HY240: Δομές Δεδομένων
Χειμερινό Εξάμηνο – Ακαδημαϊκό Έτος 2022-23
Παναγιώτα Φατούρου

2ο Σετ Ασκήσεων

Ημερομηνία Παράδοσης: Δευτέρα, 31 Οκτωβρίου 2022

Τρόπος Παράδοσης: Οι ασκήσεις μπορούν να παραδοθούν σε ηλεκτρονική μορφή χρησιμοποιώντας το πρόγραμμα turnin (δείτε οδηγίες στην ιστοσελίδα του μαθήματος), μέχρι τη Δευτέρα, 31 Οκτωβρίου 2022, ώρα 11:59 πμ ή στη βοηθό του μαθήματος που θα κάνει ώρες γραφείου τη μέρα της παράδοσης για να παραλάβει τις ασκήσεις. Ασκήσεις που παραδίδονται μετά τις 11:59 πμ της Δευτέρας, 31/10/2022, θεωρούνται εκπρόθεσμες. Εκπρόθεσμες ασκήσεις γίνονται δεκτές σε ηλεκτρονική μορφή και η αποστολή τους πρέπει να γίνει χρησιμοποιώντας το πρόγραμμα turnin

.

Άσκηση 1 [35 μονάδες]

α. Μια αεροπορική εταιρεία (π.χ. στην Aegean) αποθηκεύει πληροφορίες για τις πτήσεις της. Το πρόγραμμα διαχείρισης δεδομένων της αεροπορικής εταιρείας χρησιμοποιεί μια δομή δεδομένων, η οποία αποθηκεύει μια εγγραφή για κάθε πτήση. Κάθε εγγραφή έχει ένα μοναδικό αναγνωριστικό. Ας υποθέσουμε ότι η δομή δεδομένων που χρησιμοποιεί η εταιρεία είναι μια **δυναμική λίστα** της οποίας τα στοιχεία έχουν την εξής μορφή:

```
struct flight {
        int id;
        int year;
        struct flight *next;
        struct  flight *ynext;
}
```

Θεωρήστε ότι κάθε πτήση που πραγματοποιήθηκε πριν από κάποιο έτος Χ έχει αναγνωριστικό μικρότερο από κάθε πτήση που πραγματοποιήθηκε αργότερα και ότι **η λίστα είναι ταξινομημένη ως προς τα αναγνωριστικά των πτήσεων**. Επομένως, συναντάμε πρώτα στη λίστα εγγραφές για τις πτήσεις στον πρώτο χρόνο λειτουργίας της αεροπορικής εταιρείας, στη συνέχεια εκείνες των πτήσεων που πραγματοποιήθηκαν μέσα στο δεύτερο χρόνο, κ.ο.κ. Συμπεραίνουμε ότι οι εγγραφές για τις πτήσεις κάθε έτους σχηματίζουν μια υπολίστα μέσα στη λίστα, την οποία ονομάζουμε *αλυσίδα*. Ο δείκτης ynext του πρώτου στοιχείου μιας αλυσίδας δείχνει στο πρώτο στοιχείο της επόμενης αλυσίδας στη λίστα.

i. Ζητείται ο πιο αποδοτικός αλγόριθμος που μπορείτε να σχεδιάσετε για τη συνάρτηση Search(L, id), η οποία αναζητά στη λίστα την εγγραφή που αντιστοιχεί στην πτήση με αναγνωριστικό id. Ο αλγόριθμός σας θα πρέπει να έχει χρονική πολυπλοκότητα O(n+m), όπου n είναι ο συνολικός αριθμός αλυσίδων στην L και m είναι το πλήθος των εγγραφών στη αλυσίδα που αντιστοιχεί στο έτος που πραγματοποιήθηκε η πτήση με αναγνωριστικό id. [10M]

ii. Μετά από 35 χρόνια λειτουργίας της αεροπορικής εταιρείας, το Διοικητικό της Συμβούλιο αποφάσισε πως θα αλλάξει ο τρόπος διαχείρισης των πτήσεων έτσι ώστε να υπάρχουν δύο δομές δεδομένων που θα αποθηκεύουν πληροφορίες πτήσεων, μια που θα περιέχει τις εγγραφές για παλαιότερες πτήσεις, δηλαδή πτήσεις που πραγματοποιήθηκαν πριν από το έτος 2015, και μια η οποία θα αποθηκεύει πληροφορίες για πιο πρόσφατες πτήσεις. [5M]

Μετά την απόφαση του Διοικητικού Συμβουλίου, θα πρέπει, ως μέρος του προγράμματος διαχείρισης δεδομένων της εταιρείας, να υλοποιηθεί μια λειτουργία που θα επεξεργάζεται την αρχική δομή δεδομένων και θα δημιουργεί από αυτήν δύο αντίστοιχες δομές, τέτοιες ώστε: στην πρώτη θα περιέχονται εγγραφές για πτήσεις που πραγματοποιήθηκαν πριν από το 2015, ενώ στη δεύτερη θα περιέχονται οι εγγραφές που αντιστοιχούν στις πτήσεις που έχουν πραγματοποιηθεί μετά από αυτό το έτος. Περιγράψτε αλγόριθμο υλοποίησης της λειτουργίας αυτής. Εξηγήστε ποια είναι η χρονική πολυπλοκότητα του αλγορίθμου σας.

β. Θεωρήστε δύο σύνολα S_1 και S_2, τα οποία υλοποιούνται με δύο δυναμικές ταξινομημένες λίστες (L_1 και L_2, αντίστοιχα). Υποθέστε ότι **η πρώτη λίστα είναι ταξινομημένη σε αύξουσα διάταξη, ενώ η δεύτερη σε φθίνουσα διάταξη**. Ζητείται ψευδοκώδικας που θα υλοποιεί τη λειτουργία SetMinus() επί των συνόλων S_1 και S_2. Η SetMinus() δέχεται ως παραμέτρους δύο δείκτες, έναν στο πρώτο στοιχείο της L1 και έναν στο πρώτο στοιχείο της L2 και επιστρέφει έναν δείκτη στο πρώτο στοιχείο μιας νέας λίστας η οποία περιέχει εκείνα τα στοιχεία του S_1 που δεν ανήκουν στο S_2. Επομένως, η νέα λίστα περιέχει τα στοιχεία του συνόλου S_1-S_2.

i. Περιγράψτε ψευδοκώδικα για τον καλύτερο μη-αναδρομικό αλγόριθμο που μπορείτε να σκεφτείτε για να λύσετε το πρόβλημα, υπό την υπόθεση ότι η πρώτη λίστα είναι απλά συνδεδεμένη και η δεύτερη διπλά συνδεδεμένη. [10M]

ii. Περιγράψτε ψευδοκώδικα για αναδρομικό αλγόριθμο που θα λύνει το πρόβλημα, υπό την υπόθεση ότι και οι δύο λίστες είναι απλά συνδεδεμένες. [10M]

Η χρονική πολυπλοκότητα των αλγορίθμων σας πρέπει να είναι O(n1+n2), όπου n1 είναι το πλήθος των στοιχείων της λίστας που είναι ταξινομημένη σε αύξουσα διάταξη και n2 το πλήθος των στοιχείων της άλλης. Οι αλγόριθμοί σας δεν επιτρέπεται να χρησιμοποιούν επιπρόσθετες βοηθητικές δομές δεδομένων.

Παράδειγμα: Έστω πως S1 = {1, 3, 5, 7, 9, 11, 15, 20, 25, 36, 40} και S2 = {38, 36, 20, 15, 12, 11, 9, 3}. Επομένως, L1 = 1 → 3 → 5 → 7 → 9 → 11 → 15 → 20 → 25 → 36 → 40 και
L2 = 38 → 36 → 20 → 15 → 12 → 11 → 9 → 3. Το αποτέλεσμα της εκτέλεσης της SetMinus() πρέπει να είναι η δημιουργία της λίστας L3 = 1 → 5 → 7 → 25 → 40. Η SetMinus() επιστρέφει έναν δείκτη στο πρώτο στοιχείο αυτής της λίστας, δηλαδή έναν δείκτη στο στοιχείο 1.

Άσκηση 2 [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 σ' αυτήν. 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).

α. Δεδομένης μια στοίβας, παρουσιάστε **μη-αναδρομική συνάρτηση**, RemoveKey(Stack S, Key k), η οποία θα διατρέχει τη στοίβα και θα διαγράφει όλους εκείνους τους κόμβους που έχουν κλειδί ίσο με k. Θεωρήστε πως δεν υπάρχει κάποιος περιορισμός στο πόσα στοιχεία της στοίβας μπορεί να έχουν το ίδιο κλειδί (και επομένως, θα μπορούσαν πολλά συνεχόμενα ή και όλα τα στοιχεία της στοίβας να έχουν κλειδί ίσο με k). Ο αλγόριθμος θα πρέπει να έχει χρονική πολυπλοκότητα O(n), όπου n είναι το πλήθος των στοιχείων της στοίβας. Η RemoveKey μπορεί να χρησιμοποιεί μια επιπρόσθετη βοηθητική δομή (π.χ. μια ουρά ή μια στοίβα) και βοηθητικές μεταβλητές (π.χ. ακεραίους). Η συνάρτηση θα πρέπει να επιστρέφει μια στοίβα, η οποία θα περιέχει όλα τα στοιχεία της αρχικής στοίβας που δεν

διεγράφησαν, με την ίδια σειρά όπως ήταν αρχικά. Η συνάρτηση θα πρέπει να χρησιμοποιεί τη βιβλιοθήκη στοιβών/ουρών που περιγράφεται στην εκφώνηση και δεν γνωρίζει αν η αρχική στοίβα υλοποιείται με στατικό ή δυναμικό τρόπο. [10M]

β. Παρουσιάστε **αναδρομική έκδοση** του αλγορίθμου που περιγράφτηκε στο ερώτημα α. Αποφύγετε τη χρήση βοηθητικής δομής δεδομένων αν δεν είναι απαραίτητη. [5M]

γ. Θεωρήστε μια ουρά Q, κάθε κόμβος της οποίας αποθηκεύει έναν ακέραιο number. Παρουσιάστε μη-αναδρομική συνάρτηση ZigZag(Queue Q, int k), η οποία θα αναζητά το min{n,k}-οστό παλαιότερο στοιχείο που έχει εισαχθεί στην Q, όπου n είναι το συνολικό πλήθος των στοιχείων της Q. Ας υποθέσουμε πως το στοιχείο αυτό αποθηκεύει την τιμή num. Στη συνέχεια, ο αλγόριθμος θα πρέπει να βρίσκει το στοιχείο που βρίσκεται min{num,k} θέσεις αργότερα στην Q και θα το διαγράφει (δηλαδή η συνάρτηση θα πρέπει να βρίσκει και να διαγράφει εκείνο το στοιχείο, y, της Q για το οποίο ισχύει ότι μεταξύ του x και του y μεσολαβούν num-1 στοιχεία στην Q και το y έχει εισαχθεί στην Q μετά από το x). Η συνάρτησή σας θα πρέπει να έχει χρονική πολυπλοκότητα O(n), όπου n είναι το πλήθος των στοιχείων της Q. Η ZigZag δεν έχει γνώση του n και άρα αν ο αλγόριθμός σας χρειάζεται το n, θα πρέπει να το υπολογίζει. Μετά τη διαγραφή του y, η ουρά θα πρέπει να περιέχει τα στοιχεία που απομένουν στην ίδια σειρά όπως αυτά εμφανίζονταν στην Q. Η ZigZag μπορεί να χρησιμοποιεί επιπρόσθετες βοηθητικές ουρές ή στοίβες και βοηθητικές μεταβλητές (π.χ. ακεραίους), αλλά ζητείται η λύση να χρησιμοποιεί το ελάχιστον δυνατό αριθμό τέτοιων δομών. [10M]

δ. Παρουσιάστε αναδρομική έκδοση του αλγορίθμου που περιγράφτηκε στο ερώτημα γ. [5M]

Σημειώσεις: Ι. Οι αλγόριθμοι 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.

Άσκηση 3 [35 Μονάδες]

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, RemovePairs(pointer L, int K), η οποία θα βρίσκει ζευγάρια κόμβων των οποίων το άθροισμα των num πεδίων τους είναι ίσο με έναν αριθμό K και θα τα διαγράφει από τη λίστα.

α. Παρουσιάστε ψευδοκώδικα για την RemovePairs για τον πιο γρήγορο αλγόριθμο που μπορείτε να σκεφτείτε, θεωρώντας ότι **η λίστα δεν είναι ταξινομημένη**. Ποια είναι η χρονική πολυπλοκότητα του αλγορίθμου σας και γιατί; [20M]

β. Παρουσιάστε ψευδοκώδικα για την RemovePairs για τον πιο γρήγορο αλγόριθμο που μπορείτε να σκεφτείτε, θεωρώντας ότι **η λίστα είναι ταξινομημένη**. Ποια είναι η χρονική πολυπλοκότητα του αλγορίθμου σας και γιατί; [15M]