Τμήμα Μηχανικών Πληροφορικής & Υπολογιστών

Μάθημα: Σχεδίαση & Ανάλυση Αλγορίθμων Ιούνιος 2020

Ερωτήματα

Ερώτημα 1. (4 βαθμοί)
Έστω συναρτήσεις $f, g, h : \mathbb{N} \to \mathbb{R}_+$. Να δείξετε ότι

1. $f(n) = \Theta(f(n))$,
2. $f(n) = \Theta(g(n))$ αν και μόνο αν $\Theta(f(n)) = \Theta(g(n))$,
3. αν $f(n) = \Theta(g(n))$ και $g(n) = \Theta(h(n))$ τότε $f(n) = \Theta(h(n))$.

Απάντηση

1. Το ζητούμενο ισχύει τετριμμένα αφού για κάθε $n \in \mathbb{N}$ και κάθε συνάρτηση $f(n)$ ισχύει ότι
$$a_1 \cdot f(n) \le f(n) \le a_2 \cdot f(n),$$
με οποιεσδήποτε σταθερές $a_1, a_2 > 0$ τέτοιες ώστε $a_1 \le 1 \le a_2$.

2. ($\Rightarrow$) Εφόσον $f(n) = \Theta(g(n))$ ισχύει ότι υπάρχουν σταθερές $a_1, a_2, n_1 > 0$, τέτοιες ώστε
$$a_1 \cdot g(n) \le f(n) \le a_2 \cdot g(n), \quad \forall n \ge n_1. \tag{1}$$
Έστω $v(n)$ μία συνάρτηση της κλάσης $\Theta(f(n))$. Τότε υπάρχουν σταθερές $k_1, k_2, n_2 > 0$, τέτοιες ώστε
$$k_1 \cdot f(n) \le v(n) \le k_2 \cdot f(n), \quad \forall n \ge n_2.$$
Αντικαθιστώντας στην παραπάνω ανισότητα το κάτω και το πάνω φράγμα της $f(n)$ από την (1) έχουμε
$$\begin{aligned} &k_1 \cdot a_1 \cdot g(n) \le v(n) \le k_2 \cdot a_2 \cdot g(n), \quad \forall n \ge \max\{n_1, n_2\}, \\ \Rightarrow &v(n) = \Theta(g(n)) \\ \Rightarrow &\Theta(f(n)) \subseteq \Theta(g(n)). \end{aligned} \tag{2}$$
Παρατηρήστε ότι (1) συνεπάγεται
$$\lambda_1 \cdot f(n) \le g(n) \le \lambda_2 \cdot f(n), \quad \forall n \ge n_1, \tag{3}$$
όπου $\lambda_1 = \frac{1}{a_2} > 0, \lambda_2 = \frac{1}{a_1} > 0$. Θεωρούμε μία οποιαδήποτε συνάρτηση $u(n) = \Theta(g(n))$. Μέσω της (3) μπορούμε να δείξουμε, με αντίστοιχο τρόπο όπως στην απόδειξη της (2), ότι
$$u(n) = \Theta(f(n)) \Rightarrow \Theta(g(n)) \subseteq \Theta(f(n)). \tag{4}$$
Από (2), (4) έχουμε
$$\Theta(g(n)) = \Theta(f(n)).$$
($\Leftarrow$) Στο Ερώτημα 1.1 δείξαμε ότι $f(n) = \Theta(f(n))$. Άρα από την υπόθεση που μας λέει ότι οι κλάσεις $\Theta(f(n))$ και $\Theta(g(n))$ ταυτίζονται ($\Theta(f(n)) = \Theta(g(n))$) έχουμε ότι $f(n) = \Theta(g(n))$.

3. Εφόσον $g(n) = \Theta(h(n))$ αυτό σημαίνει ότι υπάρχουν σταθερές $b_1, b_2, n_3 > 0$ τέτοιες ώστε
$$b_1 \cdot h(n) \le g(n) \le b_2 \cdot h(n), \quad \forall n \ge n_3. \tag{5}$$
Από (1), (5) έχουμε ότι
$$c_1 \cdot h(n) \le f(n) \le c_2 \cdot h(n), \quad \forall n \ge \max\{n_1, n_3\},$$
όπου $c_1 = a_1 b_1 > 0, c_2 = a_2 b_2 > 0$. Άρα $f(n) = \Theta(h(n))$.

Ερώτημα 2. (3 βαθμοί)
Να δείξετε ότι
$$\Theta(\lg(n-1)) = \Theta(\log n),$$
όπου

- $\log n$: ο δεκαδικός λογάριθμος του n, και,
- $\lg n$: ο δυαδικός λογάριθμος του n.

Υπόδειξη: Για την απόδειξη του ζητούμενου, μπορείτε να θεωρήσετε ότι ισχύουν οι προτάσεις του Ερωτήματος 1 (ακόμα και αν δεν έχετε καταφέρει να τις αποδείξετε.)

Απάντηση
Έχουμε
$$n-1 < n \Rightarrow \lg(n-1) < \lg n. \tag{6}$$
Επίσης, για $n \geq 4$,
$$n-1 \geq \sqrt{n} \Rightarrow \lg(n-1) \geq \frac{1}{2}\lg n. \tag{7}$$
(6), (7) συνεπάγεται ότι
$$\lg(n-1) = \Theta(\lg n). \tag{8}$$
Από την ταυτότητα αλλαγής βάσης λογαρίθμων έχουμε
$$\lg n = \frac{\log n}{\log 2} = \frac{1}{\log 2}\log n \Rightarrow \lg n = \Theta(\log n), \tag{9}$$
αφού $\frac{1}{\log 2} > 0$. Από την πρόταση του Ερωτήματος 1.3 έχουμε ότι το αριστερό μέλος της (8) είναι ίσο με το δεξί μέλος (9), δηλαδή
$$\lg(n-1) = \Theta(\log n).$$
Η παραπάνω σχέση με τη χρήση της πρότασης του Ερωτήματος 1.2 συνεπάγεται $\Theta(\lg(n-1)) = \Theta(\log n)$.

Ερώτημα 3. (4 βαθμοί)
Έστω ένα υπολογιστικό σύστημα το οποίο υποστηρίζει αριθμητικές πράξεις ΜΟΝΟ ανάμεσα σε ακεραίους. Επίσης στο σύστημα αυτό ΔΕΝ επιτρέπεται η πράξη του πολλαπλασιασμού - οι αριθμητικοί τελεστές που μπορούν να χρησιμοποιηθούν για δύο ακεραίους a, b είναι:

- $+$: πρόσθεση (χρησιμοποιείται για τον υπολογισμό του αθροίσματος $a+b$),
- $-$: αφαίρεση (χρησιμοποιείται για τον υπολογισμό της διαφοράς $a-b$),
- $\div$: πηλίκο (χρησιμοποιείται για τον υπολογισμό του πηλίκου της διαίρεσης του a με το b - συμβολίζεται $a \div b$),
- mod: υπόλοιπο (χρησιμοποιείται για τον υπολογισμό του υπόλοιπου της διαίρεσης του a με το b - συμβολίζεται $a \bmod b$).

1. Με τη χρήση αποκλειστικά και μόνο των παραπάνω αριθμητικών τελεστών (όχι απαραίτητα όλων) να εκπονήσετε αλγόριθμο ο οποίος να υπολογίζει το γινόμενο δύο φυσικών αριθμών k, n. Η πολυπλοκότητα του αλγόριθμου σας πρέπει να είναι $\Theta(\lg k)$.
Υπόδειξη: Πριν την κωδικοποίηση του αλγόριθμου να περιγράψετε με λόγια την κεντρική ιδέα που αυτός υλοποιεί.
2. Να αποδείξετε την παραπάνω πολυπλοκότητα του αλγόριθμου σας.
3. Μπορεί να βελτιωθεί ο αλγόριθμος που εκπονήσατε; Εξηγείστε.

Απάντηση

1. Ο ζητούμενος αλγόριθμος περιγράφεται από την αναδρομική συνάρτηση **product**.

```
int product(int k, int n)
if k = 1 then
   return n;
end if
y ←product(k ÷ 2, n);
if k mod 2 = 0 then
   return y + y;
else
```

```
    return y + y + n;
end if
```

2. Παρατηρούμε ότι η παραπάνω συνάρτηση εκτελεί σταθερό αριθμό στοιχειωδών υπολογιστικών βημάτων (αριθμητικών, λογικών πράξεων και εκχωρήσεων) σε κάθε κλήση και χρησιμοποιεί τη λύση ενός στιγμιότυπου μεγέθους $\lfloor \frac{k}{2} \rfloor$ του αρχικού αν $k > 1$. Για $k = 1$, εκτελεί μία σύγκριση. Επομένως ο αριθμός των στοιχειωδών πράξεων δίνεται από την αναδρομική σχέση

$$T(k) = \begin{cases} T(\lfloor \frac{k}{2} \rfloor) + \Theta(1), & k > 1, \\ 1, & k = 1. \end{cases}$$

   Επομένως $T(k) = \Theta(\lg k)$ (απόδειξη ίδια με αυτή της δυαδικής αναζήτησης - δες φυλλάδιο ασκήσεων).

3. Ο παραπάνω αλγόριθμος μπορεί να βελτιωθεί στην περίπτωση που $k > n$. Δηλαδή η αρχική κλήση της συνάρτησης μπορεί να γίνει από τις ακόλουθες γραμμές κώδικα

```
if k ≤ n then
ginomeno ←product(k, n);
else
ginomeno ←product(n, k);
end if
```

   Στην περίπτωση αυτή επιτυγχάνεται πολυπλοκότητα $O(\min\{\lg k, \lg n\})$.