Πανεπιστήμιο Θεσσαλίας
Τμήμα Πληροφορικής με Εφαρμογές στη Βιοϊατρική

Δομές Δεδομένων
Ακαδημαϊκό Έτος 2018-2019
http://emarkou.users.uth/greek/teach/data_structures
Ε. Μάρκου

1ο Σετ Ασκήσεων

Ημερομηνία Παράδοσης: Παρασκευή 10 Μαϊου 2019.

Τρόπος Παράδοσης: Χειρόγραφες ή εκτυπωμένες σελίδες που τις αφήνετε στη γραμματεία του τμήματος (κατά τις ώρες λειτουργίας της).

Άσκηση **1** **[50 μονάδες]**

Έστω ότι μια βιβλιοθήκη σας παρέχει πρόσβαση σε ουρές χαρακτήρων. Η βιβλιοθήκη σας επιτρέπει να ορίσετε μια ουρά και να καλέσετε τις 5 βασικές λειτουργίες σε αυτή. Για παράδειγμα, ο ορισμός μιας ουράς γίνεται γράφοντας:

Queue Q;

Για την ουρά υποστηρίζονται οι λειτουργίες:

(1) void MakeEmptyQueue(Queue Q) /* αρχικοποιεί την ουρά Q */

(2) boolean IsEmptyQueue(Queue Q) /* επιστρέφει 1 αν η ουρά Q είναι κενή και 0 διαφορετικά */

(3) char Front(Queue Q) /* επιστρέφει το χαρακτήρα που βρίσκεται στο εμπρός μέρος της ουράς Q */

(4) char Dequeue(Queue Q) /* επιστρέφει το χαρακτήρα που βρίσκεται στο εμπρός μέρος της ουράς Q και τον διαγράφει από την Q */

(5) void Enqueue(Queue Q, char x) /* εισάγει τον χαρακτήρα x στην ουρά Q */

Σχεδιάστε αλγόριθμο ο οποίος δέχεται μια ουρά που περιέχει μια ακολουθία από χαρακτήρες (η ακολουθία δεν είναι απαραίτητα σε λεξικογραφική διάταξη) και μετά την εκτέλεσή του οι χαρακτήρες έχουν αποθηκευτεί σε λεξικογραφική διάταξη σε μια ουρά. Ο αλγόριθμος θα πρέπει να χρησιμοποιεί το μικρότερο δυνατό πλήθος ουρών. Μπορείτε να το κάνετε χρησιμοποιώντας μόνο την αρχική ουρά;

Προσοχή**:** Ο αλγόριθμος θα πρέπει να χρησιμοποιεί μόνο ουρές, καθώς και ενδεχόμενα κάποιες βοηθητικές μεταβλητές (σταθερό πλήθος, ανεξάρτητο του πλήθους των χαρακτήρων). Δεν επιτρέπεται η χρήση πινάκων ή άλλων δομών δεδομένων που δεν ανήκουν στη βιβλιοθήκη για την αποθήκευση ή την επεξεργασία των δεδομένων.

Άσκηση **2** **[50 μονάδες]**

Δίνεται η ακόλουθη υλοποίηση για αραιούς δισδιάστατους πίνακες. Κάθε αραιός δισδιάστατος πίνακας A[1..n, 1..m] υλοποιείται με έναν boolean δισδιάστατο πίνακα B[1..n, 1..m] (κάθε στοιχείο του B έχει τιμή 0 ή 1) και μια δυναμική συνδεδεμένη λίστα που περιέχει τα μη-μηδενικά στοιχεία. Για τον boolean δισδιάστατο πίνακα ισχύουν τα ακόλουθα. Για κάθε i, j, με $1 \leq i \leq n$, $1 \leq j \leq m$, B[i][j] = 1 αν και μόνο αν A[i][j] <> 0, και B[i][j] = 0 αν και μόνο αν A[i][j] = 0. Τα μη-μηδενικά στοιχεία στη λίστα αποθηκεύονται ως εξής: «τα μη-μηδενικά στοιχεία οποιασδήποτε γραμμής i απαντώνται στη λίστα πριν από τα μη-μηδενικά στοιχεία οποιαδήποτε γραμμής k > i και τα μη-μηδενικά στοιχεία οποιασδήποτε στήλης j απαντώνται πριν από τα μη-μηδενικά στοιχεία οποιασδήποτε στήλης l > j». Δηλαδή η σειρά με την οποία είναι αποθηκευμένα τα στοιχεία στη λίστα είναι η σειρά που τα συναντάμε στον πίνακα A αν τον σαρώσουμε από αριστερά προς τα δεξιά και από πάνω προς τα κάτω.

Για παράδειγμα, δίνεται ο αραιός πίνακας A:

0	7	0	0	0
1	2	0	0	3
0	0	4	0	0
12	0	0	0	0

Ο πίνακας B που αντιστοιχεί στον A είναι ο ακόλουθος:

0	1	0	0	0
1	1	0	0	1
0	0	1	0	0
1	0	0	0	0

και η λίστα έχει την εξής μορφή: 7 → 1 → 2 → 3 → 4 → 12

Παρουσιάστε αλγόριθμους που θα υλοποιούν τις ακόλουθες λειτουργίες στον αραιό πίνακα:

i. **Insert(***sparse_table A, int i, int j, int x***):** εισάγει το (μή-μηδενικό) στοιχείο x στη i-οστή γραμμή και τη j-οστή στήλη του αραιού πίνακα A.

ii. **Delete(***sparse_table A, int x***):** αναζητά το (μή-μηδενικό) στοιχείο x στον αραιό πίνακα A και αν υπάρχει το διαγράφει (δηλαδή το αντικαθιστά με 0).

iii. A**ddArrays(***sparse_table A1, A2,* A*3***):** προσθέτει τους αραιούς πίνακες A1 και A2, και ειστρέφει το αποτέλεσμα στον αραιό πίνακα A3.