Φύλλο εργασίας 1

ΝΤΕΝΤΑΪ ΜΕΪΓΚΑΝ 12509 ΚΑΙ ΒΙΡΓΙΝΙΑ ΓΚΕΝΟΥΔΗ 12632

April 30, 2025

Contents

1 Άσκηση 1

(i) Απόδειξη ότι $n^2 + 3n \in O(n^2)$

Έχουμε:

$$n^2 + 3n \leq n^2 + 3n^2 = 4n^2, \quad \text{για κάθε } n \geq 1$$

Άρα, επιλέγοντας $c = 4$ και $n_0 = 1$, ισχύει:

$$n^2 + 3n \leq 4n^2$$

Άρα:

$$n^2 + 3n \in O(n^2)$$

(ii) Απόδειξη ότι $2^n \in \omega(n^{10})$

Υπολογίζουμε:

$$\lim_{n\to\infty} \frac{n^{10}}{2^n} = 0$$

Άρα:

$$2^n \in \omega(n^{10})$$

(iii) Απόδειξη ότι $n \log n \in O(n^{3/2})$

Για μεγάλα n ισχύει:

$$\log n \leq n^{0.5}$$

Άρα:

$$n \log n \leq n^{1.5}$$

Άρα:

$$n \log n \in O(n^{3/2})$$

(iv) Απόδειξη ότι $n^5 + 2n^3 \in \Omega(n^5)$

Για κάθε $n \geq 1$ ισχύει:

$$n^5 + 2n^3 \geq n^5$$

Άρα:

$$n^5 + 2n^3 \in \Omega(n^5)$$

(v) Ταξινόμηση συναρτήσεων

Η κατάταξη από τη λιγότερο αποδοτική (μεγαλύτερη αύξηση) προς την περισσότερο αποδοτική (μικρότερη αύξηση) είναι:

1. $n!$
2. $2^{n/2}$
3. $n^{\log n}$
4. $e^{\sqrt{n}}$
5. $n^3 + n^2$
6. $n^2 \log n$
7. $10n$
8. $\sqrt{n}$
9. $\log^3 n$
10. $\log n$

2 Άσκηση 2

Μια ακολουθία A περιέχει $n-2$ μοναδικούς ακεραίους στο διάστημα $[0, n-1]$.

Θέλουμε να βρούμε τους δύο αριθμούς που λείπουν, σε $O(n)$ χρόνο και με $O(1)$ επιπλέον χώρο.

Ιδέα

Αν είχαμε όλους τους αριθμούς από 0 έως $n-1$, το συνολικό άθροισμά τους θα ήταν:

$$S = \frac{n(n-1)}{2}$$

και το συνολικό άθροισμα των τετραγώνων τους:

$$S_2 = \frac{(n-1)n(2n-1)}{6}$$

Υπολογίζοντας:

- το άθροισμα των στοιχείων της ακολουθίας A (έστω S_A),
- και το άθροισμα των τετραγώνων των στοιχείων της A (έστω S_{A2}),

προκύπτει:

$$x + y = S - S_A$$
$$x^2 + y^2 = S_2 - S_{A2}$$

όπου x και y οι δύο χαμένοι αριθμοί.

Επίλυση

Χρησιμοποιούμε την ταυτότητα:

$$(x+y)^2 = x^2 + y^2 + 2xy$$

Άρα:

$$2xy = (x+y)^2 - (x^2 + y^2)$$

οπότε υπολογίζουμε το xy.

Έχοντας $x + y$ και xy, λύνουμε τη δευτεροβάθμια εξίσωση:

$$t^2 - (x+y)t + xy = 0$$

Οι δύο ρίζες της εξίσωσης είναι οι δύο χαμένοι αριθμοί.

Χαρακτηριστικά

- Ο αλγόριθμος έχει χρόνο εκτέλεσης $O(n)$.
- Ο αλγόριθμος χρησιμοποιεί $O(1)$ πρόσθετο χώρο.

3 Άσκηση 3

Έστω n διεργασίες και ένας αριθμός k τέτοιος ώστε $1 < k < n$.

Το κόστος της i-οστής διεργασίας είναι:

- k, αν το i είναι πολλαπλάσιο του k
- 1, διαφορετικά

Απόδειξη ότι το συνολικό κόστος είναι $O(n)$

Οι διεργασίες που έχουν κόστος k είναι όσες είναι πολλαπλάσια του k. Ο αριθμός τους είναι περίπου n/k.

Οι υπόλοιπες διεργασίες έχουν κόστος 1, και είναι περίπου $n - n/k$.

Το συνολικό κόστος υπολογίζεται ως:

$$\text{Συνολικό Κόστος} = (n - n/k) \times 1 + (n/k) \times k$$

Αναπτύσσοντας:

$$= n - \frac{n}{k} + n$$

$$= n(2 - \frac{1}{k})$$

Επειδή το $k > 1$, το $\frac{1}{k}$ είναι μικρό και άρα $2 - \frac{1}{k} \approx 2$.

Άρα:

$$\text{Συνολικό Κόστος} = O(n)$$

Επιμερισμένο κόστος ανά διεργασία

Το επιμερισμένο κόστος ανά διεργασία είναι:

$$\text{Επιμερισμένο Κόστος} = \frac{\text{Συνολικό Κόστος}}{n} = 2 - \frac{1}{k}$$

Άρα το επιμερισμένο κόστος είναι σταθερό:

$$O(1)$$

Συμπέρασμα: Το συνολικό κόστος είναι $O(n)$ και το επιμερισμένο κόστος ανά διεργασία είναι $O(1)$.

4 Άσκηση 4

4.1 Συνάρτηση areEqual(listA, listB)

- Αν το μέγεθος των λιστών είναι διαφορετικό, επιστρέφει false.
- Αλλιώς, ξεκινά ταυτόχρονα από το πρώτο στοιχείο και των δύο λιστών.
- Για κάθε στοιχείο:
  - Αν τα στοιχεία διαφέρουν, επιστρέφει false.
  - Πηγαίνει στο επόμενο στοιχείο και στις δύο λίστες.
- Αν ολοκληρώσει τη σύγκριση χωρίς λάθος, επιστρέφει true.

4.2 Συνάρτηση isSymmetric(list)

- Θέτει δύο δείκτες: έναν στην αρχή (head) και έναν στο τέλος (tail) της λίστας.
- Όσο οι δείκτες δεν συναντιούνται:
  - Αν τα στοιχεία που δείχνουν οι δείκτες είναι διαφορετικά, επιστρέφει false.
  - Διαφορετικά, ο head πηγαίνει στο επόμενο στοιχείο και ο tail στο προηγούμενο στοιχείο.
- Αν όλοι οι έλεγχοι περάσουν, επιστρέφει true.

5 Άσκηση 5

5.1 Συνάρτηση isBalanced(node)

- Αν ο κόμβος είναι null, επιστρέφει 0.
- Υπολογίζει το ύψος του αριστερού υποδέντρου.
- Αν αυτό επιστρέψει -1, επιστρέφει -1.
- Υπολογίζει το ύψος του δεξιού υποδέντρου.
- Αν αυτό επιστρέψει -1, επιστρέφει -1.
- Αν η διαφορά των υψών είναι μεγαλύτερη από 1, επιστρέφει -1.
- Αλλιώς επιστρέφει το μέγιστο ύψος των δύο υποδέντρων συν 1.

Η συνάρτηση isBalanced επιστρέφει true αν το αποτέλεσμα της checkHeight είναι διαφορετικό από -1.

5.2 Συνάρτηση isSymmetric(root)

- Αν η ρίζα είναι null, επιστρέφει true.
- Αλλιώς, καλεί τη isMirror(left, right).
- Η isMirror(t1, t2) λειτουργεί ως εξής:
  - Αν και τα δύο είναι null, επιστρέφει true.
  - Αν μόνο το ένα είναι null, επιστρέφει false.
  - Αν οι τιμές τους είναι διαφορετικές, επιστρέφει false.
  - Αλλιώς, ελέγχει:
    - isMirror(t1.left, t2.right)
    - isMirror(t1.right, t2.left)