HY240: Δομές Δεδομένων
Εαρινό Εξάμηνο – Ακαδημαϊκό Έτος 2016-17
Παναγιώτα Φατούρου

2ο Σετ Ασκήσεων

Ημερομηνία Παράδοσης: Δευτέρα, 13 Μαρτίου 2017
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, 13 Μαρτίου 2017, ώρα 13:00-14:00. Ασκήσεις που παραδίδονται μετά τις 14:00 της Δευτέρας, 13/03/2017 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 [30 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).

Παλίνδρομο αλφαριθμητικό είναι εκείνο που διαβάζεται και ανάποδα, όπως π.χ. η λέξη ΣΕΡΡΕΣ ή η επιγραφή ΝΙΨΟΝ ΑΝΟΜΗΜΑΤΑ ΜΗ ΜΟΝΑΝ ΟΨΙΝ ή το αλφαριθμητικό Never odd or even.

α. Παρουσιάστε αλγόριθμο (υπό τη μορφή ψευδοκώδικα) που θα ελέγχει αν ένα αλφαριθμητικό είναι παλίνδρομο χρησιμοποιώντας ΜΙΑ ΜΟΝΟ ουρά. Ο αλγόριθμός σας επιτρέπεται να χρησιμοποιεί μετρητές ή άλλες βοηθητικές μεταβλητές (αν χρειάζονται). Θεωρήστε πως ο αλγόριθμος, κατά την έναρξή του, δεν γνωρίζει το μήκος του αλφαριθμητικού. [15Μ]

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

Σημειώσεις: Ι. Οι αλγόριθμοι και για τα δύο ερωτήματα θα πρέπει να χρησιμοποιούν δομές δεδομένων από τη βιβλιοθήκη που σας παρέχεται, 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'.

Άσκηση 2 [30 Μονάδες]

α. Θεωρήστε τη λειτουργία Intersection(S_1, S_2) (τομή), όπου S_1 και S_2 είναι δύο σύνολα. Παρουσιάστε αλγόριθμο που να υλοποιεί την Intersection(), δεδομένου ότι τα σύνολα υλοποιούνται με **ταξινομημένες απλά-συνδεδεμένες λίστες.** Η Intersection() πρέπει να παίρνει ως όρισμα ένα δείκτη στο πρώτο στοιχείο της λίστας που υλοποιεί το σύνολο S_1 και ένα δείκτη στο πρώτο στοιχείο της λίστας που υλοποιεί το S_2 και θα πρέπει να επιστρέφει ένα δείκτη στο πρώτο στοιχείο μιας τρίτης λίστας η οποία, για κάθε κλειδί του S1 που υπάρχει και στο S2, θα περιέχει ένα νέο στοιχείο με το κλειδί αυτό. Οι λίστες που αναπαριστούν τα σύνολα S_1 και S_2 θα πρέπει να παραμένουν αναλλοίωτες κατά την εκτέλεση του αλγορίθμου.

Ο αλγόριθμος σας θα πρέπει να έχει πολυπλοκότητα $O(n_1+ n_2)$, όπου n_1 και n_2 είναι οι πληθικοί αριθμοί των συνόλων S_1 και S_2, αντίστοιχα. Εξηγήστε γιατί ο αλγόριθμός σας επιτυγχάνει την πολυπλοκότητα αυτή. [15M]

β. Παρουσιάστε ψευδοκώδικα για τη λειτουργία Intersection(S_1, S_2), δεδομένου ότι τα σύνολα υλοποιούνται με μη-**ταξινομημένες απλά-συνδεδεμένες λίστες.** Τι πολυπλοκότητα έχει ο αλγόριθμός σας και γιατί; [15M]

Άσκηση 3 [40 μονάδες]

α. Έστω ότι μια διπλά-συνδεδεμένη λίστα περιέχει στοιχεία του ακόλουθου τύπου: [20M]

```
struct node {
    int key;
    int distance;
    struct node *prev;
    struct node *next;
}
```

Η λίστα είναι ταξινομημένη σε αύξουσα διάταξη **βάσει του πεδίου distance** κάθε στοιχείου της. Θεωρήστε ότι η λίστα περιέχει πληροφορίες για τις αποστάσεις κάποιων πόλεων μια γεωγραφικής περιοχής, π.χ. της Πελοποννήσου, από την Αθήνα. Έτσι, το πεδίο key αποθηκεύει ένα μοναδικό αναγνωριστικό για κάθε πόλη, ενώ η τιμή του πεδίου distance είναι η απόσταση της τρέχουσας πόλης **από την προηγούμενή της** στη λίστα. Για το πρώτο στοιχείο της λίστας, το πεδίο distance αποθηκεύει την απόσταση της πόλης από την Αθήνα. Θεωρήστε ότι λόγω κάποιων έργων που πραγματοποιούνται στο οδικό δίκτυο της Πελοποννήσου, ο μόνος τρόπος να επισκεφτεί κάποιος τις πόλεις (στην παρούσα φάση) είναι με τη σειρά που ορίζεται από τη διάταξή τους στη λίστα.

Ζητείται αλγόριθμος, ο οποίος θα παίρνει ως παράμετρο ένα δείκτη Η στον header κόμβο της λίστας και έναν ακέραιο Κ. Θα διασχίζει τη λίστα μέχρι να βρει το στοιχείο με κλειδί Κ. Στη συνέχεια θα αναζητεί και θα τυπώνει τα αναγνωριστικά όλων των πόλεων που βρίσκονται σε απόσταση το πολύ dist/2 από την πόλη με κλειδί Κ, όπου dist είναι η απόσταση της πόλης με κλειδί Κ από την Αθήνα. Ο αλγόριθμός σας, θα πρέπει να τυπώνει, εναλλάξ, το αναγνωριστικό μιας πόλης που βρίσκεται σε μεγαλύτερη απόσταση από εκείνη με κλειδί Κ από την Αθήνα και το αναγνωριστικό μιας πόλης που βρίσκεται σε μικρότερη.

β. Έστω ότι μια λίστα περιέχει στοιχεία του ακόλουθου τύπου: [20M]

```
struct node {
    int val;
    boolean flag;
    struct node *next;
```

}

Η λίστα είναι ταξινομημένη σε αύξουσα διάταξη **βάσει του πεδίου val** κάθε στοιχείου της. Ζητείται αλγόριθμος που, δοθείσης της λίστας L, θα επανατοποθετεί τα στοιχεία της L με τέτοιο τρόπο ώστε η λίστα που θα προκύπτει να αποτελείται από δύο μέρη. Το πρώτο μέρος θα πρέπει να περιέχει, ταξινομημένα **σε αύξουσα διάταξη** (ως προς το πεδίο val), όλα τα στοιχεία της L για τα οποία ισχύει ότι το πεδίο flag έχει την τιμή 0. Το δεύτερο μέρος θα πρέπει να περιέχει, ταξινομημένα **σε φθίνουσα διάταξη** (ως προς το πεδίο val), όλα τα στοιχεία της L για τα οποία ισχύει ότι το πεδίο flag έχει την τιμή 1.

Ο αλγόριθμος που θα σχεδιάσετε δεν θα πρέπει να καλεί τη malloc() για δέσμευση νέων κόμβων αλλά μπορεί να χρησιμοποιεί βοηθητικούς δείκτες. **Ο αλγόριθμός σας θα πρέπει να έχει πολυπλοκότητα O(n), όπου n είναι το πλήθος των στοιχείων της L** (πιο συγκεκριμένα, μια μόνο διάσχιση της L θα πρέπει να είναι αρκετή ώστε ο αλγόριθμος να επανατοποθετεί τα στοιχεία με τον τρόπο που ζητείται).

Παράδειγμα: Θεωρήστε τη λίστα:

<5,1> → <15,1> → <25,0> → <30,1> → <40,0> → <45,0> → <50,1> → <55,0> → <60,1>

Μετά την εκτέλεση του αλγορίθμου σας, αυτή θα πρέπει να έχει τη μορφή που φαίνεται παρακάτω:

<25,0> → <40,0> → <45,0> → <55,0> → <60,1> → <50,1> → <30,1> → <15,1> → <5,1>

1ο μέρος λίστας που θα προκύψει — 2ο μέρος λίστας που θα προκύψει