A=1, B=2]

A) Απαντήστε με ΣΩΣΤΟ ή ΛΑΘΟΣ:

1. Η πολυπλοκότητα χειρότερης περίπτωσης της QuickSort είναι $O(n \log n)$.
2. Ο κατακερματισμός κούκου χρησιμοποιεί 2 συναρτήσεις κατακερματισμού.
3. Ο συντελεστής φόρτωσης (load factor) ενός πίνακα κατακερματισμού ισούται με το μέγεθος του πίνακα κατακερματισμού διά του πλήθους των στοιχείων που έχουν εισαχθεί στον πίνακα κατακερματισμού.
4. Η συνάρτηση κατακερματισμού murmur είναι κρυπτογραφική συνάρτηση κατακερματισμού.
5. Στον κατακερματισμό κούκου η λειτουργία της εύρεσης ενός κλειδιού και η λειτουργία της διαγραφής ενός κλειδιού είναι πάντα $O(1)$.

B) Δίνεται η ακόλουθη αναπαράσταση, σε Python, ενός κατευθυνόμενου γράφου ως λίστα γειτονικότητας (adjacency list):

```
from collections import defaultdict
graph = defaultdict(list,
    {"a":["b","c","f"],
     "b":["c","d","e"],
     "c":["d"],
     "d":[ ],
     "e":["d"],
     "f":["b","e"]})
```


Πίνακας 1

Καταγράψτε το παραπάνω γράφημα ως πίνακα γειτονικότητας στον Πίνακα 1. Γράψτε κώδικα που να υλοποιεί τη συνάρτηση `is_adjacent(graph, v1, v2)` και που να επιστρέφει `True` αν υπάρχει ακμή από την κορυφή `v1` προς την κορυφή `v2`, αλλιώς να επιστρέφει `False`.

Θέμα 2 [A=1, B=1]

A) Δίνεται μια λίστα n τιμών. Γράψτε κώδικα σε Python (ή ψευδοκώδικα) που να υπολογίζει σε γραμμικό χρόνο την τιμή που εμφανίζεται περισσότερες φορές από το μισό του πλήθους των στοιχείων της λίστας. Υποθέστε ότι υπάρχει πάντα μια τέτοια τιμή στα δεδομένα. Για παράδειγμα για τη λίστα τιμών `[2,2,1,2,3,1,2,2]` η τιμή που ζητείται είναι η τιμή 2.

B) Προτείνετε λύση που να επιλύει το πρόβλημα του προηγούμενου ερωτήματος σε γραμμικό χρόνο χωρίς να χρησιμοποιηθεί επιπλέον δομή αποθήκευσης ενδιάμεσων τιμών, πέρα από την αρχική ακολουθία τιμών.

