HY240: Δομές Δεδομένων
Χειμερινό Εξάμηνο – Ακαδημαϊκό Έτος 2020-21
Παναγιώτα Φατούρου

2ο Σετ Ασκήσεων

Ημερομηνία Παράδοσης: Δευτέρα, 26 Οκτωβρίου 2020
Τρόπος Παράδοσης: Οι ασκήσεις μπορούν να παραδοθούν σε ηλεκτρονική μορφή χρησιμοποιώντας το πρόγραμμα turnin (δείτε οδηγίες στην ιστοσελίδα του μαθήματος), μέχρι τη Δευτέρα, 26 Οκτωβρίου 2020, ώρα 10:00. Ασκήσεις που παραδίδονται μετά τις 10:00 της Δευτέρας, 26/10/2020, θεωρούνται εκπρόθεσμες. Εκπρόθεσμες ασκήσεις γίνονται δεκτές σε ηλεκτρονική μορφή και η αποστολή τους πρέπει να γίνει χρησιμοποιώντας το πρόγραμμα turnin.

Άσκηση 1 [40 μονάδες]

α. Θεωρήστε μια απλά συνδεδεμένη λίστα, κάθε κόμβος της οποίας είναι ένα struct με πεδία έναν ακέραιο num και ένα δείκτη next στο επόμενο στοιχείο της λίστας. Έστω ότι L είναι ένας δείκτης στο πρώτο στοιχείο της λίστας. Παρουσιάστε **μη-αναδρομική** συνάρτηση, RemoveKey(pointer L, Key k), η οποία θα διατρέχει τη λίστα και θα διαγράφει όλους εκείνους τους κόμβους που έχουν κλειδί ίσο με k. Θεωρήστε πως δεν υπάρχει κάποιος περιορισμός στο πόσα στοιχεία της λίστας μπορεί να έχουν το ίδιο κλειδί (και επομένως, θα μπορούσαν πολλά συνεχόμενα ή και όλα τα στοιχεία της λίστας να έχουν κλειδί ίσο με k). Ο αλγόριθμός σας θα πρέπει να διατρέχει το πολύ μια φορά τη λίστα και άρα θα έχει χρονική πολυπλοκότητα O(n), όπου n είναι το πλήθος των στοιχείων της λίστας. [15M]

β. Θεωρήστε μια απλά συνδεδεμένη λίστα, κάθε κόμβος της οποίας είναι ένα struct με πεδία έναν ακέραιο num και ένα δείκτη next στο επόμενο στοιχείο της λίστας. Έστω ότι L είναι ένας δείκτης στο πρώτο στοιχείο της λίστας. Παρουσιάστε **μη-αναδρομική** συνάρτηση, η οποία θα διατρέχει τη λίστα μέχρι να βρει το τελευταίο στοιχείο v της L, θα διαβάζει το πεδίο num του v, θα βρίσκει το στοιχείο που βρίσκεται num θέσεις νωρίτερα στη λίστα και θα το διαγράφει (δηλαδή η συνάρτηση θα πρέπει να βρίσκει και να διαγράφει εκείνο τον κόμβο u της λίστας για τον οποίο ισχύει ότι μεταξύ του u και του v μεσολαβούν num-1 κόμβοι). Η συνάρτησή σας θα πρέπει να έχει χρονική πολυπλοκότητα O(n), όπου n είναι το πλήθος των στοιχείων της λίστας. Αν η λίστα περιέχει λιγότερα στοιχεία από num, η συνάρτησή σας θα πρέπει **να βρίσκει και να διαγράφει** τον κόμβο της λίστας που βρίσκεται num%n στοιχεία πριν τον κόμβο v. Ο αλγόριθμός σας δεν γνωρίζει το n και άρα θα πρέπει να το υπολογίζει. [15M]

γ. Παρουσιάστε τις συναρτήσεις Insert() και Delete(L,k) για μια απλά συνδεδεμένη λίστα L με κόμβο φρουρό. Η Insert(L,k) εισάγει έναν νέο κόμβο με κλειδί k στη λίστα (αν δεν υπάρχει ήδη), ενώ η Delete(L,k) αναζητά κόμβο με κλειδί k στη λίστα και αν τον βρει, τον διαγράφει. Θεωρήστε πως τα κλειδιά των στοιχείων της λίστας είναι μοναδικά. [10M]

Άσκηση 2 [35 μονάδες]

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).

Διπλό αλφαριθμητικό είναι εκείνο που είναι της μορφής ss όπου s είναι ένα οποιοδήποτε αλφαριθμητικό. Για παράδειγμα το αλφαριθμητικό ΚΑΠΟΥΚΑΠΟΥ είναι διπλό (συγκεκριμένα, στο παράδειγμα αυτό ισχύει ότι s = ΚΑΠΟΥ).

α. Παρουσιάστε αλγόριθμο (υπό τη μορφή ψευδοκώδικα) που θα ελέγχει αν ένα αλφαριθμητικό είναι διπλό χρησιμοποιώντας ΜΙΑ ΜΟΝΟ ουρά. Ο αλγόριθμός σας επιτρέπεται να χρησιμοποιεί μετρητές ή άλλες βοηθητικές μεταβλητές (αν χρειάζονται). Θεωρήστε πως ο αλγόριθμος, κατά την έναρξη του, γνωρίζει το μήκος του αλφαριθμητικού (π.χ. αυτό αποτελεί μια από τις παραμέτρους του αλγορίθμου). [15Μ]

β. Παρουσιάστε αλγόριθμο (υπό τη μορφή ψευδοκώδικα) που θα ελέγχει αν ένα αλφαριθμητικό είναι **διπλό** χρησιμοποιώντας **ΔΥΟ στοίβες** (και αναδρομή). Ο αλγόριθμός σας επιτρέπεται να χρησιμοποιεί βοηθητικές μεταβλητές (αν χρειάζεται). Θεωρήστε πως ο αλγόριθμος, κατά την έναρξή του, δεν γνωρίζει το μήκος του αλφαριθμητικού. [20Μ]

Σημειώσεις: Ι. Οι αλγόριθμοι και για τα δύο ερωτήματα θα πρέπει να χρησιμοποιούν **μόνο τις δομές δεδομένων από τη βιβλιοθήκη που καθορίζονται από την εκφώνηση**, καθώς και ενδεχόμενα 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. ΙΙ. 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'. **ΙΙΙ. Μπορείτε να χρησιμοποιήσετε βοηθητικές ρουτίνες για να διαβάζετε το αλφαριθμητικό από το stdin και να το αποθηκεύεται στη δομή και άλλες (αναδρομικές ή μη, ανάλογα με το τι ζητείται στην εκφώνηση) για να το επεξεργάζεστε.**

Άσκηση 3 [25 Μονάδες]

Να περιγράψετε **αναδρομικό** αλγόριθμο, ο οποίος θα λειτουργεί ως εξής. Θα παίρνει ως είσοδο δύο απλά-συνδεδεμένες λίστες (δηλαδή δύο δείκτες στο πρώτο στοιχείο της κάθε μιας) οι οποίες είναι ταξινομημένες σε φθίνουσα διάταξη. Ο αλγόριθμος θα πρέπει να συνενώνει τα στοιχεία των δύο λιστών δημιουργώντας μια λίστα που είναι ταξινομημένη σε αύξουσα διάταξη και περιέχει τα στοιχεία και των δύο λιστών. Θεωρήστε πως οι δύο λίστες έχουν τον ίδιο αριθμό στοιχείων. Η χρονική πολυπλοκότητα του αλγορίθμου σας πρέπει να είναι O(n), όπου n είναι το πλήθος των στοιχείων της κάθε λίστας **(το οποίο δεν είναι γνωστό στον αλγόριθμο)**. Ο αλγόριθμος σας δεν επιτρέπεται να χρησιμοποιεί επιπρόσθετες βοηθητικές δομές δεδομένων για να κάνει τη συνένωση. Μελετήστε την πολυπλοκότητα του αλγορίθμου σας (με χρήση της κατάλληλης αναδρομικής σχέσης).