HY240: Δομές Δεδομένων
Εαρινό Εξάμηνο – Ακαδημαϊκό Έτος 2017-18
Παναγιώτα Φατούρου

2ο Σετ Ασκήσεων

Ημερομηνία Παράδοσης: Δευτέρα, 12 Μαρτίου 2018
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, τη Δευτέρα, 12 Μαρτίου 2018, ώρα 13:00-14:00. Ασκήσεις που παραδίδονται μετά τις 14:00 της Δευτέρας, 12/03/2017 θεωρούνται εκπρόθεσμες. Εκπρόθεσμες ασκήσεις γίνονται δεκτές σε ηλεκτρονική μορφή και η αποστολή τους πρέπει να γίνει χρησιμοποιώντας το πρόγραμμα turnin. **Ασκήσεις μέσω ηλεκτρονικού ταχυδρομείου δεν γίνονται δεκτές.**

Άσκηση 1 [40 μονάδες]

α. Έστω ότι μια βιβλιοθήκη σας παρέχει πρόσβαση σε στοίβες και ουρές χαρακτήρων. Η βιβλιοθήκη σας επιτρέπει να ορίσετε μια στοίβα (ή μια ουρά) και να καλέσετε τις 5 βασικές λειτουργίες σε αυτή. Για παράδειγμα, ο ορισμός μιας στοίβας (ή μιας ουράς) γίνεται γράφοντας: Stack S; (αντίστοιχα, Queue Q;). Για τη στοίβα υποστηρίζονται οι εξής λειτουργίες: (1) void MakeEmptyStack(Stack S), (2) boolean IsEmptyStack(Stack S), (3) Type Top(Stack S), (4) Type Pop(Stack S), (5) void Push(Stack S, char x). Αντίστοιχα, για την ουρά υποστηρίζονται οι λειτουργίες: (1) void MakeEmptyQueue(Queue Q), (2) boolean IsEmptyQueue(Queue Q), (3) char Front(Queue Q), (4) char Dequeue(Queue Q), (5) void Enqueue(Queue Q, char x).

Να παρουσιαστεί αλγόριθμος που θα διαβάζει ένα αλφαριθμητικό a από το πληκτρολόγιο και θα ελέγχει αν αυτό είναι της μορφής a = www', όπου w' είναι το αντίστροφο αλφαριθμητικό του w. Για παράδειγμα, ένα αλφαριθμητικό αυτής της μορφής είναι το maymayyam. Ο αλγόριθμός σας επιτρέπεται να χρησιμοποιεί δύο μόνο δομές (π.χ. δύο ουρές ή δύο στοίβες ή μια ουρά και μια στοίβα). Ο αλγόριθμός σας επιτρέπεται επίσης να χρησιμοποιεί βοηθητικές μεταβλητές. [20M]

Υπόδειξη: Ο αλγόριθμός σας θα πρέπει να χρησιμοποιεί δομές δεδομένων από τη βιβλιοθήκη που σας παρέχεται, καθώς και ενδεχομένως κάποιες βοηθητικές μεταβλητές. Δεν επιτρέπεται η χρήση πινάκων ή άλλων δομών δεδομένων που δεν ανήκουν στη βιβλιοθήκη για την αποθήκευση ή την επεξεργασία των αλφαριθμητικών ή των ακολουθιών χαρακτήρων. Δεν γνωρίζετε αν οι στοίβες και οι ουρές που παρέχει η βιβλιοθήκη υλοποιούνται με στατικό ή με δυναμικό τρόπο. Θεωρήστε πως σας παρέχεται μια βοηθητική ρουτίνα char getchar() για την ανάγνωση αριθμών ή χαρακτήρων και ότι κάθε προδιατεταγμένη παράσταση τελειώνει με το χαρακτήρα '\n'. Ο αλγόριθμος διαβάζει την είσοδο του που είναι μια γραμμή χαρακτήρων από το πληκτρολόγιο (δηλαδή το αλφαριθμητικό δεν είναι ήδη αποθηκευμένο σε κάποια δομή όταν ξεκινά η εκτέλεση του αλγορίθμου).

β. Να παρουσιαστεί υλοποίηση τεσσάρων στοιβών σε έναν πίνακα A[1..2n] έτσι ώστε μια στοίβα να υπερχειλίζει μόνο αν υπάρχουν δύο στοίβες για τις οποίες ισχύει ότι ο συνολικός αριθμός στοιχείων που περιέχουν ισούται με n. Όλες οι λειτουργίες θα πρέπει να εκτελούνται σε χρόνο Θ(1). [20M]

Άσκηση 2 [30 Μονάδες]

Θεωρήστε μια απλά-συνδεδεμένη μη-ταξινομημένη λίστα, κάθε στοιχείο της οποίας είναι ένα struct με δύο πεδία, έναν ακέραιο num και έναν δείκτη next στο επόμενο στοιχείο της λίστας.

α. Παρουσιάστε έναν αλγόριθμο (ψευδοκώδικα) ο οποίος θα παίρνει ως όρισμα ένα δείκτη L στην αρχή της λίστας και θα δημιουργεί μια νέα **απλά-συνδεδεμένη** λίστα L' που θα περιέχει όλα τα στοιχεία της L με τον ακόλουθο τρόπο. Τα στοιχεία της L για τα οποία το πεδίο num είναι περιττός αριθμός πρέπει να εμφανίζονται στην L' πριν από όλα τα στοιχεία των οποίων το πεδίο num είναι άρτιος αριθμός. Τα στοιχεία με περιττό αριθμό στο πεδίο num θα πρέπει να διατηρούν τη σειρά εμφάνισης που είχαν στην αρχική λίστα. Ομοίως, για τα στοιχεία με άρτιο αριθμό στο πεδίο num. Για παράδειγμα αν η αρχική λίστα L είναι $14 \rightarrow 13 \rightarrow 15 \rightarrow 20 \rightarrow 17 \rightarrow 24 \rightarrow 26 \rightarrow 11 \rightarrow 30 \rightarrow 53 \rightarrow 44$, η τελική λίστα L' θα πρέπει να είναι $13 \rightarrow 15 \rightarrow 17 \rightarrow 11 \rightarrow 53 \rightarrow 14 \rightarrow 20 \rightarrow 24 \rightarrow 26 \rightarrow 30 \rightarrow 44$.

Ο αλγόριθμος σας θα πρέπει να έχει πολυπλοκότητα O(n), όπου n είναι ο αριθμός των στοιχείων της L. Εξηγήστε γιατί ο αλγόριθμός σας επιτυγχάνει την πολυπλοκότητα αυτή. Ο αλγόριθμός σας θα πρέπει να διατηρεί την L αναλλοίωτη, δηλαδή η L δεν πρέπει να αλλάζει καθώς ο αλγόριθμός σας εκτελείται. [15M]

β. Παρουσιάστε έναν αλγόριθμο (ψευδοκώδικα) ο οποίος θα παίρνει ως όρισμα ένα δείκτη L στην αρχή της λίστας και θα δημιουργεί μια νέα **διπλά-συνδεδεμένη** λίστα L' που θα περιέχει όλα τα στοιχεία της L με τον ακόλουθο τρόπο. Τα στοιχεία της L για τα οποία το πεδίο num είναι περιττός αριθμός πρέπει να εμφανίζονται στην L' πριν από όλα τα στοιχεία των οποίων το πεδίο num είναι άρτιος αριθμός. Τα στοιχεία με περιττό αριθμό στο πεδίο num θα πρέπει να διατηρούν τη σειρά εμφάνισης που είχαν στην αρχική λίστα. *Τα στοιχεία με άρτιο αριθμό στο πεδίο num θα πρέπει να εμφανίζονται στην L' με αντίθετη σειρά από αυτή που είχαν στην αρχική λίστα.* Για παράδειγμα αν η αρχική λίστα L είναι $14 \rightarrow 13 \rightarrow 15 \rightarrow 20 \rightarrow 17 \rightarrow 24 \rightarrow 26 \rightarrow 11 \rightarrow 30 \rightarrow 53 \rightarrow 44$, η τελική λίστα L' θα πρέπει να είναι $13 \rightarrow 15 \rightarrow 17 \leftrightarrow 11 \leftrightarrow 53 \leftrightarrow 44 \leftrightarrow 30 \leftrightarrow 26 \leftrightarrow 24 \leftrightarrow 20 \leftrightarrow 14$.

Ο αλγόριθμος σας θα πρέπει να έχει πολυπλοκότητα O(n), όπου n είναι ο αριθμός των στοιχείων της L. Εξηγήστε γιατί ο αλγόριθμός σας επιτυγχάνει την πολυπλοκότητα αυτή. Ο αλγόριθμός σας θα πρέπει να διατηρεί την L αναλλοίωτη, δηλαδή η L δεν πρέπει να αλλάζει καθώς ο αλγόριθμός σας εκτελείται. [15M]

Άσκηση 3 [30 μονάδες]

Ζητείται να σχεδιάσετε κάποιο είδος **απλά-συνδεδεμένων** λιστών προκειμένου να χρησιμοποιηθεί για την αναπαράσταση πολυ-συνόλων ακεραίων, έτσι ώστε κάθε λίστα να αποθηκεύει τα στοιχεία ενός πολυσυνόλου. Οι λίστες πρέπει να είναι τέτοιου είδους που να είναι δυνατόν να υλοποιηθούν οι ακόλουθες δύο λειτουργίες επιτυγχάνοντας τη χρονική πολυπλοκότητα που ζητείται σε κάθε περίπτωση:

α. Traverse(pointer p): δεδομένου ενός δείκτη p σε **οποιοδήποτε** στοιχείο της λίστας (όχι απαραίτητα στο πρώτο), η Traverse() διασχίζει τη λίστα και τυπώνει κάθε στοιχείο της. Η Traverse() θα πρέπει να έχει *γραμμική (O(n))* χρονική πολυπλοκότητα.

β. Merge(pointer p1, pointer p2): δεδομένων δύο δεικτών σε δύο **οποιαδήποτε** στοιχεία δύο τέτοιων λιστών, η Merge() δημιουργεί μια λίστα που περιέχει την ένωση των δύο συνόλων που αναπαριστούν οι αρχικές λίστες. Οι αρχικές λίστες παύουν να υπάρχουν μετά το πέρας της εκτέλεσης της Merge(). Η Merge() πρέπει να εκτελείται σε *σταθερό (O(1))* χρόνο.

Να περιγράψετε τη μορφή που πρέπει να έχουν οι λίστες προκειμένου να μπορούν να υλοποιηθούν οι παραπάνω λειτουργίες όπως ζητείται. Να παρουσιαστεί ψευδο-κώδικας για τις λειτουργίες αυτές και να εξηγηθεί γιατί επιτυγχάνονται οι χρονικές πολυπλοκότητες που ζητούνται.