Array Initialization in O(1) time

Πρόβλημα: Δίνεται array A [0...N-1], το οποίο πρέπει να αρχικοποιηθεί με την τιμή 0, σε κάθε θέση του. Να δώσετε ένα τρόπο για να αποφευχθεί η αρχικοποίηση.

Απάντηση:
Χρησιμοποιούμε επιπλέον πίνακες AUX και S, και οι δυο μεγέθους N, χωρίς να χρειάζονται αρχικοποίηση. Ο S υλοποιεί μια στοίβα η οποία κρατά όλες τις θέσεις του A που έχουν έγκυρες τιμές. Οι θέσεις του πίνακα AUX χρησιμοποιούνται για τυχαία προσπέλαση της στοίβας με τρόπο που θα φανεί παρακάτω. Επίσης χρησιμοποιείται η μεταβλητή TOP που δείχνει στην κορυφή της στοίβας. Αρχικά TOP=-1, συνθήκη ισοδύναμη με την αρχικοποίηση του A με μηδενικά.

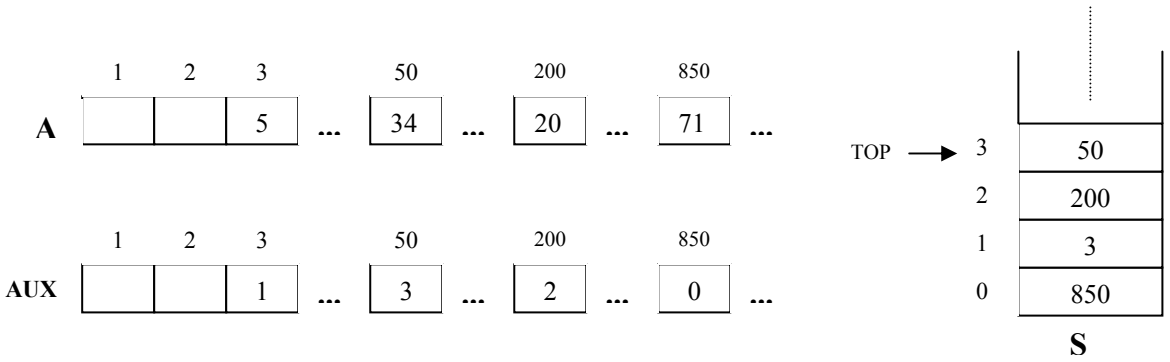
Κατά τη διάρκεια της εκτέλεσης, διατηρούμε την εξής συνθήκη:

$$A[i] = s, \text{ αν και μόνο αν } 0 \le AUX[i] \le TOP \;\&\&\; S[AUX[i]] = i \qquad (1)$$

Οπότε, όταν χρειαστεί να γράψουμε ή να διαβάσουμε μια τιμή από το A[i], ελέγχουμε την τήρηση της παραπάνω συνθήκης. Αν επαληθεύεται, μπορούμε να συνεχίσουμε να κάνουμε την πράξη. Διαφορετικά το i, πρέπει να εισαχθεί στη στοίβα και να ενημερωθεί γι' αυτό και ο AUX, δηλαδή

```
TOP = TOP+1;
S[TOP]= i;
AUX[i] = TOP;
A[i] = s;
```

Έστω ότι ο A δεν έχει καμία τιμή και ότι για πρώτη φορά αποφασίζουμε να γράψουμε στη θέση A[850]. Τότε αυξάνουμε το TOP, λαμβάνοντας TOP=0 και θέτουμε S[0]=850, δηλαδή ενημερώνουμε τη στοίβα ότι ο A έχει έγκυρη τιμή στη θέση A[850].
Το παρακάτω σχήμα δίνει τα περιεχόμενα των A, AUX, S, μετά την ακολουθία ενθέσεων A[850]=71, A[3]=5, A[200]=20, A[50]=34



Παρατηρήσεις:

Μόνο ο έλεγχος $0 \le AUX[i] \le TOP$ δεν θα αρκούσε γιατί ο AUX είναι *μη αρχικοποιημένος*. Συνεπώς θα μπορούσε, για παράδειγμα, για κάποιο i, να ισχύει $0 \le AUX[i] \le TOP$, αλλά ο AUX να περιέχει σκουπίδια στη συγκεκριμένη θέση μνήμης

και η επαλήθευση της ανισότητας να είναι γι' αυτό το λόγο ψευδής. Αυτή η περίπτωση μπορεί να απεικονιστεί στο παραπάνω σχήμα αν επιχειρήσει κάποιος να προσπελάσει τη διεύθυνση A[910] και έστω AUX[910]=1. Επειδή S[1]=3 σημαίνει ότι η θέση A[910] δεν έχει αρχικοποιηθεί και άρα περιέχει μη έγκυρα δεδομένα. Το επόμενο βήμα θα ήταν να γινόταν εισαγωγή του 910 στη στοίβα και να τεθεί A[910]=0 (τιμή αρχικοποίησης)

Η στοίβα δεν μπορεί να χρησιμοποιηθεί από μόνη της γιατί θέλουμε να επαληθεύουμε αν κάποια τιμή του i, έχει περαστεί μέσα στη στοίβα χωρίς μεγάλη επιβάρυνση. Για να πετύχουμε O(1) επιβάρυνση ανά βήμα προσπέλασης χρειαζόμαστε και τη βοήθεια του AUX. (Πόση θα ήταν η επιβάρυνση αν θα χρησιμοποιούσαμε μόνο τη στοίβα;)

Εφαρμογές:

Η παραπάνω τεχνική έχει εφαρμογή στους μεταφραστές όπου όταν χρειάζεται η αρχικοποίηση ενός τεράστιου πίνακα, αυτή αποφεύγεται και ο πίνακας απλώς παίρνει τιμές, όταν αυτό είναι αναγκαίο από το πρόγραμμα που τον χρησιμοποιεί. Μια δεύτερη εφαρμογή (βλ. Mehlhorn p.289) είναι η υλοποίηση boolean πινάκων που χρησιμοποιούνται για την αναπαράσταση ενός συνόλου. Έστω για παράδειγμα ένα σύνολο $S \subseteq U$, $U=\{1, 2, \ldots, N\}$. Για την αναπαράσταση του S χρησιμοποιείται πίνακας BV, τέτοιος ώστε, $BV[i] = 1$ ανν $i \in S$ και $BV[i] = 0$, $i \notin S$. Προφανώς για τη σωστή αναπαράσταση του συνόλου, όταν αυτό υπόκειται στις πράξεις Insert, Delete, χρειάζεται αρχικοποίηση του BV με την τιμή 0. Δεν μπορούμε να αποδεχτούμε την υπόθεση ότι δεν αρχικοποιούμε το A, και όποτε χρειαστεί πηγαίνουμε στις αντίστοιχες θέσεις και τις θέτουμε σε true. Η έλλειψη του true δεν σημαίνει false. Αυτό γιατί μια πράξη Access μπορεί προσπελάζοντας μια θέση του A να συναντήσει την τιμή true, η οποία όμως αντιστοιχεί σε σκουπίδια που υπήρχαν στην αντίστοιχη θέση μνήμης και όχι σε πραγματική τιμή και άρα να οδηγηθούμε σε λάθος.