Ερωτήματα

Ερώτημα 1.
Έστω $n \in \mathbb{N}$ και k μη-αρνητικός ακέραιος. Χωρίς τη χρήση ορίων να δείξετε ότι

1. $(n+1)^k = \Theta(n^k)$, **(1 βαθμός)**
2. $(n+k)^2 = \Theta(n^2+k^2)$. **(2 βαθμοί)**

Απάντηση

1. Προφανώς

$$n+1 > n \Rightarrow (n+1)^k \geq n^k \Rightarrow (n+1)^k = \Omega(n^k), \quad (1)$$
$$n+1 \leq n+n \Rightarrow (n+1)^k \leq (n+n)^k = (2n)^k = 2^k \cdot n^k \Rightarrow (n+1)^k = O(n^k). \quad (2)$$

(1),(2) συνεπάγονται το ζητούμενο.

2. Αν $k=0$ τότε το ζητούμενο είναι προφανές. Διαφορετικά,

$$(n+k)^2 = n^2 + 2kn + k^2 \quad (3)$$
$$\Rightarrow (n+k)^2 > n^2 + k^2 \Rightarrow (n+k)^2 = \Omega(n^2+k^2). \quad (4)$$

Για το άνω φράγμα, είτε $n \geq k$ ή $n < k$. Από την (3) έχουμε για την πρώτη περίπτωση,

$$(n+k)^2 = n^2 + 2kn + k^2 \leq 4n^2,$$

ενώ για τη δεύτερη

$$(n+k)^2 = n^2 + 2kn + k^2 < 4k^2.$$

Σε κάθε περίπτωση

$$(n+k)^2 = n^2 + 2kn + k^2 < 4(n^2+k^2) \Rightarrow (n+k)^2 = O(n^2+k^2).. \quad (5)$$

(4) , (5) συνεπάγονται το ζητούμενο.

Ερώτημα 2.
Έστω $a =< a_1, \ldots, a_n >$ μία ταξινομημένη (αύξουσα) διάταξη ακεραίων. Από αυτή παράγεται μία *κυκλικά κυλιόμενη προς τα δεξιά κατά* $k > 0$ *θέσεις* σειρά $a' =< a'_1, \ldots, a'_n >$ ως εξής: για $i = 1, \ldots, n$,

$$a'_{(i+k) \bmod n} = a_i, \text{ αν } (i+k) \bmod n \neq 0, \quad (6)$$
$$a'_n = a_i, \text{ αν } (i+k) \bmod n = 0. \quad (7)$$

Δίνεται ένας πίνακας A ο οποίος περιέχει στις θέσεις από 1 ως n μία κυκλικά κυλιόμενη προς τα δεξιά σειρά ακεραίων κατά $k > 0$ θέσεις. Θεωρήστε για ευκολία ότι $n = 2^p, p \in \mathbb{N}$. Να εκπονήσετε έναν αλγόριθμο που να δέχεται σαν είσοδο τον πίνακα A και να βρίσκει (επιστρέφει) το μεγαλύτερο στοιχείο του πίνακα σε κάθε μία από της παρακάτω περιπτώσεις:

1. θεωρώντας το k γνωστό, **(1 βαθμός)**
2. θεωρώντας το k άγνωστο - ο αλγόριθμος σας πρέπει να είναι τύπου *διαίρει και βασίλευε* και τάξης $O(n)$, **(2 βαθμός)**
3. θεωρώντας το k άγνωστο και επιπλέον ισχύει ότι όλα τα στοιχεία της σειράς είναι διαφορετικά μεταξύ τους,- ο αλγόριθμος σας πρέπει να είναι τάξης $o(n)$. **(4 βαθμοί)**

Υπόδειξη: Για καθένα από τους παραπάνω (τρεις) αλγόριθμους πριν την κωδικοποίηση, να περιγράψετε με λόγια την κεντρική ιδέα που αυτός υλοποιεί. Μετά την κωδικοποίηση, να παραθέσετε απόδειξη της πολυπλοκότητας του.

Απάντηση

1. Η αρχική σειρά είναι ταξινομημένη και επομένως το μεγαλύτερο στοιχείο σε αυτή βρίσκεται στην θέση n. Εφόσον το k είναι γνωστό, με βάση τις (6) και (7), το στοιχείο αυτό στην κυλιόμενη σειρά μετατίθεται στη θέση j όπου

$$j = \begin{cases} n, & \text{αν } k \bmod n = 0, \\ (n+k) \bmod n & \text{διαφορετικά.} \end{cases}$$

Ο Αλγόριθμος (1) υλοποιεί την ιδέα. Όπως είναι προφανές εκτελείται σταθερός αριθμός στοιχειωδών πράξεων και επομένως

Αλγοριτημ 1 Γνωστό k

```
1: int Find_Max(int A, int n, int k)
2: if k mod n = 0 then
3:    return A[n];
4: else
5:    return A[(n + k) mod n];
6: end if
```

ο αλγόριθμος είναι τάξης $\Theta(1)$.

2. Στην περίπτωση αυτή μπορούμε να διαμερίσουμε τα στοιχεία του πίνακα περίπου στη μέση και θα αναζητήσουμε το μεγαλύτερο στοιχείο σε καθένα από τα δύο μέρη (φάση Διαίρει.) Στη συνέχεια θα συγκρίνουμε τα δύο μεγαλύτερα αυτά στοιχεία μεταξύ τους και θα επιστρέψουμε το μεγαλύτερο (φάση Συνδύασε (Βασίλευε).) Ο Αλγόριθμος (2) υλοποιεί την ιδέα. Ο αριθμός των

Αλγοριτημ 2 Άγνωστο k

```
1: int Find_Max_Dc(int A, int lo, int hi )
2: if lo = hi then
3:    return A[lo];
4: end if
5: mid ← ⌊(lo+hi)/2⌋;
6: max_left ← Find_Max_Dc(A, lo, mid);
7: max_right ← Find_Max_Dc(A, mid + 1, hi);
8: if max_left > max_right then
9:    return max_left;
10: else
11:    return max_right;
12: end if
```

στοιχειωδών πράξεων δίνεται από την αναδρομική σχέση

$$T(n) = \begin{cases} 2T(\frac{n}{2}) + \Theta(c), & \text{αν } n \geq 2, \\ 1 & \text{αν } n \leq 1. \end{cases}$$

Σύμφωνα με το κεντρικό θεώρημα (πρώτη περίπτωση) η ασυμπτωτική συμπεριφορά της παραπάνω συνάρτησης είναι $\Theta(n)$.

3. Η ιδέα του διαχωρισμού των στοιχείων σε δύο μέρη περίπου ίδιου μεγέθους μπορεί να χρησιμοποιηθεί πιο αποδοτικά. Υπάρχουν δύο περιπτώσεις σε σχέση με την παράμετρο k : α) $k \bmod n = 0$ και β) $k \bmod n \neq 0$. Στην πρώτη περίπτωση ο πίνακας A είναι ταξινομημένος ενώ στη δεύτερη όχι. Σε κάθε μία από τις δύο αυτές περιπτώσεις μελετάμε τη σχέση που έχει ένα στοιχείο που βρίσκεται σε μία τυχαία θέση j του πίνακα, όπου $1 \leq j \leq n$.

   α) Στην περίπτωση αυτή ο πίνακας A περιέχει την αρχική σειρά και είναι ταξινομημένος. Επομένως για κάθε $1 \leq j \leq n$ ισχύει ότι $A[1] \leq A[j] \leq A[n]$.

   β) Ο πίνακας δεν είναι ταξινομημένος και μπορεί να χωριστεί σε δύο τμήματα: στο πρώτο τμήμα ανήκουν τα στοιχεία από τη θέση 1 ως τη θέση $\lambda = \text{argmax}\{A[i] : i = 1, \ldots, n\}$ και στο τμήμα από τη θέση λ μέχρι τη θέση n. Θυμηθείτε ότι η θέση λ δεν είναι γνωστή· είναι η θέση την οποία αναζητούμε (συγκεκριμένα αναζητούμε το στοιχείο $A[\lambda]$). Αν πάρουμε μία τυχαία θέση j του πίνακα τότε αυτό βρίσκεται ή στο πρώτο τμήμα ή στο δεύτερο ή $j = \lambda$. Στην πρώτη περίπτωση $A[1] < A[j]$ και η θέση λ βρίσκεται στο τμήμα από $A[j]$ μέχρι $A[n]$. Στη δεύτερη περίπτωση $A[1] > A[j]$ και η θέση λ βρίσκεται στο τμήμα από $A[1]$ μέχρι $A[j]$. Παρατηρούμε ότι σε κάθε τμήμα το τελευταίο στοιχείο δεν μπορεί να είναι το μεγαλύτερο της σειράς γιατί τότε για την πρώτη περίπτωση θα είχαμε την περίπτωση (α) ενώ για τη δεύτερη περίπτωση θα είχαμε τη σειρά σε φθίνουσα διάταξη (αντίφαση με δεδομένο ότι η πρωτότυπη σειρά των αριθμών είναι ταξινομημένη σε αύξουσα σειρά)

Οι παραπάνω παρατηρήσεις οδηγούν στον εξής αλγόριθμο. Επιλέγουμε μία τιμή για τον δείκτη j που να διαχωρίζει τον πίνακα A περίπου στη μέση. Αν ισχύουν οι ανισότητες της περίπτωσης (α) ο πίνακας είναι ταξινομημένος και επομένως επιστρέφεται το στοιχείο $A[n]$. Στην περίπτωση αυτή ο αλγόριθμος ολοκληρώνεται. Στην περίπτωση (β) πρέπει να επιλύσουμε αναδρομικά ένα πρόβλημα με το μισό αριθμό στοιχείων (δεδομένου ότι ο δείκτης j είναι τέτοιος ώστε να διαχωρίζεται ο πίνακας περίπου στη μέση). Παρατηρήστε ότι εφόσον και στα δύο τμήματα περιλάβουμε το τελευταίο στοιχείο πάντα το υποπρόβλημα που θα εξετάζεται αναδρομικά θα ανήκει στην περίπτωση (β).

Η διαδικασία έχει ίδια πολυπλοκότητα με αυτή της δυαδικής αναζήτησης, δηλαδή $O(\lg n)$ η οποία είναι $o(n)$. Η συνάρτηση **Find_Max_Bs** υλοποιεί την περίπτωση (β) αναδρομικά. Η συνάρτηση ολοκληρώνεται όταν ο πίνακας αποτελείται ουσιαστικά από δύο στοιχεία. Ο αλγόριθμος απεικονίζεται παρακάτω.

If $A[lb] < A[ub]$ **then**
  return $A[ub]$
else
  return Find_Max_Bs$(A, 1, n)$
end if

Algorithm 3 Άγνωστο k περίπτωση (β)

```
1: int Find_Max_Bs(int A, int lb, int ub)
2: mid ← ⌊(ub+lb)/2⌋
3: if lb = mid then
4:   return A[lb]
5: else
6:   if A[lb] < A[mid] then
7:     return Find_Max_Bs(A, mid, ub);
8:   else
9:     return Find_Max_Bs(A, lb, mid);
10:  end if
11: end if
```