Πανεπιστήμιο Θεσσαλίας
Τμήμα Πληροφορικής με Εφαρμογές στη Βιοϊατρική

Δομές Δεδομένων και Αλγόριθμοι
Ακαδημαϊκό Έτος 2013-2014
http://users.ucg.gr/~emarkou/greek/teach/data_structures
Ε. Μάρκου

2ο Σετ Ασκήσεων

Ημερομηνία Παράδοσης: Παρασκευή 2 Μαΐου 2014.

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

Άσκηση **1** **[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) /* αρχικοποιεί τη στοίβα S */

(2) boolean IsEmptyStack(stack S) /* επιστρέφει 1 αν η στοίβα S είναι κενή και 0 διαφορετικά */

(3) char Top(Stack S) /* επιστρέφει το χαρακτήρα που βρίσκεται στην κορυφή της στοίβας S */

(4) char Pop(Stack S) /* επιστρέφει το χαρακτήρα που βρίσκεται στην κορυφή της στοίβας S και τον διαγράφει από την S */

(5) void Push(Stack S, char x) /* εισάγει τον χαρακτήρα x στη στοίβα S */

Αντίστοιχα, 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 */

Θεωρήστε πως στη βιβλιοθήκη έχει δηλωθεί ένας πίνακας S από στοίβες: stack S[1..N],

καθώς και ένας πίνακας Q από ουρές: queue Q[1..N], τις οποίες μπορείτε να χρησιμοποιείτε χωρίς να τις δηλώσετε. Θεωρήστε επίσης ότι το N είναι αρκετά μεγάλο ώστε να καλύπτει τις ανάγκες σε στοίβες και ουρές των αλγορίθμων που ζητούνται στη συνέχεια.

i. Παλίνδρομη Συμβολοακολουθία

Μια συμβολοακολουθία a ονομάζεται παλίνδρομη αν έχει τη μορφή $a = wcw^T$, όπου w^T είναι η αντίστροφη συμβολοακολουθία της w και c είναι ένας χαρακτήρας. Για παράδειγμα, η αντίστροφη συμβολοακολουθία της "yam" είναι η "may". Έτσι, π.χ., η συμβολοακολουθία "yamamay" είναι παλίνδρομη. Να παρουσιαστεί αλγόριθμος που θα διαβάζει μια συμβολοακολουθία από το πληκτρολόγιο και θα ελέγχει αν αυτή είναι παλίνδρομη.

ii. Ταξινόμηση ακολουθίας χαρακτήρων

Σχεδιάστε αλγόριθμο ο οποίος θα διαβάζει μια ακολουθία από χαρακτήρες από το πληκτρολόγιο (η ακολουθία δεν είναι απαραίτητα σε λεξικογραφική διάταξη) και θα χρησιμοποιεί ουρές για να τυπώσει τους χαρακτήρες αυτούς σε λεξικογραφική διάταξη. Ο αλγόριθμος θα πρέπει να χρησιμοποιεί το μικρότερο δυνατό πλήθος ουρών. Υπόδειξη: Ο αλγόριθμος θα πρέπει να τοποθετεί προσεκτικά χαρακτήρες από την είσοδο στις διάφορες ουρές, έτσι ώστε η υποακολουθία χαρακτήρων που βρίσκεται σε κάθε ουρά να είναι λεξικογραφικά διατεταγμένη. Ο αλγόριθμος θα πρέπει να χρησιμοποιεί μια καινούρια ουρά μόνο εάν αυτό είναι απαραίτητο.

Προσοχή**:** 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 δεδομένων.

Άσκηση **2** **[35 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 $1 \le i \le n$, $1 \le j \le m$, 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 Α:

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

Ο πίνακας Β που αντιστοιχεί στον Α 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: $7 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 12$

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

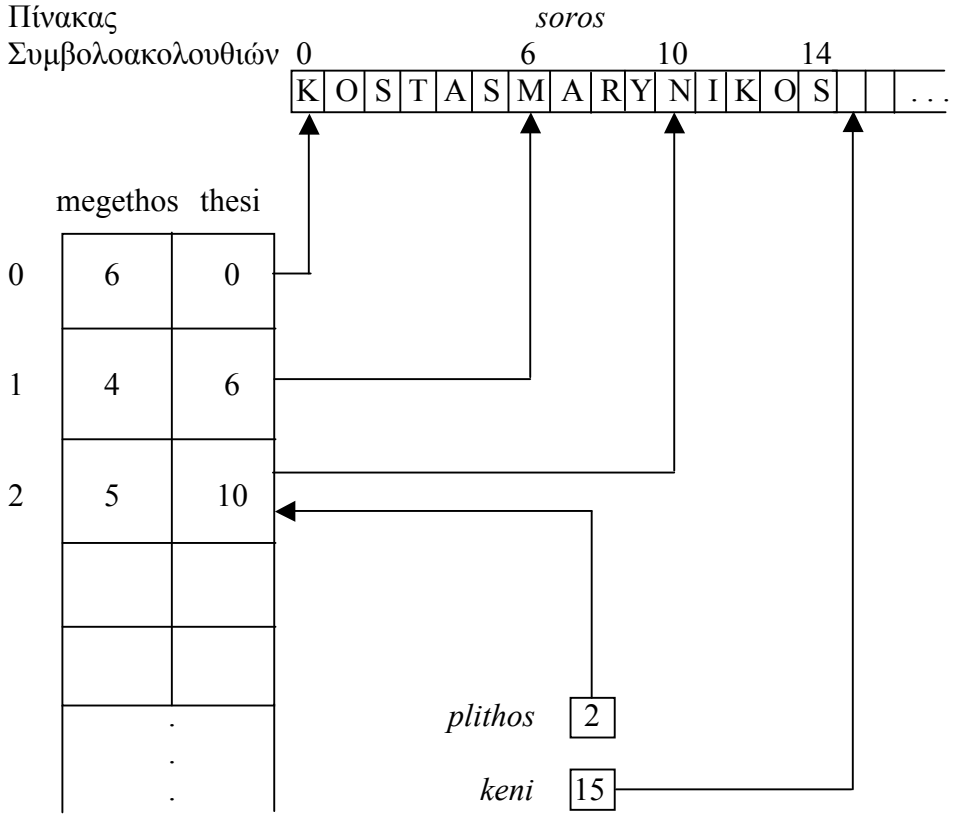
i. **Insert(***int x, int i, int j***):** εισάγει το στοιχείο x στη i-οστή γραμμή και τη j-οστή στήλη του αραιού πίνακα.

ii. **Delete(***int x***):** αναζητά το x στον αραιό πίνακα και αν υπάρχει το διαγράφει.

iii. A**ddArrays(***sparse_table A1, A2,* A*3***):** προσθέτει τους αραιούς πίνακες A1 και A2, δηλαδή δημιουργεί έναν νέο αραιό πίνακα A3 για τον οποίο ισχύει ότι A3[i][j] = A1[i][j] + A2[i][j], $\forall$ i,j με $1 \leq i \leq n$ και $1 \leq j \leq m$. Θεωρούμε πως όλοι οι αραιοί πίνακες, A1, A2 και A3, υλοποιούνται με τον τρόπο που περιγράφτηκε πιο πάνω.

Άσκηση **3** **[30 μονάδες]**

Μια άλλη υλοποίηση των συμβολοακολουθιών είναι εκείνη που χρησιμοποιεί ένα πίνακα *soros* και ένα βοηθητικό πίνακα για τον εντοπισμό των συμβολοακολουθιών στον πίνακα *soros*. Κάθε συμβολοακολουθία τοποθετείται στον πίνακα *soros* και παριστάνεται σαν μια δομή με μέλη *thesi* και *megethos*. Τα μέλη της δομής καθορίζουν τη θέση της συμβολοακολουθίας στον πίνακα *soros*. (βλ. σχήμα).



Για παράδειγμα, αν s είναι μια συμβολοακολουθία με s.thesi = 10 και s.megethos = 5, τότε οι χαρακτήρες της s, "NIKOS" βρίσκονται στις θέσεις 10 μέχρι 10 + 5 – 1 = 14 του πίνακα *soros*.

Επίσης χρησιμοποιούνται τα μέλη:
plithos η οποία δείχνει την τελευταία θέση που είναι κατειλημμένη στο βοηθητικό πίνακα,
keni που 'δείχνει' στην πρώτη κενή θέση του πίνακα *soros*.

Να γραφτούν σε C οι κατάλληλες δηλώσεις και οι συναρτήσεις όλων των βασικών πράξεων του ΑΤΔ συμβολοακολουθία για την παραπάνω υλοποίηση.
Ποια είναι κατά τη γνώμη σας τα πλεονεκτήματα και μειονεκτήματα της παραπάνω υλοποίησης σε σχέση με αυτή που υπάρχει στις διαφάνειες; Να σχολιάσετε τη χωρική και χρονική πολυπλοκότητα της χειρότερης περίπτωσης των δύο υλοποιήσεων.

Να προτείνετε και να υλοποιήσετε την **antigrafi_seiras** έτσι ώστε αυτή να εκτελείται σε χρόνο *O(1)*.

Υπόδειξη:
Μπορείτε να έχετε τον πίνακα soros και τον βοηθητικό πίνακα καθολικά δηλωμένους (στο κύριο πρόγραμμα). Με αυτό τον τρόπο οι βασικές συναρτήσεις θα έχουν μόνο τα απαραίτητα ορίσματα και θα δρούν στους καθολικούς πίνακες.