Αλγόριθμοι και Πολυπλοκότητα

Γιώργος Αγγέλης
el18030

1η γραπτή σειρά ασκήσεων

Άσκηση 2

(α)

Γράφουμε τον αλγόριθμο σε ψευδογλώσσα:

```
max = max(A[1…n])                              //find max element of A

count_flips = 0                                //count the number of flips

find_max_function(A[1…n]):                     //function to find max element

    i = index(max(A[1…n]))                     //find the position of max in the array

        if i==n continue                       //no need to flip - optimization1

        if i==1 go to prethematic_flip(A(n))   //one flip needed - optimization2

        prethematic_flip(i)                    //execute a prethematic flip with upper
                                               //bound the max element in order to
        count_flips++                          //move it to the first place of the array

        prethematic_flip(A(n))                 //execute a prethematic flip of the
                                               //entire array to move the max element
        count_flips++                          //to the last place of the array

main():

    x=0

    while n-x>2:                               //while we have not reached the
                                               //second element of the array we keep
        find_max_function(A[1…n-x]):           //flipping the subarrays in order to sort
                                               //the entire array.
        x++

if A[1] > A[2]:                                //in case the first two elements of the
                                               //array are not sorted perform one
    prethematic_flip(A[2])                     //prethematic_flip() on the second
                                               //element - optimization3
```

Σημείωση: η συνάρτηση prethematic_flip(a) την οποία χρησιμοποιούμε και πιο πάνω και παρακάτω θεωρούμε ότι παίρνει μία παράμετρο, έστω a, η οποία είναι το άνω όριο στο οποίο θα γίνει η προθεματική περιστροφή και εντέλει θα βρεθεί σε πρώτη θέση.

Συνολικά θα γίνουν count_flips προθεματικές περιστροφές, όπου count_flips ≤ 2*(n-2) + 1 = 2*n - 3 (λόγω του **optimization 3**) στην χειρότερη περίπτωση.

Επεξήγηση:

Για να ταξινομήσουμε έναν πίνακα μεγέθους N με προθεματικές περιστροφές <u>χωρίς τις παραπάνω βελτιστοποιήσεις (optimizations)</u> χρειαζόμαστε 2*(N-1) τέτοιες περιστροφές, αφού στην τελευταία επανάληψη (που οφείλεται στο δεύτερο στοιχείο του) θα ταξινομηθεί και το πρώτο. Άρα γενικά χρειαζόμαστε στην χειρότερη περίπτωση και χωρίς βελτιστοποιήσεις 2*(N-1) περιστροφές. Με τις βελτιστοποιήσεις πετυχαίνουμε:

- **optimization1**: σε περίπτωση που το max στοιχείο του (υπο-)πίνακα βρίσκεται ήδη στην τελευταία θέση να μην κάνουμε τις δύο περιστροφές που αντιστοιχούν στην επανάληψη αυτή χωρίς λόγο.
- **optimization2**: σε περίπτωση που το max στοιχείο του (υπο-)πίνακα βρίσκεται στην πρώτη θέση να μην κάνουμε την πρώτη περιστροφή που αντιστοιχεί στην επανάληψη αυτή χωρίς λόγο.
- **optimization3**: για τα δύο τελευταία στοιχεία που θα ελέγξει ο αλγόριθμος (που θα είναι τα δύο πρώτα πλέον στον πίνακα) στην περίπτωση που αυτά δεν είναι ταξινομημένα μας αρκεί ένα prethematic_flip() στο A[2], που θα φέρει τα στοιχεία στην σωστή σειρά.

(β)

Ο παραπάνω αλγόριθμος θα χρησιμοποιηθεί και εδώ με μία διαφοροποίηση. Εφόσον κάθε στοιχείο του τελικού πίνακα έχει περιστραφεί δύο ή καμία φορά γνωρίζουμε ότι θα έχει το σωστό πρόσημο. Συνεπώς το μόνο πρόβλημα στον παραπάνω αλγόριθμο έχει να κάνει με το τι συμβαίνει στην περίπτωση που υπάρξει max στοιχείο (υπο-)πίνακα στην πρώτη θέση. Εκεί αρκεί να κάνουμε μία προθεματική περιστροφή μόνο για το max = A[1], που θα οδηγήσει σε -A[1], το οποίο με τη σειρά του θα λάβει αρνητικό (σωστό, δηλαδή) πρόσημο στην επόμενη περιστροφή που θα το στείλει στην τελευταία θέση του πίνακα. Συνεπώς εδώ θα έχουμε το πολύ 3*(n-3) + 3 = 3*n - 6 περιστροφές.

(γ)

1.
Έστω ότι δεν υπάρχει ζεύγος στοιχείων (A_t[i], A_t[i + 1]) στον ενδιάμεσο πίνακα A_t, που δεν είναι ο [−1, −2, . . . , −n]. Θα δείξουμε ότι μπορούμε να κατασκευάσουμε ένα νέο συμβατό ζεύγος έπειτα από 2 το πολύ προσημασμένες προθεματικές περιστροφές.

- Αν όλα τα στοιχεία έχουν αρνητικό πρόσημο, επειδή ο A_t δεν είναι ο [−1, −2, . . . , −n], για κάποιο στοιχείο του, έστω |A_t[x]| = k, θα υπάρχει αριστερότερα άλλο στοιχείο, έστω A_t[y], με y<x, όπου |A_t[y]| = k+1. Οπότε εκτελούμε prethematic_flip(A_t[y]) και άρα πλέον A_t[1] = k+1 και έχουμε: […, -(k+1), …, -k, …,] → [(k+1), …, -k, …,]. Κατόπιν εκτελούμε prethematic_flip(A_t[x-1]) (δηλαδή φέρνουμε το στοιχείο του πίνακα με απόλυτη τιμή (k+1) στην πρώτη θέση). Οπότε έχουμε [(k+1), …, -k, …,] → […, -(k+1), -k, …,] και προκύπτει το ζεύγος (-(k+1), -k).

- Αν όλα τα στοιχεία έχουν θετικό πρόσημο βλέπε ερώτημα (β).

- Αν υπάρχει τουλάχιστον ένα στοιχείο με θετικό πρόσημο βρίσκουμε το θετικό x = max(A_t[i]), για τα i όπου A_t[i]>0.

  • Αν είναι και το max όλων των απόλυτων τιμών, δηλαδή, x = max(A_t[i]) για κάθε i, με δύο κινήσεις το μεταφέρουμε στο τέλος αφού αυτό θεωρείται καταχρηστικά δημιουργία συμβατού ζεύγους.

- Ειδάλλως, θα υπάρχει στοιχείο με απόλυτη τιμή x+1 και αρνητικό πρόσημο (δηλ. -(x+1)) είτε αριστερά είτε δεξιά του x.

  - Αριστερά:
    [{…, -(x+1), …, x}, …,] → [{-x, …}, (x+1), …,] → […, x, (x+1), …,] οπότε με δύο περιστροφές δημιουργήθηκε κατάλληλο ζεύγος: (x, (x+1)).

  - Δεξιά:
    [{…, x, …, -(x+1)}, …,] → [{(x+1), …}, -x, …,] → […, -(x+1), -x, …,] οπότε με δύο περιστροφές δημιουργήθηκε κατάλληλο ζεύγος: (-(x+1), -x)

2.

<u>Βήμα 1:</u>

Ζεύγη ήδη ταξινομημένων στοιχείων αντικαθίστανται με έναν άλλο αριθμό, προκειμένου να οδηγηθούμε σε ένα μικρότερο πρόβλημα.

<u>Βήμα 2:</u>

Έστω ότι ο πίνακας είναι της μορφής [−1,−2,…,−m]. Προκειμένου να ταξινομηθεί εκτελούμε τον αλγόριθμο:

```
i=0

for i in range(m):

        prethematic_flip(A(m))

        prethematic_flip(A(m-1))
```

Στο τέλος προκύπτει ο πίνακας ταξινομημένος. Αυτό συμβαίνει γιατί αναδρομικά το τελευταίο στοιχείο έρχεται πρώτο αλλάζοντας πρόσημο κάθε φορά.

<u>Βήμα 3:</u>

Κατασκευάζουμε συμβατό ζεύγος όπως είδαμε στο Ερώτημα 1 και επαναλαμβάνουμε το <u>Βήμα 1</u> μέχρι να ταξινομηθεί πλήρως ο πίνακας.

Συνολικά θα γίνουν 2i κινήσεις (προθεματικές περιστροφές) για i επαναλήψεις, i<n.

Άσκηση 3

Για τη λύση θα χρησιμοποιήσουμε stack.

Αλγόριθμος:

Έχουμε τον πίνακα A[1…n] και έστω ο πίνακας B[1…n]. Ορίζουμε μία στοίβα s. Τοποθετούμε στην αρχή της στοίβας το πρώτο στοιχείο του πίνακα A και ορίζουμε B[1] = FALSE.

Διατρέχουμε τον πίνακα A για κάθε i από 1 έως N.

> Όσο η στοίβα δεν είναι άδεια και το πρώτο στοιχείο της στοίβας είναι μικρότερο από το A[i] διώχνουμε το πρώτο στοιχείο της στοίβας.
>
> Αν η στοίβα αδειάσει τότε δεν βρήκαμε μεγαλύτερο στοιχείο και θέτουμε B[i] = FALSE
>
> Αν βρεθεί μεγαλύτερο στοιχείο θέτουμε B[i] = τρέχουσα κεφαλή της στοίβας
>
> Βάζουμε στη στοίβα το τρέχων στοιχείο του A.

Ο πίνακας B περιέχει το ζητούμενο.

Η υπολογιστική πολυπλοκότητα του παραπάνω αλγορίθμου είναι O(n). Η ορθότητα του αλγορίθμου προκύπτει από το ότι σε κάθε επανάληψη γίνεται εξαντλητικός έλεγχος σε όλα τα προηγούμενα στοιχεία του εκάστοτε A[i].

ΤΕΛΟΣ