Φροντιστήριο Επανάληψης - Λύσεις

Άσκηση 1

(i) Ας υποθέσουμε ότι $n\sqrt{n} \in O(1000n)$. Τότε υπάρχουν n_0, c, τέτοια ώστε

$$n\sqrt{n} \leq 1000cn \text{ για κάθε } n \geq n_0.$$

Επομένως, υπάρχουν n, c_0, τέτοια ώστε

$$\sqrt{n} \leq 1000c \text{ για κάθε } n \geq n_0.$$

Εφόσον το $\sqrt{n}$ είναι μια συνάρτηση η οποία τείνει στο άπειρο καθώς το n τείνει στο άπειρο, είναι αδύνατη η ύπαρξη σταθεράς c που να το φράσσει. Αυτό μας οδηγεί σε αντίφαση και επομένως $n\sqrt{n} \notin O(1000n)$.

(ii) Θέλουμε να εντοπίσουμε n_0, c, τέτοια ώστε

$$2^{n+1} \leq c3^n \text{ για κάθε } n \geq n_0.$$

Προφανώς, για $c = 3$

$$2^{n+1} \leq 3^{n+1} \text{ για κάθε } n \geq 1$$

και το ζητούμενο έπεται.

(iii) Αν $f_1(n) \in \Omega(g_1(n))$ και $f_2(n) \in \Omega(g_2(n))$ τότε $f_1(n) + f_2(n) \in \Omega(\min\{g_1(n), g_2(n)\})$.

Ας υποθέσουμε ότι $f_1(n) \in \Omega(g_1(n))$ και $f_2(n) \in \Omega(g_2(n))$. Τότε υπάρχουν n_1, n_2, c_1, c_2 τέτοια ώστε

$$f_1(n) \geq c_1 \cdot g_1(n), \text{ για κάθε } n \geq n_1 \text{ και}$$

$$f_2(n) \geq c_2 \cdot g_2(n), \text{ για κάθε } n \geq n_2 \text{ και}$$

Από τα πιο πάνω έχουμε:

$$f_1(n)+f_2(n) \geq c_1 \cdot g_1(n) + c_2 \cdot g_2(n), \text{ για κάθε } n \geq n_1, n_2 \text{ και}$$

$$\geq 2 \cdot \min(c_1,c_2) \cdot \min(g_1(n), g_2(n)) \text{ για κάθε } n \geq n_1, n_2.$$

Θέτοντας $c = 2 \cdot \min(c_1,c_2)$ και $n_0 = \max(n_1, n_2)$ έχουμε ότι

$$f_1(n)+f_2(n) \geq c \cdot \min(g_1(n), g_2(n)), \text{ για κάθε } n \geq n_0.$$

Επομένως $f_1(n) + f_2(n) \in \Omega(\min\{g_1(n), g_2(n)\})$ και το ζητούμενο έπεται.

Άσκηση 2

(α) Ο χρόνος εκτέλεσης είναι της τάξης $\Theta(n^4)$. Αποδείξτε το!.

(β) Ο χρόνος εκτέλεσης του εσωτερικού βρόχου είναι ίσος με $5n \in O(n)$. Επομένως ο χρόνος εκτέλεσης της αναδρομικής διαδικασίας δίνεται από την πιο κάτω αναδρομική εξίσωση:

$$T(1) = 1$$

$$T(n) = T(n/4) + n$$

Λύνουμε την εξίσωση με τη μέθοδο της αντικατάστασης:

$$\begin{aligned} T(n) &= T(n/4) + n \\ &= T(n/16) + n/4 + n \\ &= \dots \\ &= T(n/4^i) + n/4^{i-1} + \dots + n \end{aligned}$$

Θέτουμε $k = log_4 n$ (δηλαδή $4^k = n$) και αντικαθιστούμε k για i παίρνοντας ότι:

$$\begin{aligned} T(n) &= T(1) + n/4^{k-1} + \dots + n \\ &= n\,(1/4^k + \dots + 1/4^1 + 1/4^0) \\ &= n\,\frac{1/4^k - 1}{1/4 - 1} = n\frac{1/n - 1}{-3/4} = n\frac{4(n-1)}{3n} \\ &= 4(n-1)/3 \end{aligned}$$

Άσκηση 3

Για την υλοποίηση ενός AVL δένδρου χρησιμοποιούμε την εγγραφή:

```
struct AVLNode{
      int height;
      int key;
      struct AVLNode *left;
      struct AVLNode *right;
}
```

Η ζητούμενη αναδρομική διαδικασία λειτουργεί βάσει της εξής ιδέας: Δεδομένου ότι το δένδρο δεν είναι κενό,

- Αν βρίσκόμαστε σε ύψος k τότε επιστρέφουμε το κλειδί του κόμβου, διαφορετικά,
- Υπολογίζουμε αναδρομικά το ζητούμενο γινόμενο στο αριστερό υπόδενδρο (αν υπάρχει) και πολλαπλασιάζουμε το αποτέλεσμα με το ζητούμενο γινόμενο στο δεξί υπόδενδρο.

```
int Rec_Product(struct AVLNode *p, int k){
     x=1;
     if (p != NULL)
        if (p->height == k)
             x = p->key;
        else
               x = Rec_Product(p->left,k)
                      *Rec_Product(p->right,k);
     return x;
   }
```

(β) Υποθέτουμε την ύπαρξη υλοποίησης στοίβας, και συγκεκριμένα των πράξεων `MakeEmpty()`, `Push(x,S)`, και `Pop(S)`.

Η μη-αναδρομική διαδικασία έχει ως εξής:

1. Θέτουμε prod ίσο με 1 και αρχικοποιούμε μια κενή στοίβα.
2. Εφόσον ο δείκτης δεν είναι NULL και η στοίβα δεν είναι κενή προχωρούμε στο βήμα 3, διαφορετικά επιστρέφουμε την τιμή του prod και τερματίζουμε τη διαδικασία.
3. Αν ο δείκτης δεν είναι NULL, τότε αν το ύψος του κόμβου είναι ίσο με k πολλαπλασιάζουμε την τιμή του prod επί το κλειδί του κόμβου και κάνουμε τον δείκτη μας ίσο με NULL έτσι ώστε να σταματήσουμε τη διερεύνηση στη συγκεκριμένη κατεύθυνση. Παρόμοια, σταματούμε τη διερεύνηση κάνοντας το p = NULL αν βρεθούμε σε κόμβο με ύψος μικρότερο του k. Διαφορετικά, τοποθετούμε στη στοίβα τον δείκτη p->right και προχωρούμε στο p->left. Επαναλαμβάνουμε από το βήμα 2.
4. Αν ο δείκτης είναι NULL ανασύρουμε τον κόμβο κορυφής της στοίβας και συνεχίζουμε με επεξεργασία του δείκτη από το βήμα 2

```
int Product(AVLNode *p, int k){
  prod = 1;
  MakeEmpty(S);
  while(p != NULL OR !IsEmpty(S)){
       if (p)
            if (p->height == k)
                 prod = prod * p->key;
            if (p->height <= k)
                 p = NULL;
            else
                 p = p->left;
                 Push(p->right,S);
       else
            p = Pop(S);
  }
  return prod;
}
```