1ο Φροντιστήριο ΗΥ240

Άσκηση 1

Αποδείξτε τη μεταβατική και τη συμμετρική ιδιότητα του Θ.

Λύση:

Μεταβατική Ιδιότητα (ορισμός): Αν $f(n) = \Theta(g(n))$ και $g(n) = \Theta(h(n))$ τότε $f(n)=\Theta(h(n))$.

Για να ισχύει $f(n)= \Theta(h(n))$ πρέπει να δείξουμε ότι $f(n)= O(h(n))$ και ότι $f(n)= \Omega(h(n))$.

- Αποδεικνύω πρώτα ότι $f(n)= O(h(n))$.

Αφού $f(n) = \Theta(g(n)) \Rightarrow f(n)= O(g(n))$,

άρα $\exists\ c_1 \in R^+$ και $n_1 \geq 0$ τ.ω $f(n) \leq c_1 \cdot g(n)$, $\forall\ n \geq n_1$ (1)

Αφού $g(n) = \Theta(h(n)) \Rightarrow g(n)= O(h(n))$,

άρα $\exists\ c_2 \in R^+$ και $n_2 \geq 0$ τ.ω $g(n) \leq c_2 \cdot h(n)$, $\forall\ n \geq n_2$ (2)

Επιλέγω $n_0 = \max\{n_1,n_2\}$ και τότε (1) + (2) $\Rightarrow f(n) \leq c_1 \cdot g(n) \leq c_1 \cdot c_2 \cdot h(n)$, $\forall\ n \geq n_0$

Επιλέγω $c= c_1 \cdot c_2$ και τότε $\exists\ c\ (=c_1 \cdot c_2)$ και $n_0 = \max\{n_1,n_2\}$ τ.ω $f(n) \leq c \cdot h(n)$,

$\forall\ n \geq n_0$, Άρα $f(n) = O(h(n))$

- Αποδεικνύω τώρα ότι $f(n)= \Omega(h(n))$.

Αφού $f(n) = \Theta(g(n)) \rightarrow f(n)= \Omega(g(n))$,

άρα $\exists\ c_1 \in R^+$ και $n_1 \geq 0$ τ.ω $f(n) \leq c_1 \cdot g(n)$, $\forall\ n \geq n_1$ (1)

Αφού $g(n) = \Theta(h(n)) \rightarrow g(n)= \Omega(h(n))$,

άρα $\exists\ c_2 \in R^+$ και $n_2 \geq 0$ τ.ω $g(n) \leq c_2 \cdot h(n)$, $\forall\ n \geq n_2$ (2)

Επιλέγω $n_0 = \max\{n_1,n_2\}$ και τότε (1) + (2) $\rightarrow f(n) \geq c_1 \cdot g(n) \geq c_1 \cdot c_2 \cdot h(n)$, $\forall\ n \geq n_0$

Επιλέγω $c= c_1 \cdot c_2$ και τότε $\exists\ c\ (=c_1 \cdot c_2)$ και $n_0 = \max\{n_1,n_2\}$ τ.ω $f(n) \geq c \cdot h(n)$,

$\forall\ n \geq n_0$, Άρα $f(n) = \Omega(h(n))$

Συμμετρική ιδιότητα (ορισμός): f(n) = Θ(g(n)) αν και μόνο αν g(n) = Θ(f(n)).

Αν f(n) = Θ(g(n)) τότε πρέπει g(n) = Θ(f(n)).

(Αν g(n) = Θ(f(n)) τότε πρέπει f(n) = Θ(g(n)) , είναι συμμετρική!)

Για να δείξω ότι g(n) = Θ(f(n)) πρέπει να δείξω ότι g(n) = O(f(n)) και g(n) = Ω(f(n)) (1)

Αφού f(n) = Θ(g(n)) τότε f(n) = O(g(n)). Άρα $\exists$ $c_1 \in R^+$ και $n_1 \geq 0$ τ.ω $f(n) \leq c_1 \cdot g(n)$, $\forall$ $n \geq n_1 \leftrightarrow g(n) \geq 1/c_1 \cdot f(n)$, $\forall$ $n \geq n_1$

Άρα αν επιλέξω $c = 1/c_1$ και $n_0 = n_1$ προκύπτει ότι $g(n) \geq c \cdot f(n)$, $\forall$ $n \geq n_0$.

Άρα g(n) = Ω(f(n)). (2)

Αφού f(n) = Θ(g(n)) τότε f(n) = Ω(g(n)). Άρα $\exists$ $c_2 \in R^+$ και $n_2 \geq 0$ τ.ω $f(n) \leq c_2 \cdot g(n)$, $\forall$ $n \geq n_2 \leftrightarrow g(n) \leq 1/c_2 \cdot f(n)$, $\forall$ $n \geq n_2$

Άρα αν επιλέξω $c = 1/c_2$ και $n_0 = n_2$ προκύπτει ότι $g(n) \leq c \cdot f(n)$, $\forall$ $n \geq n_0$.

Άρα g(n) = O(f(n)). (3)

(2) + (3) → g(n) = Θ(f(n)) (όπως ζητείται)

(απόδειξη του → ομοίως και για το αντίστροφο)

Άσκηση 2

i) Ισχύει ότι $\sqrt{n^5} \log(\sqrt{n^5}) = O(n^3)$?
ii) Ισχύει ότι $\log(\sqrt{n}) = \Theta(\log(n))$?

Λύση:

i) Εξετάζω εάν $\sqrt{n^5} \log(\sqrt{n^5}) \in O(n^3)$

Αναζητώ $c \in R^+$ και ακέραιο $n_0 \geq 0$ τ.ω $\sqrt{n^5} \log(\sqrt{n^5}) \leq c \cdot n^3$, $\forall$ $n \geq n_0$

$\leftrightarrow$ $n^{5/2} \cdot \log n^{5/2} \leq c \cdot n^3 \leftrightarrow n^{2.5} \cdot 2.5 \cdot \log(n) \leq c \cdot n^3 \leftrightarrow 2.5 \cdot n^{2.5} \cdot \log(n) \leq 2.5 \cdot n^{2.5} \cdot \sqrt{n} = 2.5\, n^3$, $\forall$ $n \geq 4$

Αν επιλέξω $c = 2.5$ και $n_0 = 4$ προκύπτει ότι $\sqrt{n^5} \log(\sqrt{n^5}) \leq c \cdot n^3$, $\forall$ $n \geq n_0$

Άρα $\sqrt{n^5} \log(\sqrt{n^5}) \in O(n^3)$

ii) Εξετάζω εάν log $(\sqrt{n}) \in O(\log(n))$

Αναζητώ c $\in R^+$ και ακέραιο $n_0 \geq 0$ τ.ω log $(\sqrt{n}) \leq c \cdot \log(n)$, $\forall\ n \geq n_0$

$\leftrightarrow$ log $n^{1/2} \leq c \cdot \log(n) \leftrightarrow$ ½ log(n) $\leq c \cdot \log(n) \leftrightarrow 1/2 \leq c$

Αν επιλέξω c = 1/2 και $n_0 = 1$ προκύπτει ότι log $(\sqrt{n}) \leq c \cdot \log(n)$, $\forall\ n \geq n_0$

Άρα log $(\sqrt{n}) \in O(\log(n))$

(Ομοίως εξετάζω και το εάν log $(\sqrt{n}) \in \Omega(\log(n))$)

<u>Άσκηση 3</u>

a) Αποδείξτε επαγωγικά ότι αν T(0) = 0 και T(n) = 2T(n-1) + 1, n>0 τότε $T(n) = 2^n - 1$

b) Θεωρήστε τη συνάρτηση f : N→ N που ορίζεται ως εξής : f(0) =1, f(1)= 2 και

$f(n)= 4\ f(n-2) +2^n$ αν n>1. Αποδείξτε επαγωγικά ότι για κάθε ακέραιο n≥3 ισχύει

$f(n) \leq 3 \cdot n \cdot 2^{n-2}$

Λύση:

a)

<u>Βάση επαγωγής:</u> για n=1, T(1)= 2T(1-1) + 1 = 2T(0) +1 = 0+1=1

$T(1) = 2^1 -1 = 2-1 =1$, άρα ισχύει για n=1

<u>Επαγωγική Υπόθεση:</u> έστω ότι ισχύει για n= k-1. Θα δείξω ότι ισχύει για n = k.

<u>Επαγωγικό Βήμα:</u> $T(k) = 2T(k-1) +1 = 2(2^{k-1} – 1) +1 = 2^k -2 +1 =$ <u>2^k-1</u> (απεδείχθη)

b)

<u>Βάση επαγωγής:</u> για n=3, $f(3) = 4\ f(1) +2^3 = 4\cdot\ 2 +8 =16$

$f(3) \leq 3\cdot 3 \cdot 2^1 =18$,άρα ισχύει αφού 16<18

<u>Επαγωγική Υπόθεση:</u> έστω ότι ισχύει για κάθε $3 \leq n \leq k-1$.Θα δείξω ότι ισχύει για n=k

<u>Επαγωγικό Βήμα:</u> $f(k)= 4\cdot f(k-2) +2^k \leq 4(3(k-2) \cdot 2^{k-4}) +2^k$

$= 2^2 (3(k-2) \cdot 2^{k-4}) +2 \cdot 2^{k-1} \leq 2^2 (3(k-2) \cdot 2^{k-4}) +3 \cdot 2^{k-1}$

$= 2^2 (3(k-2) \cdot 2^{k-4}) +3 \cdot 2 \cdot 2^{k-2} = 3 \cdot 2^{k-2} (k-2) +3 \cdot 2^{k-2} \cdot 2$

$= 3 \cdot 2^{k-2} \cdot k =$ <u>$3 \cdot k \cdot 2^{k-2}$</u> (απεδείχθη)

Άσκηση 4

Βρείτε την τάξη (βάσει των συμβολισμών Ο,Ω,Θ) της χρονικής πολυπλοκότητας Τ(n) του ακόλουθου αλγόριθμου

```
Procedure f (integer n){
        for(i=1; i ≤  n; i++)
                for(k=n; k ≤ n+5; k++)
                        x=x+1;
}
```

Λύση:

Για i=1:

k =n
k=n+1
k=n+2
k=n+3
k=n+4
k=n+5

Για i=2

k =n
k=n+1
k=n+2
k=n+3
k=n+4
k=n+5

.
.
.
.
.

Για i=n

k =n
k=n+1
k=n+2
k=n+3
k=n+4
k=n+5

Οπότε η εντολή «x=x+1» εκτελείται : $T(n) = n * 6 = \Theta(n)$

Άσκηση 5

Δίνεται ο αλγόριθμος Binary Search, ο οποίος χρησιμοποιείται για την αναζήτηση ενός στοιχείου σε έναν ήδη ταξινομημένο πίνακα.

```
Integer BinarySearch(A[0…N-1], value, low, high) {
        if( high < low)
                return -1; //not found
        mid = low+(high-low)/2;
        if (A[mid] > value)
                return BinarySearch(A, value, low, mid-1);
        else if (A[mid] < value)
                return BinarySearch(A, value, mid+1, high);
        else
                return mid; //found
}
```

a) Παρουσιάστε σύντομη περιγραφή του τρόπου λειτουργίας του αλγορίθμου.
b) Ιχνηλατήστε την BinarySearch (A,6,0,9) για την περίπτωση που A= [0,1,2,3,4,5,6,7,8,9]. Πρέπει να παρουσιαστούν όλες οι αναδρομικές κλήσεις της BinarySearch με τη σειρά που καλούνται καθώς και οι τιμές των παραμέτρων A, value, low,high σε κάθε κλήση. Πρέπει επίσης να παρουσιαστεί ο χώρος στη μνήμη που κατανέμεται για την εκτέλεση των αναδρομικών κλήσεων της BinarySearch.
c) Παρουσιάστε αναδρομική σχέση που να περιγράφει τη χρονική πολυπλοκότητα T(n) της BinarySearch για την περίπτωση που $n = 2^k$, για κάποιο k (δηλαδή για την περίπτωση που το n είναι μια δύναμη του 2). Τι τάξης είναι η πολυπλοκότητα της BinarySearch, αποδείξτε τον ισχυρισμό σας.

Λύση:

a) Η BinarySearch ακολουθεί την ιδέα του διαίρει και κυρίευε. i)το πρόβλημα διαιρείται σε μικρότερα υποπροβλήματα, ii) το πρόβλημα επιλύεται στα υποπροβλήματα (και πάλι υποδιαιρούνται σε μικρότερα υποπροβλήματα) η διαίρεση αυτή σταματά ,στη BinarySearch, όταν βρεθεί το στοιχείο που έχει δοθεί προς αναζήτηση(value), iii) οι λύσεις των υποπροβλημάτων συνδυάζονται ώστε να παραχθεί η τελική λύση.
   Η συνάρτηση λειτουργεί ως εξής, παίρνει ορίσματα ένα ταξινομημένο πίνακα A, μία τιμή προς αναζήτηση value,ένα κάτω όριο low και ένα πάνω όριο high τα δύο όρια αυτά υποδηλώνουν τα όρια του πίνακα όπου θα γίνει η αναζήτηση,(δηλαδή «ψάξε» στον A[low…high] το value) .
   Μέσα στη συνάρτηση βρίσκει το μεσαίο στοιχείο του πίνακα A[low…high], το mid,και ελέγχει αν το στοιχείο στη θέση A[mid] είναι μεγαλύτερο ή μικρότερο από το value.Στην περίπτωση που είναι μικρότερο κάνει αναδρομική κλήση της BinarySearch(A, value, mid+1, high) δηλαδή ψάχνει το value στο άνω μισό του πίνακα A, ενώ αν είναι μεγαλύτερο κάνει αναδρομική κλήση της BinarySearch(A, value, low, mid-1) δηλαδή ψάχνει το value στο κάτω μισό του πίνακα A. Αν το στοιχείο δε βρεθεί στον πίνακα(δηλαδή φτάνουμε στο σημείο

όπου high < low τότε η συνάρτηση επιστρέφει -1, αν το στοιχείο βρεθεί τότε επιστρέφεται η θέση του πίνακα που βρέθηκε, δηλαδή mid.

b)

Η ιχνηλάτηση:

```
Sorted Array: [0,1,2,3,4,5,6,7,8,9]

BinarySearch ([0,1,2,3,4,5,6,7,8,9], 6, 0,9):
-------------------------------------------------------------
if (9 < 0) --> False-Not Found
value=6
low=0
high=9
mid = 0+(9-0)/2 = 4
if(4>6)--> FALSE
else if(4<6)--> TRUE
      BinarySearch ([0,1,2,3,4,5,6,7,8,9], 6, 5, 9):
      ---------------------------------------------------------
      if (9 < 5) --> False-Not Found
      value=6
      low=5
      high=9
      mid = 5+(9-5)/2 = 7
      if(7>6)--> TRUE
            BinarySearch ([0,1,2,3,4,5,6,7,8,9], 6, 5, 6):
            ------------------------------------------------
                  if (6 < 5) --> False-Not Found
                  value=6
                  low=5
                  high=6
                  mid = 5+(6-5)/2 = 5
                  if(5>6)--> FALSE
                  else if(5<6)--> TRUE
                    BinarySearch ([0,1,2,3,4,5,6,7,8,9], 6, 6, 6):
                    ------------------------------------------------
                        if (6 < 6) --> False-Not Found
                        value=6
                        low=6
                        high=6
                        mid = 6+(6-6)/2 = 6
                        if (6>6)--> TRUE
                        else if(6<6)--> FALSE
                        else
                              return 6 //Found
```

το στοιχείο βρίσκεται στη θέση 6 του πίνακα A[0,1,2,3,4,5,6,7,8,9] -->A[6]=6

Για τη μνήμη:

A= [0,1,2,3,4,5,6,7,8,9] σε όλες τις περιπτώσεις

Value : 6	
Low: 0	BinarySearch(A,6,0,9)
High: 9	
Mid: 4	
Value : 6	
Low: 5	BinarySearch(A,6,5,9)
High: 9	
Mid: 7	
Value : 6	
Low: 5	BinarySearch(A,6,5,5)
High: 6	
Mid: 5	
Value : 6	
Low: 6	BinarySearch(A,6,6,6)
High: 6	
Mid: 6	

c)

T(1)=1

$T(n) = T(n/2) + c_1 = T(n/2^2) + c_1 + c_2$ = (προσθέτω τον όρο Τα(n/2) +c log n φορές)

$= T(n/2^3) + c_1 + c_2 + c_3 = \ldots = T(n/2^{\log n}) + c_1 + c_2 + c_3 + \ldots + c_{\log n}$

$(T(n/2^{\log n}) = T(n/n) = T(1) = 1)$

$= 1 + c_1 + c_2 + c_3 + \ldots + c_{\log n}$

Επιλέγω $c = \max\{ c_1, c_2, c_3, \ldots, c_{\log n}\}$

$T(n) \leq 1 + c(1+1+...+1) =$ [((1+1+...+1) logn φορές)]

$=1 + c * \log n$, άρα $T(n) \in O(\log n)$. (1)

Επιλέγω $c = \min\{c_1, c_2, c_3, \ldots, c_{\log n}\}$

$T(n) \geq 1 + c(1+1+\ldots+1) = [((1+1+\ldots+1) \log n \text{ φορές })]$

$= 1 + c * \log n$, άρα $T(n) \in \Omega(\log n)$. (2)

(1) + (2) $\rightarrow$ $T(n) \in \Theta(\log n)$