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

2ο Σετ Ασκήσεων

Ημερομηνία Παράδοσης: Δευτέρα, 16 Μαρτίου 2015
Τρόπος Παράδοσης: Οι ασκήσεις μπορούν να παραδοθούν είτε σε ηλεκτρονική μορφή χρησιμοποιώντας το πρόγραμμα submit (δείτε οδηγίες στην ιστοσελίδα του μαθήματος), είτε στους βοηθούς του μαθήματος στο εργαστήριο των μεταπτυχιακών, την Δευτέρα, 16 Μαρτίου 2015, ώρα 15:00-16:00. Ασκήσεις που παραδίδονται μετά τις 16:00 της Δευτέρας, 16/3/2015 θεωρούνται εκπρόθεσμες. Εκπρόθεσμες ασκήσεις γίνονται δεκτές σε ηλεκτρονική μορφή και η αποστολή τους πρέπει να γίνει χρησιμοποιώντας το πρόγραμμα submit. Εναλλακτικά, εκπρόθεσμες ασκήσεις μπορούν να παραδοθούν στη διδάσκουσα ή να σταλούν στο hy240b@csd.uoc.gr μέσω ηλεκτρονικού ταχυδρομείου.

Άσκηση 1 [40 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).

Επιλύστε τα προβλήματα που παρουσιάζονται στη συνέχεια:

α. Στην προδιατεταγμένη παράσταση, κάθε πράξη εφαρμόζεται στους δύο τελεστέους που την ακολουθούν. Για παράδειγμα, '+ 1 2' αναπαριστά το άθροισμα του αριθμού 1 με τον αριθμό 2. Ο ορισμός είναι αναδρομικός και επομένως κάθε πράξη μπορεί να αναφέρεται σε τελεστέους που είναι το αποτέλεσμα άλλων πράξεων. Π.χ., '- * + 1 2 – 3 4 5' είναι η προδιατεταγμένη αναπαράσταση της '(1+2)*(3-4)-5'. Παρουσιάστε αλγόριθμο, ο οποίος θα διαβάζει μια έκφραση σε προδιατεταγμένη μορφή και θα επιστρέφει το αποτέλεσμα του υπολογισμού της. [20%]

β. Ζητείται να υλοποιηθούν οι λειτουργίες που υποστηρίζει μια ουρά χρησιμοποιώντας δύο στοίβες. Συγκεκριμένα, ζητούνται αλγόριθμοι που να υλοποιούν τις λειτουργίες της ουράς, χρησιμοποιώντας δύο στοίβες. Μελετήστε την πολυπλοκότητα κάθε λειτουργίας που παρουσιάσατε. [20%]

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.

ΙΙ. Στο ερώτημα (α), θεωρήστε πως σας παρέχεται μια βοηθητική ρουτίνα ch = getchar() για την ανάγνωση αριθμών ή χαρακτήρων. Επίσης, σας παρέχονται οι βοηθητικές συναρτήσεις IsOperand(char ch) και IsOperator(char ch), οι οποίες επιστρέφουν TRUE αν το ch είναι τελεστέος και τελεστής, αντίστοιχα. Θεωρήστε επίσης, πως όλοι οι αριθμοί κάθε προδιατεταγμένης παράστασης είναι μονοψήφιοι.

Άσκηση 2 [20 μονάδες]

Έστω πως μια λίστα περιέχει στοιχεία του ακόλουθου τύπου:

```
struct node {
    int pid;
    int val;
    struct node *next;
}
```

Το σύνολο τιμών του πεδίου pid κάθε στοιχείου της λίστας είναι το {0, 1, 2}. Η λίστα είναι ταξινομημένη σε αύξουσα διάταξη, βάσει του πεδίου val κάθε στοιχείου της. Ζητείται αλγόριθμος που δοθείσης της λίστας L, θα κατασκευάζει τρεις άλλες λίστες L0, L1, L2, με στοιχεία ίδιου τύπου όπως η L, κάθε μια εκ των οποίων θα είναι επίσης ταξινομημένη σε αύξουσα διάταξη, βάσει του πεδίου val κάθε στοιχείου της. Για κάθε $j \in \{0, 1, 2\}$, η λίστα Lj θα πρέπει να περιέχει όλα τα στοιχεία της L των οποίων το πεδίο pid ισούται με j. Ο αλγόριθμος που θα σχεδιάσετε δεν θα πρέπει να προκαλεί κάποια αλλοίωση στην L. **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 δημιουργεί και τις τρεις νέες λίστες).

Άσκηση 3 [40 Μονάδες]

Ένα πολυωνύμο της μορφής $a_n x^n + \ldots + a_1 x + a_0$ μπορεί να αποθηκευτεί σε μια απλά συνδεδεμένη λίστα με στοιχεία του ακόλουθου τύπου:

```
struct node {
        int coefficient;                    //συντελεστής όρου
        int degree;                         // βαθμός όρου
        struct node *next;
}
```

Κάθε στοιχείο μιας τέτοιας λίστας αποθηκεύει πληροφορία για έναν όρο του πολυωνύμου. Κάθε λίστα που αποθηκεύει ένα πολυώνυμο είναι ταξινομημένη, σε φθίνουσα διάταξη, ως προς το πεδίο degree κάθε στοιχείου της.

Ζητείται να παρουσιαστούν αλγόριθμοι για τις εξής δύο λειτουργίες επί των πολυωνύμων:

α. AddPol που προσθέτει δύο πολυώνυμα και αποθηκεύει το αποτέλεσμα σε μια νέα λίστα (η οποία είναι επίσης ταξινομημένη ως προς το πεδίο degree κάθε στοιχείου της). Ο αλγόριθμος που θα σχεδιάσετε θα πρέπει να έχει χρονική πολυπλοκότητα $O(n_1+n_2)$ όπου n_1, n_2 είναι οι βαθμοί των πολυωνύμων που θα προστεθούν. [20%]

β. MultPol που υπολογίζει το γινόμενο δύο πολυωνύμων και αποθηκεύει το αποτέλεσμα σε μια νέα λίστα (που είναι επίσης ταξινομημένη ως προς το πεδίο degree κάθε στοιχείου της). Τι χρονική πολυπλοκότητα έχει ο αλγόριθμος που σχεδιάσατε και γιατί; [20%]