Τμήμα Μηχανικών Πληροφορικής & Υπολογιστών

Μάθημα: Σχεδίαση & Ανάλυση Αλγορίθμων　　　　Σεπτέμβριος 2019

Ερωτήματα

Ερώτημα 1 (20%)**.**

1. Θεωρούμε συναρτήσεις $f_i(n), g_i(n) : \mathbb{N} \to \mathbb{N}$ τέτοιες ώστε $f_i = O(g_i)$ όπου $i = 1, 2$. Να δείξετε ότι $f_1(n) + f_2(n) = O(g_1(n) + g_2(n))$.
2. Έστω $f(n) = n + \sqrt{n}, g(n) = \sqrt{n}\log_{10} n$. Ποια(ες) από τις παρακάτω προτάσεις είναι αληθείς;

   (α') $f(n) = o(g(n))$

   (β') $f(n) = \omega(g(n))$

   Να αιτιολογήσετε τις απαντήσεις σας.

Απάντηση

1. Για $i = 1, 2$,

$$f_i(n) = O(g_i(n)) \Rightarrow \exists a_i, n_0^i > 0, f_i(n) \le a_i g_i(n), \forall n \ge n_0^i. \tag{1}$$

Προσθέτοντας κατά μέλη την (1) για $i = 1, 2$, παίρνουμε

$$f_1(n) + f_2(n) \le a_1 g_1(n) + a_2 g_2(n), \forall n \ge n_0, \tag{2}$$

όπου $n_0 = \max\{n_0^1, n_0^2\}$. Έστω $a_0 = \max\{a_1, a_2\}$. Από (2) έχουμε το ζητούμενο:

$$f_1(n) + f_2(n) \le a_0(g_1(n) + g_2(n)), \forall n \ge n_0.$$

2.

$$\begin{aligned}\lim \frac{f(n)}{g(n)} &= \lim \frac{n + \sqrt{n}}{\sqrt{n}\log_{10} n} \\ &= \lim \frac{\sqrt{n}}{\log_{10} n} + \lim \frac{1}{\log_{10} n} \\ &= \lim \frac{\left[\sqrt{n}\right]'}{[\log_{10} n]'} \end{aligned} \tag{3}$$

Ως γνωστόν

$$\left[\sqrt{n}\right]' = \frac{1}{2\sqrt{n}}, \tag{4}$$

$$[\log_{10} n]' = \frac{1}{n \ln 10} \tag{5}$$

Αντικαθιστώντας από τις (4) και (5) στηγ (3), παίρνουμε

$$\lim \frac{f(n)}{g(n)} = \lim \frac{\frac{1}{2\sqrt{n}}}{\frac{1}{n\ln 10}} = lim \frac{\ln 10}{2}\sqrt{n} = \infty.$$

Επομένως είναι αληθής η (β) και επειδή $o(g(n)) \cap \omega(g(n)) = \emptyset$ είναι ψευδής η (α).

Ερώτημα 2 (20%)**.** Να δώσετε μία ασυμπτωτική εκτίμηση για τις συναρτήσεις

1. $T(n) = 2T(\frac{n}{3}) + \lg n$,

2. $T(n) = T(\frac{n}{4}) + T(\frac{2n}{4}) + n$.

Απάντηση

1. Θα εφαρμόσουμε το Κεντρικό Θεώρημα. Παρατηρούμε ότι $a = 2, b = 3$. Για την πρώτη περίπτωση του Κεντρικού Θεωρήματος, εξετάζουμε αν υπάρχει $\epsilon > 0$ τέτοιο ώστε $f(n) = \lg n = O(n^{\lg_3 2-\epsilon})$ ή ισοδύναμα

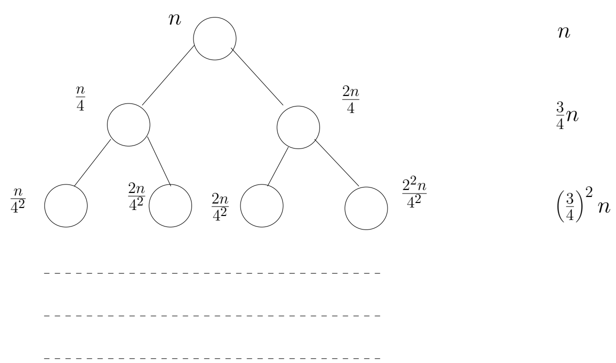
$$\lg n \leq c \cdot n^{\lg_3 2-\epsilon}, \forall n \geq n_0. \tag{6}$$

Αν θεωρήσουμε (αυθαίρετα) $c = 1$, θα πρέπει να εξετάσουμε αν υπάρχει $\epsilon > 0$, ώστε να ισχύει η (6) για κάποιο n_0. Παρατηρούμε ότι $\lg n \leq n^{\frac{1}{2}}$ για $n \geq n_0 = 4$. Άρα θα πρέπει να εξετάσσουμε αν υπάρχει $\epsilon > 0$ τέτοιο ώστε $\lg_3 2 - \epsilon = \frac{1}{2}$. Ισοδύναμα,

$$\lg_3 2 - \frac{1}{2} = \epsilon \Rightarrow \lg_3 2 - \lg_3 3^{\frac{1}{2}} = \epsilon \Rightarrow$$
$$\epsilon = \lg_3 \frac{2}{\sqrt{3}} = 0.13029.$$

Επομένως είμαστε στην πρώτη περίπτωση του Κεντρικού Θεωρήματος και άρα $T(n) = \Theta(n^{\lg_3 2})$.

2. Θα εφαρμόσουμε τη μέθοδο του δένδρου αναδρομής. Στο Σχήμα 1 απεικονίζεται το δένδρο για τη συγκεκριμένη συνάρτηση. Παρατηρούμε ότι η συνδρομή του επιπέδου k στη συνάρτηση είναι $(3/4)^k n$. Επομένως η συνάρτηση φράζεται από τα επάνω



Σχήμα 1: Δένδρο αναδρομής - Ερώτημα 2β

ως εξής:

$$T(n) \leq \sum_{k=0}^{\infty} \left(\frac{3}{4}\right)^k n = n \frac{1}{1-\frac{3}{4}} \Rightarrow$$
$$T(n) \leq 4n.$$

Επομένως $T(n) = O(n)$. Επίσης επειδή $T(n) \geq n$ έχουμε $T(n) = \Theta(n)$

Ερώτημα 3 (20%)**.** Δίνονται δύο ταξινομημένοι, κατά αύξουσα σειρά, πίνακες A, B που περιέχουν ακεραίους στις θέσεις 1 ως n και 1 ως m, αντίστοιχα. Ζητείται να εκπονηθεί αλγόριθμος ο οποίος να μετράει για κάθε στοιχείο του πίνακα B τον αριθμό των στοιχείων του πίνακα A που είναι μεγαλύτερα από αυτό. Να μελετήσετε την ασυμπτωτική συμπεριφορά του αλγόριθμου σας.

Απάντηση Ο Αλγόριθμος 1 με τη μορφή συνάρτησης υλοποιεί το ζητούμενο. Ο συνολικός αριθμός των ζητουμένων στοιχείων επιστρέφεται από τη συνάρτηση.

Η πολυπλοκότητα του Αλγόριθμου 'υπαγορεύεται' από τη γραμμή 4: είναι ίση με $\Theta(k)$ όπου $k = \min\{n, m\}$.

Ερώτημα 4 (40%)**.** Δίνεται πίνακας A ο οποίος περιέχει ακέραιους στις θέσεις 1 ως n. Ένα ζευγάρι στοιχείων του πίνακα $A[i], A[j]$ βρίσκεται σε 'αντιστροφή' αν $i < j$ αλλά $A[i] > A[j]$. Να εκπονήσετε έναν αλγόριθμο τύπου 'διαίρει και βασίλευε' ο οποίος να μετράει τον αριθμό των ζευγαριών που βρίσκεται σε αντιστροφή σε $\Theta(n \lg n)$ βήματα. Να αποδείξετε την πολυπλοκότητα του ζητούμενου αλγόριθμου.

Απάντηση Δεδομένης της σειράς των ακεραίων αριθμών, το βήμα 'διαίρει' συνίσταται στο να διαχωρίσουμε τη σειρά σε δύο σειρές η κάθε μία μεγέθους μισού της αρχικής και να μετρήσουμε τον αριθμό των ζευγαριών που βρίσκονται σε αντιστροφή στην κάθε σειρά. Στη συνέχεια θα πρέπει να μετρήσουμε των αριθμό των ζευγαριών που βρίσκονται σε αντιστροφή ανάμεσα στις δύο σειρές. Αυτό μπορεί να πραγματοποιηθεί με τη χρήση του αλγόριθμου του προηγούμενου ερωτήματος αν οι σειρές είναι ταξινομημένες. Οι σειρές ταξινομούνται με τη μέθοδο της συγχώνευσης (χρησιμοποιείται τη ρουτίνα merge (δες διαφάνειες περί Ταξινόμησης και Συγχώνευσης Πινάκων». Ο Αλγόριθμος 2 υλοποιεί την ιδέα.

Η πολυπλοκότητα του Αλγόριθμου είναι η ζητούμενη και η απόδειξη είναι αντίστοιχη με αυτή της Άσκησης 3 στο φυλλάδιο 'Διαίρει-και-Βασίλευε Ασκήσεις'.

Algorithm 1 Αλγόριθμος - Ερώτημα 3

```
int num_elements(int A[], int n, int B[], int m)
nm ← 0;
i ← 1; j ← 1;
while i ≤ nand j ≤ m do
    while A[i] ≤ A[j] do
        i + +;
        if i > n then
            break;
        end if
    end while
    nm ← mn + n − (i − 1);
    j + +;
end while
return nm;
```

Algorithm 2 Αλγόριθμος - Ερώτημα 4

```
int inv_count(int A[], int lb, int ub)
int B[MAXN], B1[MAXN], B2[MAXN];
if ub − lb ≥ 1 then
    mid ← ⌈(ub+lb)/2⌉;
    p1 ←inv_count(A, lb, mid);
    p2 ←inv_count(A, mid + 1, ub);
    B1[0] ← mid − lb + 1; B2[0] ← ub − (mid + 1) + 1;
    for j ← lb; j ≤ mid; j + + do
        B1[B1[0]] ← A[j];
    end for
    for j ← mid + 1; j ≤ ub; j + + do
        B2[B2[0]] ← A[j];
    end for
    p3 ←num_elements(B1, B1[0], B2, B2[0]);
    merge(B, B1, B2);
    for j ← lb; j ≤ ub; j + + do
        A[j] ← B[j − lb + 1];
    end for
    return p1 + p2 + p3;
end if
return 0
```