HY240: Δομές Δεδομένων
Χειμερινό Εξάμηνο – Ακαδημαϊκό Έτος 2014-15
Παναγιώτα Φατούρου

2ο Σετ Ασκήσεων

Ημερομηνία Παράδοσης: Δευτέρα, 27 Οκτωβρίου 2014
Τρόπος Παράδοσης: Οι ασκήσεις μπορούν να παραδοθούν είτε σε ηλεκτρονική μορφή χρησιμοποιώντας το πρόγραμμα turnin (δείτε οδηγίες στην ιστοσελίδα του μαθήματος), είτε στους βοηθούς του μαθήματος στο εργαστήριο των μεταπτυχιακών, την Δευτέρα, 27 Οκτωβρίου 2014, ώρα 15:00-16:00. Ασκήσεις που παραδίδονται μετά τις 16:00 της Δευτέρας, 27/10/2014 θεωρούνται εκπρόθεσμες. Εκπρόθεσμες ασκήσεις γίνονται δεκτές σε ηλεκτρονική μορφή και η αποστολή τους πρέπει να γίνει χρησιμοποιώντας το πρόγραμμα turnin. Εναλλακτικά, εκπρόθεσμες ασκήσεις μπορούν να παραδοθούν στη διδάσκουσα του μαθήματος ή να σταλούν στο hy240a@csd.uoc.gr μέσω ηλεκτρονικού ταχυδρομείου.

Άσκηση 1 [30 μονάδες]

α. Έστω ότι μια βιβλιοθήκη σας παρέχει πρόσβαση σε στοίβες χαρακτήρων. Η βιβλιοθήκη σας επιτρέπει να ορίσετε μια στοίβα και να καλέσετε τις 5 βασικές λειτουργίες σε αυτή. Για παράδειγμα, ο ορισμός μιας στοίβας γίνεται γράφοντας: Stack S; Για τη στοίβα υποστηρίζονται οι εξής λειτουργίες: (1) void MakeEmptyStack(stack S) (2) boolean IsEmptyStack(stack S) (3) char Top(Stack S) (4) char Pop(Stack S) (5) void Push(Stack S, char x).

Έλεγχος ορθότητας έκφρασης με παρενθέσεις και αγκύλες. [15%]

Παρουσιάστε αλγόριθμο ο οποίος, χρησιμοποιώντας στοίβες θα ελέγχει αν μια έκφραση που περιέχει παρενθέσεις και αγκύλες είναι σωστή.

Παραδείγματα: Η έκφραση ((([])[()][()])[]()) είναι σωστή, ενώ οι εκφράσεις ([([]()))] και (([())) δεν είναι.

β. Έστω ότι A είναι ένας **ταξινομημένος** πίνακας ακεραίων N θέσεων του οποίου τα στοιχεία έχουν αρχικοποιηθεί με την τιμή `INT_MAX` (που είναι ο μεγαλύτερος ακέραιος που μπορεί να αναπαρασταθεί στην πλειοψηφία των σημερινών μηχανών). Το πρόγραμμα που χρησιμοποιεί τον A αποθηκεύει στα στοιχεία του A ακέραιους αριθμούς μικρότερους της τιμής `INT_MAX`.

Θεωρήστε πως κάποια χρονική στιγμή, ο πίνακας περιέχει n ≤ N έγκυρα στοιχεία, δηλαδή την χρονική στιγμή εκείνη, οι n πρώτες θέσεις του A περιέχουν ακέραιους μικρότερους του `INT_MAX`, ενώ οι υπόλοιπες θέσεις του πίνακα περιέχουν την τιμή `INT_MAX`. Παρουσιάστε αλγόριθμο ο οποίος δοθέντος του πίνακα A θα βρίσκει το n, δηλαδή θα βρίσκει την τελευταία θέση του πίνακα που περιέχει έγκυρο στοιχείο. Η χρονική πολυπλοκότητα του αλγορίθμου σας θα πρέπει να είναι O(log N). [15%]

Άσκηση 2 [40 μονάδες]

α. Παρουσιάστε **αναδρομικό** αλγόριθμο ο οποίος δεδομένης μιας ταξινομημένης **κατ' αύξουσα διάταξη** λίστας L με n στοιχεία θα δημιουργεί μια νέα λίστα η οποία θα περιέχει τα στοιχεία της L **σε φθίνουσα διάταξη.** Θεωρήστε πως η L είναι απλά συνδεδεμένη και πως παρέχεται ένας δείκτης στο πρώτο στοιχείο της λίστας.

Παρουσιάσετε αναδρομική σχέση που να περιγράφει το πλήθος των στοιχειωδών εντολών που εκτελεί ο αλγόριθμος σας. Επιλύστε τη σχέση αυτή και μελετήστε την τάξη της χρονικής πολυπλοκότητας του αλγορίθμου σας. Ο αλγόριθμός σας μπορεί να χρησιμοποιεί ένα σταθερό πλήθος από βοηθητικές μεταβλητές (π.χ. δείκτες). [20%]

β. Δίνεται μια απλά-συνδεδεμένη μη-ταξινομημένη λίστα. Παρουσιάστε ψευδοκώδικα ο οποίος θα ταξινομεί τη λίστα εφαρμόζοντας την τεχνική ταξινόμησης της InsertionSort(). Μελετήστε την πολυπλοκότητα του αλγορίθμου σας. [20%]

Άσκηση 3 [30 Μονάδες]

Παρουσιάστε αλγόριθμους για να υλοποιήσετε τις ακόλουθες λειτουργίες σε **απλά συνδεδεμένες** λίστες:

i. **Συνένωση μη ταξινομημένων λιστών (ListMerge).** Η ListMerge() θα παίρνει ως ορίσματα δύο λίστες και θα πρέπει να επιστρέφει έναν δείκτη σε μια νέα λίστα η οποία θα περιέχει τόσο τα στοιχεία της πρώτης λίστας όσο και τα στοιχεία της δεύτερης. Ο αλγόριθμός σας θα πρέπει να έχει χρονική πολυπλοκότητα O(1). Εξηγήστε τι είδους μεταβλητές θα πρέπει να διατηρείτε για κάθε λίστα (οι οποίες θα αποτελέσουν και τα ορίσματα της ListMerge) ώστε να επιτευχθεί η ζητούμενη πολυπλοκότητα. [10%]

ii. **Συνένωση ταξινομημένων λιστών (SortedListMerge).** Η SortedListMerge() θα πρέπει να παίρνει ως ορίσματα έναν δείκτη που θα δείχνει στο πρώτο στοιχείο της πρώτης λίστας και έναν δείκτη που θα δείχνει στο πρώτο στοιχείο της δεύτερης λίστας και θα πρέπει να επιστρέφει έναν δείκτη σε μια νέα **ταξινομημένη** λίστα η οποία θα περιέχει τόσο τα στοιχεία της πρώτης λίστας όσο και τα στοιχεία της δεύτερης. Ο αλγόριθμός σας θα πρέπει να έχει χρονική πολυπλοκότητα O(n), όπου n είναι το μέγιστο του πλήθους των στοιχείων της πρώτης λίστας και του πλήθους των στοιχείων της δεύτερης λίστας. [20%]