HY240: Δομές Δεδομένων
Χειμερινό Εξάμηνο – Ακαδημαϊκό Έτος 2023-24
Παναγιώτα Φατούρου

4ο Σετ Ασκήσεων

Ημερομηνία Παράδοσης: 7 Ιανουαρίου 2024
Τρόπος Παράδοσης: Οι ασκήσεις μπορούν να παραδοθούν σε ηλεκτρονική μορφή χρησιμοποιώντας το πρόγραμμα turnin μέχρι τις 7 Ιανουαρίου 2024, ώρα 23:59. Ασκήσεις που παραδίδονται μετά τις 23:59 τις 7 Ιανουαρίου 2024, θεωρούνται εκπρόθεσμες.

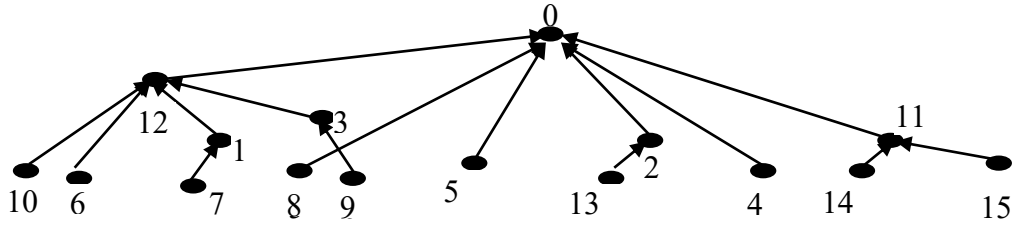
Άσκηση 1 [50 μονάδες]

α. Θεωρήστε ένα κοκκινόμαυρο δένδρο για το οποίο ισχύουν τα εξής: (α) το αριστερό υποδένδρο της ρίζας έχει ελάχιστο πλήθος εσωτερικών κόμβων και ύψος 2, (β) το ύψος του αριστερού υποδένδρου της ρίζας είναι κατά δύο μικρότερο από το ύψος του δεξιού υποδένδρου της ρίζας και (γ) το δεξιό υποδένδρο της ρίζας είναι τέλειο. Τα κλειδιά που αποθηκεύονται στους κόμβους του δένδρου, όταν τοποθετηθούν σε αύξουσα διάταξη, αποτελούν πρόθεμα της ακολουθίας κλειδιών 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, κλπ.

(i) Σχεδιάστε το δένδρο. [5M]

(ii) Εισάγεται τα κλειδιά 19,21,23,25, 27 διαδοχικά, ξεκινώντας από το δένδρο που σχεδιάσατε στο ερώτημα (i). [10M]

β. Ξεκινώντας από τα 16 μονοσύνολα $S_j = \{j\}$, όπου $0 \le j \le 15$, παρουσιάστε μια ακολουθία λειτουργιών Union-Find η οποία αν εφαρμοστεί στα μονοσύνολα να προκύπτει το up-δένδρο που φαίνεται στο Σχήμα 2. Είναι η ακολουθία που παρουσιάσατε μοναδική και γιατί; [15M]



Σχήμα 2

γ. Εισάγετε τα κλειδιά 25, 20, 15, 14, 12, 10, 8, 6, 4, 2, 1 (μ' αυτή τη σειρά), σε ένα αρχικά άδειο AVL δένδρο. Στη συνέχεια, διαγράψτε τα κλειδιά (μ' αυτή τη σειρά) από το δένδρο που προέκυψε μετά τις εισαγωγές. [12M]

δ. Εξετάστε αν ο πίνακας [30, 27, 19, 23, 26, 14, 15, 21, 11, 3, 4, 7] αναπαριστά σωρό (θα πρέπει να εξετάσετε αν ισχύει η ιδιότητα της μερικής διάταξης στο δένδρο που αναπαριστά ο πίνακας). Σε περίπτωση που ο πίνακας δεν αναπαριστά σωρό (min-heap) παρουσιάστε πως θα τον μετατρέπατε σε σωρό εφαρμόζοντας τη διαδικασία InitializeHeap() της Heapsort() που έχετε διδαχθεί στο μάθημα. Θεωρήστε ότι το μικρότερο στοιχείο του σωρού αποθηκεύεται στην αριστερότερη θέση του πίνακα.

[8M]

Σημειώσεις: (α) Στα παραπάνω ερωτήματα **πρέπει να παρουσιαστεί όλη η διαδικασία** που ακολουθείται προκειμένου να προκύψει το τελικό δένδρο. Οι περιστροφές, όπου απαιτούνται, θα πρέπει να περιγραφούν αναλυτικά και οι περιπτώσεις που χρησιμοποιείτε (όπως περιγράφονται στους αλγορίθμους που έχετε διδαχθεί) πρέπει να αναφερθούν. **(β)** Στο ερώτημα δ. θα πρέπει να παρουσιάσετε όλα τα βήματα που θα εφαρμόσετε (και με τη σωστή σειρά, βάσει του αλγορίθμου InitializeHeap()) προκειμένου να αποκαταστήσετε την ιδιότητα της μερικής διάταξης (εάν χρειάζεται).

Άσκηση 2 [15 Μονάδες]

Θεωρήστε έναν πίνακα κατακερματισμού Α, m θέσεων, στον οποίο οι συγκρούσεις επιλύονται με τη μέθοδο του διπλού κατακερματισμού (double hashing). Έστω ότι h_1(Key K) είναι η πρωτεύουσα συνάρτηση κατακερματισμού και h_2(Key K) είναι η δευτερεύουσα συνάρτηση κατακερματισμού. Γράψτε ψευδοκώδικα που θα υλοποιεί τη συνάρτηση HashTable-Delete(HashTable A, Key K), η οποία θα πρέπει να διαγράφει το κλειδί Κ από τον πίνακα κατακερματισμού Α. Επιπρόσθετα, παρουσιάστε ψευδοκώδικα για τις ρουτίνες HashTable-Insert(HashTable A, Key K), η οποία εισάγει το κλειδί Κ στον πίνακα κατακερματισμού Α και HashTable-Search(HashTable A, Key K), η οποία αναζητά το κλειδί Κ στον πίνακα κατακερματισμού Α.

Άσκηση 3 [15 μονάδες]

Παρουσιάστε αλγόριθμο (υπό τη μορφή ψευδοκώδικα) που θα προσθέτει δύο στοιχεία ενός σωρού Η (δηλαδή ζητείται μια υλοποίηση της HeapAddElements(H, j, k) η οποία θα προσθέτει το κλειδί του στοιχείου με δείκτη j του σωρού και εκείνο του στοιχείου με δείκτη k και θα αντικαθιστά το κλειδί στη θέση j με το άθροισμα. Μετά την εκτέλεση του αλγορίθμου σας, θα πρέπει ο σωρός να έχει την ιδιότητα της μερικής διάταξης (δηλαδή ο αλγόριθμός σας θα πρέπει να εκτελεί κατάλληλες ενέργειες μετά την πρόσθεση των στοιχείων ώστε να επαναφέρει την ιδιότητα της μερικής διάταξης,, αν αυτή έχει καταστρατηγηθεί. Ο αλγόριθμός σας θα πρέπει να εκτελείται σε χρόνο O(log n)).

Άσκηση 4 [20 Μονάδες]

α. Παρουσιάστε αλγόριθμο (σε μορφή ψευδοκώδικα) ο οποίος δοθέντος ενός δείκτη στη ρίζα ενός δένδρου AVL και ενός κλειδιού Κ θα επιστρέφει το ύψος του κόμβου με κλειδί Κ στο δένδρο. Ο αλγόριθμός σας θα πρέπει να έχει χρονική πολυπλοκότητα O(logn). Αν δεν υπάρχει κόμβος με κλειδί κ στο δένδρο, ο αλγόριθμος θα πρέπει να επιστρέφει την τιμή 0. [10M]

β. Παρουσιάστε αλγόριθμο (σε μορφή ψευδοκώδικα) ο οποίος θα διαβάζει n αριθμούς από το stdin και θα τους τυπώνει στο stdout ταξινομημένους κατ' αύξουσα διάταξη. Θεωρήστε ότι η μοναδική δομή δεδομένων που μπορείτε να χρησιμοποιήσετε για την αποθήκευση και διαχείριση των αριθμών είναι ένα δένδρο AVL (και άρα δεν επιτρέπεται η χρήση πινάκων, λιστών ή άλλων δομών δεδομένων για την αποθήκευση και την ταξινόμηση των αριθμών). Θεωρήστε ότι δίνονται οι συναρτήσεις AVL_Insert(), AVL_Delete() και AVL_LookUp() που πραγματοποιούν εισαγωγή, διαγραφή και αναζήτηση στο δένδρο AVL. Θεωρήστε επίσης ότι δίνεται μια συνάρτηση getNumber(), η οποία διαβάζει και επιστρέφει τον επόμενο αριθμό από το stdin. Η συνάρτηση getNumber() επιστρέφει NULL όταν δεν υπάρχουν άλλοι αριθμοί στο stdin. **Τι πολυπλοκότητα έχει ο αλγόριθμος που σχεδιάσατε και γιατί;** [10M]