Εργασία 1

Απαντήσεις

Ονοματεπώνυμο: Παύλος Τομάζος **Α.Μ.:** 1115202200188
Ονοματεπώνυμο: Ευάγγελος Αργυρόπουλος **Α.Μ.:** 1115202200010

Θέμα 1

1. $T(n) = T(n-1) + n \quad \rightarrow \quad \Theta(n^2)$
   Από την αναδρομική εξίσωση $T(n) = T(n-1) + n \Rightarrow T(n) - T(n-1) = n$ προκύπτουν τα ακόλουθα:

$$\left.\begin{array}{l} T(2) - T(1) = 2 \\ T(3) - T(2) = 3 \\ T(4) - T(3) = 4 \\ \dots \\ T(n) - T(n-1) = n \end{array}\right\} \overset{(+)}{\Longrightarrow} T(n) - \cancel{T(n-1)} + \ldots - \cancel{T(2)} + \cancel{T(2)} - T(1) = 2 + 3 + \cdots + n$$

$$\Longrightarrow T(n) - T(1) = (1 + 2 + 3 + ... + n) - 1 \Longrightarrow T(n) - T(1) = \sum_{i=1}^{n} i - 1$$

$$\Longrightarrow T(n) - T(1) = \frac{n(n+1)}{2} - 1$$

   Θεωρούμε $T(1) = c$ όπου $c \in \mathbb{R}$ σταθερά, οπότε:

$$T(n) - c = \frac{n(n+1)}{2} - 1 \Longrightarrow T(n) = \frac{1}{2}n^2 + \frac{1}{2}n + c - 1 = \Theta(n^2).$$

2. $T(n) = T(\sqrt{n}) + \log n \quad \rightarrow \quad \Theta(\log n)$
   Έστω $m = \log n$ και $S(m) = T(2^m)$. Έχουμε ότι:

$$T(2^m) = T(2^{\frac{m}{2}}) + \log 2^m \Rightarrow T(2^m) = T(2^{\frac{m}{2}}) + m \Rightarrow S(m) = S(\frac{m}{2}) + m$$

   Η τελευταία εξίσωση είναι της μορφής:

$$S(m) = \alpha S(\frac{m}{\beta}) + f(m), \quad \alpha = 1, \quad \beta = 2, \quad f(m) = m$$

   Έχουμε ότι $\log_b a = \log_2 1 = 0$. Οπότε, ισχύει: $f(m) = \Omega(m^{\log_b a + \epsilon}) \Rightarrow m = \Omega(m^{\epsilon})$, για αυθαίρετο $\epsilon \in (0, 1]$. Επιπλέον, για $\frac{1}{2} \leq c < 1 \overset{m \to \infty}{\Longrightarrow} \frac{m}{2} \leq cm \iff \alpha f(\frac{m}{\beta}) \leq cf(m)$, ικανοποιώντας και το 2^o κριτήριο της $3^{ης}$ περίπτωσης του θεωρήματος Master.

   Επομένως, $S(m) = \Theta(f(m)) = \Theta(m) \overset{m = \log n}{\Longrightarrow} T(n) = \Theta(\log n)$.

3. $T(n) = 4T(\frac{n}{2}) + n^3 \quad \rightarrow \quad \Theta(n^3)$
Η εξίσωσή είναι της μορφής:

$$T(n) = \alpha T(\frac{n}{\beta}) + f(n), \quad \alpha = 4, \quad \beta = 2, \quad f(n) = n^3$$

Έχουμε ότι $\log_b a = \log_2 4 = 2$. Οπότε, ισχύει: $f(n) = \Omega(n^{\log_b a + \epsilon}) \iff n^3 = \Omega(n^{2+\epsilon})$, για αυθαίρετα επιλεγμένο $\epsilon \in (0, 1]$. Επιπλέον, για $\frac{1}{2} \leq c < 1 \overset{n\to\infty}{\Longrightarrow} \frac{n^3}{2} \leq cn^3 \iff 4\left(\frac{n}{2}\right)^3 \leq cn^3 \iff \alpha f(\frac{n}{\beta}) \leq cf(n)$, ικανοποιώντας και το 2^o κριτήριο της $3^{ης}$ περίπτωσης του θεωρήματος Master.

Επομένως, $T(n) = \Theta(f(n)) = \Theta(n^3)$.

4. $T(n) = \frac{4}{3}T(n^{3/4}) + \log n \quad \rightarrow \quad \Theta(\log n \cdot \log(\log n))$
Έστω $m = \log n$ και $S(m) = T(2^m)$. Έχουμε ότι:

$$T(2^m) = \frac{4}{3}T(2^{3m/4}) + \log 2^m \Rightarrow T(2^m) = \frac{4}{3}T(2^{3m/4}) + m$$

Επομένως, $S(m) = \frac{4}{3}S(\frac{3}{4}m) + m$, που είναι της μορφής:

$$S(m) = \alpha S(\frac{m}{\beta}) + f(m), \quad \alpha = \frac{4}{3}, \quad \beta = \frac{4}{3}, \quad f(m) = m$$

Έχουμε ότι $\log_b a = \log_{\frac{4}{3}} \frac{4}{3} = 1$, οπότε $f(m) = m = \Theta(m) = \Theta(m^1) = \Theta(m^{\log_b a})$. Άρα, σύμφωνα με την $2^η$ περίπτωση του θεωρήματος Master, προκύπτει ότι:

$S(m) = \Theta(m \log m) \overset{m=\log n}{\Longrightarrow} T(n) = \Theta(\log n \cdot \log(\log n))$

5. $T(n) = 16T(\frac{n}{8}) + n^{2/3} \quad \rightarrow \quad \Theta(n^{4/3})$
Η εξίσωση είναι της μορφής:

$$T(n) = \alpha T(\frac{n}{\beta}) + f(n), \quad \alpha = 16, \quad \beta = 8, \quad f(n) = n^{\frac{2}{3}}$$

Έχουμε ότι $\log_b a = \log_8 16 = \frac{\log_2 16}{\log_2 8} = \frac{4}{3}$, οπότε $f(n) = n^{\frac{2}{3}} = O(n^{\frac{4}{3}-\epsilon})$ για οποιοδήποτε αυθαίρετο $\epsilon \in (0, \frac{2}{3}]$, οπότε σύμφωνα με την $1^η$ περίπτωση του θεωρήματος Master, προκύπτει ότι:

$T(n) = \Theta(n^{\log_b a}) = \Theta(n^{\frac{4}{3}})$.

Θέμα 2

1. $T(n) = 5T(n-1) - 6T(n-2), \quad T(0) = 3, \quad T(1) = 11$
Υποθέτοντας ότι $T(n) = x^n$, έχουμε:

$$\begin{aligned} T(n) - 5T(n-1) + 6T(n-2) = 0 \quad &\Rightarrow \quad x^n - 5x^{n-1} + 6x^{n-2} = 0 \\ &\Rightarrow \quad x^{n-2}(x^2 - 5x + 6) = 0 \\ &\Rightarrow \quad x_1 = 2, \quad x_2 = 3 \end{aligned}$$

Συνεπώς, η T θα είναι της μορφής: $T(n) = \lambda_1 x_1^n + \lambda_2 x_2^n \Rightarrow T(n) = \lambda_1 2^n + \lambda_2 3^n$.

$$\begin{cases} T(0) = 3 \\ T(1) = 11 \end{cases} \Longrightarrow \begin{cases} \lambda_1 2^0 + \lambda_2 3^0 = 3 \\ \lambda_1 2^1 + \lambda_2 3^1 = 11 \end{cases} \Longrightarrow \begin{cases} \lambda_1 + \lambda_2 = 3 \\ 2\lambda_1 + 3\lambda_2 = 11 \end{cases} \Longrightarrow \begin{cases} \lambda_1 = -2 \\ \lambda_2 = 5 \end{cases}$$

Επομένως, $T(n) = -2 \cdot 2^n + 5 \cdot 3^n = -2^{n+1} + 5 \cdot 3^n = \Theta(3^n)$

2. $T(n) = 5T(n-1) - 8T(n-2) + 4T(n-3), \quad T(0) = 0, \quad T(1) = 1, \quad T(2) = 5$

   Υποθέτοντας ότι $T(n) = x^n$, έχουμε:

$$\begin{aligned} T(n) - 5T(n-1) + 8T(n-2) - 4T(n-3) = 0 \quad &\Rightarrow \quad x^n - 5x^{n-1} + 8x^{n-2} - 4x^{n-3} = 0 \\ &\Rightarrow \quad x^{n-3}(x^3 - 5x^2 + 8x - 4) = 0 \end{aligned}$$

   Από σχήμα Horner, παρατηρώντας την προφανή ρίζα του χαρακτηριστικού πολυωνύμου ($r = 1$) έχουμε:

1	-5	8	-4	1
	1	-4	4	
1	-4	4	0	

   Οπότε, $(x-1)(x^2 - 4x + 4) = 0 \iff (x-1)(x-2)^2 = 0 \Rightarrow x_1 = 1, x_2 = 2$. Επομένως, εφόσον το πολυώνυμο τρίτου βαθμού έχει διπλή ρίζα το 2 η T θα είναι της μορφής:

$$T(n) = \lambda_1 1^n + \lambda_2 2^n + \lambda_3 n 2^n$$

$$\begin{cases} T(0) = 0 \\ T(1) = 1 \\ T(2) = 5 \end{cases} \Longrightarrow \begin{cases} \lambda_1 + \lambda_2 = 0 \\ \lambda_1 + 2\lambda_2 + 2\lambda_3 = 1 \\ \lambda_1 + 4\lambda_2 + 8\lambda_3 = 5 \end{cases} \Longrightarrow \begin{cases} \lambda_1 = -\lambda_2 \\ \lambda_2 + 2\lambda_3 = 1 \\ 3\lambda_2 + 8\lambda_3 = 5 \end{cases} \Longrightarrow \begin{cases} \lambda_1 = -\lambda_2 \\ -3\lambda_2 - 6\lambda_3 = -3 \\ 3\lambda_2 + 8\lambda_3 = 5 \end{cases} \Longrightarrow$$

$$\begin{cases} \lambda_1 = -\lambda_2 \\ -3\lambda_2 - 6\lambda_3 = -3 \\ 2\lambda_3 = 2 \end{cases} \Longrightarrow \begin{cases} \lambda_1 = -\lambda_2 \\ \lambda_2 = -2\lambda_3 + 1 \\ 2\lambda_3 = 2 \end{cases} \Longrightarrow \begin{cases} \lambda_1 = 1 \\ \lambda_2 = -1 \\ \lambda_3 = 1 \end{cases}$$

   Συνεπώς: $T(n) = 1 - 2^n + n \cdot 2^n = \Theta(n \cdot 2^n)$

Θέμα 3

Θα χρησιμοποιήσουμε τον αριθμό μητρώου 1115202200010 για τους σκοπούς της άσκησης, οπότε τα τρία τελευταία ψηφία που χρειαζόμαστε θα είναι 010 ($x = 0, y = 1, z = 0$). Επομένως,

$$P = (2000 + 130x + 52y)(1500 + 264x + 83z) = (2000 + 130 \cdot 0 + 52 \cdot 1)(1500 + 264 \cdot 0 + 83 \cdot 0) = 2052 \cdot 1500$$

Παρακάτω δίνεται αναλυτικά το δέντρο αναδρομής που προκείπτει από την εφαμοργή του αλγορίθμου Karatsuba-Ofman που υλοποιεί την ιδέα του Gauss των διαφανειών L7 σελ.14, για τον υπολογισμό του γινομένου P. Θεωρούμε ότι ο υπολογισμός αθροισμάτων και το γινόμενο δύο μονοψήφιων αριθμών πραγματοποιούνται σε σταθερό χρόνο. Ακολουθήσαμε, μεθοδικά την σημειογραφία των διαφανειών, δηλαδή:

- $z_1 = \alpha \cdot \gamma$
- $z_2 = \beta \cdot \delta$
- $z_3 = (\alpha + \beta) \cdot (\gamma + \delta)$
- $z = z_3 - z_1 - z_2$
- $P = z_1 \cdot 10^n + z \cdot 10^{n/2} + z_2$

Κάθε κόμβος στο δέντρο αναδρομής περιέχει αναλυτικά τις πληροφορίες του εκάστοτε γινομένου που υλοποιεί. Φύλλα του δένδρου είναι τα γινόμενα που αφορούν μονοψήφιους αριθμούς, ενώ για τους υπόλοιπους κόμβους, η αναδρομή συνεχίζεται στο σημείο υπολογισμού των $\alpha, \beta, \gamma, \delta$. Εκεί υπολογίζονται τα z_1, z_2, z_3 που χρειαζόμαστε ώστε να υπολογίσουμε το z και έπειτα το γινόμενο που μας ενδιαφέρει. Όλοι οι κόμβοι έχουν την παρακάτω μορφή:

Γινόμενο προς υπολογισμό
Έκφραση γινομένου ως $(\alpha \cdot 10^{n/2} + \beta)(\gamma \cdot 10^{n/2} + \delta)$
Οι τιμές των α, β, γ, δ
Υπολογισμός του z συνδοιάζοντας τα αποτελέσματα της αναδρομής
Υπολογισμός του αρχικού γινομένου

Γινόμενο προς υπολογισμό
Αποτέλεσμα σε σταθερό χρόνο

Δέντρο Αναδρομής

$P = 2052 \cdot 1500$
$(20 \cdot 10^2 + 52)(15 \cdot 10^2 + 0)$
$\alpha = 20, \beta = 52, \gamma = 15, \delta = 0$
$z = z_3 - z_1 - z_2 = 780$
$P = z_1 \cdot 10^4 + z \cdot 10^2 + z_2 = 3.078.000$

$z_1 = 20 \cdot 15$
$(2 \cdot 10 + 0)(1 \cdot 10 + 5)$
$\alpha = 2, \beta = 0, \gamma = 1, \delta = 5$
$z' = z'_3 - z'_1 - z'_2 = 10$
$z_1 = z'_1 \cdot 10^2 + z' \cdot 10 + z'_2 = 300$

$z'_1 = 2 \cdot 1$
2

$z'_2 = 0 \cdot 5$
0

$z'_3 = (2 + 0)(1 + 5)$
12

$z_2 = 52 \cdot 0$
$(5 \cdot 10 + 2)(0 \cdot 10 + 0)$
$\alpha = 5, \beta = 2, \gamma = 0, \delta = 0$
$z' = z'_3 - z'_1 - z'_2 = 0$
$z_2 = z'_1 \cdot 10^2 + z' \cdot 10 + z'_2 = 0$

$z'_1 = 5 \cdot 0$
0

$z'_2 = 2 \cdot 0$
0

$z'_3 = (5 + 2)(0 + 0)$
0

$z_3 = (20 + 52)(15 + 0)$
$(7 \cdot 10 + 2)(1 \cdot 10 + 5)$
$\alpha = 7, \beta = 2, \gamma = 1, \delta = 5$
$z' = z'_3 - z'_1 - z'_2 = 37$
$z_3 = z'_1 \cdot 10^2 + z' \cdot 10 + z'_2 = 1.080$

$z'_1 = 7 \cdot 1$
7

$z'_2 = 2 \cdot 5$
10

$z'_3 = (7 + 2)(1 + 5)$
54

Θέμα 4

Algorithm 1 : Strong Reversals → $O(n \log n)$

```
procedure MERGE-AND-COUNT(a, b)          ▷ Sorted arrays a, b with sizes m, n respectively
    r ← 0
    i ← 1
    j ← 1
    while i ≤ m and j ≤ n do                        ▷ Loop to count strong reversals
        if 2b_j < a_i then
            r ← r + m − i + 1        ▷ Add remaining array a elements to inversion count
            j ← j + 1
        else
            i ← i + 1
        end if
    end while
    i ← 1
    j ← 1
    for k ← 1 to m + n do
        if i ≤ m and (j > n or a_i < b_j) then
            c_k ← a_i
            i ← i + 1
        else
            c_k ← b_j
            j ← j + 1
        end if
    end for
    return r, c
end procedure


procedure SORT-AND-COUNT(A, low, high)
    if high = low then         ▷ if the array has only one element return 0 and the array A
        return 0, A
    end if
    mid ← ⌊array_size/2⌋
    (r_A1, A1) ← SORT-AND-COUNT(A, low, mid)      ▷ recursively sort the first half of the array
    (r_A2, A2) ← SORT-AND-COUNT(A, mid + 1, high)                  ▷ Same for it's second half
    (r, A) ← MERGE-AND-COUNT(A1, A2)          ▷ merge the two halves and count the inversions
    return r_A1 + r_A2 + r, A
end procedure
```

Επεξήγηση Αλγορίθμου:
Για την μέτρηση των σημαντικών αντιστροφών ενός πίνακα A, βασιστήκαμε στην γνωστή ταξινόμηση «Διαίρει και Βασίλευε» Merge Sort. O αλγόριθμος διαιρεί τον αρχικό πίνακα σε δύο υποπίνακες ($T(n/2) + T(n/2)$) και εκτελεί δύο ανεξάρτητες μεταξύ τους επαναλήψεις. Η μία επανάληψη αφορά μέτρηση σημαντικών αντιστροφών ($O(n)$) και η άλλη συγχώνευση των δύο ταξινομημένων υποπινάκων ($O(n)$). Συνολικά $O(n) + O(n) = O(n)$. Εύκολα λοιπόν, μπορούμε να διαπιστώσουμε πως η αναδρομική εξίσωση που περιγράφει την διαδικασία είναι η $T(n) = 2T(n/2) + O(n)$ και από θεώρημα Master έχουμε $T(n) = O(n \log n)$. Έχουμε παραδώσει και αντίστοιχο python script.

Θέμα 5

Algorithm 2 : Max difference of items in array $\rightarrow O(n)$

```
procedure MIN-MAX(arr, low, high)
    if low = high then                                  ▷ Check if there is only one element
        return arr[low], arr[high]
    end if
    if high = low + 1 then                              ▷ Check if there are two elements
        if arr[low] < arr[high] then
            return arr[low], arr[high]
        else
            return arr[high], arr[low]
        end if
    end if
    mid ← ⌊(low + high)/2⌋                              ▷ Find the middle element
    min_1, max_1 ←MIN-MAX(arr, low, mid)                ▷ Recur for the left sub-array
    min_2, max_2 ←MIN-MAX(arr, mid + 1, high)           ▷ Recur for the right sub-array
    if min_1 < min_2 then
        min ← min_1
    else
        min ← min_2
    end if
    if max_1 < max_2 then
        max ← max_2
    else
        max ← max_1
    end if
    return min, max
end procedure

procedure MAX-DIFF(arr)                                 ▷ We suppose that the array is not empty
    min, max ←MIN-MAX(arr, 1, N)                        ▷ N is the size of the Array
    return max − min
end procedure
```

Συνοπτική Ανάλυση Αλγορίθμου:

- Για τις περιπτώσεις όπου οι πίνακες έχουν 1 ή 2 στοιχεία ο αλγόριθμος θα επιστρέφει κατευθείαν το ελάχιστο και το μέγιστο (αν είναι ένα το στοιχείο τότε θα επιστρέφει το στοιχείο αυτό δύο φορές (0 συγκρίσεις) ενώ αν είναι δύο, πρώτα θα τα συγκρίνει και μετά θα επιστρέφει πρώτα το ελάχιστο και μετά το μέγιστο (1 σύγκριση)).

- Αν ο πίνακας έχει παραπάνω από δύο στοιχεία, ακολουθούμε μία μέθοδο ¨Διαίρει και Βασίλευε' με την οποία διασπάμε τον πίνακα στη μέση και καλούμε αναδρομικά τον αλγόριθμο για τους δύο υποπίνακες που προκύπτουν. Η αναδρομή θα μας επιστρέφει το ελάχιστο και το μέγιστο του κάθε υποπίνακα. $(T(n/2) + T(n/2))$

- Αφού έχουμε βρεί το ελάχιστο και το μέγιστο από τους δύο υποπίνακες, τα συκρίνουμε για να βρούμε το ολικό ελάχιστο και μέγιστο και τα επιστρέφουμε (2 συγκρίσεις).

Απόδειξη πολυπλοκότητας:
Σύμφωνα με την παραπάνω περιγραφή, εύκολα προκύπτει η αναδρομική εξίσωση που διέπει τον αλγόριθμο:

$$T(n) = \begin{cases} 0, \quad n = 1 \\ 1, \quad n = 2 \\ 2 \cdot T\left(\frac{n}{2}\right) + 2, \quad n > 2 \end{cases}$$

Σύμφωνα με το θεώρημα Master, εφόσον $\log_b a = \log_2 2 = 1$, τότε $f(n) = 2 = O(n^{1-\epsilon})$ για οποιοδήποτε αυθαίρετα επιλεγμένο $\epsilon \in (0, 1]$, οπότε $T(n) = O(n^{\log_2 2}) = O(n)$. Επομένως, πράγματι ικανοποιείται η ζητούμενη προδιαγραφή για την γραμμική απόδοση του αλγόριθμου.