Αναδρομή
Ασκήσεις

Δημήτριος Μάγος

29 Απριλίου 2023

Άσκηση 1. Το πλήθος των διαφορετικών ομάδων καθεμία αποτελούμενη από k αντικείμενα που μπορούν να σχηματιστούν από ένα σύνολο n διακριτών αντικειμένων συμβολίζεται με $C(n,k)$. Είναι γνωστό ότι $C(n,0) = C(n,n) = 1, C(n,k) = 0$ για $k > n$ ενώ για $k < n$ έχουμε

$$C(n,k) = C(n-1,k) + C(n-1,k-1).$$

Να εκπονήσετε αναδρομικό αλγόριθμο ο οποίος να δέχεται σαν είσοδο μη-αρνητικούς ακεραίους k, n και να επιστρέφει την τιμή $C(n,k)$.

Λύση. Ο Αλγόριθμος 1 υπολογίζει το ζητούμενο.

Αλγόριθμος 1 Υπολογισμός του $C(n,k)$

Απαιτείται: Ακέραιοι $n, \geq 0$.
Επιστρέφεται: Αριθμός συνδυασμών n ανά k.

```
1: function Comb(int n, int k )
2:     if k > n then
3:         return 0;
4:     end if
5:     if k = n  or k = 0 then
6:         return 1;
7:     end if
8:     return Comb(n − 1, k) + Comb(n − 1, k − 1);
9: end function
```

■

Άσκηση 2. Μία συμβολοσειρά αποτελούμενη από ψηφία του συνόλου $\{0,1\}$ ονομάζεται *παλίνδρομη* αν ταυτίζεται με την αντίστροφή της. Να εκπονήσετε αλγόριθμο ο οποίος να δέχεται σαν είσοδο μία σειρά από αριθμούς και να αποφασίζει αν η συμβολοσειρά αυτή είναι παλίνδρομη.Να διατυπώσετε την αναδρομική σχέση που περιγράφει τον αριθμό των ΣΥΒ.

Λύση. Ο Αλγόριθμος 2 υπολογίζει το ζητούμενο. Αν τα στοιχεία βρίσκονται στις n πρώτες θέσεις του πίνακα A τότε ο Αλγόριθμος 2 καλείται ως

$$decide \leftarrow \text{Palindrome}(A, 1, n);$$

Παρατηρούμε ότι σε κάθε κλήση της συνάρτησης εκτελείται σταθερός αριθμός ΣΥΒ. Στην αναδρομική κλήση που εκτελείται στη Γραμμή 5 του αλγόριθμου, εξετάζεται ένα στιγμιότυπο με

$$hi - 1 - (lo + 1) + 1 = hi - lo - 1$$

στοιχεία. Το πλήθος των στοιχείων του αρχικού στιγμιότυπου είναι $n = hi - lo + 1$ και επομένως στο στιγμιότυπο που εξετάζεται στην αναδρομική κλήση έχει $n-2$ στοιχεία. Άρα ο αριθμός των ΣΥΒ περιγράφεται από την αναδρομική σχέση

$$T(n) \leq \begin{cases} T(n-2) + \Theta(1), & n \geq 2, \\ \Theta(1), & \text{διαφορετικά .} \end{cases}$$

■

Αλγόριθμος 2 Αποφασίζεται αν μία σειρά είναι παλίνδρομη

Απαιτείται: Ακέραιοι lo, hi $(lo \leq hi)$ πίνακας A που περιέχει στις θέσεις από lo ως hi στοιχεία από το σύνολο $\{0, 1\}$

Επιστρέφεται: `True` αν ο πίνακας A αποτελεί μία παλίνδρομη σειρά, `False` διαφορετικά.

```
1:  function Palindrome(int A[], int lo, int hi )
2:      if lo < hi then
3:          if A[lo] = A[hi] then
4:              return Palindrome(A, lo + 1, hi − 1);
5:          else
6:              return False;
7:          end if
8:      else
9:          return True;
10:     end if
11: end function
```

Άσκηση 3. Μία σειρά από διαφορετικούς ακεραίους $a_1, \ldots, a_n$ ονομάζεται *μονότροπη (unimodal)* αν υπάρχει ένα στοιχείο $a_k, 1 \leq k \leq n$, για το οποίο ισχύει

$$a_1 < a_2 < \cdots < a_{k-1} < a_k > a_{k+1} > \cdots > a_{n-1} > a_n.$$

Από τον παραπάνω ορισμό προκύπτει ότι το σημείο καμπής μπορεί να είναι το πρώτο ή το τελευταίο στοιχείο της σειράς. Το στοιχείο a_k ονομάζεται *σημείο καμπής*.
Δίνεται μία μονότροπη σειρά ακεραίων αποθηκευμένη στις θέσεις $1, \ldots, n$ του πίνακα A. Να εκπονήσετε αναδρομικό αλγόριθμο που να επιστρέφει τη θέση του σημείου καμπής και να διατυπώσετε την αναδρομική σχέση που περιγράφει τον αριθμό των ΣΥΒ που εκτελούνται.

Λύση. Θα εξετάσουμε το στοιχείο σε μία τυχαία θέση j $(1 \leq j \leq n)$ του πίνακα A. Αν $A[j] < A[j + 1]$ τότε το στοιχείο καμπής βρίσκεται από τη θέση $j + 1$ και μετά. Διαφορετικά, το ζητούμενο στοιχείο βρίσκεται από τη θέση j και πριν. Ο Αλγόριθμος 3 υλοποιεί την ιδέα.

Αλγόριθμος 3 Υπολογισμός σημείου καμπής σε μονότροπη σειρά

Απαιτείται: ακέραιοι lo, hi $(lo \leq hi)$ πίνακας A με στοιχεία που σχηματίζουν μονότροπη σειρά

Επιστρέφεται: σημείο καμπής

```
1:  function Unimodal(int A[], int lo, int hi )
2:      if lo = hi then
3:          return A[lo];
4:      end if
5:      j ← ⌊(lo+hi)/2⌋;
6:      if A[j] < A[j + 1] then
7:          return Unimodal(A, j + 1, hi);
8:      else
9:          return Unimodal(A, lo, j);
10:     end if
11: end function
```

Σε κάθε κλήση της συνάρτησης Unimodal εκτελείται ένας σταθερός αριθμός ΣΥΒ και μετά είτε έχουμε κλήση της συνάρτησης για την επίλυση ενός στιγμιότυπου με

$$hi - (\lfloor \frac{lo+hi}{2} \rfloor + 1) + 1 \tag{1}$$

στοιχεία ή ενός στιγμιότυπου με

$$\lfloor \frac{lo+hi}{2} \rfloor - lo + 1 \tag{2}$$

στοιχεία. Γνωρίζουμε ότι

$$hi - lo + 1 = n \Rightarrow hi = n - 1 + lo \Rightarrow hi + lo = n - 1 + 2lo \tag{3}$$

Συνδυάζοντας την (3) με την (2) έχουμε

$$\lfloor \frac{lo+hi}{2} \rfloor - lo + 1 = \lfloor \frac{n-1+2lo}{2} \rfloor - lo + 1 \tag{4}$$

$$= \lfloor \frac{n-1}{2} \rfloor + 1 = \lfloor \frac{n+1}{2} \rfloor = \lceil \frac{n}{2} \rceil. \tag{5}$$

Συνεπώς η (1) συνεπάγεται

$$hi - (\lfloor \frac{lo+hi}{2} \rfloor + 1) + 1 = \lfloor \frac{n}{2} \rfloor \tag{6}$$

Από τις (5), (6) έχουμε ότι για τη συνάρτηση που δίνει τον αριθμό των ΣΥΒ ισχύει

$$T(n) \leq \begin{cases} T(\lceil \frac{n}{2} \rceil) + \Theta(1), & n \geq 2, \\ \Theta(1), & \text{διαφοετικά.} \end{cases} \tag{7}$$

■

Άσκηση 4. 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 $i = 1, \ldots, n$,

$$a'_{(i+k) \bmod n} = a_i, \text{ αν } (i+k) \bmod n \neq 0, \tag{8}$$

$$a'_n = a_i, \text{ αν } (i+k) \bmod n = 0. \tag{9}$$

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,
2. θεωρώντας το k άγνωστο,
3. θεωρώντας το k άγνωστο και επιπλέον ισχύει ότι όλα τα στοιχεία της σειράς είναι διαφορετικά μεταξύ τους.

Για κάθε έναν αλγόριθμο που εκπονήσατε, να διατυπώσετε τη σχέση που περιγράφει τον αριθμό των ΣΥΒ που εκτελούνται.

Λύση.

Η αρχική σειρά είναι ταξινομημένη και επομένως το μεγαλύτερο στοιχείο σε αυτή βρίσκεται στην θέση n. Εφόσον το k είναι γνωστό, με βάση τις (8) και (9), 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 (4) υλοποιεί την ιδέα. Όπως είναι προφανές εκτελείται σταθερός αριθμός

Αλγόριθμος 4 Γνωστό k

Απαιτείται: ακέραιοι $n, k > 0$, ταξινομημένος πίνακας A με στοιχεία στις θέσεις από 1 ως n κυλιόμενα κατά k θέσεις.
Επιστρέφεται: μεγαλύτερο στοιχείο της σειράς

```
1: function 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
7: end function
```

στοιχειωδών πράξεων και επομένως ο αλγόριθμος είναι τάξης $\Theta(1)$.

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. Στη συνέχεια θα συγκρίνουμε τα δύο μεγαλύτερα αυτά στοιχεία μεταξύ τους και θα επιστρέψουμε το μεγαλύτερο. Ο Αλγόριθμος (5) υλοποιεί την ιδέα. Ο αριθμός των στοιχειωδών

Αλγόριθμος 5 Άγνωστο k

Απαιτείται: ακέραιοι $lo, hi, (1 \leq lo \leq hi)$, ταξινομημένος πίνακας A με στοιχεία στις θέσεις από lo ως hi κυλιόμενα κατά k θέσεις (;agnvsto k).
Επιστρέφεται: μεγαλύτερο στοιχείο της σειράς

```
1: function 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 + 1hi);
8:     if max_left > max _right then
9:         return max_left;
10:    else
11:        return max_right;
12:    end if
13: end function
```

πράξεων δίνεται από την αναδρομική σχέση

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

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 περίπου 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). Ο αλγόριθμος απεικονίζεται σε συντομία παρακάτω.

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

Η συνάρτηση Find_Max_Bs υλοποιεί την περίπτωση (β) αναδρομικά. Η συνάρτηση ολοκληρώνεται όταν ο πίνακας αποτελείται ουσιαστικά από δύο στοιχεία.

Η σχέση που περιγράφει τον αριθμό των ΣΥΒ είναι

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

■

Algorithm 6 Άγνωστο k διαφορετικά στοιχεία, περίπτωση (β)

Απαιτείται: ακέραιοι $lo, hi, (1 \leq lo \leq hi)$, ταξινομημένος πίνακας A με διαφορετικά στοιχεία στις θέσεις από lo ως hi κυλιόμενα κατά k θέσεις (;agnvsto k).

Επιστρέφεται: μεγαλύτερο στοιχείο της σειράς

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